Департамент внутренней и кадровой политики Белгородской области
Областное государственное автономное профессиональное образовательное учреждение
«Белгородский индустриальный колледж»

Рассмотрено
ЦК «Информатики и ПОВТ»
Протокол заседания № 1
от «30» августа 2019 г.
Председатель цикловой комиссии
_____Третьяк И.Ю.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине

ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

по специальности
09.02.06 Сетевое и системное администрирование

Квалификация – сетевой и системный администратор

Разработчик преподаватель:
Шершнева Марина Александровна, ОГАПОУ «Белгородский индустриальный колледж

Белгород 2019

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1. Краткая характеристика дисциплины, ее цели и задачи. Место лабораторных работ в курсе дисциплины

Дисциплина ОП.04 Основы алгоритмизации и программирования является частью рабочей основной образовательной программы в соответствии с ФГОС по специальности СПО 09.02.06 «Сетевое и системное администрирование».

Дисциплина изучается в III-V семестрах. В целом рабочей программой предусмотрено 90 часов на выполнение лабораторных работ, что составляет 46 % от обязательной аудиторной нагрузки, которая составляет 196 часов, при этом максимальная нагрузка составляет 220 часов, из них 6 часа приходится на самостоятельную работу обучающихся.

Цель настоящих методических рекомендаций: оказание помощи обучающимся в выполнении лабораторных работ по дисциплине ОП.04 Основы алгоритмизации и программирования, качественное выполнение которых поможет обучающимся освоить обязательный минимум содержания дисциплины и подготовиться к промежуточной аттестации в форме экзамена.

1.2. Организация и порядок проведения лабораторных работ

Лабораторные работы проводятся после изучения теоретического материала. Введение лабораторных работ в учебный процесс служит связующим звеном между теорией и практикой. Они необходимы для закрепления теоретических знаний, а также для получения практических навыков и умений. При проведении лабораторных работ задания, выполняются студентом самостоятельно, с применением знаний и умений, усвоенных на предыдущих занятиях, а также с использованием необходимых пояснений, полученных от преподавателя. Обучающиеся должны иметь методические рекомендации по выполнению лабораторных работ, конспекты лекций, средство для вычислений.

1.3. Общие указания по выполнению лабораторных работ

Курс лабораторных работ по дисциплине ОП.04 Основы алгоритмизации и программирования предусматривает проведение 45 (90 часов) работ, посвященных изучению:

После выполнения лабораторной работы обучающийся к следующему занятию оформляет отчет, который должен содержать:

- название лабораторной работы, ее цель;
- краткие, теоретические сведения;
- номер варианта и задание лабораторной работы, согласно варианту;
- алгоритм и код реализованной программы или описание хода выполнения лабораторной работы;
- результаты выполнения программы;
- вывод о проделанной работе.

Перед выполнением лабораторной работы необходимо получить вводные инструкции преподавателя и внимательно ознакомиться с описанием лабораторной работы.

Внимание! Включать ПК и выполнять какие-либо действия с другим оборудованием допускается ТОЛЬКО с разрешения преподавателя!

При обнаружении признаков неисправности, таких как: появление искрения, дыма, специфического запаха, немедленно отключить все источники электроэнергии и сообщить о случившемся преподавателю.

Лаборатории должны иметь средства пожаротушения. На первом занятии изучаются правила техники безопасности и проводится вводный инструктаж с последующей проверкой его усвоения, о чем свидетельствует запись в журнале по технике безопасности кабинета/лаборатории, подписываемый преподавателем, проводившем инструктаж, и всеми обучающимися.

1.4. Критерии оценки результатов выполнения лабораторных работ

Критериями оценки результатов работы обучающихся являются:

– уровень усвоения обучающимся учебного материала;

– умение обучающегося использовать теоретические знания при выполнении практических задач;

– сформированность общеучебных и профессиональных компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

ПК 1.2. Осуществлять выбор технологии, инструментальных средств и средств вычислительной техники при организации процесса разработки и исследования объектов профессиональной деятельности.

ПК 2.3. Обеспечивать сбор данных для анализа использования и функционирования программно-технических средств компьютерных сетей.

ПК 2.4. Взаимодействовать со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности.

Критерии оценивания лабораторной работы

Оценка	**Критерии оценивания**
5	Работа выполнена в полном объеме с соблюдением необходимой последовательности проведения, содержит результаты и выводы, все записи, таблицы, рисунки, чертежи, графики выполнены аккуратно. Обучающийся владеет теоретическим материалом, формулирует собственные, самостоятельные, обоснованные, представляет полные и развернутые ответы на дополнительные вопросы.
4	Работа выполнена в полном объеме с соблюдением необходимой последовательности проведения, содержит результаты и выводы, все записи, таблицы, рисунки, чертежи, графики выполнены аккуратно. Обучающийся владеет теоретическим материалом, допуская незначительные ошибки на дополнительные вопросы.
3	Работа выполнена в полном объеме, содержит результаты и выводы, все записи, таблицы, рисунки, чертежи, графики выполнены аккуратно. Обучающийся владеет теоретическим материалом на минимально допустимом уровне, допуская ошибки на дополнительные вопросы.
2	Работа выполнена не полностью. Студент практически не владеет теоретическим материалом, допускает ошибки при ответе на дополнительные вопросы.

Тематика лабораторных работ по ОП.04 Основы алгоритмизации и программирования

Номер лабораторной работы	Тема лабораторной работ	Кол-во часов
1	Алгоритмы в математике	2
2	Анализ сложности алгоритмов	2
3	Алгоритм поиска минимального и максимального элементов в массиве	2
4	Сортировка массивов	2
5	Схема примитивной рекурсии	2
6	Операция минимизации	2
7	Частично рекурсивные функции. Общерекурсивные функции	2
8	Знакомство со средой программирования	2
9	Составление программ линейной структуры.	2
10	Составление программ разветвляющейся структуры.	2
11	Составление программ циклической структуры	2
12	Обработка одномерных массивов.	2
13	Обработка двумерных массивов.	2
14	Работа со строками.	2
15	Работа с данными типа множество.	2
16	Файлы последовательного доступа.	2
17	Типизированные файлы.	2
18	Нетипизированные файлы.	2
19	Организация процедур.	2
20	Организация функций.	2
21	Применение рекурсивных функций.	2
22	Программирование модуля.	2
23	Создание библиотеки подпрограмм.	2
24	Использование указателей для организации связанных списков	2
25	Изучение интегрированной среды разработчика	2
26	Создание проекта с использованием компонентов для работы с текстом.	2
27	Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.	2
28	События компонентов (элементов управления), их сущность и назначение.	2
29	Создание процедур на основе событий.	2
30	Создание проекта с использованием кнопочных компонентов.	2
31	Создание проекта с использованием компонентов стандартных диалогов и системы меню.	2
32	Разработка функциональной схемы работы приложения.	2
33-34	Разработка оконного приложения с несколькими формами.	4
35	Разработка игрового приложения.	2
36-37	Создание процедур обработки событий. Компиляция и запуск приложения.	4
38-39	Разработка интерфейса приложения.	4
40	Тестирование, отладка приложения.	2
41	Классы ООП: виды, назначение, свойства, методы, события. Объявления класса.	2
42	Создание наследованного класса.	2
43-44	Программирование приложений.	4
45	Перегрузка методов.	2
ИТОГО:		**90**

Лабораторная работа № 1
«Алгоритмы в математике»

Цель работы: приобретение навыков работы в построении алгоритмов.

Теоретические сведения:

Алгоритм при решении уравнений

Алгоритм нахождения неизвестного слагаемого

Для того чтобы вычислить неизвестное слагаемое, необходимо из суммы вычесть известное слагаемое

Алгоритм нахождения неизвестного уменьшаемого

Для того чтобы вычислить неизвестное уменьшаемое, необходимо к разности прибавить вычитаемое

Алгоритм нахождения неизвестного вычитаемого

Для того чтобы вычислить неизвестное вычитаемое, необходимо из уменьшаемого вычесть разность

Алгоритм нахождения неизвестного множителя

Для того чтобы вычислить неизвестный множитель, необходимо произведение разделить на известный множитель

Алгоритм нахождения неизвестного делимого

Для того чтобы вычислить неизвестное делимое, необходимо частное умножить на делитель

Алгоритм нахождения неизвестного делителя

Для того чтобы вычислить неизвестный делитель, необходимо делимое разделить на частное

Алгоритмические действия с положительными и отрицательными числами

Алгоритм сложения двух отрицательных чисел

1.Определить, являются ли слагаемые отрицательными числами
2.Сложить модули слагаемых
.Поставить перед полученной суммой знак «минус»

Алгоритм сложения чисел с разными знаками

1.Определить модуль какого из чисел больше
2.Вычесть из большего модуля меньший
.Поставить перед полученным числом знак того слагаемого, модуль которого больше

Алгоритм умножения чисел с разными знаками

1.Определить, являются ли умножаемое и множитель отрицательными числами.
2.Перемножить модули этих чисел
.Поставить перед полученным произведение знак «минус»

Алгоритм деления отрицательного числа на отрицательное число

1.Определить, являются ли делимое и делитель отрицательными
2.Разделить модуль делимого на модуль делителя

Алгоритм деления чисел с разными знаками

1.Определить, являются ли делимое и делитель числами с разными знаками
2.Разделить модуль делимого на модуль делителя
.Поставить перед полученным числом знак «минус»

Задание:

Реализовать в виде блок-схемы следующий алгоритм:

1. Алгоритм нахождения неизвестного слагаемого
2. Алгоритм нахождения неизвестного уменьшаемого
3. Алгоритм нахождения неизвестного вычитаемого

4. Алгоритм нахождения неизвестного множителя
5. Алгоритм нахождения неизвестного делимого
6. Алгоритм нахождения неизвестного делителя
7. Алгоритмические действия с положительными и отрицательными числами
8. Алгоритм сложения двух отрицательных чисел
9. Алгоритм сложения чисел с разными знаками
10. Алгоритм умножения чисел с разными знаками
11. Алгоритм деления отрицательного числа на отрицательное число
12. Алгоритм деления чисел с разными знаками

Лабораторная работа №2
Оценка сложности алгоритмов

Целью лабораторной работы является приобретение навыков исследования временной сложности алгоритмов и определения ее асимптотических оценок.

Требования к содержанию, оформлению и порядку выполнения

В содержательной части отчета по выполнению лабораторной работы требуется привести описание алгоритма, выбранного согласно своему варианту, провести его анализ и определить асимптотические оценки его временной сложности. Алгоритм рекомендуется оформлять с помощью блок-схем.

Теоретическая часть

Теоретические сведения, необходимые для выполнения лабораторной работы, представлены в разд. 3.2 – 3.4.

Общая постановка задачи

Требуется провести анализ и оценку временной сложности заданного алгоритма. Варианты заданий представлены в таблице в следующем разделе.

В качестве дополнительных заданий рекомендуется программно реализовать заданный алгоритм.

Список индивидуальных данных

Данные для выполнения лабораторной работы сведены в табл.Л2.1.

Таблица Л2.1.

Варианты заданий к лабораторной работе № 2

Вариант	*Алгоритм*
1	Тривиальный алгоритм возведения в степень (рис. 3.1)
2	Рекурсивный алгоритм возведения в степень (рис. 3.2)
3	Алгоритм быстрого возведения в степень (рис. 3.3 а)
4	Алгоритм быстрого возведения в степень (рис. 3.3 б)
5	Алгоритм вычисления значения многочлена (рис. 3.4)
6	Алгоритм вычисления значения многочлена по схеме Горнера (рис. 3.5)
7	Алгоритм сортировки обменом (рис. 4.5)
8	Алгоритм сортировки выбором (рис. 4.7)
9	Алгоритм сортировки вставками (рис. 4.9)
10	Алгоритм быстрой сортировки (рис. 4.14, 4.16)

Пример выполнения работы

Рассмотрим 3 вариант. Требуется провести анализ и оценку временной сложности алгоритма быстрого возведения в степень.

Алгоритм предназначен для решения следующей задачи: дано число x и натуральное (целое неотрицательное) число $n \geq 0$. Вычислить значение функции $f(x)=x^n$.

Число n можно представить в виде:

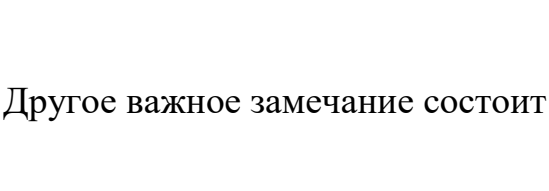
где  – двоичная цифра,  – ближайшее целое не превосходящее $\log_2 n$ (количество двоичных знаков, необходимых для записи числа минус 1).

Тогда:

Введем обозначения: $c_i = x^{2^i}$, $0 \le i \le m$, тогда предыдущую формулу можно представить в виде:

$$x^n = \prod_{i=0}^{m} c_i^{d_i}.$$

Легко заметить, что $c_0 = x$, и $c_i = c_{i-1}^2$ для всех $1 \le i \le m$.

Другое важное замечание состоит в том, что: 

Алгоритм быстрого возведения в степень основан на выше приведенных соображениях. Его блок-схема приведена на рис. Л2.1.

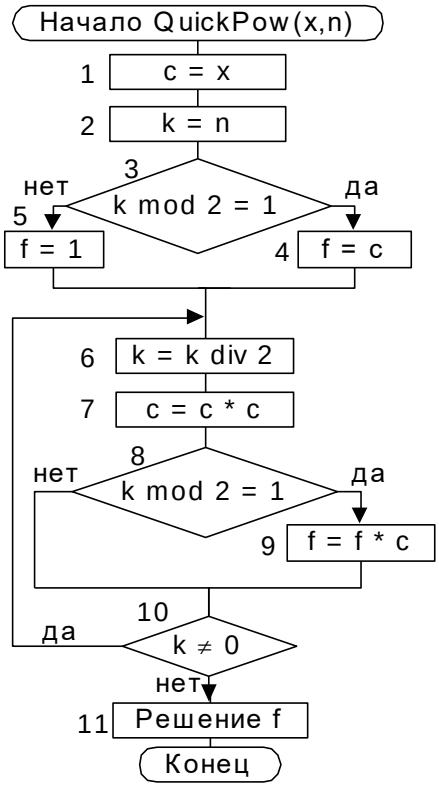


Рис. Л2.1. Алгоритм быстрого возведения в степень

Проведем его анализ. Сначала отметим, что, степень, в которую возводится число, может служить мерой объема входных данных и все операторы присваивания и сравнения имеют некоторое постоянное время выполнения, независящее от размера входных данных.

Наибольшее влияние на временную сложность алгоритма оказывают циклические конструкции. Рассмотрим группу блоков (6) – (10) алгоритма. Эти блоки составляют цикл с постусловием. Тело цикла (блоки (6) – (9)) имеет временную сложность порядка *O(1)*. Кроме этого в цикле выполняется проверка условия завершения цикла (блок 10), которая так же имеет время выполнения порядка *O(1)*. Определим количество повторений тела цикла в

зависимости от *n*. Таблица Л2.2 содержит динамику изменения параметра цикла *k* в зависимости от номера итерации и параметра *n*.

Таблица Л2.2

№ итерации	Значение параметра цикла *k*														
	n=0	*n=1*	*n=2*	*n=3*	*n=4*	*...*	*n=7*	*n=8*	*...*	*n=15*	*n=16*	*...*	*n=31*	*n=32*	*...*
1	0	1	2	3	4	…	7	8	…	15	16	…	31	32	…
2		0	1	1	2	…	3	4	…	7	8	…	15	16	…
3			0	0	1	…	1	2	…	3	4	…	7	8	…
4					0	…	0	1	…	1	2	…	3	4	…
5								0	…	0	1	…	1	2	…
6											0	…	0	1	…
7														0	…

Анализируя данные, приведенные в таблице Л2.2, можно отметить, что на 1-й итерации $k_1 = n \text{ div } 2^0 = n$, на этой и каждой последующей итерации k уменьшается в два раза, по сравнению с предыдущем значением, до тех пор, пока k не станет равно 1 (пусть это будет итерация m+1). На следующей итерации: $k_{m+2} = k_{m+1} \text{ div } 2 = 0$, что обеспечивает условие завершения цикла. Таким образом, для произвольного n>0 имеем:

$k_1 = n \ div \ 2^0 = n;$

$k_2 = n \ div \ 2^1;$

$k_3 = n \ div \ 2^2;$

....

$k_i = n \ div \ 2^{i-1};$

....

$k_{m+1} = n \ div \ 2^m = 1;$

$k_{m+2} = n \ div \ 2^{m+1} = 0.$

Всего получается m+2 итераций, где m определяется соотношением $2^m \le n < 2^{m+1}$. Из последнего соотношения следует: $m = \lfloor \log n \rfloor$ – ближайшее целое не превосходящее $\log_2 n$. Итого: количество повторений цикла равняется $\lfloor \log n \rfloor + 2$, а временная сложность группы блоков (6) – (10) составляет O(log n).

Время выполнения конструкции ветвления (блоки (3), (4) и (5)) состоит из времени вычисления логического выражения (блок (3)) и наибольшего из времени, необходимого для выполнения операторов, исполняемых при значении логического выражения **true** (блок (4)) и при значении **false** (блок (5)). Каждый из этих блоков имеет время выполнения порядка 0(1), соответственно вся конструкция в целом имеет временную сложность порядка O(1). Группа блоков (1) – (2), так же как и блок (11), имеет время выполнения порядка O(1). Группы блоков (1) – (2), (3) – (5), (6) – (10) и (11) выполняются последовательно. Согласно правилу сумм, общая временная сложность алгоритма T(n) = O(max(1,1,log n,1)) = O(log n).

Следует отметить, что в этом алгоритме имеет место ситуация: T(n) = □ (log n), и следовательно T(n) = □ (log n).

Вывод. В лабораторной работе проведен анализ алгоритма быстрого возведения в степень. Его временная сложность имеет верхнюю оценку O(log n). Нижняя оценка составляет □ (log n), следовательно, имеет место оценка □ (log n). Данный алгоритм имеет лучшую оценку временной сложности чем тривиальный алгоритм, который, "на вскидку" имеет временную сложность порядка O(n).

Контрольные вопросы к защите

1. Понятие временной сложности алгоритма.
2. Определение асимптотических оценок временной сложности.
3. Основные принципы получения асимптотических оценок.
4. Правила анализа алгоритмов с целью определения их временной сложности.

Лабораторная работа № 3

Алгоритм поиска минимального и максимального элементов в массиве

Цель:

Теоретические сведения:

Алгоритм нахождения максимального элемента массива выполняется следующим образом:

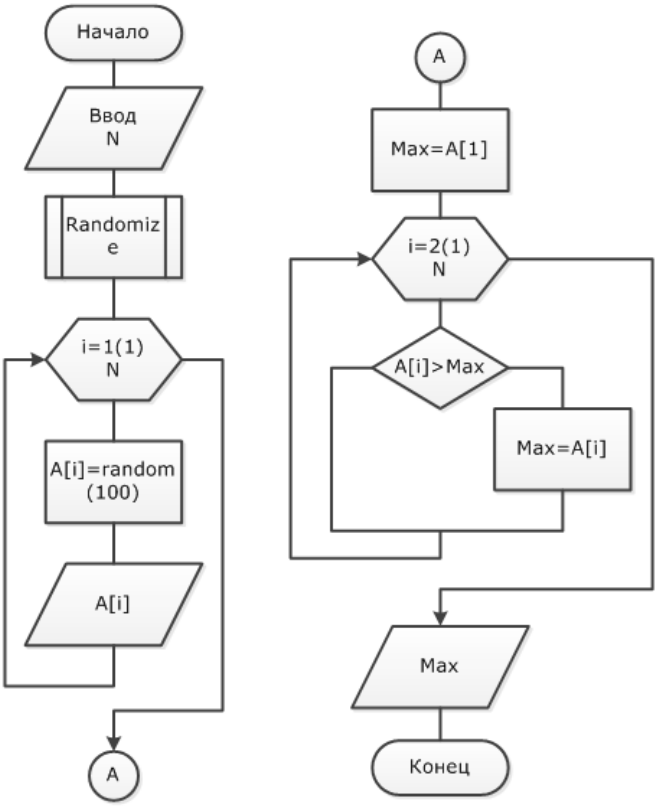
Сначала указываем, что первый элемент массива считается максимальным, иначе говоря – *Max = A[i]*.

Потом начинаем процесс сравнивания последующих элементов массива с максимальным элементом в массиве.

Тут возможно два случая:

1. Если максимальный элемент больше следующего, то ничего не меняем.
2. Если максимальный элемент меньше следующего, то он становиться максимальным.

После этого выводим на экран максимальный элемент.



Минимальный элемент находится по аналогии с максимальным.

Задание:

Дан одномерный массив состоящий из n действительных чисел, найти среди них минимальный элемент массива, и его положение в массиве; максимальный элемент и его положение в массиве и сумму максимального и минимального элементов.

Лабораторная работа № 4
Сортировка массивов

Цель: Изучить алгоритмы сортировки массивов и научиться использовать их при обработке данных.

Теоретические сведения

Метод сортировки выбором

Исходный массив длиной N разбивается на две части: итог и остаток. Участок массива, называемый **итогом**, располагается с начала массива и должен быть упорядоченным, а участок массива, называемый **остатком,** располагается вплотную за итогом и содержит исходные числа не отсортированной части исходного массива.

Текстуальный алгоритм сортировки выбором:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.

Шаг 2. В остатке массива ищется минимальный элемент и он меняется местом с первым элементом остатка (i-ым элементом массива). После чего значение i увеличивается на единицу, тем самым расширяя итоговый участок массива (отсортированную часть исходного массива).

Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

Метод сортировки пузырька

Аналогично, как и в методе выбора, исходный массив длиной N разбивается на две части: отсортированную (итог) и не отсортированную (остаток). Первый элемент остатка является i-ым элементом массива.

Текстуальный алгоритм сортировки пузырьком:

Шаг 1. Пусть k=N-1 , т.е. итоговый участок состоит из одного элемента.

Шаг 2. Берется первый элемент остатка и перемещается на место в итоговый участок массива так, чтобы итог остался упорядоченным. Первый элемент остатка назовем перемещаемым. Перемещение выполняется путем сравнения перемещаемого элемента с последующим элементом. Если последующий элемент больше сравниваемого элемента, то процесс перемещения этого элемента закончен.

Шаг 3. После того, как первый элемент остатка переместился в итоговый участок, уменьшается на единицу значение переменной k, тем самым увеличивая отсортированную часть массива. Если k>0, то управление передается на Шаг 2, в противном случае - работа алгоритма завершена.

Конец алгоритма.

Метод сортировки включением

Этот метод похож на метод пузырька. Происходит такое же разбиение массива на отсортированную и не отсортированную части, но перемещение первого элемента остатка на принадлежащее ему место в итоге делается не сравнением двух соседних элементов, а с помощью метода двоичного поиска, который удобно оформить в виде отдельной процедуры.

Текстуальный алгоритм методом включением:

Шаг 1. Пусть i=1 , т.е. итоговый участок состоит из одного элемента.

Шаг 2. Берется первый элемент остатка и перемещается в отсортированную часть массива так, чтобы итоговый участок остался упорядоченным.

Шаг 3. После того, как первый элемент остатка переместился в итоговый участок, увеличивается на единицу значение переменной i, тем самым увеличивая отсортированную часть массива. Если $i < N$, то управление передается на Шаг 2, в противном случае - работа алгоритма завершена.

Конец алгоритма.

Задания:

1. Реализовать алгоритм сортировки выбором в виде блок-схемы.
2. Реализовать алгоритм сортировки пузырька в виде блок-схемы.
3. Реализовать алгоритм сортировки сключением в виде блок-схемы

Лабораторная работа № 5

Схема примитивной рекурсии

Цель:

Теоретические сведения:

Оператор примитивной рекурсии Оператор примитивной рекурсии позволяет определить циклические вычисления специального вида. Пусть заданы: - n-местная вычислимая функция g и - (n+2)-местная вычислимая функция h. Тогда (n+1)-местная функция f строится примитивной рекурсией из функций g и h, если для всех натуральных значений x 1, x 2,...,x n, y имеем: f(x 1, x 2,...,x n,0)=g(x 1, x 2,...,x n) f(x 1, x 2,...,x n,y+1)=h(x 1, x 2,...,x n,y, f(x 1, x 2,...,x n,y)).

Соотношения называются схемой примитивной рекурсии. Они определяют оператор примитивной рекурсии R. Эта схема, собственно, и есть алгоритм вычисления функции f. Оператор R определен на множестве частичных вычислимых функций, f=R(g,h).Соотношения называются схемой примитивной рекурсии. Они определяют оператор примитивной рекурсии R. Эта схема, собственно, и есть алгоритм вычисления функции f. Оператор R определен на множестве частичных вычислимых функций, f=R(g,h).

Понятно, что функция f существует и единственна. Это следует из того, что определение функции, построенной примитивной рекурсией, фактически содержит схему (алгоритм) ее вычисления. Распишем детально эту схему, используя термальные представления. Понятно, что функция f существует и единственна. Это следует из того, что определение функции, построенной примитивной рекурсией, фактически содержит схему (алгоритм) ее вычисления. Распишем детально эту схему, используя термальные представления.

Пусть необходимо вычислить значение функции f при y=k. Тогда из определения (1) схемы примитивной рекурсии имеем последовательность функциональных термов t 0, t 1,…, t k, вычисляющих значение функции f(x 1, x 2,...,x n, k): t 0 : f 0 =f(x 1,x 2,...,x n,0)=g(x 1,x 2,...,x n) t 1 : f 1 =f(x 1,x 2,...,x n,1)=h(x 1,x 2,...,x n,0,g(x 1,x 2,...,x n)) t 2 : f 2 =f(x 1,x 2,...,x n,2)= h(x 1,x 2,...,x n,1, f(x 1,x 2,...,x n,1)) t k : f k =f(x 1,x 2,...,x n, k)=h(x 1,x 2,...,x n,k-1,f(x 1,x 2,...,x n,k-1)) Здесь представлен способ вычисления f, следовательно, функция f определена однозначно. Если одна из функций f(x 1,x 2,...,x n,m), m

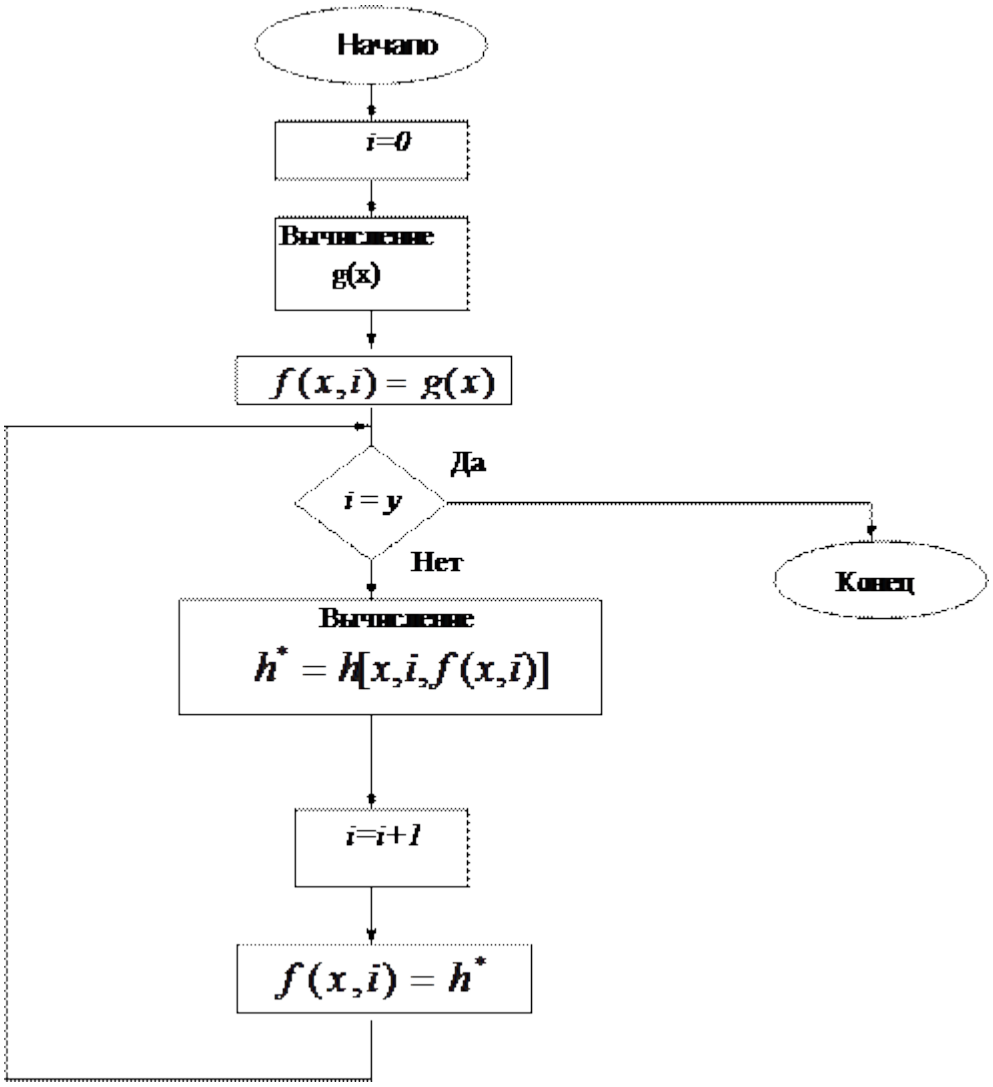
ПРИМЕР. Рассмотрим примитивно рекурсивную схему Рассмотрим примитивно рекурсивную схему x+0=xx+0=x x+(y+1)=(x+y)+1=s(x+y)x+(y+1)=(x+y)+1=s(x+y) Следовательно, функция (x+y) строится применением оператора примитивной рекурсии R к примитивно рекурсивным функциям g(x)=x и h(x,y,z)=z+1. Следовательно, функция x+y примитивно рекурсивна. Следовательно, функция (x+y) строится применением оператора примитивной рекурсии R к примитивно рекурсивным

функциям g(x)=x и h(x,y,z)=z+1. Следовательно, функция x+y примитивно рекурсивна.

Оператор примитивной рекурсии R_n, определяющий значение функции f, записывается в виде следующей схемы (для простоты f будем считать двуместной):

$$f(x,0) = g(x)$$
$$f(x,y+1) = h[x,y,f(x,y)]$$

При этом значение X считается фиксированным. Работа оператора R_n заключается в последовательном вычислении значения $f(x,i)$ для $i = 0,1,2,\ldots,y+1$. Более детально алгоритм вычисления функции $f(x,y)$ по схеме примитивной рекурсии показан на блок-схеме



Пример 2. Вычисление функции $f(n) = n!$ с помощью оператора примитивной рекурсии.

$$f(0) = 1 = g;$$
$$f(n+1) = (n+1) \cdot f(n)$$

Пусть требуется вычислить 4!. По схеме примитивной рекурсии имеем:

$$f(0) = 1;$$
$$f(1) = 1 \cdot f(0) = 1 \cdot 1 = 1;$$
$$f(2) = 2 \cdot f(1) = 2 \cdot 1 = 2;$$
$$f(3) = 3 \cdot f(2) = 3 \cdot 2 = 6;$$
$$f(4) = 4 \cdot f(3) = 4 \cdot 6 = 24.$$

Видно, что всякий раз при вычислении $f(n+1)$ через $f(n)$ значение $f(n)$ уже определено.

Функция принято называть примитивно-рекурсивной, если она может быть получена из простейших с помощью конечного числа применений операторов суперпозиции и примитивной рекурсии.

Очевидно, что примитивно-рекурсивные функции являются всюду определенными, т.к. простейшие функции всюду определены, а операторы суперпозиции и примитивной рекурсии не сужают область определения.

Задания:

Операция минимизации

Лабораторная работа № 7
Частично рекурсивные функции. Общерекурсивные функции

Частично-рекурсивная функция – функция, которая может быть построена из простейших с помощью конечного числа применений оператора суперпозиции, примитивной рекурсии и минимизации.

Частично-рекурсивная функция является не всюду определенной, причем там, где она не определена, процесс ее вычисления продолжается бесконечно.

Общерекурсивная функция–всюду определенная частично-рекурсивная функция.

Связь между алгоритмами и рекурсивными функциями выражается **тезисом Черча**: какова бы ни была вычислимая неотрицательная целочисленная функция от неотрицательных целочисленных аргументов, существует тождественно равная ей частично-рекурсивная функция.

Класс частично-рекурсивных функций (ЧРФ) шире чем класс общерекурсивных функций (ОРФ), который в свою очередь шире классса примитивно-рекурсивных функций (ПРФ) (см. рис. 1).

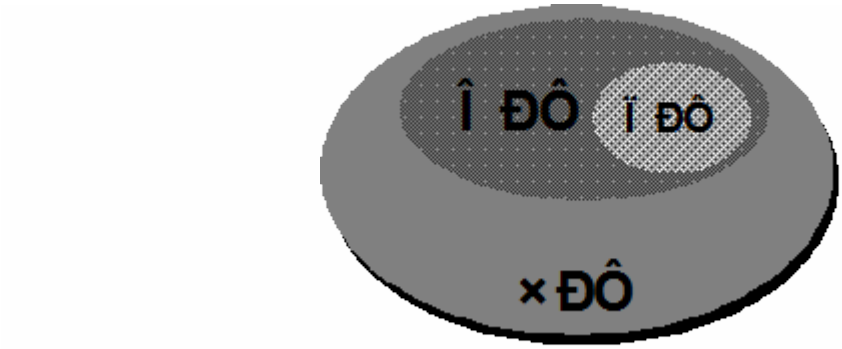


Рисунок 1 – Соотношение между классами частично-рекурсивных, общерекурсивных и примитивно-рекурсивных функций

Задание на лабораторную работу

1. Разработать алгоритм вычисления $f(n)$ в виде рекурсивной функции.
2. Проверить модель алгоритма на множестве тестовых примеров.
3. Определить к какому классу рекурсивных функций принадлежит $f(n)$: примитивно-рекурсивна, частично-рекурсивна или общерекурсивна.

Варианты заданий

1. Наибольший общий делитель двух чисел, $D(x,y), D(0,0)=0.$
2. Функция, отличная от нуля в конечном числе точек.
3. Номер наибольшего простого делителя числа $n.$
4. Функция, вычисляющая целую часть квадратного корня от аргумента, $y=\lfloor\sqrt{x}\rfloor$

Контрольные вопросы

1. Что такое вычислимая, арифметическая, частичная или всюду определенная функция?

2. Определить операторы суперпозиции и примитивной рекурсии.

3. Перечислить простейшие функции теории рекурсивных функций.

4. Что такое примитивно-рекурсивные функции?

5. Показать примитивную рекурсивность известных арифметических функций.

6. Показать примитивную рекурсивность арифметизованных логических функции. Примитивная рекурсивность отношений и предикатов.

7. Определить оператор минимизации, в каких случаях он работает бесконечно?

8. Что такое частично-рекурсивная функция и общерекурсивная?

9. Сформулировать тезис Черча.

10. Определите соотношение между примитивно, частично и общерекурсивными функциями.

Лабораторная работа № 8

Знакомство со средой программирования

Цель:

Теоретические сведения

Команды редактора

Команды управления движением курсора

-> - (<- -) перемещение курсора на символ вправо (влево);
^ (∨) - перемещение курсора на строку вверх (вниз);
Home (End) - перемещение курсора в начало (конец) текущей строки;
Page Up (Page Down) - перемещение курсора на страницу вверх (вниз);
Примечание. Страница - это число строк текста, составляющих один экран (21 строка).
Ctrl + Home (Ctrl + End) - перемещение курсора в левый верхний угол (левый нижний угол);

Команды вставки и удаления текста

Insert - включение и выключение режима вставки;
Примечание. Если режим вставки включен, то на экране курсор имеет вид мигающей черты. В режиме вставки набираемый символ вводится в позицию, в которой стоит курсор, а все символы (начиная с символа, стоящего в позиции курсора ранее),расположенные правее, сдвигаются вправо. Если режим вставки выключен, то набираемый символ заменит тот символ, который находится в позиции курсора, таким образом можно старый текст заменить на новый.
Delete - удаление символа, стоящего в позиции курсора;
Backspace - удаление символа, стоящего перед курсором;
Ctrl + N - вставка пустой строки над строкой, где находится курсор;
Ctrl + Y - удаление строки, где находится курсор.

Режим помощи

Необходимо познакомить с режимом помощи - Help (F1). Показать, как входить в режим помощи, перемещаться по нему. Подробное знакомство с этим режимом учащиеся проводят самостоятельно.

Запуск программы на выполнение

1. Выполняем **проверку <F9>**
2. Производим компиляцию **<Alt>+<F9>**
3. **Запуск** программы **<Ctrl>+<F9>**
4. Просмотр содержимого окна консоли **<Alt>+<F5>**

Сохранение программы

Для того, чтобы сохранить программу, необходимо:
- выйти в главное меню и выбрать режим File;
- нажать <Enter> и из появившегося окна выбрать режим Save as..., после нажатия клавиши <Enter> появится окно, в котором наберите имя файла. Например, a:\prim1_1.pas; здесь a:\ - это название диска, на котором будем сохранять файл, prim1_1 - имя файла (оно может содержать не более 8 символов), pas - расширение, сообщающее о том, что файл содержит программу, написанную на языке Паскаль.

Примечание. список символов, которые нельзя употреблять в именах файлов:
* = + [] \ | ; : , . < > / ?. А также не следует использовать в именах файлов символ пробела и буквы русского алфавита.
После того, как имя файла набрано, нажмите клавишу <Enter>.
Примечание. Следует отметить, что для быстрого сохранения файла можно воспользоваться командами Save или Save all меню File.

Выход из системы программирования Турбо Паскаль

Для того, чтобы закончить работу, необходимо:
- выйти в главное меню и выбрать режим File;
- нажать <Enter> и из появившегося окна выбрать режим Quit <Alt>-<X>, после чего нажать либо <Enter>, либо комбинацию <Alt>-<X>.

Рассмотрим на примере конкретной программы, запрашивающей имя и возраст.

```
Program lab1_1;
Var     a: Integer;
        b: String;

Begin
Writeln ('Введите имя ');
Readln (b);
Writeln ('Ваш возраст? ');
Readln (a);
Writeln (b,',вам', a,'лет?');
End.
```

Имя этой программы lab1_1 (заметим, что в имени программы не должно быть пробелов, оно должно начинаться с буквы, состоять только из латинских букв, цифр и некоторых символов, не допускается использование символов точки и запятой).

Раздел описания переменных – следующий раздел. Он начинается со служебного слова **Var**, после которого идет последовательность каких-либо переменных, разделенных точкой с запятой.[1]

Тип переменных указывается после перечисления всех переменных, после знака двоеточие. В нашем примере описаны две переменные: ***a*** имеет целый тип (integer), ***b*** имеет строковый тип (String).

Раздел операторов – следующий раздел, начинающийся со служебного слова **Begin**, после которого идут операторы языка. В конце раздела операторов стоит служебное слово **End**, после которого ставится точка.

```
Program lab1_1;
Var     a: Integer;
        b: String;

Begin
Writeln ('Введите имя ');
Readln (b);
Writeln ('Ваш возраст? ');
Readln (a);
Writeln (b,',вам',a,'лет?');
End.
```

Первый встречающийся оператор - это **Write***ln*('текст'); - записать (вывести) на экран текст, заключенный между апострофами (у нас – ***Введите имя***), *ln* добавляется в конце этого оператора для того, чтобы курсор автоматически переходил на следующую строку.

Следующий оператор - это **Read***ln*(b); - читать данные с клавиатуры. В данном случае необходимо ввести какой-либо текст, для этого мы используем переменную b – строкового типа. Тогда переменной b будет присваиваться значение, которое вы введете с клавиатуры. Например, вы ввели свое имя ***Василий***,

Следующий оператор опять **Write***ln*('текст'); - записать (вывести) на экран текст «Ваш возраст».

Следующий оператор - это **Read***ln*(a); - читать данные с клавиатуры. В данном случае необходимо ввести число, для этого мы используем переменную a – целого типа. Переменной *a* присваивается значение, равное числу которое вы ввели. Например, вы ввели свой возраст 12, тогда *a* = 12.

Следующий оператор - это снова оператор *writeln*('текст', z) - он выведет на экран сначала значение переменной b – «Василий», текст «,вам», а за ним значение переменной *a – 12* , и текст «лет?». После запуска программы вы получите фразу-вопрос: ***Василий. вам 12 лет?***

```
Program lab1_3;      {заголовок программы}
Var  a,b,c: integer;   {раздел описания переменных}
BEGIN                  {начало раздела операторов}
        c:=5;
        b:=4;          {занесение в ячейки c и b начальных значений}
        a:=c*b;        {вычисление значения переменной a}
        writeln(a);    {вынесение на экран значения a}
END.                   {конец программы}.
```

Комментарии

Тексты, заключённые в скобки { }, являются комментариями, не являются частью программы и могут быть опущены

Комментарий – это произвольная последовательность любых символов, поясняющая текст программы. Комментарий разрешается вставлять в любое место программы, где по смыслу может стоять пробел. Как правило, комментарии пишутся в начале программы как поясняющий текст о предназначении программы, и в тексте программы. В качестве ограничителей комментария используются фигурные скобки « {» и «} »: Только используя комментарии при написании программ, вы сможете считать себя грамотным пользователем среды разработки Pascal!

Задание. Написать программу вычисления a = c*b.

Лабораторная работа № 9

Составление программ линейной структуры.

Цель: *научиться составлять простейшие программы линейной структуры на языке Pascal.*

Теоретические сведения.

Линейный алгоритм не содержит логических условий и имеет одну ветвь вычислений. Структура программы:

PROGRAM <имя программы>;
USES Перечисление подключаемых модулей>;
TYPE <описание типов>;
VAR <Перечисление используемых переменных с указанием типов>;
CONST <Перечисление констант>;
BEGIN
<оператор 1>;
<оператор 2>;
<оператор п>;
END.

Задания:

1. Вычислить длину окружности и площадь круга одного и того же радиуса *R.*
2. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее арифметическое модулей этих чисел.
3. Вычислить расстояние между двумя точками с данными координатами *(x1. y1)* и *(x2. y2).*
4. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
5. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.($S=\pi r^2$; $L=2\pi r$);
6. Треугольник задан координатами своих вершин. Найти периметр треугольника;
7. Дано действительное число *x.* Не пользуясь никакими другими арифметическими операциями, кроме умножения, сложения и вычитания, вычислить за минимальное число операций $2x^4-3x^3+4x^2-5x+6$;
8. Вычислить корни квадратного уравнения $ax^2+ bx+ c = 0$ с заданными коэффициентами *a, b и c* (предполагается, что a, b, c не 0 и что дискриминант уравнения неотрицателен).
9. Дано значение a. Не используя никаких функций и никаких операций, кроме умножения, получить значение a^8 за три операции и a^{10} за четыре операции.
10. Написать программу, которая выводит на экран первые четыре степени числа я.
11. Три сопротивления *R1, R2, R3* соединены параллельно. Найти сопротивление соединения.
12. Составить программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде v км/ч, скорость течения реки v1 км/ч, время движения по озеру t1 ч, а против течения реки – t2 ч.

13. Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту H и одинаковый радиус основания R?. ($V_{кон} = \frac{1}{3}\pi r^2 h; V_{цил} = \pi r^2 h;$)

14. Даны два действительных числа *x* и *y*. Вычислить их сумму, разность, произведение и частное.

15. Даны действительные числа x и y. Получить $\frac{|x| - |y|}{1 + |xy|}$.

Лабораторная работа №10

Составление программ разветвляющейся структуры.

Цели: *Научиться работать с условным оператором If...Then...Else.*

Теоретические сведения

Алгоритмическая структура ветвления программируется в ТР с помощью условного оператора:

IF <условие> THEN <оператор 1>
ELSE <оператор 2>;

Пример: вычислить x=a/Ь

```
Var x,a,Ь: real;
If b>0 then x:=a/b
else Write('решения нет');
```

Кроме этого возможно использование неполной формы условного оператора:

IF <условие> THEN <оператор 1>;

Запишем еще раз, чтобы понять, как работает эта конструкция:

IF <условие> THEN <оператор 1>; <оператор 2>; <оператор 3>;

Существует два варианта:

1. если условие истинно, то программа «уходит в сторону» на оператор 1, он выполняется, а затем продолжается выполнение операторов 2, 3...

2. если условие ложно, оператор 1 НЕ выполняется, а выполняются сразу операторы 2, 3..., т.е. следующие операторы программы, не зависящие от условия.

Задание:

1. Даны две точки A(x1,y1) и B(x2,y2). Составить программу, определяющую, которая из точек находится ближе к началу координат.

2. Даны действительные числа x и y, не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее - их удвоенным произведением.

3. Даны целые числа in, n. Если числа не равны, то заменить каждое из них тем же числом, равным большему из исходных, а если равны, го заменить числа нулями.

4. Определить, делителем каких чисел a, b. c является число k.

5. Услуги телефонной сети оплачиваются по следующему правилу: за разговоры до A минут в месяц - B рублей, а разговоры сверх установленной нормы оплачиваются из расчета C рублей за минуту. Написать программу, вычисляющую плату за пользование телефоном для введенного времени разговоров за месяц.

6. Грузовой автомобиль выехал из одного города в другой со скоростью V1 км/ч. Через t ч в этом же направлении выехал легковой автомобиль со скоростью V2 км/ч. Составить программу, определяющую догонит ли легковой автомобиль грузовой через t, ч после своего выезда.

7. Определить правильность даты, введенной с клавиатуры (.число - от 1 до 31, месяц - от 1 до 12). Если введены некорректные данные, то сообщить об этом.

8. Составить программу, определяющую результат гадания на ромашке - «любит - не любит», взяв за исходное данное количество лепестков n.

9. Рис расфасован в два пакета. Масса первого - m кг. второго - n кг. Составить программу. определяющую:

a. какой пакет тяжелее - первый пли второй;

b. массу более тяжелого пакета.

10. Написать программу, которая анализирует данные о возрасте и относит человека к одной из четырех групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры.

11. Написать программу нахождения суммы большего и меньшего из трех чисел.

12. На оси OX расположены три точки a, b, c. Определить, какая из точек b или c расположена ближе к точке a.

13. Написать программу решения уравнения $ax^3 + b\,x = 0$ для произвольных a, b.

14. Заданы размеры A, B прямоугольного отверстия и размеры x, y,z кирпича. Определить, пройдет ли кирпич через отверстие.

15. Подсчитать количество отрицательных и положительных чисел среди чисел *a, b, c.*

Лабораторная работа № 11

Составление программ циклической структуры.

Цели: *Содействовать формированию у студентов навыков работы с циклами; Обеспечить закрепление способов работы с операторами циклов*

Теоретические сведения.

В ТР существует 3 вида циклов:

1. *Цикл с параметром*

```
For I := m  1 o k Do
Begin
Оператор 1;
Оператор 2;
……
Оператор n;
End;
```

Если после Do выполняется 1 оператор, операторные скобки Begin…End можно опустить.

Свойства цикла с параметром:

• Счетчик автоматически изменяется на единицу при каждом следующем исполнении алгоритма.

• Переменные I, m, k должны быть порядкового типа.

• Счетчику присваивается начальное значение: I := m.

• Если начальное значение совпадает с конечным, то тело цикла выполнится 1 раз.

• Если начальное значение больше конечного значения, тело цикла не выполнится ни разу.

• При выходе из цикла значение счетчика совпадает с конечным значением.

2. Цикл с предусловием

```
While<усл.>Do
Begin
Оператор 1;
Оператор 2:
.....
Оператор n;
End;
```

3. Цикл с постусловием

```
Repeat  Begin
Оператор 1;
Оператор 2;
……
Оператор n; End;
Until <усл.>;
```

Например: Составить программу для вычисления факториала.

```
program rr;
var i,n,f:integer;
begin
 write('vv n=');
 readln(n);
 f:=1;
 i:=1;
 while i<=n do begin
 f:=f*i;
 i:=i+1;
 end;
 writeln('f=',f);
 readln;
end.
```

Задания по теме:

Составить программу с использованием цикла While, Repeat или For, которая является решением данной задачи.

1. Дано натуральное число n и последовательность действительных чисел a1,a2 ,...,an. В этой последовательности все отрицательные числа увеличить на 0,5, а все неотрицательные числа увеличить на 0,1.

2. Дано натуральное число n и последовательность действительных чисел a1,a2 ,...,an. Получить сумму положительных и количество отрицательных членов этой последовательности.

3. Вывести на экран *к* членов последовательности Фибоначчи: 1 1 2 3 5 8 13 21... ($a_к = a_{к-1} + a_{к-2}$);

4. Даны два числа m и n. Вынести на экран НОД(m, n). (Алгоритм Евклида)

5. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?

6. Дано натуральное число n? Найти первую цифру числа n

7. Дано натуральное число N . Вычислить: $s = 1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\ldots-\frac{1}{2}$;

8. Дано натуральное число n. Вычислить: $s= \frac{2}{1}+\frac{3}{2}+\frac{4}{3}+\ldots+\frac{m+1}{m}$.

9. Дано натуральное число n. Вычислить: s= 1!+2!+3!+…+n! (n>1).

10. Дано натуральное число n? Сколько цифр в числе n?

11. Найти сумму всех n-значных чисел, кратных к (к = 1, 2, 3).

12. Составить программу, которая запрашивает пароль (например, 4-значное число) до тех пор, пока он не будет правильно введен.

13. Дано натуральное число n? Чему равна сумма его цифр?

14. Определить, является ли число «совершенным» (Число называется «совершенным», если оно равно сумме всех своих положительных делителей, например, совершенным числом является число 6=1+2+3).

15. Вывести на экран все простые числа от 1 до n.(Число является простым, если оно делится только на себя и на единицу, например, простыми числами являются числа 3,5,7...).

Лабораторная работа № 12
Тема: «Обработка одномерных массивов»

Цель работы: *научиться правильно описывать различные массивы, уметь инициализировать массивы, распечатывать содержимое массива; научится решать задачи с использованием массивов.*

Теоретические сведения:

Массив - это структурированный тип данных, который используется для описания упорядоченной совокупности фиксированного числа элементов одного типа, имеющих общее имя. Для обозначения элементов массива используются имя переменной-массива и индекс.

Перед выполнением работы необходимо изучить правила описания и использования переменных типа массив, типизированных констант типа массив.

Примеры:

Пример 1: Дан двумерный массив. В каждой строке все его элементы, не равные нулю, переписать (сохраняя порядок) в начало строки, а нулевые элементы - в конец массива. Новый массив не заводить.

Этапы решения задачи:

Суть одного из алгоритмов решения данной задачи состоит в том чтобы "просматривать" массив построчно и находить в каждой строке пару (0:число), а затем менять их местами между собой и так до тех пор пока в строке таких пар не окажется.

Блок схема алгоритма целиком:

Приведем программу на языке Паскаль:

```
program example1;
 var
  V:array[1..100,1..100] of integer;
   m,n, i,j, c: integer;
     flag: boolean;
begin
    write('Введите размерность массива m-n> '); readln(m,n);
      for i:= 1 to m do
        for j:= 1 to n do begin
          write('V[',i,',',j,']= '); readln(V[i,j]);
 end;
  for i:=1 to m do
   repeat
      flag:= true;
        for j:=1 to n-1 do
       if (v[i,j]=0) and (v[i,j+1]<>0) then begin
      c:=v[i,j]; v[i,j]:=v[i,j+1]; v[i,j+1]:=c;
   flag:= false;
 end;
  until flag;
   for i:= 1 to m do begin
     for j:= 1 to n do write(V[i,j]:2);
     writeln
  end;
 readln;
end.
```

Задания

1. Найти все элементы массива целых чисел, удовлетворяющих условию: остаток от деления на 5 равен 3.

2. Расположить элементы данного массива в обратном порядке (первый элемент меняется с последним, второй - с предпоследним и т.д. до середины; если массив содержит нечетное количество элементов, то средний остается без изменения).

3. В данном массиве поменять местами элементы, стоящие на нечетных местах, с элементами, стоящими на четных местах.

4. Дана последовательность целых чисел a_1, a_2, …,a_n. Выяснить, какое число встречается раньше- положительное или отрицательное.

5. Дана последовательность действительных чисел a_1, a_2, …,a_n. Заменить все его члены, большие заданного К, этим числом.

6. Дан целочисленный массив с количеством элементов n. Сжать массив, выбросив из него каждый второй элемент.

7. Найти количество элементов массива целых чисел, больших квадрата первого элемента этого массива. Если таких нет, выдать сообщение «поиск неудачен».

8. Найти наибольший элемент массива вещественных чисел.

9. Подсчитать количество элементов массива целых чисел, меньших 0.

10. Найти все элементы массива целых чисел, больше заданного числа.

11. Найти номера четных элементов массива вещественных чисел, меньших заданного числа.

12. В целочисленной последовательности есть нулевые элементы. Вывести номера этих элементов.

13. Дан целочисленный массив с количеством элементов n. Напечатать те его элементы, индексы которых являются степенями двойки (1, 2, 4, 8, 16…).

14. В массив А занесены натуральные числа. Найти сумму тех элементов, которые кратны заданному К.

15. Задать последовательность из N вещественных чисел. Определить, сколько среди них чисел меньших К, равных К и больших К.

Лабораторная работа № 13
Тема: «Обработка двумерных массивов».

Цель: *Научиться использовать структурированный тип данных (массивы) при решении задач на языке Turbo Pascal.*

Теоретические сведения:

1. Описание двумерных массивов.

<имя_массива> :ARRAY[<тип_индексов>] OF <тип_элементов>;

где

<имя_массива> - допустимый в языке идентификатор;

<тип_индексов> - дискретный тип, чаще всего ограниченный;

<тип_элементов> - любой допустимый в языке тип данных, в том числе составной.

Например:

```
Type AR= array [1..8, 1..50] of real; {массив вещественных}
{чисел. Содержит не более 8 строк}
{и 50 столбцов}
AI= array [1995..1999, 1..12] of integer; {Массив}
{целых чисел. Содержит 5 строк и 12 столбцов.}
{Целесообразно использовать для описания}
{данных за каждый месяц указанных лет}
ACHESS= array [‘a’ .. ‘h’, 1..8] of string [3]; {Массив}
{строк символов. Содержит 8 строк и 8 столбцов.}
{Целесообразно использовать для описания}
{шахматной позиции}
Var X: AR;   Pribyl: AI;  Pozic: ACHESS;
```

2. Обращение к отдельным элементам двумерных массивов (см. описание массивов, данное в предыдущем пункте).

Обращение к первому элементу первой строки массива: X [1,1], Pribyl [1,1], Pozic [1,1].

Обращение к элементу, стоящему в i-й строке и k-м столбце: X [i, k], Pribyl [i, k], Pozic [i, k].

Обращение к последнему элементу последней строки: X [8, 50], Pribyl [1999, 12], Pozic [‘h’, 8].

3. Подсчет суммы элементов одномерного массива (фрагмент программы).

```
S: = 0;      For k: = 1 to n do S: = S + T[k];
```

4. Заполнение двумерного массива.

а) Ввод значений элементов с клавиатуры (фрагмент программы):

```
For i: = 1 to m do
For k: = 1 to n do
begin
write ('Введите значение элемента’, i, ‘-й строки, ‘, k, ‘-го столбца:');
readln (T [i, k])
end;
```

б) Заполнение массива случайными числами из интервала [a, b] (фрагмент программы):

```
For i: = 1 to m do
For k: = 1 to n do T [i, k]: = random (b-a) +a;
```

5. Вывод на экран двумерного массива построчно (предполагается, что в массиве не более 15 столбцов).

```
For i: = 1 to m do
begin
For k: = 1 to n do write (T [i, k]: 5)
end;
writeln;
```

6. Подсчет суммы всех элементов двумерного массива (фрагмент программы).

```
S: = 0;
For i: = 1 to m do
For k: = 1 to n do   S: = S + T [i ,k];
```

Задания:

1**.** Найти сумму всех четных элементов двумерного массива целых чисел a(30,20);

2. Задана матрица размером n×m. Найти максимальный по модулю элемент матрицы.

3. Создайте двумерный массив целых чисел и найдите сумму всех его нечетных элементов.

4. Введите с клавиатуры целочисленные элементы матриц 3×3, умножьте каждый элемент матрицы на 3 и выведите результат на экран.

5. Составить программу, которая формирует двумерный массив случайных чисел и вычисляет значение среднего арифметического его элементов, больших 80.

6. Дана целочисленная матрица A размером n *x* m. Получить целочисленную матрицу B, для которой B[i,j]=A[i,j]-3A[i,j]**.**

7. Дана квадратная матрица A. Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы.

8. Дана матрица *A* размером n x *m.* Наибольшее из значений элементов первой и последней строк.

9. Определить наименьший элемент каждой четной строки двумерной матрицы A.

10. Дана квадратная матрица A. Найти элементы, равные заданному к, заменить их на число 10.

ов каждой строки. 11. Вычислить сумму и число положительных элементов матрицы *A [N, N]*.

12. Дана матрица *A* размером n x *m.* Определить *k* — количество особых элементов массива A, считая его элемент особым, если он больше суммы остальных элементов его столбца.

13. Для целочисленной квадратной матрицы найти число элементов, кратных *k.*

14. Дана целочисленная квадратная матрица. Вычислить сумму элементов, оба индекса которой четные.

15. Дана прямоугольная матрица. Найти сумму элемент

Лабораторная работа №14
Тема: «Работа со строками».

Цель: *Научиться использовать структурированный тип данных (строки) при решении задач на языке Turbo Pascal*

Теоретические сведения:

Для обработки строковых величин в Турбо Паскале существуют специальные процедуры и функции:

Length(st) – значением функции является длинна строковой переменной **st**.

Copy(st,m,n) – значением функции является подстрока из **n** символов, вырезанных из строки **st**, начиная с позиции указанной параметром **m**.

Delete(st,m,n) – данная процедура удаляет **n** символов из строки **st**, начиная с позиции указанной параметром **m**.

Concat(st1,st2,…stn) – соединение строк. Можно использовать конструкцию **st1+ st2+…+ stn**.

Insert(st1, st2,m) – втавка в строку **st2** строки **st1**, начиная с позиции **m**. Общая длина строки не превышает длину строки **st2**.

Pos(st1, st2) - значением функции будет номер позиции в которой в строке **st2** первый раз встречается строка **st1**.

Str(x,st) – заданное числовое значение преобразуется в строку символов. Значение присваивается переменной **st**.

Val(st,x,c) – строка символов **st**, состоящая из цифр, преобразуется в число. Значение передаётся переменной **x**. Параметр определяется средствами Турбо Паскаля.

Пример:

Составить алгоритм, подсчитывающий количество тех слов в строке из N букв, в которых третьей является заданная буква b. Слова разделены пробелами. Других знаков препинания нет.

Алгоритм может быть следующим: строка просматривается и всюду, где нужная буква стоит третьей после пробела или начала фразы, а перед ней находится не пробел, счётчик увеличивается на 1. Перед этим проверяется, есть ли в фразе хотя бы две буквы.

Задача может иметь и другое решение.

```
Program  bukvy (input,output):
Uses crt;
Var strk:string;
      bukva:string[1];
      I, k:integer;
Begin
Clrscr; {ввод строки};
Writeln('введите строку символов');
Readln (bukva);
clrscr;
k:=0; {проверка строки на наличие хотя бы трёх первых символов}
if length(strk)=3
   then
        if (copy(strk, I, 1)<>'') and (copy(strk,3,3) =bukva)
          then k:=k+1;
for I:=1 to length(strk)-3 do {поиск буквы в остальных словах строки}
   if (copy(strk, I, 1)='') and (copy(strk, I+1)<>'') and (copy(strk, I+2, 1)<>'') and
(copy(strk, I+3,1)=bukva)
     then k:=k+1;
{печать исходной строки и количества слов}
```

```
writeln('В заданной строке:');
writeln (strk);
writeln ('слов с искомой буквой', bukva:3,k);
repeat until keypressed;
end.
```

Задания:

1. Подсчитать сколько раз в заданной строке встречается заданная буква.
2. В заданной строке заменить все сочетания подстроки «на» на подстроку «над».
3. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.
4. В заданной фразе после каждой буквы «о» вставить сочетание «ok».
5. Решить предыдущую задачу при условии, что вставляемое сочетание вводится с клавиатуры.
6. Дана строка символов, среди которых есть двоеточие (**:**). Определить, сколько символов ему предшествует.
7. Написать программу, которая перевернёт введённое с клавиатуры слово или фразу.
8. Дана строка, содержащая текст, заканчивающийся точкой. Вывести на экран слова, содержащие три буквы.
9. Дана строка. Преобразовать ее, удалив каждый символ *.
10. Дана строка. Подсчитать, сколько в ней букв *z, k, t.*
11. Дана строка символов. Подсчитать наибольшее количество идущих подряд символов abc.
12. Выяснить, верно ли, что в строке символов имеются пять идущих подряд букв s.
13. В строке символов определить число вхождений группы букв.
14. Заменить в строке символов каждую группу букв child группой букв children.
15. Дана строка символов. Подсчитать число вхождений символов +, -, * в строку.

Лабораторная работа № 15

Тема: «Работа с данными типа множество»

Цель:- *познакомить с понятием "множество" в языке программирования Pascal;*

- *выработать навыки работы со структурой данных множество.*

Теоретические сведения

Под множеством понимают ограниченный, неупорядоченный набор различных элементов одного типа. В отличие от массивов к элементам множества нет прямого доступа (по индексам этих элементов, как в массивах). Поэтому ввод-вывод множеств производится с использованием операций объединения (при вводе) и проверки принадлежности (при выводе). Под мощностью множества понимают количество элементов, содержащихся в данном множестве.

Перед выполнением работы необходимо ознакомится с правилами описания и использования переменных типа множество, типизированных констант типа множество, переменных, заданных перечислением, изучить допустимые операции над переменными этих типов.

Примеры описания множеств:

```
L=set of ‘A’..’Z’;
C=(Red, Green);
H=set of 1..31;
N=set of ‘0’..’9’;
M=set of [‘.’,’,’,’!’,’?’,’-’,’+’,’=’];
D=set of 0..9;
U=set of char;
```

Процедуры работы с множествами:

Include(S,i) – включает новый элемент i в множество S;

Exclude(S,i) – исключает элемент i из множества S.

Пример программы

```
program r;
var m1: set of 1..50;
     m2: set of 1..50;
     m3: set of 1..50;
   i:integer;
begin
  m1:=[1,4]; m2:=[4,3];
  m3:=m1+m2;
  for i:=1 to 4 do                {вывод элементов множества}
   if i in m3 then write (i:3);
end.
```

Задания:

1. Для заполнения карточек «Спортлото» необходимо получить набор из к случайных чисел:
   - числа должны находиться в диапазоне 1..36;
   - числа не должны повторяться.

2. Дан текст из строчных латинских букв, за которым следует точка. Напечатать: все буквы, входящие в текст не менее двух раз.

3. Дан текст из строчных латинских букв, за которым следует точка. Напечатать: все буквы, входящие в текст по одному разу.

4. Имеются три множества символьного типа, которые заданы своими конструкторами:
Y1=['A','B','D','R','H']
Y2=['R','A','H','D']
Y3=['A','R'].
Сформировать новое множество Y=(Y1+Y2)*Y3. Распечатать элементы множества Y.

5. Имеются три множества символьного типа, которые заданы следующим образом
Y1=['A','B','D','R','H']
Y2=['R','A','H','D']
Y3=['A','R'].
Сформировать новое множество Y=Y1+Y2*Y3. Распечатать элементы множества Y.

6. Опишите множества R и L, содержащие русские и латинские буквы. В цикле вводите русские и латинские буквы и выводите соответствующее сообщение. Выход из цикла —введенная буква Z.

7. Опишите множество Pr (1..20) и поместите в него все простые числа в диапазоне1..20. В цикле организуйте ввод чисел в диапазоне 1..20 и определите, простые они или нет. Выход из цикла — введенное значение, равное 99.

8. Известны сорта роз, выращиваемых тремя цветоводами: «Анжелика», «Виктория», «Гагарин», «Ave Maria», «Катарина», «Юбилейная». Определить те сорта, которые имеются у каждого из цветоводов, которые есть хотя бы у одного из цветоводов, которых нет ни у одного из цветоводов.

9. В озере водится несколько видов рыб. Три рыбака поймали рыб, представляющих некоторые из имеющихся видов. Определить:
- какие виды рыб есть у каждого рыбака;
- какие рыбы есть в озере, но нет ни у одного из рыбаков.

10. Составить программу, которая вычисляет сумму тех элементов двумерного массива, номера строк и столбцов которых принадлежат соответственно непустым множествам *Si* и S_2.

11. Задано некоторое множество M и множество T того же типа. Подсчитать, сколько элементов из множеств T и *M* совпадает.

12. Опишите множества Ml (1,2,3,4) и M2 (3,4,1). Получите результирующее множество M3= Ml -— M2. Определите, имеется ли в M3 элемент 2.

13. Опишите множества Ml (1,2) и M2 (5,6). Получите результирующее множество M3=M1+M2. Определите, имеется ли в M3 элемент 7.

14. Описать множество гласных и согласных букв русского языка. Определить количество гласных и согласных букв в предложении, введенном с клавиатуры.

Лабораторная работа №16
Тема: «*Файлы последовательного доступа*»

Цель: *способствовать формированию знаний и умений по работе с файловыми переменными, а также приобретению навыков работы с текстовыми файлами: файлами последовательного доступа в языке Turbo Pascal.*

Ход работы:

1. Изучить теоретическую часть.
2. Проработать примеры работы с текстовыми файлами.
3. Выполнить задания, опираясь на теоретический материал и примеры.
4. Ответить на контрольные вопросы.
5. Предъявить преподавателю результаты работы программы и исходные коды.
6. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, блок-схема, программный код, результаты тестирования программы, выводы).

🕮 Теоретическая часть

Файлом последовательного доступа называется файл, к элементам которого обеспечивается доступ в такой же последовательности, в какой они записывались. *Фалом прямого доступа* называется файл, доступ к элементам которого осуществляется по адресу элемента.

При организации данных в файл последовательного доступа *нельзя одновременно* читать данные из файла и записывать данные в файл, т.к. для чтения некоторого элемента последовательного файла указатель обработки помещен на данный элемент, а для записи нового элемента этот указатель одновременно должен быть в конце файла.

В Паскале имеется 3 класса файлов:

Текстовые

Типизированные

Нетипизированные

Формат описания файловой переменной

Var <имя файловой переменной>: file of <тип данных>;

Для работы *с любым типом файлов* используются следующие процедуры и функции:

Процедуры и функции для работы с любыми файлами	Описание
ASSIGN(F,Name)	*Связь файловой переменной F с внешним файлом Name.* Name-переменная или константа типа String,Char. Имя файла должно быть написано в соответствии с правилами DOS. F- переменная любого файлового типа.
RESET(F[,SIZE])	*Открытие существующего файла.* Открывается существующий файл, с которым связана файловая переменная F и указатель текущего компонента файла перемещается на начало файла. Необязательный параметр SIZE используется только в нетипизированных файлах и задает размер пересылаемого элемента информации в байтах.
REWRITE(F[,SIZE])	*Открытие нового пустого файла.* Открывается новый пустой файл, и ему присваивается имя, заданное процедурой ASSIGN. Если файл с таким именем уже существует, то он уничтожается. Необязательный параметр SIZE используется только в нетипизированных файлах и задает размер пересылаемого элемента информации в байтах.
ERASE(F)	*Уничтожение файла.* Удаляет неоткрытый внешний файл, задаваемый переменной F.
RENAME (F,NewName)	*Переименование файла.* Переименовывает неоткрытый внешний файл, задаваемый переменной F, новое имя задается строкой NewName.

CLOSE(F)	*Закрытие файла.*
EOF(F)	*Конец файла.*

Текстовые файлы

Текстовые файлы - файлы на диске, состоящие из символов ASCII. Для разделения строк используются символы <конец строки>. Текстовые файлы являются файлами с последовательным доступом. В любой момент времени доступна только одна запись файла. Другие записи становятся доступными лишь в результате последовательного продвижения по файлу. Текстовые файлы внутренне разделены на строки, длины которых различны. Особенностью работы с текстовыми файлами является то, что параметры, значения которых вводятся и выводятся с помощью процедур *READ* или *WRITE*, могут быть не только типа *String* или *Char*, но и других типов (целых, вещественных - при вводе, целых, вещественных, логических - при выводе).

При работе с текстовыми файлами необходимо, прежде всего, объявить соответствующую файловую переменную:

Var F:text;

Начало работы с текстовым файлом - стандартное: файловой переменной ставится в соответствие имя файла (процедура *Assign*), а затем открывается новый текстовый файл (процедура *Rewrite*) или открывается существующий текстовый файл (процедура *Reset*).

Процедуры и функции для работы с текстовыми файлами:

Процедуры и функции для работы с текстовыми файлами	Описание
APPEND(F)	*Открытие существующего файла.* Открывается существующий файл, с которым связана файловая переменная F и указатель текущего компонента файла перемещается на конец файла. F- переменная любого файлового типа. После открытия текстовый файл можно дополнять информацией, начиная с конца строки.
READ(F,<список>)	*Чтение из файла.*

READLN(F,<список>)	*Чтение строки из файла.*
SETTEXTBUF(F,BUF[,SIZE])	*Назначение буфера ввода- вывода.* Для файла, с которым связана файловая переменная F, назначается буфер ввода-вывода в виде переменной BUF любого типа.
WRITE(F,<список>)	*Запись в файл.*
WRITELN(F,<список>)	*Запись строки в файл.*
EOLN(F)	*Коней строки файла.*
EOF(F)	*Конец файла.*

Приступая к решению задач, следует вспомнить, что:

1. В программе, которая выводит результаты в файл или читает исходные данные из файла, должна быть объявлена файловая переменная типа.
2. Для доступа к конкретному файлу файловую переменную нужно связать с этим файлом (при помощи процедуры Assign).
3. Для того, чтобы файл был доступен, его надо открыть (для чтения с помощью процедуры Reset, для записи – rewrite).
4. Запись в файл производится процедурой Write, чтение – Read.
5. По завершении работы с файлом его нужно закрыть Close.
6. Файл, созданный программой, в которой тип файловой переменной объявлен как Text, можно посмотреть с помощью редактора текста.

Для создания и заполнения файла требуется следующая последовательность действий:

1. Описать файловую переменную.
2. Описать переменную того же типа, что и файл.
3. Произвести назначение (Assign).
4. Открыть файл для записи (Rewrite).
5. Записать в файл данные (write).
6. Закрыть файл (Close).

Для последовательного чтения данных из файла требуется выполнить следующие действия:

1. Описать файловую переменную.

2. Описать переменную того же типа.

3. Выполнить назначение (Assign).

4. Открыть файл для чтения (Reset).

5. В цикле читать из файла (Read).

6. Закрыть файл (Close).

Примеры программ для работы с файлами текстового типа

Пример1. Фрагмент программы, записывающей в текущую папку файл File01.txt с текстом Text in file.

```
var f: text; //Описание файловой переменной (логического файла) f
begin
  assign(f,'File01.txt'); //Соответствие f и физического файла File01.txt
  rewrite(f); //Создание и открытие файла для записи
  writeln(f,'Text in file'); //Запись в файл текста Text in file
  close(f); //Закрытие файла
  readln;
end.
```

Пример 2. Фрагмент программы, считывающей текст из файла File01.txt, находящегося в текущей папке, и выводящей этот текст на экран.

```
var f: text; //Описание файловой переменной (логического файла) f
    s: string; //Описание строки s
begin
  assign(f,'File01.txt'); //Соответствие f и физического файла File01.txt
  reset(f); //Открытие файла
  readln(f,s); //Считывание текста из файла в строку s
  close(f); //Закрытие файла
  writeln('Text from file:    ',s);//Вывод на экр текста, записанного в ф-л
  readln;
end.
```

Пример 3. Удаление или переименование файла

```
    Var
     F: file ;
     Ch: char ;
     St: string;
    begin
     Write('Введите имя файла: ');
     Readln(St);   {Чтение имени}
     Assign(F, St);  {Назначить имя файловой переменной}
Write('Удалить файл (У), Переименовать(П), Выход(В)');
     Readln(Ch) ;
     case Ch of
          'У ', ' у' : Erase(F);   {Удаление файла}
           'П','п' : begin
              Write('Введите новое имя файла: ');
              Readln(St) ;
              Rename(F, St); {Переименование файла}
              end;
          'В','в' : Halt(1);
     end;
     end.
```

В приведенном примере альтернативный выбор тех или иных действий зависит целиком от того, что будет введено с клавиатуры. Этот вариант программы не позволяет обработать ошибочные ситуации в случае, если файла с именем *St* не существует на диске.

Пример 4. Вариант программы для проверки существования файла на диске

☞Для того чтобы файловые операции выполнялись без ошибок, необходимо использовать специальную функцию *IOresult*. Функция работает без параметров и возвращает значение типа *integer*, представляющее статус последней выполненной операции ввода-вывода. Использование этой функции в программах возможно лишь в том случае, если на время выполнения файловых операций отключена стандартная проверка операций ввода-вывода. Для этих целей используется либо специальная опция в интегрированной системе, либо директива компилятора *{$I}*, которая может задаваться внутри текста программы.

```
    Var
     F: file ;
     St: string;
    begin
     Write('Введите имя файла : ');
     Readln(St) ;
     Assign(F, St) ;
     {$I-}        {Отключить стандартную обработку ошибок}
     Reset(F);     {Открыть файл}
     {$I+}         {Включить стандартную обработку ошибок}
     if IOresult = 0 then
          begin
          Writeln('Файл существует и нормально открыт');
          Close(F); {Закрыть файл}
          end
     else
          Writeln('Файла с именем '+St+' на диске нет');
end.
     end;
     end.
```

После корректного выполнения операции ввода-вывода функция *IOresult* возвращает значение, равное нулю, в остальных случаях функция возвращает соответствующий код ошибки.

Пример 5. Создать текстовый файл, в который записать 3 предложения. Прочитать этот файл, вывести его содержимое на экран. Определить длину каждого предложения.

```
Program File_text;
   var
      f1 : text;
      st : string;
      n: byte;
 begin
   assign   (f1, 'file1.txt'); {связать с файлом file1.txt
файловую переменную f1  }
   rewrite  (f1); { создать новый файл  с именем file1.txt }
   writeln ( f1, 'Очень полезно изучать'); { записать
предложения в файл}
   writeln ( f1, ' всем  студентам ');
   writeln (f1, ' язык  Pascal ');
   close (f1);    { закрыть файл для записи }
   reset (f1); { открыть файл для чтения }
   while not eof (f1) do { пока не конец файла  f1}
     begin
        readln (f1, st);    {читаем строку из файла f1 }
        writeln(st);         { выводим на экран }
        n:= length (st);     {определяем длину строки }
        writeln ('  длина =',n);
      end;
    close (f1); { закрыть файл  для чтения}
end.
```

Пример 6. Написать программу, которая вводит с клавиатуры список фамилий учащихся, а затем выводит его, кроме тех учащихся, у которых фамилия начинается с буквы 'Ш'".

Так как заранее количество данных не известно, то для их хранения используем файл.

```
Program L;
Var
I,N : Integer;
F : text;
S : String;
Begin
     [redacted];{Связываем переменную F с файлом}
     [redacted]('Введите количество учащихся');
     Readln(N); {Вводим количество учащихся}
     [redacted]; {Создаем файл для записи в него данных}
     For [redacted] Do {Для всех учащихся}
          Begin
               Writeln('Введите фамилию');
               Readln(S);
               [redacted] {запись строки с фамилией в файл}
          End;
     [redacted]; {закрываем файл}
     [redacted]; {открываем существующий файл}
     Writeln; Writeln('Список учащихся:');
     [redacted] Do {пока не конец файла, делай}
          Begin
               [redacted]; {считывание строки из файла}
               If S[1]<>'Ш' Then Writeln(S)
          End;
     [redacted] {закрываем файл}
End.
```

☞Задание 1. Изучить и проработать примеры работы с текстовыми файлами. В Примере 6 дописать требуемые процедуры, функции и операторы, предъявить результат работы программы.

☞**Задание 2.**Выполнить по вариантам :

1. Дан текстовый файл f. Записать в файл g компоненты файла f в обратном порядке.
2. Даны текстовые файлы f и g. Записать в файл h сначала компоненты файла f, затем - компоненты файла g с сохранением порядка.
3. **Дан текстовый файл f. Найти количество слов, начинающихся на букву «а»**
4. Дан текстовый файл f. Получить все его строки, содержащие более 20 символов.
5. Дан текстовый файл f. Найти слова, заканчивающие на of.
6. Дан текстовый файл f. Получить самую длинную строку файла. Если в файле имеется несколько строк с наибольшей длиной, то получить одну из них.
7. Дан текстовый файл f. Исключить пробелы, стоящие в концах его строк. Результат поместить в файл g.
8. Даны текстовый файл и строка s. Получить все строки файла f содержащие в качестве фрагмента строку s.
9. Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.
10. **Дан текстовый файл. Ввести слово с клавиатуры. Определить сколько раз встречается данное слово в файле.**
11. Создать текстовый файл. Вывести его на экран. Вывести на экран слова, начинающиеся с буквы "В".
12. Создать текстовый файл. Определить количество вопросительных предложений и количество слов в первом предложении.
13. Создать текстовый файл. Вывести его на экран. Вывести на экран последнее предложение. Определить количество слов в нем.
14. **Создать текстовый файл на диске с: , вводя информацию построчно. Подсчитать количество строк в файле.**
15. Создать текстовый файл. Скопировать из одного файла в другой только определенные символы (например, ряд гласных) и посчитать их общее количество.

Контрольные вопросы:

1. Понятие файла. Описание файлового типа данных.
2. Файлы последовательного и прямого доступа. Средства обработки файлов.
3. Текстовые файлы. Процедуры и функции работы с текстовыми файлами.
4. Каковы особенности работы с текстовыми файлами?
5. Каково должно быть содержание программы по созданию файла и его корректировки (замена элементов, добавление элементов, удаление элементов)?

Как подсчитать число строк в текстовом файле?

Лабораторная работа №17
Тема: «Типизированные файлы».

Цель: сформировать навык применения основных процедур работы с файловыми типами данных в языке Turbo Pascal на примере типизированных файлов целых чисел.

Теоретические сведения:

Приступая к решению задач, следует вспомнить, что:

7. В программе, которая выводит результаты в файл или читает исходные данные из файла, должна быть объявлена файловая переменная типа.
8. Для доступа к конкретному файлу файловую переменную нужно связать с этим файлом (при помощи процедуры Assign).
9. Для того, чтобы файл был доступен, его надо открыть (для чтения с помощью процедуры Reset, для записи – rewrite).
10. Запись в файл производится процедурой Write, чтение – Read.
11. По завершении работы с файлом его нужно закрыть Close.
12. Файл, созданный программой, в которой тип файловой переменной объявлен как Text, можно посмотреть с помощью редактора текста.
    Формат описания файловой переменной

Var <имя файловой переменной**>: file of <**тип данных**>;**

Процедуры и функции для работы с файлами:

Append – открывает существующий файл для добавления элементов в конце файла.

Flush – сбрасывает для текстового файла буфер вывода.

Readln – работает так же как и Read, но дополнительно выполняет пропуск всех оставшихся элементов текущей строки и переводит указатель текущей позиции файла в начало следующей строки текстового файла.

SeekEof – возвращает для текстового файла состояние Eof(конец файла).

SeekEoln – возвращает для текстового файла состояние Eoln(конец строки).

SetTextBuf – назначает для текстового файла буфер ввода-вывода.

Writeln – работает так же как и Write, но после записи указанных в процедуре значений, дополнительно записывает в текстовый файл признак конца строки Eoln.

Assign(var F; name: string); - устанавливает соответствие между файлом и файловой переменной, открывает все другие процедуры работы с файлами. F -переменная любого файлового типа, name - полное имя файла.

Close(var F); - окончание процедур работы с файлами (закрытие файла).

Rewrite(var F: file); - создаёт и открывает новый файл.

Reset(var F: file); - открывает существующий файл.

Append(var F: text); - открывает существующий текстовый файл и позиционирует указатель обработки на конец файла. После этого можно дополнять текстовый файл информацией, начиная с конца строки.

Erase(var F); - удаляет неоткрытый внешний файл любого типа, задаваемый переменной F.

Rename(var F: newname: string); - переименовывает неоткрытый файл F любого типа. Новое имя задаётся строкой newname.

Для создания и заполнения файла требуется следующая последовательность действий:

1. Описать файловую переменную.
2. Описать переменную того же типа, что и файл.
3. Произвести назначение (Assign).
4. Открыть файл для записи (Rewrite).
5. Записать в файл данные (write).
6. Закрыть файл (Close).

Для последовательного чтения данных из файла требуется выполнить следующие действия:

1. Описать файловую переменную.
2. Описать переменную того же типа.
3. Выполнить назначение (Assign).
4. Открыть файл для чтения (Reset).
5. В цикле читать из файла (Read).
6. Закрыть файл (Close).

Задания

1. Дан файл чисел f. Найти сумму его элементов. Записать сумму в файл g.
2. Дан файл чисел f. Найти в нем наибольшее число. Записать полученное число в файл g.
3. Заполнить файл последовательного доступа f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g те компоненты файла f, которые являются чётными.
4. Даны 2 файла чисел. Записать в третий файл такие элементы первого файла, которых нет во втором.
5. Дан файл чисел и некоторое число. Найти позицию вхождения этого числа в файл, либо определить, что такого числа в файле нет.
6. Составить программу, которая создает файл, состоящий из 10 целых чисел. Прочитайте файл и определите, есть ли в нем заданное с клавиатуры значение.
7. Составить программу, которая создает файл, состоящий из 10 целых чисел. Прочитайте файл и вычислите суму его элементов.

8. Дан файл чисел. Верно ли, что его элементы расположены в следующем порядке: сначала пять положительных чисел, затем пять отрицательных, затем снова пять положительных, и т.д.
9. Дан файл чисел. Записать в другой файл максимум из первых десяти элементов, затем максимум из следующих десяти элементов и т.д.
10. Даны 2 файла. Каждый элемент каждого файла хранит одну цифру числа. Сложить эти числа.
11. Составить программу, которая создает файл, состоящий из N целых чисел. Прочитайте файл и выведите только положительные элементы.
12. Записать в файл последовательного доступа N действительных чисел. Вычислить произведение компонентов файла и вывести их на печать.
13. Заполнить файл последовательного доступа N действительными числами с помощью датчика случайных чисел. Найти сумму минимального и максимального элементов файла.
14. Записать в файл последовательного доступа N действительных чисел. Найти разность первого и последнего компонентов файла.
15. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Найти количество нечётных чисел среди компонентов файла и поместить их в файл последовательного доступа g. Вывести файл g на печать.

Лабораторная работа № 18

Нетипизированные файлы

Цель:

Теоретические сведения.

Задания

1. Организовать простейшую базу данных по студентам группы. Сведения о студенте включают: ФИО, год рождения, пол, средний балл. Обеспечить ввод данных, редактирование, вывод на экран. Информацию хранить в типизированном файле.

2. Дан текстовый файл. Считая, что количество букв в одном слове не превосходит 20, определить, сколько в файле имеется слов, состоящих из одного, двух, трех и т.д. символов. Результат вывести в другой текстовый файл.

3. Дан текстовый файл, который содержит массив целых чисел. Необходимо записать массив чисел из текстового файла в нетипизированный файл, а также вычислить среднее арифметическое элементов файла.

4. Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Создать файл, содержащий сведения о нескольких автомобилях, после чего определить фамилии владельцев и номера автомобилей заданной марки. Марка автомобиля вводится пользователем.

5. Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Создать файл, содержащий сведения о нескольких автомобилях, после чего определить количество автомобилей каждой марки.

6. Создать файл, содержащий не более 100 случайных целых чисел. Выполнить сортировку чисел по возрастанию.

7. Создать файл, содержащий не более 100 случайных целых чисел. Создать новый файл, разместив все нечетные числа в начале файла, а четные – в конце, при этом порядок следования чисел сохраняется.

8. Дан текстовый файл, содержащий строки произвольной длины. Отформатировать текст и записать его в новый файл так, чтобы все строки имели одинаковую длину, равную длине самой длинной строки. Форматирование выполняется добавлением пробелов между словами.

9. Дан текстовый файл, содержащий сведения о товарах по предприятиям. Сведения включают наименование, стоимость, адрес предприятия, название предприятия и др. Сведения об одном товаре находятся в одной строке, отделены друг от друга точкой с запятой.

Считать сведения, записать в типизированный файл соответствующего типа, вывести на экран название товара с максимальной стоимостью.

10.Дан текстовый файл, который содержит массив целых чисел. Необходимо записать массив чисел из текстового файла в нетипизированный файл, а также найти минимальный элемент.

11.Считать сведения, записать в типизированный файл соответствующего типа, вывести на экран название товара с максимальной стоимостью.

12.Дан текстовый файл. Записать все строки файла в новый файл, изменив порядок следования букв в каждой строке на противоположный. Посчитать количество букв в каждой строке и количество строк.

13. Дан текстовый файл. Записать в новый файл строку с максимальным количеством букв. Вывести эту же строку на экран, изменив порядок следования букв на противоположный.

14. Создать файл, содержащий не более 50 случайных целых чисел. Записать в новый файл и вывести на экран максимальное и минимальное числа.

15. Дан текстовый файл с текстом не менее 15 строк. Записать в новый файл все слова, начинающиеся с определенной буквы. Буква вводится по запросу пользователя.

Контрольные вопросы

1.Что такое «типизированный файл»?

2.Что такое «нетипизированный файл»?

3.Что такое «текстовый файл»?

4.Что такое «указатель файла»?

5.Как установить указатель файла в требуемую позицию?

6.Для каких типов файлов можно устанавливать позицию указателя файла?

7.Какие процедуры предназначены для открытия типизированного файла?

8.Какие процедуры предназначены для открытия текстового файла?

9.Какие процедуры и в какой последовательности надо вызывать для чтения данных из типизированного файла?

10. Как задать размер блока для нетипизированного файла?

11.Какими процедурами выполняется чтение и запись для нетипизированных файлов?

Лабораторная работа №19
Тема: «Организация процедур».

Цель: Научиться решать задачи, используя внешние процедуры в Turbo Pascal.

Теоретические сведения:

Задача: Даны натуральные числа a и b. Найти их НОД (наибольший общий делитель).

```
Program NOD;
Var A, B, C: Integer;
Procedure    Evklid (M, N: Integer; Var K: Integer);
Begin
                While M<>N Do
If M>N
Then M: = M – N
Else N: = N – M
K: = M
End;
Begin
                Write ( ' a = ' );
                Readln (A);
                Write ( ' b = ' );
                Readln (B);
                Evklid (A + B, Abs (A – B), C);
                Evklid (C, A * B, c);
Writeln (' НОД = ', C)
End.
```

Задание:

1) Даны координаты вершин многоугольника (x1,y1,x2,y2,x3,y3,…x10,y10). Определить его периметр (вычисление расстояния между вершинами оформить подпрограммой).

2) Даны две дроби A\B и C\D *(A, B, C, D* — натуральные числа). Составить программу для нахождения частного первой и второй дроби.

3) Составить программу для вычисления суммы факториалов всех нечетных чисел от1 до 9.

4) Составить программу для нахождения наименьшего общего кратного двух натуральных чисел

$$НОК(A,B) = \frac{A * B}{НОД(A,B)}$$

5) Составить программу для нахождения наибольшего общего делителя четырех натуральных чисел.

6) Задан массив D. Определить следующие суммы: D[1]+D[2]+D[3]; D[3]+D[4]+D[5]; D[5]+D[6]+D[7].

7) На плоскости заданы своими координатами n точек. Составить программу, определяющую, между какими из пар точек самое большое расстояние. Координаты точек занести в массив.

8) Составить программу для вычисления суммы факториалов всех четных чисел от m до n.

9) На плоскости заданы своими координатами n точек. Создать массив размером n(n -1), элементами которого являются расстояния от каждой точки до n -1 других.

10) Заменить отрицательные элементы линейного массива их модулями, не пользуясь стандартной функцией вычисления модуля. Подсчитать количество произведенных замен.

11) Дан массив A(N) (N-четное). Сформировать массив B(N), элементами которого являются большие из двух рядом стоящих в массиве A чисел. (Например, A=(1,3,5,-2,0,4,0). Элементами массива B будут 3,5,4)

12) Дано натуральное число N. Составить программу для формирования массива, элементами которого являются цифры числа N.

13) Составить программу, определяющую, в каком из данных двух чисел больше цифр.

14) Дан массив *A(N) (N*— четное). Сформировать массив *B(M),* элементами которого являются средние арифметические соседних пар рядом стоящих в массиве *A* чисел. (Например, массив *A* состоит из элементов 1; 3; 5; -2; 0; 4; 0; 3. Элементами массива *B* будут 2; 1,5; 2; 1,5).

15) Даны числа *a, b, c, d* — длины сторон прямоугольника. Вычислить его площадь, разделив данный прямоугольник на 2 треугольника и используя формулу Герона для нахождения их площади.

Лабораторная работа № 20

Тема: «Организация функций**».**

***Цель**: Научиться использовать пользовательские функции в программе.*

Теоретические сведения:

Функции в Паскале описываются в разделе описания подпрограмм.

```
program    Имя_Программы;
 uses      Список_используемых_модулей;
 label     описание_меток;
 const     описание_констант;
 type      описание_типов;
 var       описание_переменных; могут описываться в любом порядке
 procedure описание_процедур;
 function  описание_функций;
 begin
       операторы;
 end.
```

Function <имя> (формальные параметры): <тип результата>;
Const…;
Type…;
Var…;
Begin
  <операторы>;
End;

Пример1. Найти максимальное из 4-х чисел

```
Program max_4;
  uses crt;
  var a, b, c, d, m: integer;
  function max_2(x,y:integer):integer;
    var max:integer;
  begin
    if x>y then
      max:=x
    else
      max:=y;
    max_2:=max;
  end;
Begin
  clrscr;
  readln(a, b, c, d);
  m:=max_2(a,b);
  m:=max_2(m,c);
  m:=max_2(m,d);
(или m:=max_2(max_2(max_2(a,b), c), d);)
  writeln(m);
End.
```

Пример2. Организация ввода координат вектора. Вычислить длину вектора. Для начала определим список формальных параметров: входные и выходные данные. Нам потребуется Размерность векторного пространства (k: byte) и переменная x пользовательского типа vector=array[1..100] of real.

Для ввода координат будем использовать procedure, так как выходных данных будет много (его координаты), а так как длина вектора это число, поэтому используем function.

```
program pro;
type vector=array [1..100] of real;
var k: byte;
   dl:real;
   x: vector;

procedure vvod (var y:vector);                    {Процедура ввода вектора}
  var i:byte;
      begin
         writeln('Введите координаты вектора');
         for i:=1 to k do
         readln(y[i]);
      end;

function dlvec(y:vector):real;          {Функция вычисления длины вектора}
  var i:byte;
     s:real;
      begin
       for i:=1 to k do
              s:= s+ sqr(y[i]);
       dlvec:= sqrt(s);            {Обязательно в конце нужно имени функции присвоить
                                   вычисленное значение}
      end;
begin
write(' Введите размерность векторного  пространства k=');
readln(k);
vvod(x);                                          {вызов процедуры}
dl:=dlvec(x);                                        {вызов функции}
writeln('Длина вектора X равна', dl: 8: 2);
end.
```

Практическая часть

Задание: Решить данные задачи с использованием функций.

1) Даны отрезки a, b, c, d. Для каждой тройки этих отрезков, из которых можно построить треугольник, напечатать площадь данного треугольника. Определить функцию Plo(x, y, z), печатающую площадь треугольника со сторонами x, y, z, если такой треугольник существует.

2) Даны действительные числа s, *t*. Получить g(1.2, s)+g(t, s)-g(2s-l. st),

где g (a,b)= (a^2+b^2) /($a^2+2ab+3b^2+4$)

3) Даны действительные числа s, t. Получить f (t, -2s, 1.17)+f (2.2, t, s-t), где

f (a, b, c)=(2a-b-sin c) /(5+|c|).

4) Даны действительные числа a, b, c. Получить

(max (a, a + b) + max (a , b + c)) / (1 + max ((a , bc, 1.15))

5) Даны действительные числа a, h, натуральное число n. Вычислить

f(a)+f(a+h)+…+f(a+(n-1)h)+f(a+nh), где $f(x)=(x^2+1)\cos^2 x$.

6) Даны действительные числа a, b. Получить

v= min (ab, a + b), u=min (a, b), x= min($u^2 + v^2$. 3.14).

7) Даны действительные числа s, t. Получить h(s, t)+max(h^2(s-t, st), h^3(s-t, s+t))+h(l, 1),

где $h(a, b)=a/(1+b)+b/(1+a)-(a-b)^2$.

8) Написать программу вычисления суммы 1 + 1\2 +1\3 +… + 1\n, используя функцию.

9) Написать программу вычисления суммы. $S= 1 – 1\2 + 1\3 - …. + (-1)^{n+1}/n$, используя функцию.

10) Имеется часть катушки с автобусными билетами. Номер билета шестизначный. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета — *N*, больший — *M* (билет является счастливым, если сумма первых трех его цифр равна сумме последних трех).

Лабораторная работа №21
Тема: «Применение рекурсивных функций»

Цель: *Познакомиться с одним из эффективных способов решения сложных задач – рекурсией.*

Теоретические сведения

Рекурсия- это способ организации вычислительного процесса, при котором в ходе выполнения подпрограммы обращается сама к себе.(прямая и косвенная).

Если же несколько подпрограмм вызывают друг друга, но эти вызовы "замкнуты в кольцо", то такая рекурсия называется косвенной.

Максимальное число рекурсивных вызовов подпрограммы без возвратов, которое происходит во время выполнения программы, называется глубиной рекурсии.

Структура рекурсивной процедуры может принимать три разных формы:

1)форма с выполнением действий до рекурсивного вызова (на рекурсивном спуске);

```
procedure Rec; begin
S;
if условие then Rec; end;
```

2)форма с выполнением действий после рекурсивного вызова (на рекурсивном возврате);

```
procedure Rec; begin
if условие then Rec;
S; end;
```

3)форма с выполнением действий как до, так и после рекурсивного вызова (с выполнением действий как на рекурсивном спуске, так и на рекурсивном возврате).

```
procedure Rec; begin
S1;
if условие then Rec; S2 ; end;
```

Все формы рекурсивных процедур находят применение на практике. Многие задачи, в том числе вычисление факториала, безразличны к тому, какая используется форма рекурсивной процедуры. Однако есть классы задач, при решении которых программисту требуется сознательно управлять ходом работы рекурсивных процедур и функций. Такими, в частности, являются задачи, использующие списковые и древовидные структуры данных.

Алгоритм называется рекурсивным, если он прямо или косвенно обращается к самому себе.

Часто в основе такого алгоритма лежит рекурсивное определение какого-то понятия. Например, о факториале числа N можно сказать, что ***N! = N*(N – 1)!***, если N > 0 и N! = 1 если N = 0. Это – рекурсивное определение.

Вот еще одно рекурсивное определение.

1. 3 коровы – это стадо коров.
2. Стадо из n коров – это стадо из n – 1 коровы и еще одна корова.

Любое рекурсивное определение состоит из двух частей. Эти части принято называть базовой и рекурсивной частями. Базовая часть является нерекурсивной и задает определение для некоторой фиксированной части объектов. Рекурсивная часть определяет понятие через него же и записывается так, чтобы при цепочке повторных применений она редуцировалась бы к базе.

Пример 1:

Написать рекурсивную программу поиска минимального элемента массива.

Решение. Опишем функцию Pmin, которая определяет минимум среди первых n элементов массива a. Параметрами этой функции являются количество элементов в рассматриваемой части массива - n и значение последнего элемента этой части – a[n]. При этом если n>2, то результатом является минимальное из двух чисел – a[n] и минимального числа из первых (n-1) элементов массива. В этом заключается рекурсивный вызов. Если же n=2, то результатом является минимальное из первых двух элементов массива. Чтобы найти

минимум всех элементов массива, нужно обратиться к функции Pmin, указав в качестве параметров значение размерности массива и значение последнего его элемента. Минимальное из двух чисел определяется с помощью функции Min, параметрами которой являются эти числа.

```
Program Example _1;
Const n=10;
Type MyArray=Array[1..n] of Integer;
Const a : MyArray = (4,2, -1,5,2,9,4,8,5,3);
Function Min (a, b : Integer) : Integer;
Begin
  if a>b then Min := b else Min:=a;
End;
Function Pmin(n, b : Integer) : Integer;
Begin
  if n = 2 then Pmin := Min(n,a[1]) else Pmin := Min(a[n], Pmin(n-
1,a[n]));
End;
BEGIN
  Writeln('Минимальный элемент массива  - ', Pmin(n,a[n]));
END.
```

Пример 2:

Рассмотрим математическую головоломку из книги Ж. Арсака «Программирование игр и головоломок».

Построим последовательность чисел следующим образом: возьмем целое число i>1. Следующий член последовательности равен i/2, если i четное, и 3 i+1, если i нечетное. Если i=1, то последовательность останавливается.

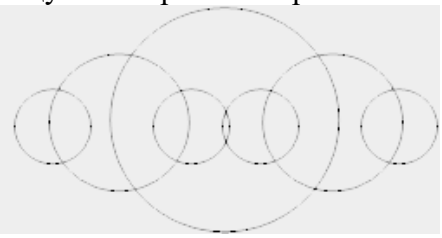
Математически конечность последовательности независимо от начального i не доказана, но на практике последовательность останавливается всегда.

Применение рекурсии позволило решить задачу без использования циклов, как в основной программе, так и в процедуре.

```
Program Arsac;
Var first: word;
Procedure posledov (i: word);
Begin
   Writeln (i);
   If i=1 then exit;
   If odd(i) then posledov(3*i+1) else posledov(i div 2);
End;
Begin
   Write (' введите первое значение '); readln (first);
   Posledov (first);
   Readln ;
End.
```

Пример 3.1:

Написать программу, строящую на экране изображение:



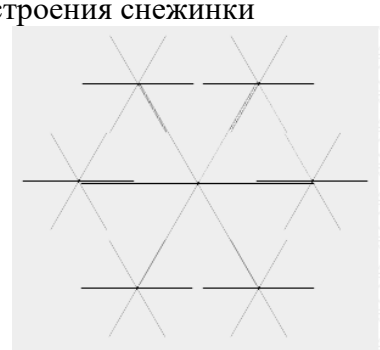
Изображение строится по следующему правилу: строится окружность с заданным радиусом r. Затем на диаметрально противоположных точках окружности (x- r и x+ r)строится вновь окружность меньшего радиуса (r=3 r/5). Для каждой меньшей окружности

на диаметрально противоположных точках вновь строится окружность меньшего радиуса, и т.д., пока радиус не уменьшится до 10.

```
program recurs;
uses graph;
var x,y,r,d,m:integer;
procedure ris(x,y,r:integer);
var i:integer;
begin
   if r<10 then exit;
   circle(x,y,r);
   for i:=1 to 1000 do; { просто цикл задержки }
   ris(x+r,y,r*3 div 5);
   ris(x-r,y,r*3 div 5);
end ;
begin {начало основной программы}
   d:=detect;
   initgraph(d,m,"e:\bp\bgi");
   x:=320;
   y:=240;
   r:=120;
   ris(x,y,r);
   readln ;
end.
```

Пример 3.2:

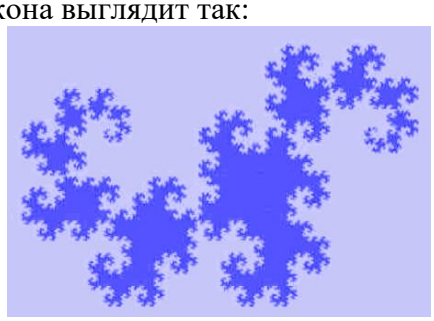
Рекурсивная программа построения снежинки



```
program sneg;
uses graph, crt;
var
   x,y,r,d,m:integer;
procedure ris(x,y,r:integer);
var
   x1,y1,t:integer;
begin
   if r<=1 then begin putpixel(x,y,15);exit end;
   for t:=0 to 6 do
   begin
      x1:=x+trunc(r*cos(t*pi/3));
      y1:=y+trunc(r*sin(t*pi/3));
      line(x,y,x1,y1);
      ris(x1,y1,r*2 div 5);
      delay(500);
   end;
end;
begin
   d:=detect;
   initgraph(d,m,"e:\bp\bgi");
   x:=320;
   y:=240;
   r:=80;
   ris(x,y,r);
   readln;
end.
```

Пример 3.3:

Рассмотрим пример решения еще одной классической задачи: «Кривая Дракона». Изображение кривой Дракона выглядит так:



Очень красиво, не правда ли. Разберемся, как же эта кривая получается.

```
program dragon;
uses graph;
var k,d,m:integer;
procedure ris(x1,y1,x2,y2,k:integer);
var xn,yn:integer;
begin
   if k>0 then
   begin
      xn:=(x1+x2) div 2 +(y2-y1) div 2;
      yn:=(y1+y2) div 2 -(x2-x1) div 2;
      ris(x1,y1,xn,yn,k-1);
      ris(x2,y2,xn,yn,k-1);
   end
   else begin line(x1,y1,x2,y2); end;
end;
begin
   readln ( k );{задаем порядок кривой}
   d:=detect;
   initgraph(d,m,"e:\bp\bgi");
   ris(200,300,500,300,k);
   readln;
end.
```

Практическая часть:

Задание:

На оценку «3»: рассмотреть работу программы из примера 1, 2.

На оценку «4»: одна из программ примера 3.1, 3.2, 3.3

На оценку «5»:

1. Напишите рекурсивную функцию, которая возвращает среднее из n элементов массива чисел.

2. Найти первые К чисел Фибоначчи с помощью рекурсии. (Последовательность Фибоначчи: 1 1 2 3 5 8 13 21... ($a_к = a_{к-1} + a_{к-2);}$)

3. Написать функцию сложения двух чисел, используя только прибавление единицы.

4. Написать функцию умножения двух чисел, используя только операцию сложения.

5. Вычислить сумму элементов одномерного массива.

6. Вычислить произведение элементов одномерного массива.

7. Вычислить, используя рекурсию, выражение

$$\sqrt{6+2\sqrt{7+3\sqrt{8+4\sqrt{9+\ldots}}}}$$

Контрольные вопросы:

1. Что такое рекурсия?
2. Что означает «глубина» рекурсии?
3. Какие структуры рекурсивной функции вам известны?

Лабораторная работа № 22
Тема: «Программирование модуля»

Цель: *Разработка структуры программы с использованием модулей. Разработка и компиляция модулей, подключение разработанных модулей. Способствовать формированию умений и навыков программирования модулей.*

Ход работы:

1. Изучить теоретическую часть
2. Выполнить задания, опираясь на примеры из теоретической части
3. Ответить на контрольные вопросы
4. Предъявить преподавателю результаты работы (результаты работы программ из примеров и индивидуальной части) и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, блок-схема, программный код, результаты тестирования программы, выводы).

🕮 Теоретическая часть

Модули в *Turbo Pascal* появились в связи с необходимостью упрощения процесса разработки сложных программ. Модуль является независимо компилируемой частью программы и представляет собой мощный инструмент создания различных прикладных программ. Компилятор *Turbo Pascal* размещает программный текст каждого модуля в отдельном сегменте памяти. Хотя максимальная длина сегмента не может превышать 64 Кбайт, число одновременно открытых модулей ограничено лишь объемом доступной памяти, что позволяет создавать достаточно большие программы. Использование модулей целесообразно для создания собственных библиотек программ (процедур и функций), а также для разделения работы между отдельными программистами в группе.

Структура модуля выглядит следующим образом:

Unit <имя>;

INTERFACE интерфейсная часть>

IMPLEMENTATION <исполняемая часть>

[Begin

<инициирующая часть>]

End

Где

Unit - служебное слово, начинающее заголовок модуля;

<имя> - имя модуля (правильный идентификатор);

INTERFACE - служебное слово, начинающее интерфейсную часть модуля;

IMPLEMENTATION - служебное слово, начинающее исполняемую часть;

Begin - служебное слово, начинающее инициирующую часть (квадратные скобки означают, что эта часть необязательна);

End - признак конца модуля. Любая из перечисленных трех частей модуля может быть пустой.

Для правильной компиляции модуля его имя должно совпадать с именем файла на диске, в котором этот модуль будет храниться. Если, например, модуль определен заголовком *Unit Complex*; то его текст должен быть сохранен в файле с именем COMPLEX.PAS. Имя модуля служит для его связи с другими модулями и основной программой. Эта связь осуществляется специальным предложением

USES <список модулей>;

где *Uses* -служебное слово, <список модулей> - имена используемых модулей, перечисленные через запятую. **Например:** USES CRT, Graph, Complx;

Если в программе используются модули, то слово *Uses* должно открывать раздел описаний основной программы или следовать сразу за служебным словом *Interface* в модуле.

Интерфейсная часть. **В** этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и блоков), которые должны стать доступными основной программе и другим модулям. При объявлении процедур и функций в интерфейсной части указывается только их заголовок, например:

```
Unit Vect;
INTERFACE
Type vector = Record
 x, y, z: real; end;
Var a, b, c : vector;
 i: real;
Procedure Summa(varf,d,e: vector);
Procedure ProizvScal(var f,d: vector; var k: real);
Procedure ProizvVect(var f,d,e: vector);
```

Если теперь в основной программе будет записана строка

```
USES Vect;
```

то в ней станут доступными тип *Vector* и процедуры *Summa*, *ProizvScal* и *ProizvVect.*

Исполняемая часть. Содержит тела процедур и функций, объявленных в интерфейсной части. Здесь же могут объявляться локальные для модуля объекты - вспомогательные типы, константы, переменные, процедуры и т. д. Ранее объявленные процедуры и функции должны быть описаны в той же последовательности, в какой их заголовки записаны в интерфейсной части. Описанию глобального блока в исполняемой части должен предшествовать заголовок, в котором разрешается опускать список формальных переменных (и тип результата для функции), поскольку они уже описаны в интерфейсной части.

Создадим модуль с именем SORT, который будет содержать несколько процедур, каждая из которых реализует один из алгоритмов сортировки массива, состоящего из N целых чисел.

Пусть массив имеет имя MAS, а его размер не превышает 10000 чисел. Тогда интерфейсная часть будет иметь вид:

```
Unit SORT;
Interface
Uses CRT;
Const Dl_MAS=10000;
Type TMAS=array[1..DlMAS] of integer;
Var MAS:TMAS; N:integer;
Procedure SORT_EXCHANGE; {обменная сортировка}
Procedure SORT_CHOICE; {сортировка выбором}
Procedure SORT_QUICK; {быстрая сортировка}
```

Теперь, если в основной программе написать директиву: Uses SORT, то в ней станут доступны константа Dl_MAS, тип TMAS, массив с именем MAS, переменная N и три процедуры: SORT_EXCHANGE, SORT_CHOICE и SORT_QUICK.

При этом необходимо помнить, что все константы и переменные, объявленные в интерфейсной части модулей, наряду с глобальными константами и переменными использующей их программы, помещаются компилятором ТП в общий сегмент данных, размер которого не должен превышать 64 Кбайта.

Инициирующая часть. Она может отсутствовать вместе с начинающим ее словом *Begin* или быть пустой. В эту часть помещают исполняемые операторы, содержащие некоторый фрагмент программы. Эти операторы исполняются до передачи управления основной программе и обычно используются для подготовки ее работы. В инициирующей части, например, могут открываться необходимые файлы, устанавливаться значения переменных и т.д.

Компиляция модулей. В *Turbo Pascal* определены три режима компиляции: COMPILE, MAKE и BUILD. Они отличаются только способом связи компилируемого модуля или основной программы с другими модулями, объявленными в предложении USES.

При компиляции в режиме COMPILE все упоминающиеся в предложении USES модули должны быть предварительно откомпилированы и результаты их компиляции помещены в одноименные файлы с расширением TPU. Например, если в программе (или модуле) есть строка **USES Vect;** то на диске в каталоге, объявленном опцией UNIT DIRECTORIES, уже должен быть файл VECT.TPU. Такой файл создается в результате компиляции самого модуля.

В режиме MAKE компилятор проверяет наличие TPU-файлов для каждого объявленного модуля. Если какой-либо из файлов не обнаружен, система пытается отыскать одноименный файл с расширением PAS (то есть файл с исходным текстом модуля) и приступает к его компиляции. В этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если внесены какие-либо изменения в исходный текст модуля, то, независимо от того, есть ли уже соответствующий TPU файл или нет, система осуществляет его компиляцию перед компиляцией основной программы. Если внесены изменения в интерфейсную часть модуля, то будут перекомпилированы также и все другие модули, обращающиеся к нему.

В режиме BUILD существующие TPU файлы игнорируются, а система ищет и компилирует соответствующие PAS-файлы для каждого из объявленных модулей. После компиляции в этом режиме можно быть уверенным, что учтены все изменения, сделанные в любом из модулей.

Пример 1.

Дан текстовый файл, содержащий записи следующей структуры:

№ рейса	Пункт отправления	Пункт назначения	День недели	Время отправления Час Мин	Цена билета
5 символов	15 символов	15 символов	1..7	0..23 0..59	Real

1) Ввести заданный список из файла в массив записей.

2) Вывести список на экран.

3) Упорядочить список по возрастанию номеров рейсов одним из алгоритмов сортировки (предусмотреть возможность выбора алгоритма). Процедуры сортировки оформить в виде модуля.

4) Вывести упорядоченный список на экран.

Разработка модуля

Включим в разрабатываемый модуль с именем SORT две процедуры сортировки: обменной сортировки с именем Sort_Exchange и сортировки выбором - Sort_Choose. Все остальные пояснения см. в комментариях.

```
Unit SORT; {Заголовок модуля}
Interface { Секция связи}
Type
        T_Time = record {Описание типа для задания времени}
        Hour: 0..23;
        Min: 0..59;
end;
T_zap = record {Описание типа для одной записи списка}
         nom: string[5];
         potpr, pnazn: string[15];
         day: 1..7;
         time: T_Time;
         price: real;
end;
Var
     Spis: array[1..1000]of T_zap; {Описание списка - массива из записей}
     Kol_Zap: word; { Число записей в файле и списке}
Procedure Sort_Exchange; { Описание заголовков процедур модуля}
Procedure Sort_Choose;
Implementation {Секция реализации}
Procedure Sort_Exchange; {Процедура обменной
сортировки}
Var
     i: word; Z: T_Zap;
     Key: boolean; {Ключ - признак обмена}
  Begin
     Repeat {Цикл проверок очередности}
     Key:=false;
     For i:=1 to Kol_Zap-1 do
     If Spis[i].nom > Spis[i+1].nom then
          Begin {Обмен записей}
             Z:=Spis[i];
             Spis[i]:=Spis[i+1];
             Spis[i+1]:=Z;
             Key:=True;
          End;
  Until Key=False; {Цикл заканчивается в том случае,
  если ни одна пара записей не переставлялась}
  End;
```

```
Procedure Sort_Choose; {Процедура сортировки выбором}
Var
    i, j, imin: word;
    nomr: string[5];
    Z: T_Zap;
  Begin
    For i:=1 to Kol_Zap-1 do
    Begin
      nomr:=Spis[i].nom;
      imin:=i;
      For j:=i+1 to Kol_Zap do
      If Spis[j].nom < nomr then
      begin
        nomr:=Spis[j].nom;
        imin:=j;
      end;
        {Обмен целыми записями}
        Z:=Spis[i];
        Spis[i]:=Spis[imin];
        Spis[imin]:=z;
    End;
  End;
End. {Конец модуля}
```

Сохраняем созданный исходный текст модуля в файле с именем Sort.pas. Затем компилируем его, в результате на диске создается файл Sort.tpu.

Разработка программы, использующей модуль Sort

Включим в программу две процедуры: процедуру ввода списка из текстового файла в массив записей с именем INPUT_SPIS и вывода списка на экран с именем OUT_SPIS.

```
Uses crt, sort; {Подключение стандартного и разработанного модулей }
Var
    nom_alg:byte; {переменная для выбора номера алгоритма}
Procedure INPUT_SPIS;
Var
    F: text;
    Name_file: string;
    i: word;
  Begin
    Writeln(' Введите имя файла со списком');
    Readln(Name_file);
    Kol_Zap:=0; i:=0;
    Assign(f,Name_file);
    Reset(f);
    While not eof(f) do
```

```
          Begin
               i:=i+1;
               With Spis[i] do
               { Чтение очередной записи из текстового
               файла по полям}
               Readln(f,nom, potpr, pnazn,
               day,time.hour,time.min,price);
          End;
      Kol_zap:=i; {Считанное число записей}
      Close(f);
    End;
Procedure OUT_SPIS;
Var i:word;
  Begin
     Writeln(' № рейса Пункт отправл. Пункт назн. День
     Время Цена');
     For i:=1 to Kol_Zap do
     With Spis[i] do
     Writeln(' ',nom,' ', potpr,' ', pnazn,' ', day,'
     ',time.hour,' ',time.min,' ',price);
    End;
BEGIN {main program}
INPUT_SPIS;
Writeln(' Исходный список');
OUT_SPIS;
Writeln(' Введите номер алгоритма сортировки: 1 или
2');
Readln(nom_alg);
If nom_alg = 1 then SORT_EXCHANGE
Else SORT_CHOOSE;
Writeln(' Упорядоченный список');
OUT_SPIS;
Readkey;_
```

Пример 2. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида *P/Q (P* — целое, *Q* — натуральное):

1. сложение;
2. вычитание;
3. умножение;
4. деление;
5. сокращение дроби;
6. возведение дроби в степень *N (N* — натуральное);
7. функции, реализующие операции отношения (равно, неравно, больше или равно, меньше или равно, больше, меньше).

Дробь представить в виде:

Type Frac=Record

P: Integer; Q: 1..32767 **End;**

```
Unit Droby;
Interface
Type
Natur=1..High(Longlnt);
Frac=Record
P: Longlnt;
Q: Natur
End;
Procedure Sokr(Var A: Frac);
Procedure Summa(A,B: Frac; Var C: Frac);
Procedure Raznost(A,B: Frac; Var C: Frac);
Procedure Proizvedenie(A,B: Frac; Var C: Frac);
Procedure Chastnoe(A,B: Frac; Var C: Frac);
Procedure Stepen(A: Frac; N: Natur; Var C: Frac};
Function Menshe(A,B: Frac): Boolean;
Function Bolshe(A,B: Frac): Boolean;
Function Ravno(A,B: Frac): Boolean;
Function MensheRavno(A,B: Frac): Boolean;
Function BolsheRavno(A,B: Frac): Boolean;
Function NeRavno(A,B: Frac): Boolean;
{Раздел реализации модуля)
Implementation
{Наибольший общий делитель двух чисел -вспомогательная
функция, ранее не объявленная)
Function NodEvklid(A,B: Natur): Natur;
Begin
While A<>B Do If A>B Then
If A Mod B<>0 Then A:=A Mod B
```

```
Else A:=B Else If B Mod AOO Then B: =B Mod A
Else B:=A;
NodEvklid:=A
End;
{Сокращение дроби}
Procedure Sokr;
Var M,N: Natur;
Begin
If A. P<>O
Then
Begin
If A.P<0
Then M:=Abs(A.P)
Else M:=A.P; {Совмещение типов, т.к. A.P -Longlnt}
N:=NodEvklid(M,A.Q);
A.P:=A.P Div N;
A.Q:=A.Q Div N
End
End;
Procedure Summa; {Сумма дробей}
Begin
{Знаменатель дроби}
C.Q:=(A.Q*B.Q) Div NodEvklid(A.Q,B.Q); {Числитель дроби}
C.P:=A.P*C.Q Div A.Q+B.P*C.Q Div B.Q; Sokr(C)
End;
```

```
Procedure Raznost; {Разность дробей}
Begin
{Знаменатель дроби}
C.Q:=(A.Q*B.Q) Div NodEvklid(A.Q,B.Q); {Числитель дроби}
C.P:=A.P*C.Q Div A.Q-B.P*C.Q Div B.Q; Sokr(C)
End;
Procedure Proizvedenie; {Умножение дробей}
Begin
{Знаменатель дроби} C.Q:=A.Q*B.Q;
{Числитель дроби}
C.P:=A.P*B.P; Sokr(C)
End;
Procedure Chastnoe; {Деление дробей}
Begin
{Знаменатель дроби}
C.Q:=A.Q*B.P; {Числитель дроби}
C.P:=A.P*B.Q;
Sokr(C)
End;
Procedure Stepen; {Возведение дроби в степень}
Var I: Natur;
Begin
C.Q:=1;
C.P:=1;
Sokr(A);
For I:=l To N Do
Proizvedenie(A,C,C)
End;
```

```
Function Menshe; {отношение '<' между дробями}
Begin
Menshe:=A.P*B.Q<A.Q*B.P End;
Function Bolshe; {отношение '>' между дробями}
Begin
Bolshe:=A.P*B.Q>A.Q*B.P End;
Function Ravno; {отношение '=' между дробями}
Begin
Ravno:=A.P*B.Q=A.Q*B.P End;
Function BolsheRavno; {отношение '>=' между дробями}
Begin
BolsheRavno:=Bolshe(A,B) Or Ravno(A,B)
End;
Function MensheRavno; {отношение '<=' между дробями}
Begin
MensheRavno:=Menshe(A,B) Or Ravno(A,B)
End;
Function NeRavno; {отношение '<>' между дробями}
Begin
NeRavno:=Not Ravno(A,B)
End;
{Раздел инициализации модуля}
Begin
End.
```

Сохраним текст разработанной программы в файле DROBY.PAS и откомпилируем наш модуль. Для этого можно воспользоваться внешним компилятором, поставляемым вместе с Турбо Паскалем. Команда будет выглядеть так: TPC DROBY . PAS. Если в тексте нет синтаксических ошибок, получим файл DROBY . PAS, иначе будет выведено соответствующее сообщение с указанием строки, содержащей ошибку. Другой вариант компиляции: в меню системы программирования Турбо Паскаль выбрать **Compile/Destination Disk,** затем **— Compile/Build.**

Теперь можно подключить модуль к программе, где планируется его использование.

Решим задачу:

Дан массив *A,* элементы которого — обыкновенные дроби. Найти сумму всех элементов и их среднее арифметическое; результаты представить в виде несократимых дробей.

```
Program Sum;
Uses Droby;
Var A: Array[1..100] Of Frac;
I,N: Integer;
S: Frac;
Begin
Write ('Введите количество элементов массива:');
ReadLn(N);
S.P:=0; S.Q:=1; {Первоначально сумма равна нулю}
For I:=1 To N Do {Вводим и суммируем дроби}
Begin
Write ('Введите числитель',I, ' -и дроби:');
ReadLn(A[I] .P) ;
Write('Введите знаменатель ', I,'-й дроби:');
ReadLn(A[I].Q);
Summa(A[I],S,S);
End;
WriteLn('Ответ: ' ,S.P, '/' ,S.Q)
End.
```

☞**Задание 1.** Изучить теоретическую часть лабораторной работы.

☞**Задание 2.** Выполнить отладку программ и получить результаты (**Пример 1**, **Пример 2**). **Пример 2** предъявить преподавателю, быть готовым объяснить любую строку модуля и программы.

☞**Задание 3.** Выполнить индивидуальное задание по вариантам, опираясь на данные из теоретической части и рассмотренные примеры:

I. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над одномерными массивами:

1) заполнение массива;
2) вывод на экран массива;
3) добавление элемента в k-ю позицию;
4) удаление k-го элемента;
5) замена элементов массива;
6) суммирование элементов массива.

Используя этот модуль, решить следующую задачу:

1. В одномерном массиве подсчитать сумму всех отрицательных элементов.
2. В одномерном массиве заменить все положительные элементы на введенное число.

II. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над комплексными числами:

1) сложение;
2) вычитание;
3) умножение;
4) деление;

5) вычисление модуля комплексного числа; 6) возведение комплексного числа в степень *n* (*n* — натуральное).

Комплексное число представить следующим типом:

```
Type Complex = Record
   R, M : Real; {действительная и мнимая часть числа}
          End;
```

Комплексное число состоит из двух вещественных чисел, действительной и мнимой частей соответственно. Пусть имеются два комплексных числа x и y: x=a+bi; y=c+di. Здесь i квадратный корень из минус единицы (мнимая единица), причем i^2= - 1. Если комплексное число z=e+fi, и z есть результат операции над x и y, то

z=x+y; e=a+c, f=b+d; (сложение)

z=x-y; e=a-c, f=b-d; (вычитание)

z=x*y; e=a*c-b*d, f=a*d+b*c; (умножение)

z=x/y; e=(a*c+b*d)/(c2+d2), f=(c*b-a*d)/(c2+d2); (деление)

Легко видеть, что при сложении и вычитании складываются и вычитаются соответственно действительные и мнимые части комплексных чисел, умножение и деление более сложны.

Используя этот модуль, решить задачи:

1. Дан массив *A* — массив комплексных чисел. Получить массив *C*, элементами которого будут модули сумм рядом стоящих комплексных чисел.

2. Дан массив *A[M]* — массив комплексных чисел. Получить матрицу *B[N, M]*, каждая строка которой получается возведением в степень, равную номеру этой строки, соответствующих элементов данного массива *A*.

III. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над векторами на плоскости: 1) сложение; 2) вычитание; 3) скалярное умножение векторов; 4) умножение вектора на число; 5) длина вектора.

Вектор представить следующим типом:

```
Type Vector = Record
                X, Y : Real
End;
```

Используя этот модуль, решить задачи:

1. Дан массив *A* — массив векторов. Отсортировать его в порядке убывания длин векторов.

2. С помощью датчика случайных чисел сгенерировать 2*N* целых чисел. *N* пар этих чисел задают *N* точек координатной плоскости. Вывести номера тройки точек, которые являются координатами вершин треугольника с наибольшим углом.

Контрольные вопросы:

1. Каковы характерные особенности организации модуля в языке Паскаль?
2. Какую структуру имеет модуль в языке Паскаль?
3. В каких случаях можно отсутствовать раздел IMPLEMENTATION?
4. Какое расширение имеет имя файла с исходным текстом модуля?
5. Охарактеризуйте три режима компиляции модулей и основной программы.
6. Для чего предназначена интерфейсная часть модуля?

Создание библиотеки подпрограмм

Использование указателей для организации связанных списков

Лабораторная работа № 25

*Тема: «*Изучение интегрированной среды разработчика*»*

Цель: *изучить интегрированную среду разработчика Visual Studio.NET, приобрести навыки работы с интегрированной средой разработчика в процессе создания простого проекта.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы.
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Интегрированная среда разработки Visual Studio.Net

Интегрированная среда разработки (integrateddevelopmentenvironment,IDE) Visual Studio.NET, едина для всех языков программирования .NETот Microsoft. Таким образом, какой бы тип проекта вы ни создавали (ATL, MFC,C#, Visual Basic.NET, FoxPro, стандартный C++ и т. п.), вы все равно будетеработать в одной и той же среде.

При запуске интегрированной среды разработки Visual Studio.NET на экране появляется ее окно, в котором предлагается выбрать язык программирования. После этого на экране появляется главное окно, основными элементами которого являются:

строка заголовка содержит название программы и имя открытого файла, а также кнопки свертывания, восстановления и закрытия;

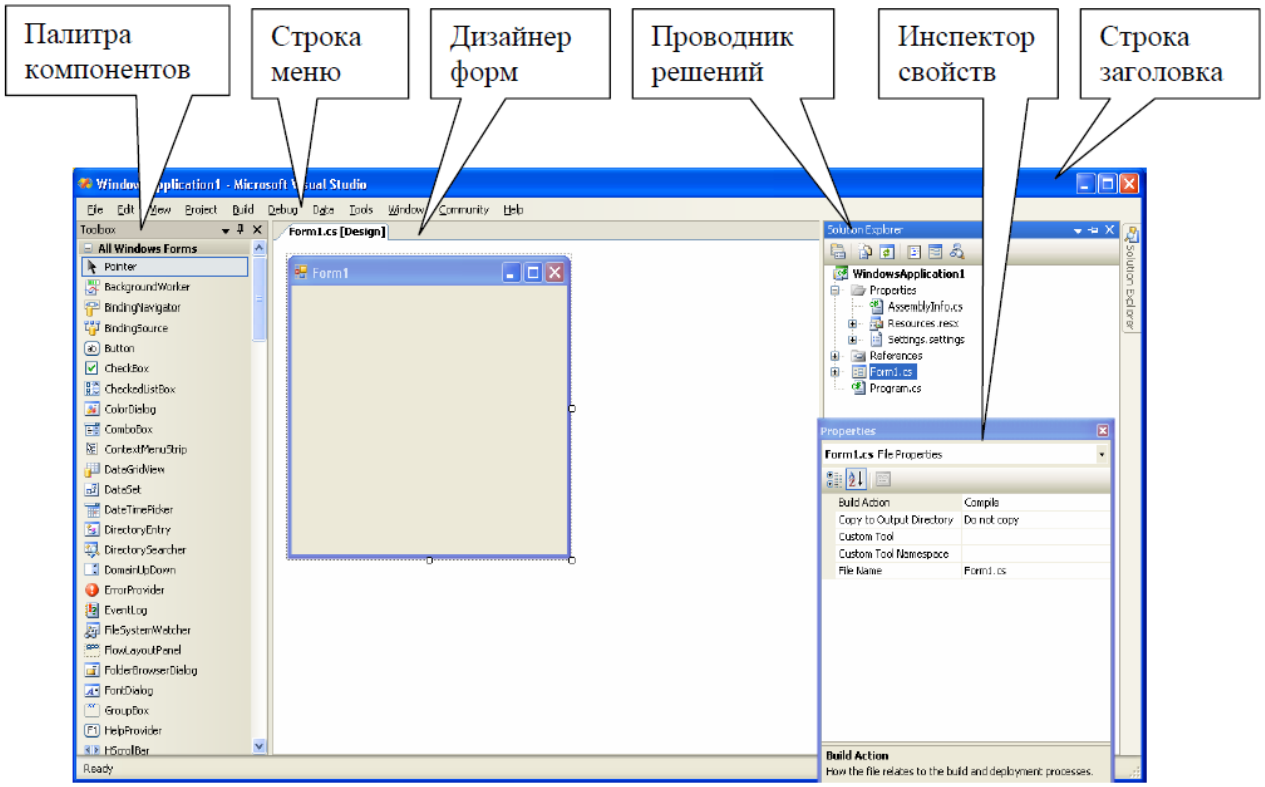
строка меню, которая представляет доступ ко всем функциям и командам программы;

панель элементов (палитра компонентов)она состоит из нескольких закладок, на которых располагаются визуальные компоненты, используемые при создании программ;

окно свойств в этом окне производится настройка основных свойств визуальных компонентов;

конструктор (дизайнер) форм содержит название и кнопки управления окном;

обозреватель решений обеспечивает выбор решения, проекта, исходного файла, ссылок на внешние сборки и прочих ресурсов, которые и образуют приложение.



Меню и команды интегрированной среды Visual Studio 2010

Чтобы выполнить команду в среде **Visual Studio**, можно воспользоваться тремя основными способами:

– с помощью главного меню,

– с помощью панелей инструментов (их видимость настраивается в меню **Вид /Панели инструментов (View \ Toolbars)**),

– с помощью контекстного меню, которое, как обычно, активизируется при нажатии правой кнопки мыши.

Меню Visual Studio 2010

Панели инструментов и палитра компонентов. Окно свойств

Панели инструментов скрываются или отображаются с помощью меню **Вид/Панели инструментов (View\Toolbars)**. Настройка панелей инструментов осуществляется с помощью команды **Вид/Панели инструментов** /Настройка (**View\Toolbars\Customize)**.

Меню		Описание
Файл		Позволяет создавать новые проекты, открывать существующие проекты или файлы с текстами программ, добавлять новые файлы в проект, сохранять изменения в файлах проекта
Правка		Позволяет работать с буфером обмена (команды **Копировать**, **Вырезать**, **Вставить**) и используется для правки текста программ при помощи команд: **Отменить**, **Вернуть**, **Поиск и замена** и др.
Вид		Предоставляет доступ к **Обозревателю решений**, **Списку ошибок** и другим окнам IDE, а также позволяет включить дополнительные панели инструментов
Проект	**(доступно в режиме открытого решения)**	Позволяет добавлять новые элементы в проект, а также изменять свойства текущего проекта
Построение		Позволяет выполнить сборку или повторную сборку всего решения или отдельного проекта в решении
Отладка		Предоставляет доступ к различным командам запуска текущего проекта внутри отладочной сессии и командам управления отладочными точками прерывания
Окно		Позволяет управлять открытыми в интегрированной среде окнами. Здесь можно скрыть окна, закрыть все открытые окна и превратить открытое окно (такое как **Обозреватель решений**) в документ с вкладками
Справка		Предоставляет доступ ко всем настройкам справочной системы VS, а также к библиотеке системы помощи

Окно **Панель элементов (Toolbox)** представляет собой палитру компонентов, распределённых по категориям:

Все формы Windows Forms (All Windows Forms)– общий список компонентов,

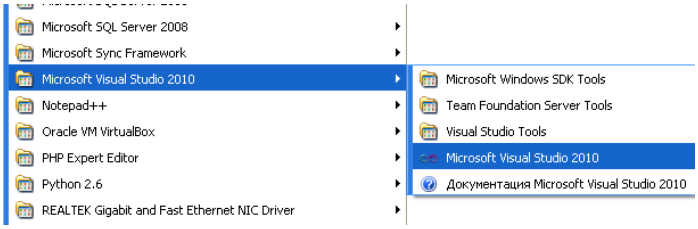
*CommonControl***s**– элементы управления общего назначения и т.д. Для добавления на форму элемента управления существует несколько путей:

− щёлкнуть мышью на нужном элементе управления в окне **Панель элементов (Toolbox)**, затем растянуть наформе рамку, которая и определит его размеры;

– дважды щёлкнуть на элементе управления в окне **Панель элементов (Toolbox)**, и он будет добавлен на форму, причем размеры и положение **Visual Studio**определит автоматически (по умолчанию);

– щёлкнуть мышью на нужном элементе управления, затем один раз щёлкнуть на форме.

Элемент будет добавлен так, чтобы его верхний левый угол совпадал с координатами курсора мыши, а размер будет задан по умолчанию. Если при этом всё время держать нажатой клавишу **Ctrl**, будут добавлено несколько экземпляров этого элемента управления (при каждом щелчке мыши).



Компоненты, не используемые для непосредственного создания пользовательского управления (невизуальные компоненты), добавляются аналогично, но вне зависимости от указанного места всегда располагаются в специальной области под формой (панели компонентов). При этом их местоположение не имеет значения.

Окно свойств (Properties) служит для установки свойств компонента. На вкладке С**войства (Properties)** редактируются свойства компонентов, на вкладке **События (Event)** задаются обработчики событий. В окне **Окно свойств (Properties)** отображены не все свойства компонентов, а лишь те, которые могут быть установлены на этапе проектирования. Все изменения в окне **Окно свойств (Properties)** вызывают изменения исходного

кода программы (для формы **Form1** – файла **Form1.Designer.cs**).

Работа с формами

Проектирование форм – ядро визуальной разработки в среде **VisualStudio.** Каждый помещаемый на форму элемент управления или любое задаваемое свойство вносит изменения в исходный код файла, связанного с формой.

Можно начать новый проект, создав пустую форму, или начать с существующей формы или добавить в проект новые формы. Проект может иметь любое число форм, кроме того, возможно динамическое создание любого числа форм во время выполнения программы.

При работе с формой на этапе проектирования можно изменять её свойства, свойства одного или нескольких элементов управления одновременно. Чтобы выбрать форму или элемент управления, можно просто щёлкнуть по нему мышью. Возможен выбор нескольких элементов управления – с помощью растягивающейся рамки либо с помощью комбинации

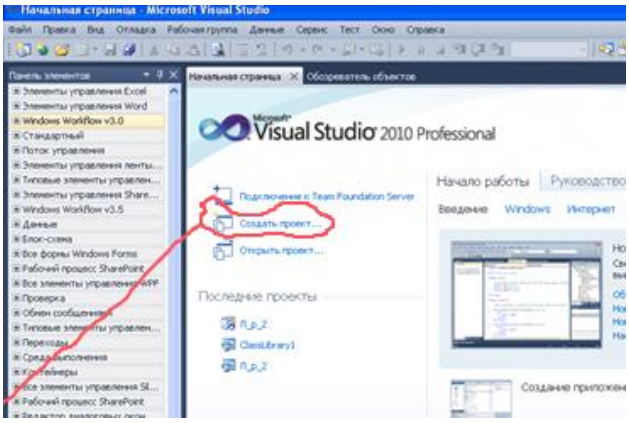
клавиш **Ctrl**+щелчок левой кнопкой мыши. Для управления взаимным расположением элементов управления используется выпадающее меню **Формат (Format)**.

В интегрированной среде разработки Visual Studio.NET проекты логически организуются в ***решения*** (solutions). Каждое решение состоит из одного или нескольких проектов. В свою очередь, каждый проект может состоять из любого количества исходных файлов, ссылок на внешние сборки и прочих ресурсов, которые и образуют приложение. Вы сможете открыть любой изданных ресурсов с помощью окна Solution Explorer - Проводника решений

Задание 1. Создать программу, в которой щелчок на кнопке приводит к изменению цвета формы и выводит на экран окно с сообщением "Привет, Мир"

Запуск и настройка Visual Studio .NET

1. Щелкните на кнопке *Пуск.*
2. Выберите пункт *Программы.*
3. Выберите пункт *Microsoft Visual Studio 2010.*
4. Выберите пункт *Microsoft Visual Studio 2010.*



Первое, что вы увидите на экране, запустив Visual Studio .NET, — начальную страницу *{Начальная страница).* По умолчанию будет открыта вкладка *Проекты.* Обычно на ней приводится список недавно открывавшихся проектов. Для открытия существующего проекта щелкнуть по кнопке *Открыть проект,* а для начала создания нового проекта — щелкнуть по кнопке *Создать проект.*

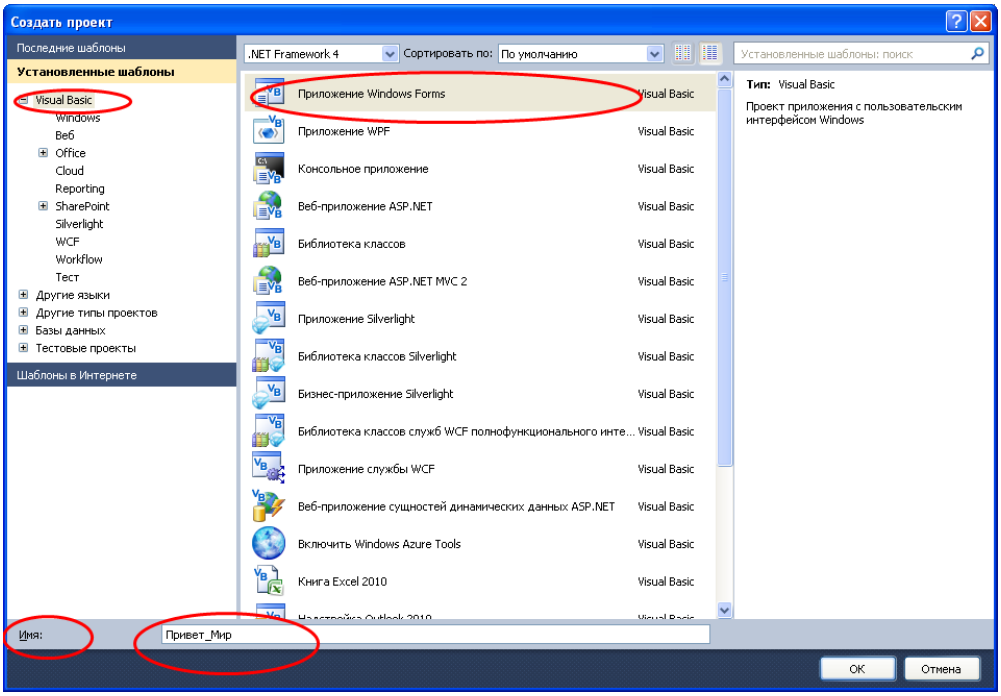
Создание первого проекта

Щелкните на кнопке *Создать проект,* чтобы создать новый проект. Откроется диалоговое окно *Создать проект.*

2. В окне *Создать проект* вы увидите, что в левой части, в панели *Типы проектов:* перечислено множество типов проектов, которые можно создавать в Visual Studio .NET. С некоторыми из этих типов вы вскоре познакомитесь. Проект «Привет, Мир» будет приложением Windows, поэтому в правой части окна (панель *Шаблоны:)* необходимо выбрать шаблон *Приложение Windows.* В окне *Создать проект* нужно выбрать не только тип проекта и шаблон, но и имя проекта и папку, в которой будут размещаться его файлы.

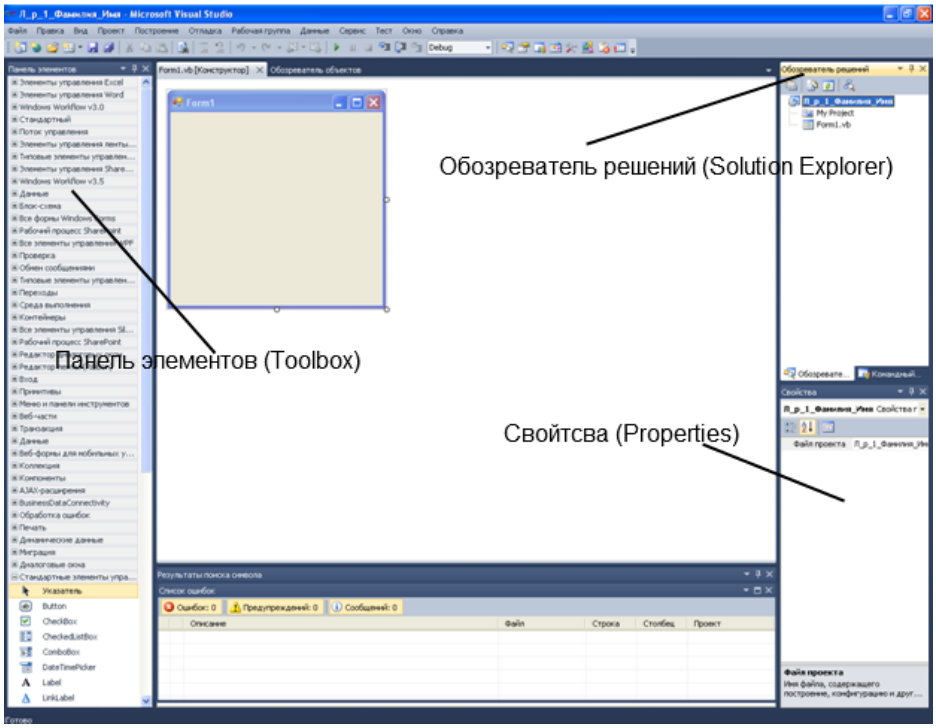
3. Выберите тип проекта *Проекты- Visual Basic* в панели *Типы проектов:*.

4. Выберите шаблон *Приложение Windows* в панели *Шаблоны:*.



5. Введите *Привет_Мир* в качестве имени проекта в текстовом поле *Имя* (в имени проекта лучше не использовать пробелы).

6. Нажмите кнопку *ОК*. Откроется новый созданный проект, и на экране будут отображены форма Forml.vb (в левой части) и Обозреватель решений (Solution Explorer) (в правой части) и др.



Конструирование графического интерфейса проекта

Окно *Конструктор форм (Form Design Window).*

Форма — это элемент графического интерфейса, с помощью которого осуществляется взаимодействие пользователя с программой. Форма представляет собой поле с размещаемыми на нем объектами разных типов — кнопками, текстовыми полями. В основной рабочей области в левой части окна будет отображена пустая форма, на которой нет кнопок и других объектов. Эта область называется окном *Конструктор форм.* Ярлычок этого окна содержит надпись *Forml.vb [Конструктор].*

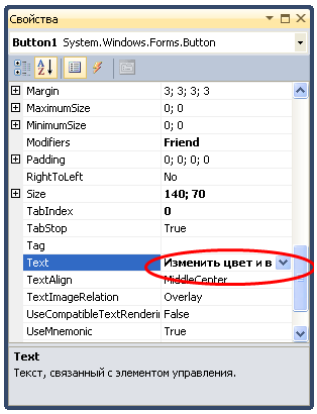
Окно *Область элементов (Toolbox).* В этом окне содержатся все объекты, которые можно поместить на форму, — кнопки, переключатели, текстовые поля, выпадающие списки и т. д.

1. Откройте меню *Вид (View)* в строке меню.
2. Выберите пункт *Панель элементов (Toolbox)* — откроется окно *Панель элементов.*
3. Двойным щелчком по пункту *Button* поместите на форму кнопку.

Перемещение кнопки и изменение ее размеров.

1. Подведите курсор мыши к кнопке, нажмите левую кнопку мыши и, удерживая ее нажатой, перетащите кнопку в центр формы, после чего отпустите кнопку мыши.
2. Чтобы изменить размер кнопки, подведите курсор мыши к одному из белых квадратиков вокруг кнопки (это метки изменения размера).
3. Нажмите левую кнопку мыши, и, удерживая ее нажатой, перемещайте метку, чтобы растянуть или сжать кнопку до требуемого раз мера.
4. В окне Свойства (Propertys) измените свойсво *Text* кнопки *Button1* на *Изменить цвет и вывести сообщение*

Создание программного кода проекта



1. Сделайте двойной щелчок по кнопке Buttonl. Откроется окно *Редактор кода.*
2. Щелкните по пустой строке над строкой End Sub.
3. Введите в пустую строку с клавиатуры строку кода, которая вызывает процедуру обработки события (точно так, как показано ниже):

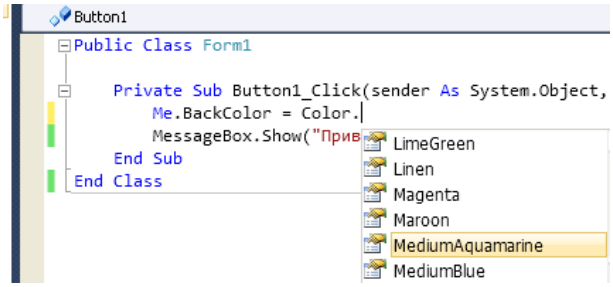
```
Form1.vb*
Button1                                   Click
Public Class Form1

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Me.BackColor = Color.MediumAquamarine
        MessageBox.Show("Привет, Мир.")
    End Sub
End Class
```

Совет. После того, как вы наберете имя объекта Color и точку, Visual Studio покажет список всех свойств объекта текстовое поле. Вы можете выбрать свойство из этого списка, дважды щелкнув на нем мышью, или набрать его самостоятельно.

Этот код будет выполняться, когда пользователь будет щелкать по кнопке на форме и генерировать событие.

Построение решения

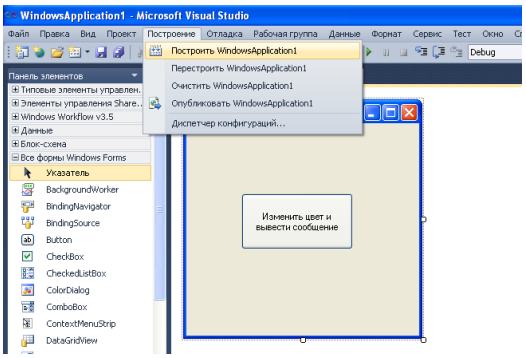
Теперь нужно построить решение. При построении решения код, написанный вами и Visual Studio .NET, компилируется в понятные компьютеру инструкции.



1. Откройте меню *Построение (Build)* в строке меню.
2. Выберите пункт *Построить решение (Build Solution)* — начнется построение решения.

Процесс построения решения будет отображаться в окне *Вывод*. Если в программном коде не было сделано ошибок, после завершения построения в окне *Вывод* будет выведено сообщение о том, что построение выполнено успешно.

В программном коде могли быть сделаны ошибки, пусть, например, вместо MessageBox вы набрали mMessageBox. Если Visual Basic может определить, где произошла ошибка, он подчеркивает это место в программном коде синей волнистой линией. Кроме того, открывается окно *Список задач (Task List),* содержащее список ошибок, которые нужно исправить. Двойной щелчок по сообщению об ошибке перемещает курсор в коде к этой ошибке.



В окне *Вывод (Output)* появится сообщение о том, что построение выполнить не удалось.

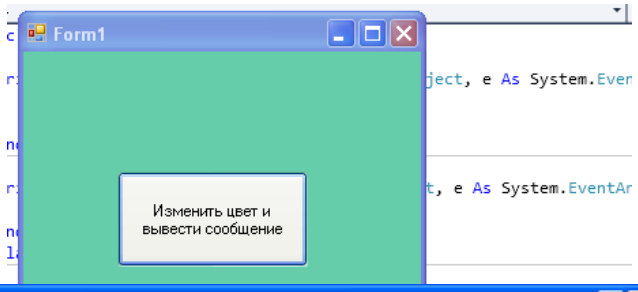
Если построение выполнить не удалось, вернитесь назад и введите строку кода точно в том виде, в каком она приведена в этом курсе. Затем еще раз попытайтесь построить решение, как показано ранее.

Запуск проекта

Теперь скомпилированную программу можно запустить.

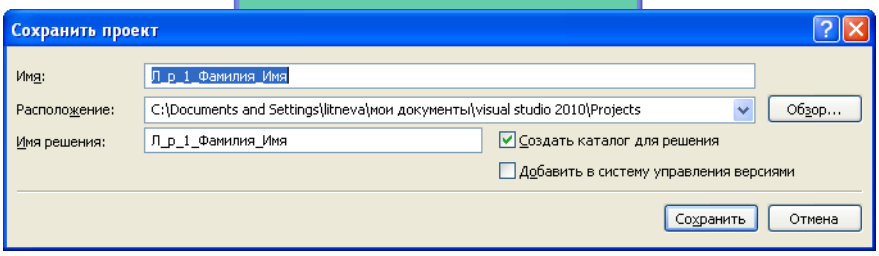
1. Откройте меню *Отладка (Debug).*

2. Выберите в этом меню пункт *Начать (Start).*(На экране появится форма проекта с кнопкой, которую вы на нее поместили.)



3. Щелкните на кнопке, изменится цвет формы и появится окно сообщения с текстом «Привет, Мир».

4. Нажмите кнопку *OK*, чтобы закрыть окно сообщения. (Окно сообщения закроется, но основное окно проекта останется открытым. Можете еще раз нажать кнопку, и окно сообщения появится снова.)



5. Нажмите кнопку со значком X | в верхнем правом углу формы, чтобы закрыть программу.

После завершения выполнения программы в окне *Вывод (Debug)* отображается отладочная информация. Чтобы посмотреть выведенные сообщения полностью, вам, вероятно, придется воспользоваться линейкой прокрутки.

Сохранение проекта

Сохранение выполняется так же, как и в других программах в операционной системе Windows.

1. Откройте меню *Файл (File).*
2. В меню *Файл (File)* выберите команду *Сохранить все (Save AH).*

Вы можете принять все значения по умолчанию для сохраняемого проекта, нажав кнопку *Сохранить(Save)*, (не забудьте только каталог в котором вы сохраняете проект, его имя указывается в поле *Расположение (Location:)*). Имеется также возможность определить свои значения. Для этого в поле *Имя (Name:)* введите имя сохраняемого проекта, в поле *Расположение (Location:)* задайте каталог, в котором должен быть сохранен проект или выберите этот каталог с помощью кнопки *Обзор (Browse...)*, а также задайте в поле *Имя решения (SolutionName:)* имя решения, содержащего ваш проект. Если вы хотите сохранить ваше решение в отдельном каталоге, оставьте галочку в поле *Создать каталог для решения (Create directory for solution)*, если же этого не требуется, то снимите галочку и нажмите кнопку *Сохранить (Save)* для сохранения.

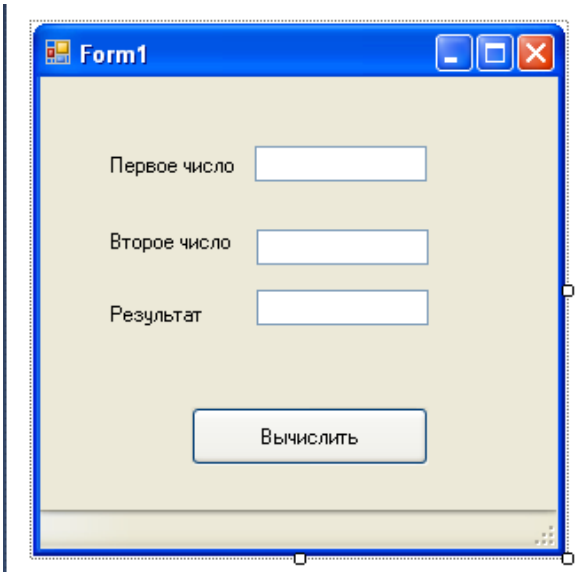
Файлы и **папки** решения. Когда вы создаете приложение на Visual Basic, создается целый набор файлов. Вы должны знать, какие это файлы и для чего они предназначены.

Перейдите в папку , в которой вы создали проект «Привет, Мир». Откройте эту папку, и вы увидите, что для проекта была создана отдельная папка, названная Привет_Мир. Зайдите в папку Привет_Мир, чтобы посмотреть файлы, созданные для вашего проекта Visual Studio .NET. В папке будут следующие файлы:

Привет_Мир.sln — это файл решения, в котором хранится описание всех файлов и настроек этого решения. Вообще-то в решение можно включить несколько проектов, но в решении «Привет, Мир» проект только один — это файл ПриветМир.vbproj. К файлу Привет_Мир.sln система обращается, когда вы открываете решение. Файл Form1.vb содержит форму и связанный с ней код.

Откройте папку ..\bin\, сделав по ее значку двойной щелчок мышью. В папке ..\bin\ содержится исполняемый файл, полученный компиляцией программы на Visual Basic. Это файл Привет_Мир.exe. Его можно запустить на другом компьютере, даже если на нем не установлена система программирования Visual Studio .NET.

Задание 2. Разработать программу, которая обеспечит возможности получения данных от пользователя, обработки полученных данных и выдачи результата. Для этого в первую очередь создайте графический интерфейс программы, примерно так, как показано на рисунке.



1. Выберите в левой области Начальной страницы пункт *Создать проект(Create:)*.В появившемся окневыберите тип *Приложение WindowsForms(WindowsForms Application)*.В поле ввода *Имя: (Name:)* введите имя своего проекта. Нажмите OK

2. Для статических строк текста, например "Первое число", поместите на форму три элемента *Label* с панели элементов. Для отображения текстаизмените значение свойства "Text" соответствующих элементов *Label1,Label2, Label3* на *Первое число, Второе число, Результат*.

3. Для создания полей ввода/вывода добавьте на форму три элемента "TextBox" с панели элементов.

4. Для написания программы нужно, как и в предыдущем примере, сделать двойной щелчок на кнопке *Вычислить* и написать в появившемся обработчике следующий код

```
Dim x, y, z As Integer
x = TextBox1.Text
y = TextBox2.Text
z = x + y
TextBox3.Text = z
```

Пояснения к приведенному коду.

В первой строке с помощью оператора Dim определены три целочисленных переменных x, y и z.

Вторая строка обеспечивает считывание числа, записанного пользователем в первое окошко ввода, в переменную x. Заметьте, что все окошки ввода различаются именами, заданными по умолчанию: TextBox1, TextBox2, TextBox3. Программист может изменить эти имена, меняя значения свойства "Name" для соответствующего элемента. Разумеется, после этого в программном коде ссылаться на эти элементы вы должны используя данные вами имена. Свойство "Text" поля ввода содержит текст, вводимый пользователем, и отображаемый на экране.

В третьей строке содержимое второго поля ввода копируется в переменную y.

Следующая строка реализует непосредсвенно вычисления, в данном случае сложение значений x и y и присвоение полученного результата переменной z.

Чтобы полученный результат стал известен пользователю, в последней строке кода производится присвоение свосту "Text" третьего окошка ввода значения переменной z, что и приводит к отображению ее значения в этом окошке.

5. Проверьте работу программы, нажав клавишу F5, введя числовые данные и нажав кнопку "Вычислить". Поэкспериментируйте с программой. Попробуйте вместо суммы вычислять значения каких-либо других арифметических выражений, например, умножения.

Контрольные вопросы:

1. Перечислите и охарактеризуйте основные элементы, которые появляются при запуске интегрированной среды разработки Visual Studio.NET.
2. Перечислите и опишите основные пункты меню и команды интегрированной среды Visual Studio 2010
3. Основные приемы настройки интегрированной среды Visual Studio 2010 (см. лекцию).
4. С помощью какого окна выбираются элементы управления для размещения их на форме.
5. Способы переключения в редактор кода.
6. Назначение оператора Dim.
7. Выполнение построения решения.

8. Выполнение отладки приложения.
9. Назвать основные файлы проекта.

Лабораторная работа № 26

Тема: «Создание проекта с использованием компонентов для работы с текстом»

Цель: *создание Windows-приложения, обеспечивающего возможность решения уравнения и построения графика функции. Приобретение навыков использования графического элемента управления* ***PictureBox*** *при проектировании интерфейса Windows-приложения в среде Visual Studio.*

Ход работы:

1. Выполнить задания, следуя указаниям (задание 1-го уровня на «3», задание 2-го уровня на «4», задание 3-го уровня на «5»,).
2. Ответить на контрольные вопросы (в устной форме).
3. Предъявить преподавателю результаты работы программы и исходные коды.
4. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

☞*Задание на лабораторную работу.*

Задание 1-го уровня

1. Создать новый проект.
2. Составить эскиз интерактивной формы (Рис.1).
3. Задать значения свойств элементов управления, размещенных на интерактивной форме.
4. Составить программу для нахождения корней функции *f(x)* на интервале *[A, B]* с шагом *E*, предусмотрев ввод исходных данных через текстовые поля интерактивной формы. Функцию *f(x)* выбрать из Табл. 6 в соответствии со своим вариантом.
5. Осуществить сборку и компиляцию модулей проекта.
6. Решить уравнение *f(x) = 0*.

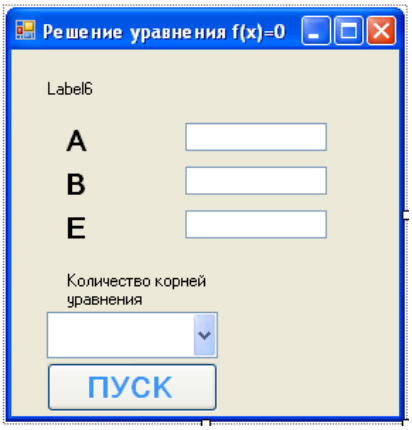
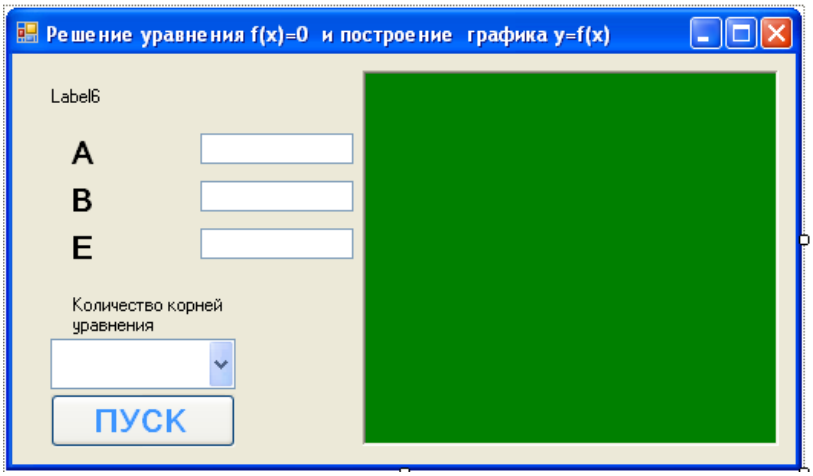


Рис. 1. Эскиз интерактивной формы

Задание 2-го уровня Реализовать построение графика и отображение графика функции в элементе управления *PictureBox* (Рис. 2).



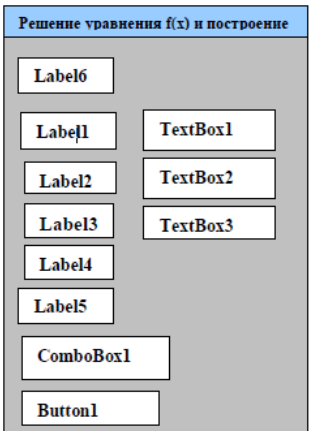


Рис. 2. Эскиз интерактивной формы с построением графика

Задание 3-го уровня. Реализовать возможность задавать пользователем функцию (полином до третьей степени), предусмотрев ввод параметров функции через текстовые поля интерактивной формы (Рис. 3).

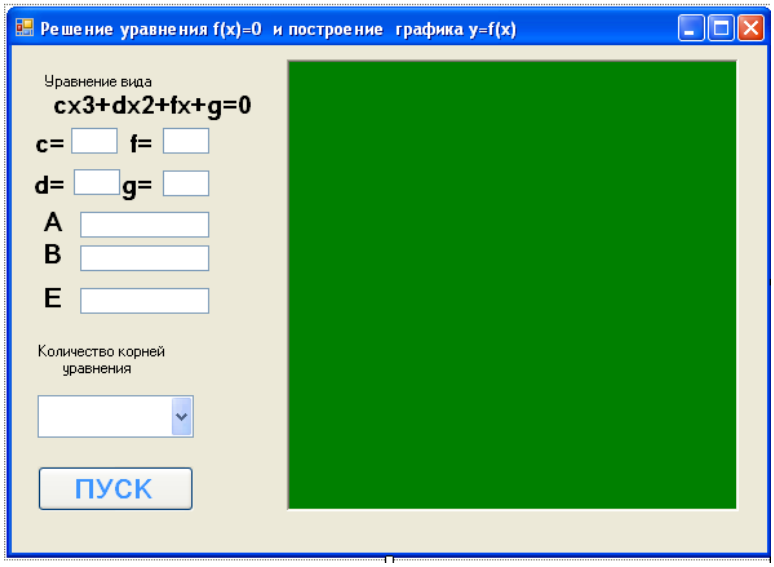


Рис. 3. Эскиз интерактивной формы с построением графика и пользовательским вводом функции

Порядок выполнения работы (1-й уровень)

1. Создать новый проект командой Создать проект (New Project) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).

2. Создать эскиз интерактивной формы.

2.1. Используя панель инструментов *ToolBox*, разместить на форме элементы управления (кнопку - *Button1*, надписи – *Label1* – *Label6*, текстовые поля - *TextBox1* *TextBox3*, поле со списком - *ComboBox1* и графическое поле - *PictureBox1*), как показано на Рис. 4. Элемент управления *ComboBox* - текстовое поле с предопределѐнным списком значений, из которого можно выбрать одно из имеющихся значений. В данной работе в *ComboBox* будут отображаться значения вычисленных корней уравнения.

Рис. 4. Размещение элементов управления на форме

3. После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель Свойства (*Properties)*, которая появляется после одинарного щелчка мышью на нужном элементе управления, расположенном на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

3.1. Установите значения свойств *Text* и *WindowState* объекта *Form1*, как показано на Рис. 5.

- **Form1.Text** = Решение уравнения f(x) = 0 и построение графика y = f(x)
  Пояснение: этот текст будет отображаться в заголовке формы.
- **Form1. WindowState** = Maximized
  Пояснение: для отображения графика функции будет использован максимальный размер окна.

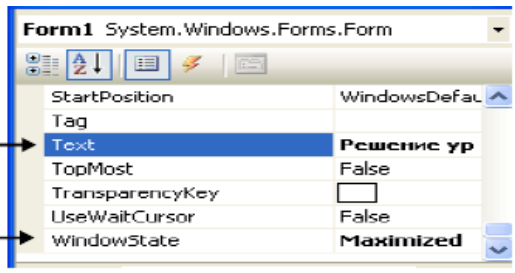


Рис. 5. Свойства Form1

3.2. Установите значения свойств элементов – надписей (*Label*), как указано в Табл. 1.

Табл. 1

Свойство	Значение
Label1.Text	A
Label2.Text	B
Label3.Text	E
Label1.Font	жирный, 16 пт.
Label2.Font	жирный, 16 пт.
Label3.Font	жирный, 16 пт.
Label4.Text	Количество корней
Label5.Text	Уравнения
Label6.Text	

3.3. Установите значения свойств элементов – текстовых полей (*TextBox*), как указано в Табл. 2.

Табл. 2

Свойство	Значение
TextBox1.Font	жирный, 16 пт.
TextBox2.Font	жирный, 16 пт.
TextBox3.Font	жирный, 16 пт.

3.4. Установите значения свойств элемента – кнопки (*Button*), как указано в Табл. 3.

Табл. 3

Свойство	Значение
Button1.BackColor	СИНИЙ
Button1.Font	жирный, 16 пт.
Button1.Text	ПУСК

Пояснение: для установки нужного цвета необходимо щелкнуть на кнопку в правом поле, перейти на вкладку Custom и выбрать из палитры цветов нужный цвет, например синий.

3.5. Установите значения свойств элемента – поля со списком (*ComboBox*), как указано в Табл. 4.

Табл. 4

Свойство	Значение
ComboBox1.DropDownStyle	DropDownList
ComboBox1.Font	жирный, 16 пт.

В результате изменения свойств вышеперечисленных объектов форма *Form1* примет вид, указанный на Рис. 1.

4. Написание программы (кода) включает в себя разработку кода для обработки событий формы и всех элементов управления. В качестве примера рассмотрим функцию $f(X) = X^2 - 2X - 10$.

4.1. Для объявления глобальных переменных выполните двойной щелчок левой кнопкой мыши на форме. В появившемся окне головного модуля *Form1.vb* выберете блок

Объявление *(Declarations)*, как показано на Рис. 6, и введите программный код, объявляющий переменные:

```
'Перечень глобальных переменных
Public Z As Boolean
Public A, B, Ep, MinF, MaxF As Double
'A - начальное значение аргумента X
'B - конечное значение аргумента X
'Ep - точность решения уравнения f(X)=0
'MinF - минимальное значение функции f(X)
'MaxF - максимальное значение функции f(X)
```

Рис. 6. Обработка события в блоке Общие (General) – Объявление (Declarations)

4.2. Для обработки события – загрузки формы (**Form1_Load**) выберете блок Load (как показано на Рис. 7) и

ведите программный код:

```
' Чистка текстовых полей исходных данных
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
Z = 0
```

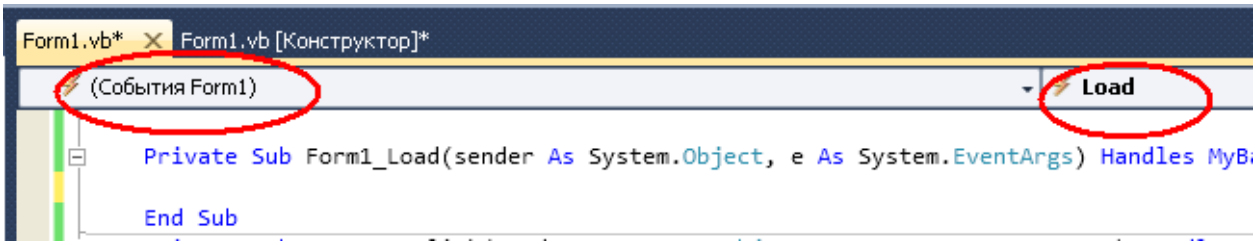


Рис. 7. Обработка события в блоке Form1 - Load

4.3. Написать программный код, обрабатывающий событие «НАЖАТИЕ КНОПКИ ПУСК» (Button1_Click).

```
Dim X, X2, XC, Y1, Y2 As Double
Dim KK As Integer
Z = 0
' Проверка корректности исходных данных : если данные не корректны,
' следует выход из подпрограммы
If TextBox1.Text = "" Or TextBox2.Text = "" Or TextBox3.Text = "" Then Exit Sub
A = Val(TextBox1.Text)
B = Val(TextBox2.Text)
Ep = Val(TextBox3.Text)
If A >= B Or Ep <= 0 Then Exit Sub
If B - A < Ep Then Exit Sub
' Если исходные данные корректны, то устанавливается значение Z=1,
' то есть разрешается перерисовка графика функции f(X)
Z = 1
' Устанавливаются начальные значения диапазона изменения f(X)
MinF = func(A)
MaxF = MinF
' Чистка открывающегося списка ComboBox1
ComboBox1.Items.Clear() ' Чистка счетчика корней уравнения f(X)=0
KK = 0
' В цикле для X от A до B с шагом Ep осуществляется анализ значений функции f(X)
For X = A To B Step Ep
    Y1 = func(X)
    ' Уточняются значения диапазона изменения f(X)
    If Y1 < MinF Then MinF = Y1
    If Y1 > MaxF Then MaxF = Y1
    X2 = X + Ep
    Y2 = func(X2)
    If Y1 * Y2 < 0 Then
        XC = (X + X2) / 2
        KK = KK + 1
        ComboBox1.Items.Add("X" & CStr(KK) & "= " & Format(XC, "0.########"))
    End If
    'Если выполнены условия существования корня уравнения,
    'то уточняется значение очередного корня уравнения f(X)=0
    ' и уточненное значение корня добавляется в список ComboBox1
Next X
'Значение счетчика (KK) корней отображается в поле элемента Label5
Label5.Text = "уравнения = " & CStr(KK)
If MaxF < 0 Then MaxF = 0
If MinF > 0 Then MinF = 0
```

7. Сборка и компиляция модулей проекта выполняется командой Построить решение (*Build WindowsApplication)* из меню Построение (*Build)*. Запустить приложение на выполнение можно командой Начать отладку (*Start Debugging)* из меню Отладка (*Debug)*. В появившейся форме (Рис. 1) ввести с клавиатуры значения исходных данных: - **A** - начало интервала табулирования функции; - **B** - конец интервала табулирования функции; - **E** - шаг вычисления корней уравнения. Для выполнения вычислений нажать кнопку ПУСК. Покажите преподавателю результаты работы.

Пояснения для выполнения задания 2-го уровня

1. Для отображения графика функции можно использовать элемент управления *PictureBox*, позволяющий размещать графические примитивы (точку, отрезок, простые геометрические фигуры). Разместите элемент управления *PictureBox* на форме, как показано на Рис. 8.

2. Установите значения свойств элемента – графического поля (*PictureBox*), как указано в Табл. 5.

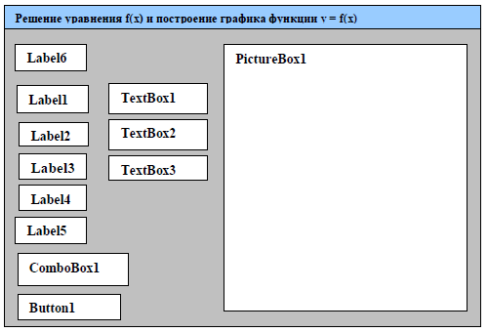


Рис. 8. Размещение элементов управления на форме

Табл. 5

Свойство	**Значение**
PictureBox1.BackColor	ЗЕЛЕНЫЙ
PictureBox1.BorderStyle	Fixed3D

3. Добавьте в обработчик события *Load* объекта *Form1* код, устанавливающий размеры *PictureBox*:

```
' Установка оптимальных размеров окна графического элемента PictureBox1
PictureBox1.Width = Me.Width * 0.75
PictureBox1.Height = Me.Height * 0.94
```

Добавьте в обработчик события *Click* объекта *Button1* код, выполняющий перерисовку содержимого в *PictureBox*:

```
' Элементу PictureBox1 дается разрешение на перерисовку графика функции f(X)
PictureBox1.Refresh()
```

5. Для обработки события **Paint**, возникающего при активизации графического элемента **PictureBox1**, необходимо выбрать блок **Paint** (Рис. 9).

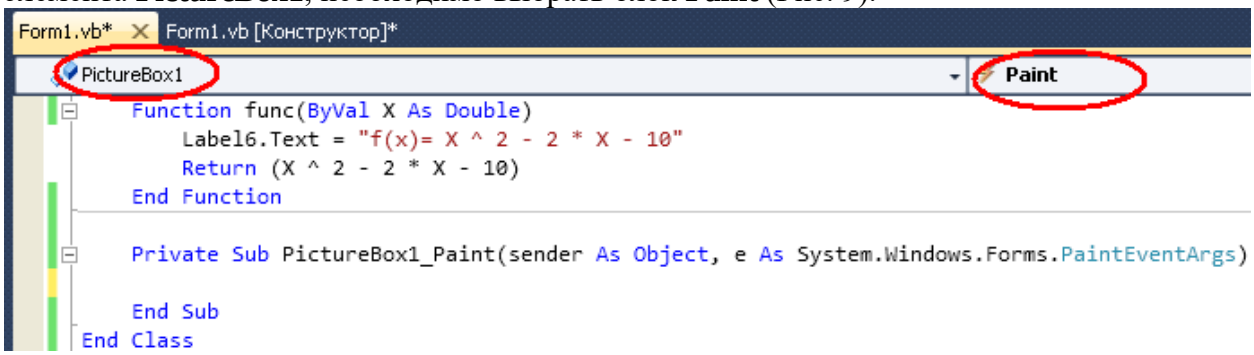


Рис. 9. Обработка события в блоке PictureBox1- Paint 35

И ввести код для подпрограммы **PictureBox1_Paint**:

```
Dim G As Graphics = e.Graphics
Dim X, DX, Y, DY As Double
Dim AG, BG, NG, N, I, DW, H0, PX, PY As Integer
' Если есть запрет на прорисовку графика функцииf(X), то выход из подпрогра
If Z = 0 Then Exit Sub
DY = MaxF - MinF
If DY <= 0 Then DY = 1
' N - количество точек для функции f(X)
N = (B - A) / Ep
AG = 1 : BG = PictureBox1.Width
If A > 0 Then AG = 10
If B < 0 Then BG = BG - 10
' NG - количество точек для графика функции f(X)
NG = BG - AG
If NG > N Then NG = N
If NG < N Then N = NG
' DX - шаг изменения аргумента X для функции f(X)
DX = (B - A) / N
' DW - шаг изменения аргумента X для графика функции f(X)
DW = (BG - AG) / NG ' H0 - высота графика функции f(X)
H0 = PictureBox1.Height
' Определяется перо для прорисовки графика функции f(X)
Dim MyPen As New Pen(Color.Red, 3)
' Определяется и заполняется массив точек графика функции f(X)
Dim Points(N) As Point
For I = 0 To NG
    X = A + DX * I
    Y = func(X)
    PX = AG + (BG - AG) * (X - A) / (B - A)
    PY = (H0 - 5) * (MaxF - Y) / DY
    Points(I) = New Point(PX, PY)
Next I
' Выполняется прорисовка графика функции f(X)
G.DrawLines(MyPen, Points)
'Определяется перо для прорисовки координатных осей графика функции f(X)
Dim MyPenXY As New Pen(Color.Blue, 3)
' Определяется и заполняется массив точек оси X
Dim PointsX(2) As Point
PX = 1
PY = (H0 - 5) * (MaxF - 0) / DY
PointsX(1) = New Point(PX, PY)
PX = PictureBox1.Width
PointsX(2) = New Point(PX, PY)
' Выполняется прорисовка оси X
G.DrawLine(MyPenXY, PointsX(1), PointsX(2))
' Определяется и заполняется массив точек оси Y
Dim PointsY(2) As Point
If A > 0 Then PX = 1 Else
If B < 0 Then PX = PictureBox1.Width - 5 Else PX = AG + (-A / DX) * DW
PY = 0
PointsY(1) = New Point(PX, PY)
PY = H0 - 5
PointsY(2) = New Point(PX, PY)
' Выполняется прорисовка оси Y
G.DrawLine(MyPenXY, PointsY(1), PointsY(2))
```

После сборки, компиляции и запуска приложения результат расчета в виде графика функции появится на форме в поле элемента **PictureBox1** (Рис. 10):

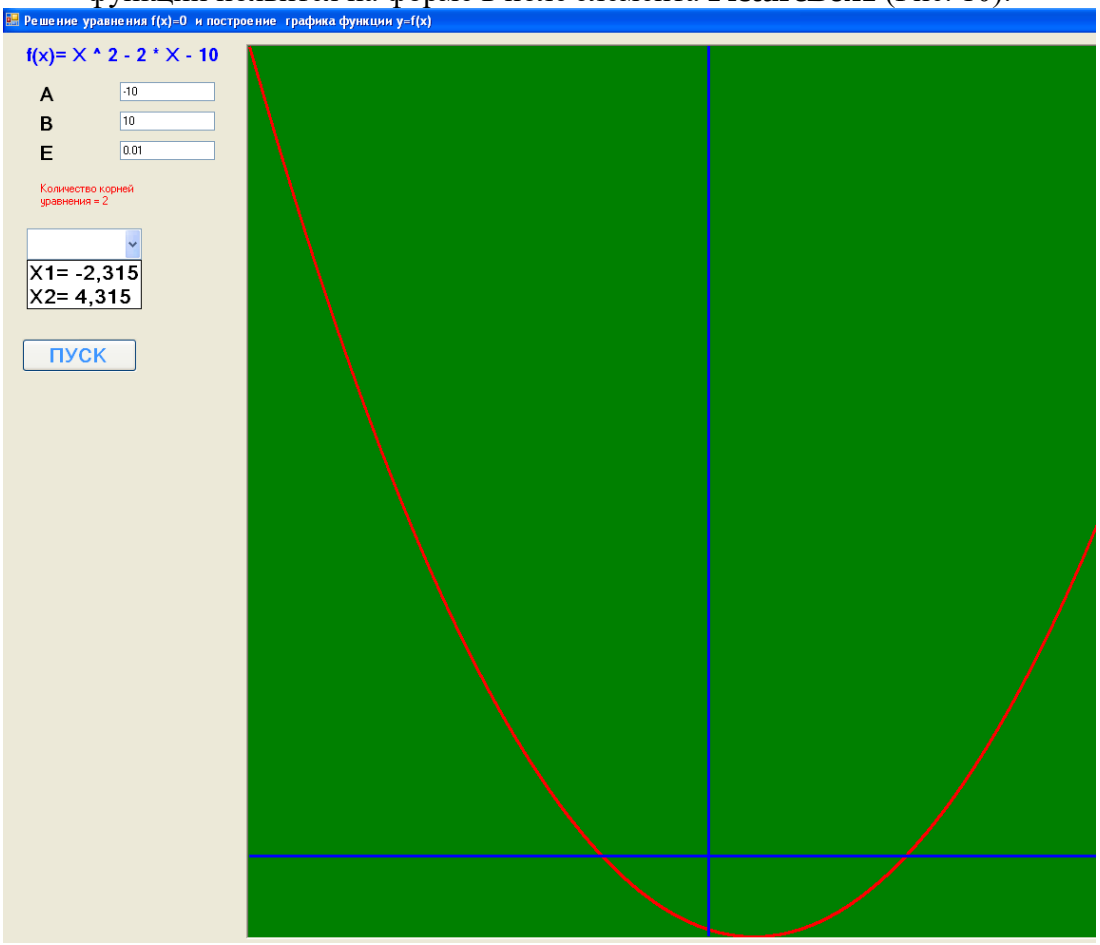


Рис. 10. Результат расчета функции $f(x) = x^2 - 2x - 10$

Выполните несколько вычислений (на разных отрезках [*A*, *B*] с разным шагом *E*), используя спроектированное приложение.

Табл. 6. Варианты заданий к лабораторной работе №6

№ вар.	Уравнение	Отрезок [a, b]	№ вар.	Уравнение	Отрезок [a,b]
1	$e^x + x - 2 = 0$	[0;1]	16	$22x - 2^x = 0$	[0;1]
2	$\frac{1}{2} - 2lnx = 0$	[1;2]	17	$e^x - 10x = 0$	[3;4]
3	$2^{-x} - \sqrt{x} = 0$	[0.1;1.1]	18	$e^{2x} = 2 - x^2$	[-1;0]
4	$Inx + \sqrt{x} = 0$	[0.1;1.1]	19	$2 - x = lgx$	[1;2]
5	$\frac{2}{x} - lnx = 0$	[2;3]	20	$\sqrt{x} + 1 = \frac{1}{x}$	[0.1;1.1]
6	$3^{-x} - \sqrt{2x} = 0$	[0;1]	21	$2x + ln(2x + 3) - 1 = 0$	[0;1]
7	$\sqrt{x+2} - 2^{-x} = 0$	[-2;-1]	22	$x2^x - 1 = 0$	[0;1]
8	$2Inx - 2x = 0$	[0.1;1.1]	23	$x^3 - x - 2 = 0$	[1;2]
9	$2^{-x} - x^2 = 0$	[0;1]	24	$x^4 - 2x - 4 = 0$	[1;2]
10	$x^2 - 2\sqrt{x} = 0$	[0,1;0,5]	25	$x + 1^x = 0$	[-1;0]
11	$x^3 - \sqrt{2x} = 0$	[0,1;2]	26	$x + lnx = 0$	[2;3]
12	$\sqrt{x} - x + 1 = 0$	[0;1.2]	27	$x^2 + ln(l + x) - 3 = 0$	[0;1]
13	$Ln\frac{x}{2} + 2\sqrt{x} = 0$	[0,1;2]	28	$x^3 + 4sinx = 0$	[-0,5;0,5]
14	$(x-1)^2 = 0,5e^x$	[0;1]	29	$lnx - sinx = 0$	[2;5]
15	$(x-1)^2 = e^{-x}$	[1;2]	30	$sinx + 2x = 1$	[0;1]

Контрольные вопросы:

1. Перечислите элементы управления для работы с текстом.
2. Поясните данный фрагмент кода:

```
If TextBox1.Text = "" Or TextBox2.Text = "" Or TextBox3.Text = "" Then Exit Sub
A = Val(TextBox1.Text)
B = Val(TextBox2.Text)
Ep = Val(TextBox3.Text)
```

3.
4. Перечислите свойства элементов управления, используемые для задания отображаемого текста и его цвета.
5. Укажите объявление глобальных переменных в коде программы.
6. Элемент управления для задания графических примитивов
7. Поясните данный фрагмент кода:

```
Dim MyPenXY As New Pen(Color.Blue, 3)
```

Лабораторная работа №27

Тема: «Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени»

Цель: *научиться извлекать даты, выбранные в элементах управления MonthCalendar и DateTimePicker, и отображать их в форме Windows Forms, получить навыки отображения диапазона дат и форматирования извлеченных дат различными способами.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Отображение дат. Использование элементов управления MonthCalendar и DateTimePicker

Использование элемента управления, отображающего календарь, значительно упрощает для пользователя выбор даты. Кроме того, такие элементы управления гарантируют, что дата будет отформатирована правильно. Календарь можно отобразить с помощью элемента управления MonthCalendar или DateTimePicker.

Элемент управления MonthCalendar позволяет отображать календарь для одного или нескольких месяцев. При этом пользователи могут выбирать отдельную дату или диапазон дат.

Элемент управления DateTimePicker имеет два состояния. По умолчанию элемент управления DateTimePicker выглядит как текстовое поле с раскрывающимся списком в виде стрелки. Когда пользователь нажимает на стрелку раскрывающегося списка, появляется календарь. При использовании этого элемента управления пользователь может выбрать только одну дату. Элемент управления DateTimePicker также позволяет отображать время вместо дат.

Процесс, используемый для извлечения даты из этих элементов управления, зависит от конкретного используемого элемента. Используйте свойство Start для элемента управления MonthCalendar и свойство Value для элемента управления DateTimePicker.

Элемент управления MonthCalendar позволяет отображать на экране одновременно до 12 месяцев. По умолчанию в этом элементе управления отображается только один месяц, однако имеется возможность указать количество месяцев, которые будут отображаться на экране, и их размещение в данном элементе управления. Чтобы обеспечить достаточное количество места в форме для новой размерности, при изменении диапазона календаря изменяются размеры элемента управления.

Константы для указания формата даты

Константа	**Описание**	**Пример**
DateFormat.GeneralDate	Отображает дату, время или оба значения. Если присутствует дата, она отображается в кратком формате. Если присутствует время, оно отображается в полном формате. Если присутствует и время, и дата, отображаются обе части.	22/11/1963 12:00:00 PM
DateFormat.LongDate	Отображает дату в полном формате, который определяется установленными на компьютере региональными параметрами.	Пятница, 22 ноября, 1963

DateFormat.ShortDate	Отображает дату в кратком формате, который определяется установленными на компьютере региональными параметрами.	11/22/1963
DateFormat.LongTime	Отображает время в полном формате, который определяется установленными на компьютере региональными параметрами.	12:00:00 PM
DateFormat.ShortTime	Отображает время в 24-часовом формате (чч:мм).	12:00

Свойства и функции системных часов

Чтобы получить от системных часов информацию о времени, можно использовать их различные свойства и функции. Информация о времени может потребоваться в программах при создании собственных календарей, часов или оповещений. В следующей таблице содержится перечень наиболее полезных функций системных часов. За дополнительной информацией обращайтесь к справочной системе Visual Studio.

Свойство или функция	**Описание**
TimeString	Возвращает от системных часов текущее время.
DateString	Возвращает от системных часов текущую дату.
Now	Возвращает закодированное значение, содержащее текущие дату и время. Наиболее полезно как аргумент для других функций системных часов.
Hour (time)	Возвращает количество часов для указанного времени (от 0 до 24).
Minute (time)	Возвращает количество минут для указанного времени (от 0 до 59).
Second (time)	Возвращает количество секунд для указанного времени (от 0 до 59).
Day (date)	Возвращает целое число, представляющее собой день месяца (от 1 до 31).
Month (date)	Возвращает целое число, представляющее собой месяц (от 1 до 12).
Year (date)	Возвращает год для указанной даты.
Weekday (date)	Возвращает целое число, представляющее собой день недели (по американской системе: 1 - это воскресенье, 2 - это понедельник, и т.д.).

☞*Задание на лабораторную работу.*

Порядок выполнения работы

1. Создать новый проект командой Создать проект (New Project) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент Приложение Windows Forms и нажмите кнопку ОК.
3. Добавьте в форму элемент Label, оставив имя по умолчанию Label1.
4. Удалите текст из свойства Text элемента управления Метка.
5. Добавьте в форму элемент управления MonthCalendar, оставив имя по умолчанию MonthCalendar1.
6. Дважды щелкните элемент управления MonthCalendar, чтобы открыть обработчик событий по умолчанию в редакторе кода.
7. В обработчике событий MonthCalendar1_DateChanged добавьте следующий код для добавления элементов в список.

```
Me.Label1.Text = CStr(Me.MonthCalendar1.SelectionRange.Start)
```

8. Вернитесь в режим конструктора и добавьте в форму элемент управления DateTimePicker, оставив имя по умолчанию DateTimePicker1.
9. Дважды щелкните элемент управления DateTimePicker, чтобы открыть обработчик событий по умолчанию в редакторе кода.
10. В обработчике событий DateTimePicker_ValueChanged добавьте следующий код для добавления элементов в список.

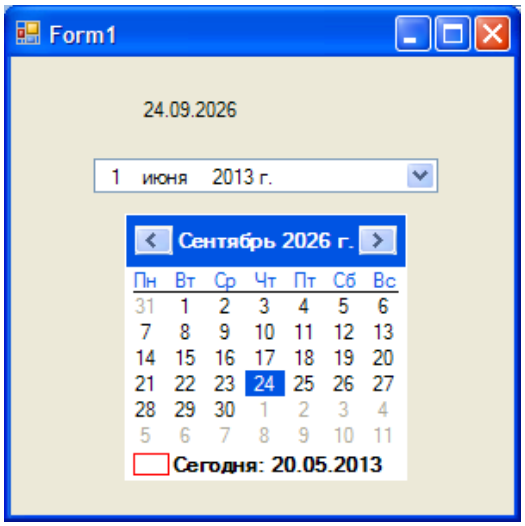
```
Me.Label1.Text = CStr(Me.DateTimePicker1.Value)
```

11. Нажмите клавишу F5 для запуска программы.

12. Когда появится форма, выберите дату в элементе управления MonthCalendar и убедитесь, что она отображается в метке.

13. Щелкните стрелку раскрывающегося списка элемента управления DateTimePicker и выберите дату.

Дата и время отображаются в метке.



Извлечение нескольких дат

Диапазон дат, выбранных в элементе управления MonthCalendar, можно извлечь с помощью свойств Start и End свойства SelectionRange. По умолчанию максимальное число дней, которые можно выбрать, равно 7, но при необходимости этот параметр можно изменить, установив значение свойства MaxSelectionCount. Чтобы определить, выбран ли диапазон дат, просто проверьте, совпадают ли даты начала и конца.

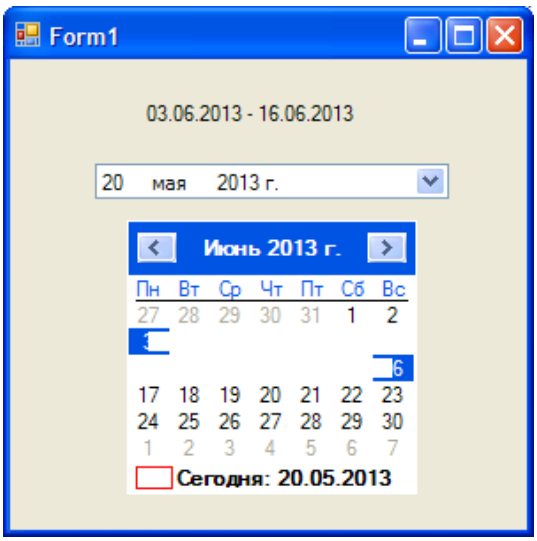
Извлечение диапазона дат из элемента управления календарем месяца

1. Замените код в обработчике событий MonthCalendar1_DateChanged следующим. Этот код устанавливает максимальное число дней (две недели), которые могут быть выбраны в элементе управления. Он отображает дату начала в метке, если выбран только один день, и отображает диапазон дат при выборе диапазона дней в элементе управления MonthCalendar.

```
Me.MonthCalendar1.MaxSelectionCount = 14

If Me.MonthCalendar1.SelectionRange.Start = _
    Me.MonthCalendar1.SelectionRange.End Then

    Me.Label1.Text = CStr(Me.MonthCalendar1.SelectionStart)

Else

    Me.Label1.Text = Me.MonthCalendar1.SelectionRange.Start & _
        " - " & Me.MonthCalendar1.SelectionRange.End

End If
```

2. Нажмите клавишу F5 для запуска программы.

3. Когда появится форма, выберите диапазон дат в элементе управления MonthCalendar и убедитесь, что диапазон дат появился в метке.



Форматирование дат

Даты, возвращаемые элементами управления MonthCalendar и DateTimePicker, можно форматировать с помощью функции FormatDateTime. Существует несколько констант, которые можно использовать для указания формата даты (см. теор. часть).

Форматирование даты в метке

1. Замените код в обработчике событий MonthCalendar1_DateChanged следующим. Этот код форматирует дату, возвращаемую в полном формате.

```
Me.MonthCalendar1.MaxSelectionCount = 14

If Me.MonthCalendar1.SelectionRange.Start = _
    Me.MonthCalendar1.SelectionRange.End Then

    Me.Label1.Text = FormatDateTime( _
        Me.MonthCalendar1.SelectionStart, _
        DateFormat.LongDate)
Else
    Me.Label1.Text = FormatDateTime( _
        Me.MonthCalendar1.SelectionRange.Start, _
        DateFormat.LongDate) & " - " & FormatDateTime( _
        Me.MonthCalendar1.SelectionRange.End, DateFormat.LongDate)
End If
```

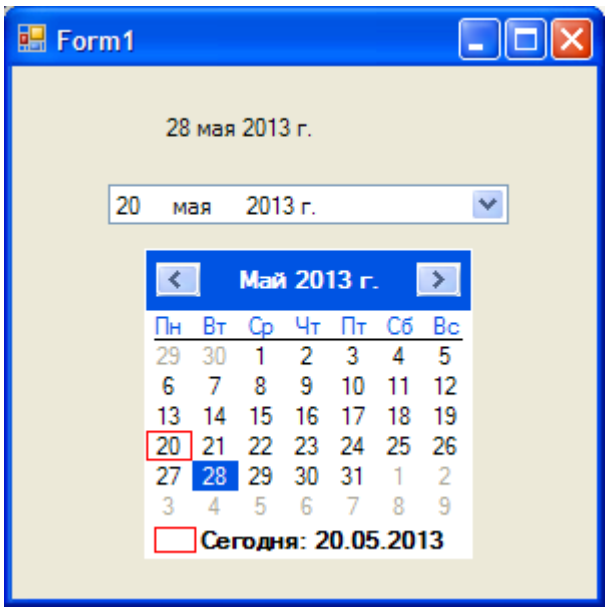
2. Замените код в обработчике событий DatePicker1_ValueChanged следующим. Этот код форматирует дату, возвращаемую в полном формате.

```
    Me.Label1.Text = FormatDateTime(Me.DateTimePicker1.Value, _
DateFormat.LongDate)
```

3. Нажмите клавишу F5 для запуска программы.

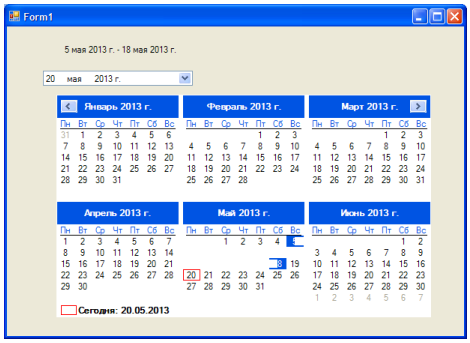
4. Когда появится форма, выберите дату или диапазон дат в элементе управления MonthCalendar. Убедитесь, что дата или диапазон дат отображается в метке в полном формате.

5. Выберите дату в элементе управления DateTimePicker и убедитесь, что дата в метке отображается в полном формате.



Чтобы отобразить несколько месяцев

- Задайте для свойства CalendarDimensions значение, равное числу месяцев, отображаемых по горизонтали и вертикали.




```
MonthCalendar1.CalendarDimensions = New System.Drawing.Size(3, 2)
```

Программа Birthday

В программе Birthday элементы управления DateTimePicker и Button используются, чтобы выяснить у пользователя дату его рождения и показать эту информацию в окне сообщения.

1. В области элементов выберите элемент управления Button, и ниже объекта выбора даты и времени добавьте объект кнопки. Эта кнопка будет использована для показа дня рождения и для проверки правильности работы объекта выбора даты и времени.
2. В окне Свойства (Properties) измените свойство Text объекта кнопки на Показать день моего рождения.
3. Дважды щелкните мышью на объекте кнопки, а потом наберите следующий фрагмент программы между операторами Private Sub и End Sub в процедуре события Button1_Click:

```
MsgBox("Ваш день рождения " & DateTimePicker1.Text)
MsgBox("День года: " & DateTimePicker1.Value.DayOfYear.ToString())
MsgBox("Сейчас: " & DateTimePicker1.Value.TimeOfDay.ToString())
```

Этот фрагмент программы показывает три последовательных окна сообщения (небольшие диалоговые окна), которые содержат информацию из объекта календаря. В первой строке используется свойство Text календаря для вывода информации о дате рождения, которую пользователь выберет в этом объекте после запуска программы. Функция MsgBox кроме текстового значения из свойства Text календаря показывает строку "Ваш день рождения". Эти два текстовых элемента объединяются в строку с помощью оператора конкатенации (слияния) строк &.

Во второй строке `DateTimePicker1.Value.DayOfYear.ToString()` объект календаря используется для вычисления дня года, отсчитывая с 1 января. Это делается с помощью свойства DayOfYear и метода ToString, который переводит числовой результат вычисления даты в текстовое значение, которое гораздо проще показать с помощью функции MsgBox.

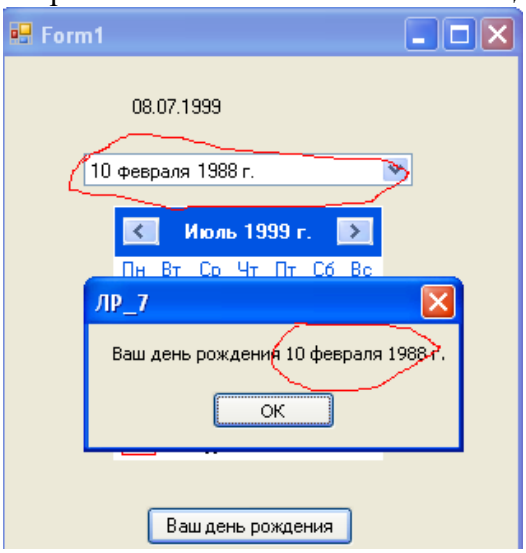
В третьей строке фрагмента, после перевода значения в строковое (или текстовое) представление, в окне сообщения показывается информация о точном времени.

Запуск программы Birthday

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа Birthday запустится в среде разработки. В окне объекта выбора даты и времени появится текущая дата.

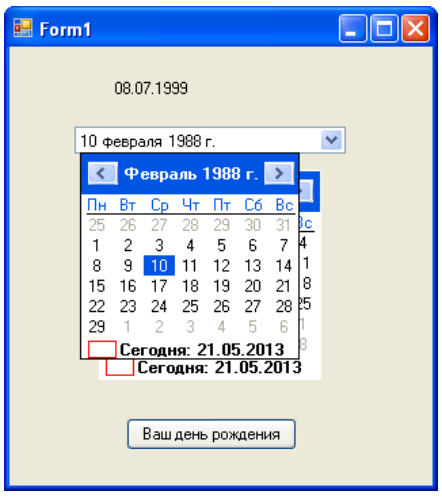
2. Нажмите стрелку раскрывающегося списка, чтобы вывести на экран представление этого объекта в виде календаря. Форма будет выглядеть как на следующей иллюстрации.

3. Выберите в элементе DatetimePicker1 число, месяц и год Вашего рождения, пользуясь стрелками прокрутки. Нажмите кнопку Показать день моего рождения. Visual Basic исполнит введенный вами код программы и покажет окно с сообщением, содержащим день и дату вашего рождения. Обратите внимание на соответствие двух дат.



4. В окне сообщения нажмите OK. Появится второе окно сообщения, указывающее, в какой день года вы родились.

5. Нажмите OK, чтобы показать последнее окно сообщения. Появятся текущие дата и время. Вы обнаружите, что объект выбора даты и времени очень удобен - он не только помнит новую, введенную вами информацию о дате или времени, но также отслеживает текущие дату и время и может показывать эту информацию в различных форматах.



Совет. Чтобы настроить объект выбора даты и времени для показа времени, а не даты, установите свойство Format этого объекта равным Time.

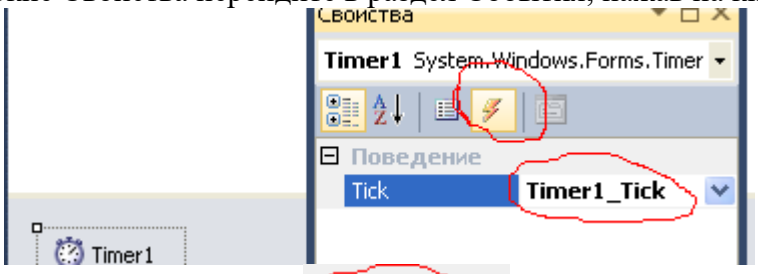
Работа с датами в VB.Net. (Дополнительно)

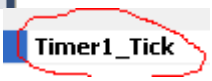
Реализовать программу которая будет узнавать текущую дату и время.

Чтобы узнать текущее время в VB.Net есть функция - **TimeString**, чтобы узнать дату - **DateString**. Кроме того есть функции работы со временем: **Hour()**(часы), **Minute()**(минуты), **Second()**(секунды). Эти функции вырезают часы, минуты, секунды из указанно времени. Например, *Minute(TimeString)* - вырезает минуты из текущего времени. И функции для работы с датами: **Day()**(год),

Month()(месяц), **Year**(год). Эти функции вырезают из текущей даты: день, месяц, год. Например, *Month(DateString)*, вырезает из текущей даты месяц.

1. Создайте новый проект командой Создать проект из меню Файл (File). Выберите элемент Приложение Windows Forms и нажмите кнопку ОК.
2. Добавьте в форму три элемента Label.
3. Чтобы размеры метки автоматически регулировались, в зависимости от текста в метке Задайте AutoSize = True.
4. Удалите текст из свойства Text элемента управления Метка: Text = "".
5. Добавьте в форму невизуальный элемент управления Timer (Enabled = True, Interval = 1000).
6. Напишите обработчик события Tick элемента Timer. Для этого выделите Элемент Таймер и в окне Свойства перейдите в раздел События, нажав на кнопку  .



7. Дважды щелкните в строке 

и введите в открывшемся окне код:

```
Label1.Text = "Время: " & TimeString
Label2.Text = "Дата: " & DateString
```

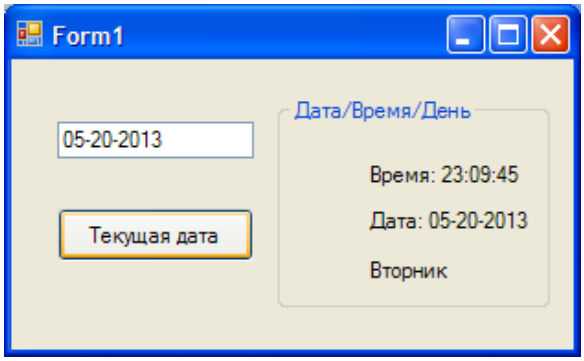
8. Для определения дня недели нужна функция ***WeekDay***. Напишите обработчик события загрузки формы:

```
Dim Den As Integer
Den = Weekday(Today) ' Определяем день недели
If Den = 1 Then Label3.Text = "Воскресенье"
If Den = 2 Then Label3.Text = "Понедельник"
If Den = 3 Then Label3.Text = "Вторник"
If Den = 4 Then Label3.Text = "Среда"
If Den = 5 Then Label3.Text = "Четверг"
If Den = 6 Then Label3.Text = "Пятница"
If Den = 0 Then Label3.Text = "Суббота"
```

9. В области элементов выберите элемент управления Button и добавьте его, задайте свойство Text =Текущая дата/Время
10. Напишите обработчик события нажатия кнопки для вывода текущей даты или времени:

```
'получить текущие дату и время
TextBox1.Text = My.Computer.Clock.LocalTime ' (microsoft.viual.basic)
TextBox1.Text = DateAndTime.Now ' (microsoft.viual.basic)
TextBox1.Text = DateTime.Now ' (mscorlib)

'получить только текущее время
TextBox1.Text = DateAndTime.Now.ToLongTimeString
TextBox1.Text = DateTime.Now.ToLongTimeString
TextBox1.Text = Format(Now, "hh:mm:ss")
TextBox1.Text = TimeString

'получить только текущюю дату
TextBox1.Text = DateAndTime.Now.Date
TextBox1.Text = DateAndTime.Today
TextBox1.Text = DateTime.Now.Date
TextBox1.Text = DateTime.Today
TextBox1.Text = Format(Now, "dd.MM.yyyy")
TextBox1.Text = DateString
```

11. Запустите проект.



Контрольные вопросы:

1. Назначение элемента управления DateTimePicker
2. Назначение элемента управления MonthCalendar
3. Назовите свойства, используемые для извлечения даты из этих элементов.
4. Свойства элемента управления MonthCalendar для извлечения диапазона дат.
5. Назначение оператора конкатенации &, пример его использования
6. Отображение нескольких месяцев элемента управления MonthCalendar.
7. Функция форматирования дат.
8. Какие существуют константы для указания формата даты, опишите их.
9. Функции для работы с датой и временем в VB.Net.
10. Объясните работу фрагмента кода

```
DateTimePicker1.Value.DayOfYear.ToString()
```

События компонентов (элементов управления), их сущность и назначение.

Создание процедур на основе событий

Лабораторная работа № 30

Тема: «Создание проекта с использованием кнопочных компонентов»

Цель: *создание Windows-приложения, аналогичного стандартному калькулятору Windows в среде Visual Studio.*

Ход работы:

1. Выполнить задания, следуя указаниям (задание 1-го уровня на «3», задание 2-го уровня на «4», задание 3-го уровня на «5»,).
2. Ответить на контрольные вопросы (в устной форме).
3. Предъявить преподавателю результаты работы программы и исходные коды.
4. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

☞*Задание на лабораторную работу.*

Задание 1-го уровня

1. Создать новый проект.

2. Составить эскиз интерактивной формы - калькулятора, способной выполнять простые арифметические действия (Рис. 1).

3. Задать значения свойств элементов управления, размещенных на интерактивной форме.

4. Для каждого элемента управления написать программный код, соответствующий событию активизации (нажатия) элемента управления.

5. Осуществить сборку и компиляцию модулей проекта.

6. Выполнить вычисления с помощью созданного калькулятора.

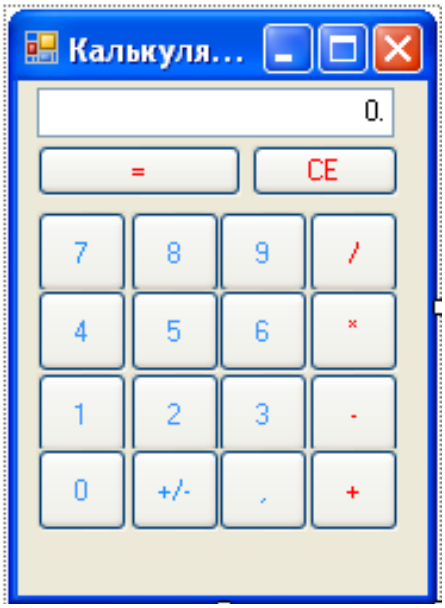


Рис. 1. Эскиз интерактивной формы

Задание 2-го уровня. Доработайте калькулятор из первого задания, добавив следующие возможности: вычисление квадратного корня, процентов, обратного числа, стирание одного символа, стирание числа (Рис. 2).

Рис. 2. Эскиз интерактивной формы (2-й уровень)

Задание 3-го уровня Доработайте калькулятор из первого и второго заданий, добавив возможности работы с памятью: стереть память, вывести из памяти, записать в память, добавить в память (Рис. 3).

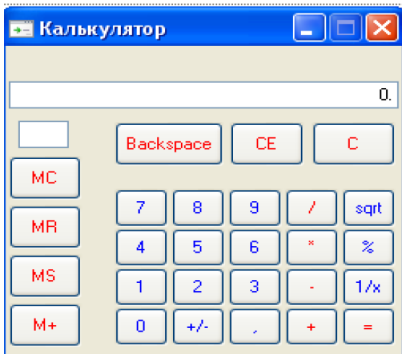


Рис. 3. Эскиз интерактивной формы (3-й уровень)

Порядок выполнения работы (1-й уровень)

1. Создать новый проект командой *Новый проект из меню Файл* (порядок создания нового проекта подробно описан в лабораторной работе № 1).

2. Создать эскиз интерактивной формы.

Используя панель инструментов *ToolBox*, разместить на форме элементы управления (кнопки - *Button1* - *Button18* и текстовое поле - *TextBox1*), как показано на Рис. 4.

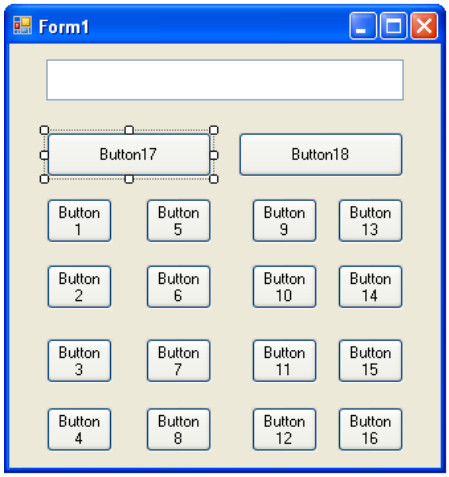


Рис. 4. Размещение элементов управления на форме

После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель Свойства (*Properties)*, которая появляется после одинарного щелчка мышью по нужному элементу управления, расположенному на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

3.1. Установите значения свойств *MaximizeBox*, *Size* и *Text* объекта *Form1*, как показано на Рис. 5.

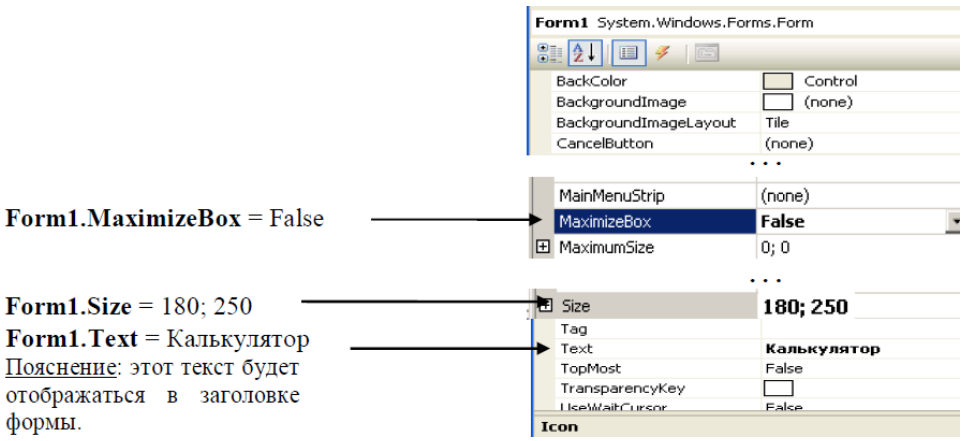


Рис. 5. Свойства Form1

3.2. Установите значения свойств элемента – текстовое поле (*TextBox*), как указано в Табл. 1.

Свойство	**Значение**
TextBox1.Name	TextBox1
TextBox1.Text	0.
TextBox1.BorderStyle	Fixed3D Пояснение: щелкнуть на кнопку в правом поле, затем с помощью окна настройки установить значение данного свойства
TextBox1.TextAlign	Right

3.3. Установите значения свойств элементов – кнопок (*Button*) как указано в Табл. 2.

Табл. 2

Свойство	**Значение**
Button1.Name	bt7
Button1.ForeColor	Голубой
Button1.Text	7
Button2.Name	bt4
Button2.ForeColor	Голубой
Button2.Text	4
Button3.Name	bt1
Button3.ForeColor	Голубой
Button3.Text	1
Button4.Name	bt0
Button4.ForeColor	Голубой
Button4.Text	0
Button5.Name	bt8
Button5.ForeColor	Голубой
Button5.Text	8
Button6.Name	bt5
Button6.ForeColor	Голубой
Button6.Text	5
Button7.Name	bt2
Button7.ForeColor	Голубой
Button7.Text	2
Button8.Name	btpm
Button8.ForeColor	Голубой
Button8.Text	+/-
Button9.Name	bt9

Button9.ForeColor	Голубой
Button9.Text	9
Button10.Name	bt6
Button10.ForeColor	Голубой
Button10.Text	6
Button11.Name	bt3
Button11.ForeColor	Голубой
Button11.Text	3
Button12.Name	btpoint
Button12.ForeColor	Голубой
Button12.Text	,
Button13.Name	btdel
Button13.ForeColor	Красный
Button13.Text	/
Button14.Name	btpr
Button14.ForeColor	Красный
Button14.Text	*
Button15.Name	btmin
Button15.ForeColor	Красный
Button15.Text	-
Button16.Name	btplus
Button16.ForeColor	Красный
Button16.Text	+
Button17.Name	btis
Button17.ForeColor	Красный
Button17.Text	=
Button18.Name	btce
Button18.ForeColor	Красный
Button18.Text	CE

В результате изменения свойств вышеперечисленных объектов форма *Form1* примет вид, указанный на Рис. 1.

4. Написание программы (кода) включает в себя разработку кода для обработки событий нажатия всех кнопок.

4.1. Выполните двойной щелчок левой кнопкой мыши на пустом месте формы. В появившемся окне головного модуля *Form1.vb* выберете блок Объявление, как показано на Рис. 6, и введите программный код, объявляющий переменные:

- *IsText* (для хранения содержимого текстовой строки);
- *IsNumber* (для хранения числа);
- *Point* (для указания разделителя дробной части);
- *op* (для хранения номера арифметической операции).

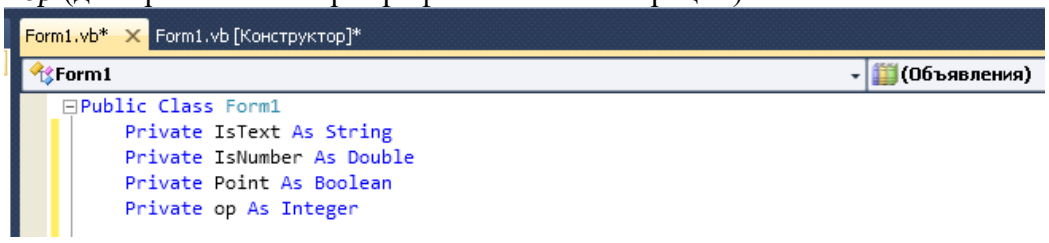


Рис. 6. Объявление переменных в блоке Form1 - Объявления

4.2. Обработка нажатия цифровых клавиш: 1, 2 … 9, 0.

4.2.1. Введите программный код для обработки события – нажатия кнопки «1» (*bt1_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *bt1* и ввести код:

```
If IsText = "0" Or IsText = "+" Or IsText = "-" Or IsText = "/" Or IsText = "*" _
    Then IsText = "1" Else IsText = IsText + "1"
TextBox1.Text = IsText
```

Пояснение: данный фрагмент кода сначала проверяет, не является ли вводимая цифра первой в числе и не была ли нажата клавиша арифметической операции (+, -, /, *), в этом случае вводимая цифра заменяет содержимое текстового поля (**TextBox1.Text**). В противном случае вводимая цифра добавляется к содержимому текстового поля (**TextBox1.Text**).

4.2.2. Введите программный код для обработки события – нажатия кнопки «2» (*bt2_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *bt2* и ввести код аналогичный коду п. 4.2.1:

```
If IsText = "0" Or IsText = "+" Or IsText = "-" Or IsText = "/" Or IsText = "*" _
    Then IsText = "2" Else IsText = IsText + "2"
TextBox1.Text = IsText
```

4.2.3. По аналогии с п. 4.2.1, 4.2.2 введите программный код для обработки нажатия оставшихся цифровых кнопок (*bt3 – bt9, bt0*). Для экономии времени можете копировать повторяющиеся фрагменты кода.

4.3. Введите программный код для обработки события – нажатия кнопки «смена знака числа (+/-)» (*btpm_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpm* и ввести код:

```
IsText = CStr(Val(IsText) * (-1))
TextBox1.Text = IsText
```

Пояснение: функция **Val()** преобразует текстовый тип в числовой; функция **CStr()** преобразует числовой тип в текстовый.

4.4. Введите программный код для обработки события – нажатия кнопки «запятая, отделяющая целую часть от дробной (,)» (*btpoint_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpoint* и ввести код:

```
If Point = False Then IsText = IsText + "."
TextBox1.Text = IsText
Point = True
```

Пояснение: данный фрагмент кода через переменную *Point* сначала проверяет, не была ли кнопка «запятая» нажата ранее при вводе текущего числа.

4.5. Обработка нажатия кнопок арифметических действий: /, *, -, +.

4.5.1. Введите программный код для обработки события – нажатия кнопки «деление (/)» (*btdel_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btdel* и ввести код:

```
IsNumber = Val(TextBox1.Text)
op = 1
IsText = "/"
TextBox1.Text = IsText
Point = False
```

Пояснение: в данном фрагменте кода переменной *op* присваивается номер арифметической операции. При этом деление соответствует первому номеру, умножение – второму, вычитание - третьему, сложение – четвертому. Значение переменной *op* будет использоваться при вычислении результата (нажатие кнопки *btis*).

4.5.2. Введите программный код для обработки события – нажатия кнопки «умножение (*)» (*btpr_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btpr* и ввести код, аналогичный коду п. 4.5.1, изменив номер операции на второй:

```
IsNumber = Val(TextBox1.Text)
op = 2
IsText = "*"
TextBox1.Text = IsText
Point = False
```

4.5.3. По аналогии с п. 4.5.1, 4.5.2 введите программный код для обработки событий – нажатия кнопок «вычитание (-)» (*btmin_Click*) и «сложение (+)» (*btplus_Click*), изменив соответственно номера операций и символы, их отображающие.

4.6. Введите программный код для обработки события – нажатия кнопки «=» (*btis_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btis* и ввести код:

```
Select Case op
    Case 1
        IsNumber = IsNumber / Val(TextBox1.Text)
    Case 2
        IsNumber = IsNumber * Val(TextBox1.Text)
    Case 3
        IsNumber = IsNumber - Val(TextBox1.Text)
    Case 4
        IsNumber = IsNumber + Val(TextBox1.Text)
End Select
IsText = CStr(IsNumber)
TextBox1.Text = IsText
IsText = ""
Point = False
```

Пояснение: в данном фрагменте кода используется конструкция **Select Case**, позволяющая выбирать выполняемую часть кода в зависимости от значения переменной *op*.

4.7. Введите программный код для обработки события – нажатия кнопки «CE» (*btce_Click*). Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке *btce* и ввести код:

```
IsNumber = 0
IsText = "0"
TextBox1.Text = IsText
Point = False
```

8. Выполните сборку и компиляцию модулей проекта.
9. Запустите проект и протестируйте.

Пояснения для выполнения задания 2-го уровня

1. Для вычисления квадратного корня используйте функцию **Math.Sqrt()**.
2. Для обработки события – нажатия кнопки «Backspace» (удаление последнего введѐнного символа) можно использовать функцию **Remove()**, удаляющую указанное число символов в текстовой переменной начиная с указанной позиции, и свойство **Length**, возвращающее число символов в значении переменной. Например, так:

IsText = IsText.Remove(IsText.Length - 1, 1).

3. Нажатие кнопки «C» должно привести к стиранию только текущего набираемого числа, например если Вы ошиблись при вводе и хотите стереть число не посимвольно кнопкой «Backspace», а сразу целиком. Отличается от кнопки «CE» тем, что не стирает предыдущие набранные числа и операции.

Пояснения для выполнения задания 3-го уровня

В текстовом поле над кнопками по работе с памятью (M+, MS, MR, MC) должен отображаться символ «M», если в памяти содержится какое-либо число.

Контрольные вопросы:

1. Назначение функций Val() и CStr().

2. Поясните данный фрагмент кода:

```
If IsText = "0" Or IsText = "+" Or IsText = "-" Or IsText = "/" Or IsText = "*" _
    Then IsText = "1" Else IsText = IsText + "1"
TextBox1.Text = IsText
```

3. Перечислите свойства кнопки, используемые для задания отображаемого текста и его цвета.

```
If Point = False Then IsText = IsText + "."
TextBox1.Text = IsText
Point = True
```

4. Поясните данный фрагмент кода:
5. Назначение конструкции **Select Case**
6. Поясните данный фрагмент кода:

```
IsNumber = Val(TextBox1.Text)
op = 1
IsText = "/"
TextBox1.Text = IsText
Point = False
```

Лабораторная работа № 31

Тема: «Создание проекта с использованием компонентов стандартных диалогов и системы меню»

Цель: *научиться использовать компоненты стандартных диалогов, системы меню и отображать их в форме Windows Forms.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Добавление меню с помощью элемента управления MainMenu

Элемент управления MainMenu - это инструмент, с помощью которого в программу можно добавить меню и настроить их в окне Свойства. С помощью MainMenu можно добавлять новые меню, изменять и удалять существующие. В меню можно добавить специальные возможности: клавиши ускоренного доступа, отметки "включено/выключено" и сочетания клавиш. После того, как меню добавлено в форму, для обработки команд меню использовать процедуры обработки событий.

Соглашения о пунктах меню

По соглашению, в приложениях для Microsoft Windows каждое название и команды меню начинаются с заглавной буквы. Часто первыми двумя командами меню в строке меню являются Файл и Правка, а последним является Справка. Часто встречаются названия команд Вид, Формат и Окно. Не имеет значения, какие меню и команды вы используете в приложении - главное, чтобы они были понятны и соответствовали своим названиям. При создании элементов меню придерживайтесь следующих рекомендаций.

- Используйте короткие и ясные названия, состоящие из одного или максимум двух слов.
- Каждому элементу меню назначайте клавишу доступа. Если возможно, используйте для этого их первую букву.
- У элементов меню, расположенных на одном уровне видимости, клавиши доступа не должны совпадать.
- Если команда используется как переключатель "включено/выключено", то когда она активна, рядом с ней должна быть видна галочка. Чтобы добавить такую пометку, в окне Свойства для этой команды свойство Checked должно иметь значение True.
- Если команда при нажатии выполняется не сразу, а от пользователя потребуется дополнительные действия, поставьте после такой команды многоточие (...). Оно указывает, что если пользователь выберет эту строку, то откроется диалоговое окно.

Добавление клавиш доступа к командам меню

В большинстве приложений команды меню можно вызывать с помощью клавиатуры. Например, чтобы в Visual Studio открыть меню Файл, нужно нажать клавишу Alt а затем Ф. Когда меню Файл откроется, то, чтобы выполнить команду Печать, достаточно нажать на клавиатуре П. Клавиши, которые вы нажимаете вместе с Alt или в открытом меню, называются **клавишами доступа** (или **клавишами быстрого доступа**). Клавишу доступа можно определить по подчеркнутому символу.

Чтобы добавить в главное меню клавишу доступа, активизируйте Конструктор меню и введите перед требуемой буквой в имени меню символ "амперсанд" (&). Во время выполнения программы соответствующая клавиша на клавиатуре будет работать как клавиша быстрого доступ к меню.

С помощью элемента управления MainMenu можно назначать создаваемым меню сочетания клавиш. Сочетания клавиш - это комбинация клавиш на клавиатуре, нажав которую пользователь может вызвать команду меню не открывая его. Например, в обычном меню Правка в приложении Windows можно выделить текст и скопировать его в буфер обмена, нажав клавиши (Ctrl)+(C). Эти сочетания настраиваются в свойстве Shortcut элемента управления MainMenu.

Использование элементов управления для диалоговых окон

В Visual Studio на закладке Windows Forms окна области элементов имеется семь стандартных элементов управления для диалоговых окон. Во многих случаях нужно написать код, который подключает эти диалоговые окна к программе, но пользовательский интерфейс уже сделан, и он соответствует стандартам для общих задач в приложениях Windows. Все семь имеющихся элементов управления для стандартных диалоговых окон перечислены в следующей таблице.

Процедуры обработки событий, которые управляют общими диалоговыми окнами

Чтобы показать в программе диалоговое окно, в процедуре обработки события для соответствующей команды меню нужно ввести оператор, состоящий из имени диалогового окна и метода ShowDialog. Если это необходимо, то перед открытием диалогового окна нужно запрограммировать свойства этого окна. Наконец, в тексте программы должен присутствовать код, который после закрытия окна выполнит те или иные операции в зависимости от выбора или действий, которые пользователь выполнит в этом диалоговом окне.

Название элемента управления	Назначение
OpenFileDialog	Получает названия диска, папки и файла для существующего файла.
SaveFile Dialog	Получает названия диска, папки и файла для нового файла.
FontDialog	Позволяет выбрать новый шрифт и его стиль.
ColorDialog	Позволяет выбрать цвет из палитры.
PrintDialog	Позволяет задать параметры печати.
PrintPreviewDialog	Отображает диалоговое окно предварительного просмотра материала для печати (как в MS Word).
PageSetupDialog	Позволяет управлять параметрами страницы: полями, размером бумаги и ее ориентацией.

Управление выбором цвета с помощью установки свойств диалогового окна выбора цвета

Диалоговое окно выбора цвета тоже можно настроить. Например, можно управлять тем набором цветов, которые будут предоставлены на выбор пользователя при открытии окна. Эти параметры можно настроить в окне Свойства в среде разработки или задать их в тексте программы перед открытием этого диалогового окна, используя метод ShowDialog. Следующая таблица содержит список наиболее часто используемых свойств элемента управления ColorDialog. Чтобы задействовать ту или иную настройку, соответствующему свойству должно быть присвоено значение True, чтобы отменить настройку - значение False.

Свойство	Значение
AllowFullOpen	Если присвоено значение True, в диалоговом окне будет работать кнопка Define Custom Colors (Определить цвет).

AnyColor	Если присвоено значение True, пользователь сможет выбрать любой цвет из показанных в диалоговом окне.
FullOpen	Если присвоено значение True, при открытии диалогового окна будет видна область Custom Colors (Дополнительные цвета).
ShowHelp	Присвойте True, если нужно включить в диалоговом окне кнопку Help (Справка).
SolidColorOnly	Если присвоено значение True, пользователь сможет выбрать только "чистые" цвета (цвета со смешением будут отключены).

☞*Задание на лабораторную работу.*

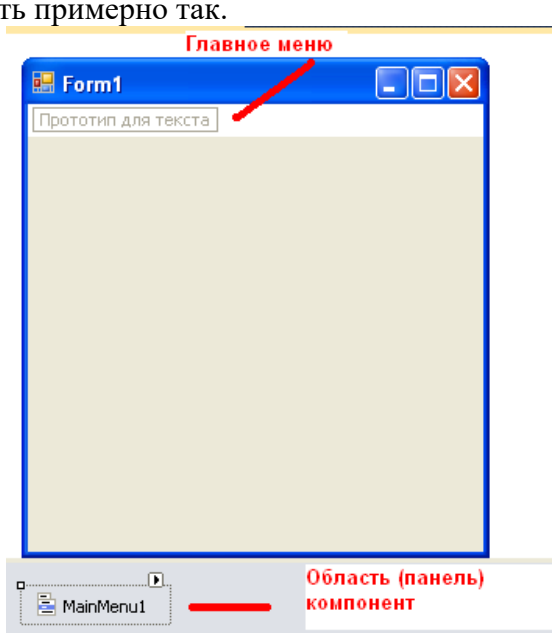
С помощью элемента управления MainMenu мы создадим меню Clock, команды которого показывает текущую дату и время.

Порядок выполнения работы

Создание меню

1. Создайте новый проект командой Создать проект (New Project) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент Приложение Windows Forms и нажмите кнопку ОК.
3. Выберите элемент управления MainMenu на закладке Windows Forms окна области элементов и нарисуйте на поле формы элемент управления.

Форма будет выглядеть примерно так.



В Visual Studio .NET невидимые объекты, в том числе меню и таймеры, показываются в среде разработки на отдельной панели, которая называется **областью компонентов**. На этой панели их можно выделять, настраивать их свойства или удалять.

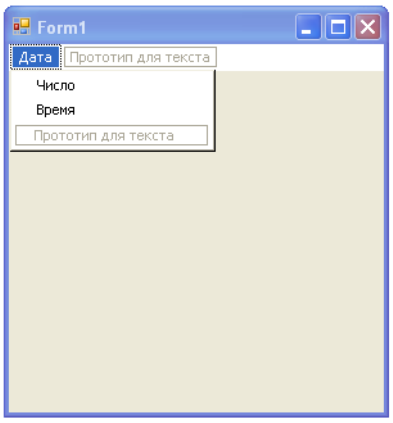
В дополнение к объекту меню, расположенному в области компонентов, создаваемое в Visual Studio.NET меню графически изображается в верхней части формы. Чтобы ввести название меню, нужно щелкнуть по полю Прототип для текста. После этого можно добавлять заголовки подменю и других меню, выбирая нужные поля с помощью стрелок и вписывая туда требуемые названия. Позже вы всегда сможете вернуться к этому встроенному Конструктору меню и отредактировать то, что уже сделано, или добавить новые пункты. Объект главного меню очень хорошо настраивается и позволяет создать решения с использованием меню, не уступающие лучшим Windows-приложениям.

4. Щелкните на Прототип для текста, введите **Дата**, а затем нажмите клавишу (Enter). Слово "Дата" стало названием первого меню, и появились два новых поля Прототип для текста, позволяющие создать элементы подменю или команды в меню Дата и дополнительные меню. В данный момент выберите подменю.

Совет. Если меню с формы исчезает, выберите объект MainMenu1 в области компонентов, и оно снова появится.

5. Чтобы в меню **Дата** создать новую команду, введите слово **Число**, а затем нажмите клавишу (Enter). Visual Studio добавит команду к меню и выделит следующий элемент подменю.

6. Наберите слово **Время**, а затем нажмите клавишу (Enter). Теперь у вас есть меню **Дата** с двумя командами меню - **Число** и **Время**. Форма будет выглядеть примерно так.



7. Чтобы закрыть Конструктор меню, щелкните на поле формы. Конструктор меню закроется, и в среде разработки будет полностью видна форма, а меню исчезнет. Чтобы увидеть меню и начать с ним работать, нужно щелкнуть на его названии.

8. В поле формы нажмите слово Дата. Меню появляется снова, с уже знакомыми полями Прототип для текста. Теперь его опять можно настраивать.

Добавление клавиш доступа

1. На форме выберите меню **Дата**, а затем щелкните на нем еще раз. В названии появится курсор редактирования текста. Появление текстового курсора означает, что название меню можно изменить или добавить символ **&**, чтобы обозначить клавишу доступа.

2. Чтобы убедиться, что курсор находится перед первой буквой, нажмите на клавишу со стрелкой влево. Курсор будет мигать перед буквой "Д" в слове Дата.

3. Введите **&** (амперсанд), чтобы указать, что буква "Д" является клавишей доступа для меню Дата.

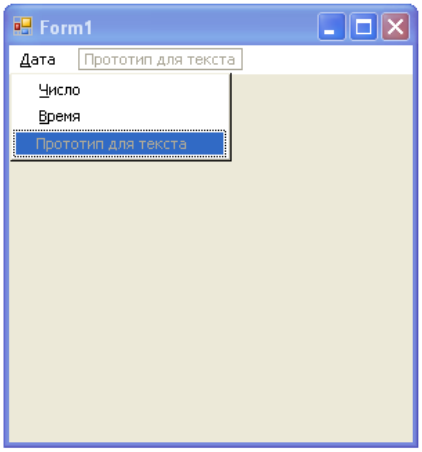
4. В списке меню выберите команду **Число**, а затем щелкните на ней еще раз, чтобы появился курсор редактирования текста.

5. Введите **&** (амперсанд) перед буквой "Ч". Теперь буква "ч" определена как клавиша доступа для команды **Число**.

6. В списке меню выберите команду **Время**, а затем щелкните на этой команде второй раз, чтобы появился курсор.

7. Введите **&** (амперсанд) перед буквой "В". Теперь буква "В" определена как клавиша доступа для команды Время.

8. Нажмите клавишу (Enter). Нажатие на (Enter) фиксирует изменения в тексте. Ваша форма будет выглядеть примерно так.



Теперь с помощью Конструктора меню давайте изменим порядок команд **День** и **Время**. Изменение порядка элементов меню - это важный навык, так как это требуется довольно часто.

Изменение порядка элементов меню

1. Выберите в форме меню **Дата**, чтобы развернуть элементы этого меню. Изменить порядок элементов очень просто - нужно просто перетащить элемент на новое место в меню.
2. Перетащите надпись **Время** на надпись **Число** и отпустите клавишу мыши. Перетаскивание элемента меню поверх другого элемента означает, что вы хотите разместить его перед вторым элементом. Visual Studio немедленно перемещает элемент меню **Время** на новое место перед элементом **Число**.

Мы закончили создавать пользовательский интерфейс для меню Дата. Теперь нужно запрограммировать процедуры, соответствующие строкам меню, для обработки выбора пользователя.

Совет. Чтобы удалить из меню ненужный элемент, щелкните на этом элементе, а затем нажмите клавишу (Delete).

Обработка выбора меню

После того, как меню и команды настроены с помощью объекта MainMenu, они также становятся объектами программы. Чтобы заставить объекты меню выполнять осмысленную работу, необходимо написать для них процедуры обработки событий. Процедуры обработки событий меню обычно содержат операторы, которые показывают информацию в пользовательском интерфейсе, обрабатывают ввод или изменяют одно или несколько свойств меню. Если для обработки команды требуется получить дополнительную информацию, то процедура обработки события обычно открывает диалоговое окно. Для этого используется один из элементов управления диалоговых окон Windows Forms или один из элементов управления для ввода.

Добавление на форму объекта надпись

1. В области элементов выберите элемент управления Label.
2. Нарисуйте в центре формы надпись среднего размера. На форме появится объект Label, и в коде программы у него будет имя Label1.
3. Задайте для этой надписи следующие свойства:

Объект	Свойство	Значение
Label1	BorderStyle	FixedSingle
	Font	Microsoft Sans Serif, жирный , 14 пунктов
	Text	(empty)
	TextAlign	MiddleCenter

Редактирование процедур событий меню

1. Щелкните на меню **Дата**, чтобы его раскрыть.

2. Чтобы открыть в редакторе кода процедуру обработки событий для команды **Время**, дважды щелкните мышью на этой команде. В редакторе кода появится процедура события MenuItem3_Click. Имя MenuItem3_Click означает, что пункт **Время** был третьим из созданных в этом проекте (вслед за **Дата** и **Число**), а слово _Click напоминает, что это процедура события, которая запускается при щелчке на этом элементе меню.

3. Добавьте в программу следующий оператор

```
Label1.Text = TimeString
```

Этот оператор присваивает текущее время (по системным часам) свойству Text объекта Label1, которое, собственно, и показывается в виде надписи.

4. Нажмите на клавишу со стрелкой вниз. Visual Basic интерпретирует строку и, если потребуется, изменит заглавные буквы и добавит или удалит пробелы. Visual Basic проверяет каждую строку в процессе ее ввода и ищет в них синтаксические ошибки. Набор строки можно закончить, нажав клавишу (Enter), стрелку вверх или стрелку вниз.

5. В Обозревателе решений нажмите кнопку Просмотреть конструктор, а затем дважды щелкните мышью на команде **Число** в меню **Дата**. В редакторе кода откроется процедура обработки событий MenuItem2_Click. Эта процедура исполняется тогда, когда пользователь щелкает в меню **Дата** на команде **Число**.

6. Добавьте в программу следующий оператор

```
Label1.Text = DateString
```

Этот оператор присваивает сегодняшнее число (по системным часам) свойству Text объекта Label1, которое показывается в виде надписи. Предыдущий текст в объекте Label1, если он имелся, будет заменен.

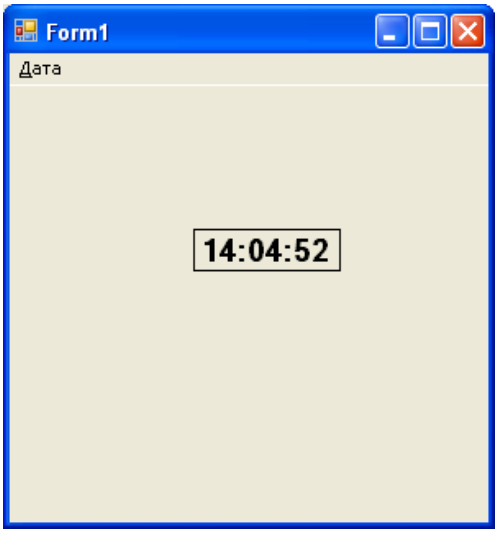
7. Чтобы закончить ввод строки, нажмите клавишу со стрелкой вниз.

Запуск программы Menu

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа Menu запустится в среде разработки.

2. В строке меню выберите пункт **Дата**. Появится меню **Дата**.

3. Выберите команду **Время**. В поле надписи появится текущее системное время, как показано ниже.



Теперь посмотрим, какое сегодня число.

4. Нажмите и отпустите клавишу (Alt). В строке меню выделится меню **Дата**.

5. Чтобы раскрыть меню **Дата**, нажмите **Д**. Меню появится на экране.

6. Чтобы показать сегодняшнее число, нажмите **Ч**. В поле надписи появится дата.

Добавление элементов управления OpenFileDialog и ColorDialog

1. Добавьте в область компонентов, в которой уже находится объект главного меню MainMenu1, два элемента управления диалоговых окон. Элемент управления

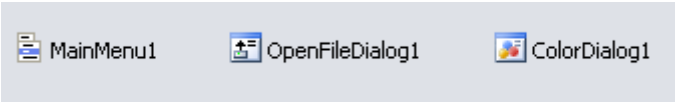
OpenFileDialog потребуется, чтобы открывать файлы с точечными изображениями, а элемент управления ColorDialog позволит изменять цвет для показа даты. В процессе разработки элементы управления диалоговых окон помещаются в области компонентов, а не на поле формы, так как во время выполнения они на форме не появляются.

2. В области элементов на закладке Windows Forms выберите элемент управления OpenFileDialog, а затем щелкните мышкой в области компонентов, где уже есть объект MainMenu1.

Совет. Если вы не видите OpenFileDialog в области элементов, то он может быть за пределами видимости. Чтобы промотать список в области элементов, щелкните на нижней стрелке прокрутки, находящейся рядом с закладкой Clipboard Ring (Буфер обмена).

В области компонентов появится объект диалогового окна для открытия файла.

3. В области элементов на закладке Windows Forms выберите элемент управления ColorDialog, а затем щелкните на области компонентов, расположенной ниже поля формы. Теперь область компонентов выглядит так.



Как и объект главного меню, объекты с диалогами открытия файла и выбора цвета можно настроить, задав их свойства.

Теперь с помощью элемента управления PictureBox создайте область для показа изображений. Этот объект показывает в поле формы содержимое из файлов изображений. На этот раз мы выведем на поле формы картинку, используя диалоговое окно для открытия файла.

Добавление объекта области показа изображения

1. В области элементов выберите элемент управления PictureBox.

2. В поле формы ниже надписи нарисуйте объект области показа изображений, и в окне Свойства для свойства SizeMode этого объекта выберите значение StretchImage.

Теперь с помощью Конструктора меню добавьте в вашу программу меню Файл.

Добавление меню Файл

1. В поле формы щелкните на меню **Дата**, затем на ячейке Прототип для текста, расположенной справа от этого меню. Теперь нужно добавить в программу меню **Файл**, в котором будут команды **Открыть**, **Закрыть** и **Выход**.

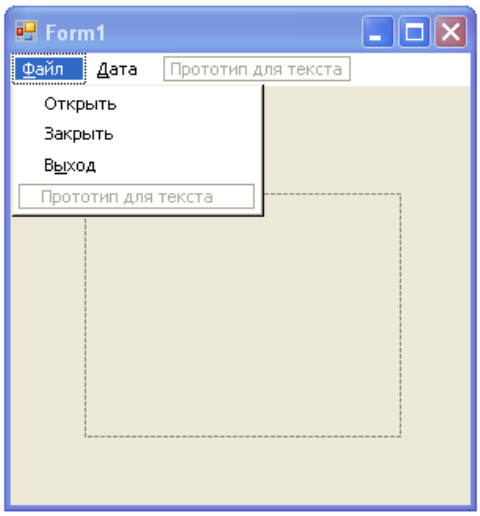
2. Чтобы создать меню **Файл** с буквой **"Ф"** в качестве клавиши доступа, введите **&Файл**.

3. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду Открыть ··· с буквой **"О"** в качестве клавиши доступа, введите **&Открыть** ···. Команда Открыть будет использоваться для загрузки точечных изображений. Так как эта команда должна будет открывать диалоговое окно, добавьте к ее имени многоточие.

4. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Закрыть** с буквой **"З"** в качестве клавиши доступа, введите **&Закрыть**. Команда **Закрыть** будет использоваться в программе для закрытия файла с изображением.

5. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Выход** с буквой **"ы"** в качестве клавиши доступа, введите **В&ыход**. Команда **Выход** будет использоваться для закрытия программы. Обратите внимание, что в этом случае в качестве клавиши доступа для команды Выход была использована третья буква, как это сделано во многих приложениях для Windows.

6. Чтобы передвинуть меню **Файл** на первое место, просто перетащите его на меню **Дата**. В Конструкторе меню целые меню можно перемещать точно так же, как и отдельные команды внутри меню. Имеет смысл сделать меню **Файл** первым меню программы. Ваша форма должна выглядеть примерно так.



В следующем упражнении мы отключим команду **Закрыть** в меню **Файл**. (Команда **Закрыть** может использоваться только после того, как файл уже был открыт в программе). Далее в этой лекции мы добавим в процедуру обработки события команды **Открыть** оператор, который в нужный момент включает команду **Закрыть**.

Отключение команды Закрыть

1. В меню **Файл** программы Menu выберите команду **Закрыть**.
2. В окне Свойства для свойства Enabled объекта MenuItem7 выберите значение False.

Теперь, чтобы продемонстрировать, как работает диалоговое окно выбора цвета, необходимо добавить в меню **Дата** команду **Цвет текста**. Диалоговое окно выбора цвета с помощью свойства Color возвращает в программу вновь выбранный цвет. Это свойство будет использоваться для изменения цвета текста в объекте Label1.

Добавление команды Цвет текста в меню Дата

1. Выберите меню **Дата**, а затем щелкните на нижней ячейке Прототип для текста.
2. Чтобы добавить в это меню команду **Цвет текста** с клавишей "Ц" в качестве клавиши доступа, введите **&Цвет текста ...**. Команда будет добавлена в меню **Дата**. Эта команда заканчивается многоточием, которое указывает, что при ее выборе откроется диалоговое окно.

Редактирование процедуры обработки событий команды Открыть

1. В меню **Файл** нашей формы дважды щелкните мышью на команде **Открыть**. В редакторе кода появится процедура обработки событий mnuOpenItem_Click.
2. Между операторами Private Sub и End Sub добавьте следующие ниже операторы. Убедитесь, что вы набрали каждую строку в точности так, как они здесь напечатаны, а после набора последней строки нажмите на клавиатуре стрелку вниз.

```
OpenFileDialog1.Filter = "Точечный рисунок (*.bmp)|*.bmp"
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
    PictureBox1.Image = System.Drawing.Image.FromFile(OpenFileDialog1.FileName)
    MenuItem6.Enabled = True
End If
```

Первые три оператора в этой процедуре обработки события ссылаются на три свойства объекта диалога открытия файла. В первом операторе свойство Filter используется для определения списка допустимых файлов. В нашем случае этот список содержит только один элемент: *.bmp. Это достаточно важно для диалогового окна Открыть, так как объект показа изображения поддерживает шесть типов файлов: точечные изображения (файлы .bmp), метафайлы Windows (файлы .emf и .wmf), значки (файлы .ico), формат Joint Photographic Experts Group (файлы .jpg и .jpeg), формат Portable Network Graphics (файлы .png) и формат Graphics Interchange Format (файлы .gif). Попытка показать в объекте изображения файл .txt приведет к возникновению ошибки в момент выполнения. Чтобы

добавить в список Filter дополнительные элементы, между ними нужно ввести символ (|). Например, при следующем значении фильтра

OpenFileDialog1.Filter = "Точечный рисунок (*.bmp)|*.bmp|Метафайл Windows (*.wmf)|*.wmf"

в диалоговом окне **Открыть** будут показаны как файлы с точечными изображениями, так и метафайлы Windows.

Второй оператор в процедуре обработки события показывает в программе диалоговое окно **Открыть**. Метод ShowDialog - это новый метод Visual Basic .NET, он похож на метод Show из Visual Basic 6, но может использоваться для любой формы Windows Forms. Метод ShowDialog возвращает результат с именем DialogResult, который указывает, какую кнопку диалогового окна нажал пользователь. Чтобы определить, щелкнул ли пользователь на кнопке **Открыть**, используется оператор If...Then, который проверяет, равно ли возвращенное значение DialogResult.OK. Если оно равно этому значению, то в свойстве FileName диалога открытия должен храниться путь к существующему файлу на диске.

В третьем операторе используется имя файла, которое было выбрано в диалоговом окне. Когда пользователь выбирает диск, папку и имя файла, а затем нажимает кнопку **Открыть**, полный путь передается в программу через свойство OpenFileDialog1.FileName. Затем используется метод System.Drawing.Image.FromFile, который копирует указанное точечное изображение в объект показа изображений. (Этот оператор разбит на две строки при помощи символа продолжения строки, так как он получился очень длинным.)

В четвертом операторе активируется команда **Закрыть** из меню **Файл**. Теперь, когда файл в программе был открыт, команда **Закрыть** должна быть доступна, чтобы пользователи могли закрыть этот файл.

Теперь введите код программы для процедуры обработки события mnuCloseItem_Click, которая выполняется при выборе команды **Закрыть** из меню **Файл**.

Редактирование процедуры события команды Закрыть

1. Снова откройте форму, а затем дважды щелкните мышью на команде **Закрыть** из меню **Файл**. В редакторе кода появится процедура обработки события для команды **Закрыть**.
2. Между операторами Private Sub и End Sub добавьте следующие операторы программы:

```
PictureBox1.Image = Nothing
MenuItem6.Enabled = False
```

Первый оператор закрывает открытое точечное изображение, удаляя информацию, хранящуюся в свойстве Image. Ключевое слово Nothing используется здесь для того, чтобы отменить связь между текущим объектом точечного изображения и свойством Image, другими словами, Nothing задает для этого свойства нулевое значение, и изображение исчезает. (Далее в этой книге Nothing будет использоваться для сброса значений и других объектов и свойств.) Во втором операторе команду **Закрыть** из меню **Файл** отключается, так как открытых файлов больше нет. Использование этого оператора в программе равнозначно настройке свойства Enabled в окне Properties (Свойства).

Редактирование процедуры событий команды Выход

1. Снова перейдите в конструктор формы, а затем дважды щелкните мышью на команде **Выход** из меню **Файл**. В редакторе кода появится процедура обработки событий для команды **Выход**.
2. Между операторами Private Sub и End Sub добавьте следующий оператор программы

```
End
```

Оператор End останавливает программу, когда пользователь заканчивает с ней работать.

Редактирование процедуры обработки событий команды Цвет текста

1. Перейдите в конструктор формы, а затем дважды щелкните мышью на новой команде **Цвет текста** из меню **Дата**. В редакторе кода появится процедура обработки событий для команды **Цвет текста**.

2. Добавьте в нее следующие операторы

```
ColorDialog1.ShowDialog()
Label1.ForeColor = ColorDialog1.Color
```

Совет. Диалоговое окно Color (Цвет) может использоваться для установки цвета любого элемента пользовательского интерфейса, который работает с цветом. Например, это может быть фоновый цвет формы, цвета геометрических фигур на форме, основной или фоновые цвет для различных объектов.

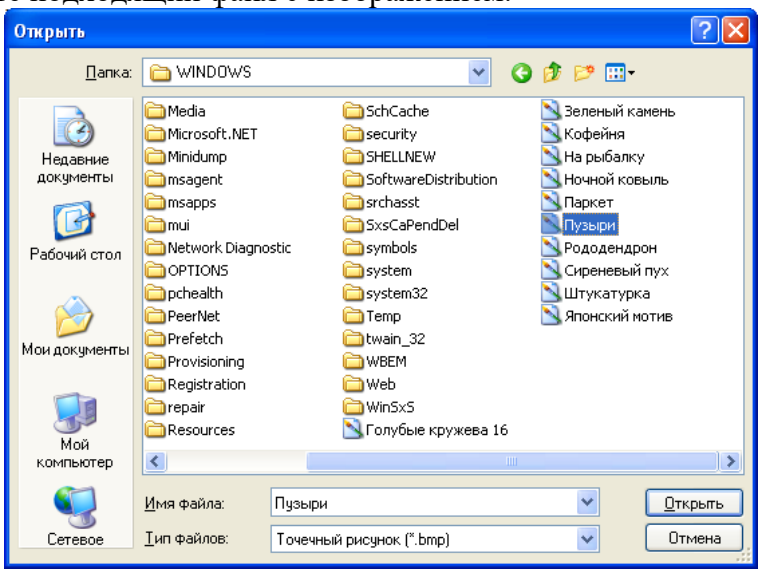
В первом операторе используется метод ShowDialog, который открывает диалоговое окно выбора цвета. Метод ShowDialog используется для открытия любой формы как диалогового окна, в том числе и формы, созданные стандартными диалоговыми окнами, которые предоставляет Visual Studio. Во втором операторе цвет, выбранный пользователем в диалоговом окне, присваивается свойству ForeColor объекта Label1. Label1 - это поле надписи, которое используется в форме для показа текущих даты и времени. При работе программы вы будете выбирать в диалоговом окне цвет, и он будет использован для текста в этой надписи.

Теперь давайте запустим программу Menu и поэкспериментируем с созданными нами меню и диалоговыми окнами.

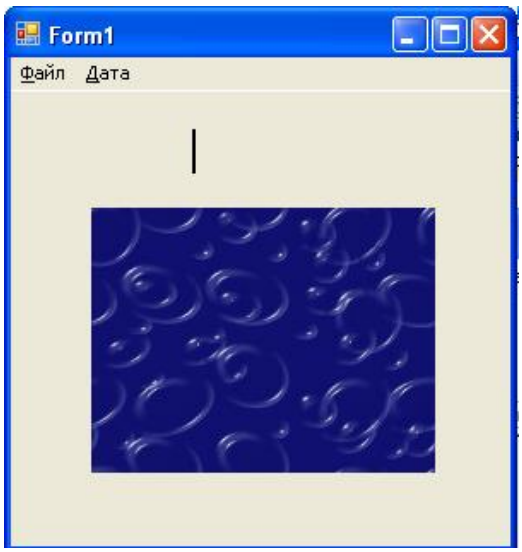
Запуск программы Menu

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа запустится, и в строке меню появятся меню **Файл** и **Дата**.

2. В меню **Файл** запущенной программы выберите **Открыть**. Появится диалоговое окно **Открыть**. Обратите внимание на текст Точечный рисунок (*.bmp) в поле Тип файлов. Выберите подходящий файл с изображением.

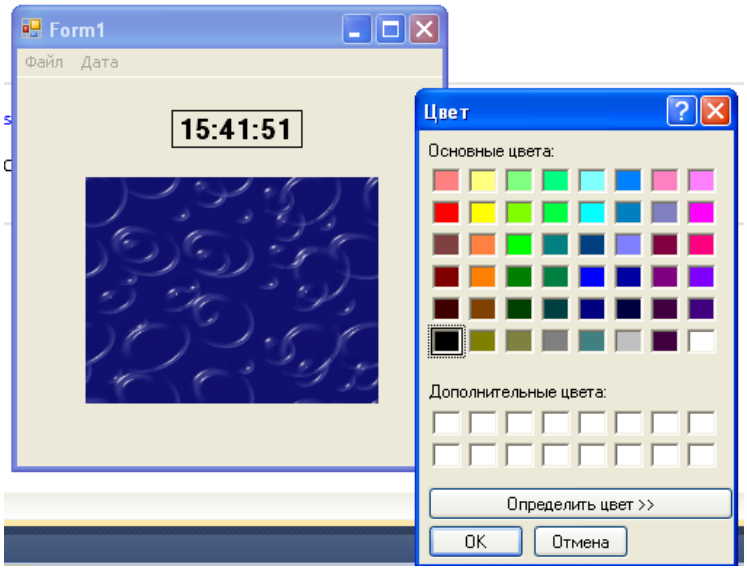


3. Выберите один из файлов .bmp, а затем нажмите кнопку **Открыть**. Изображение из этого файла появится в поле показа изображений. Форма будет выглядеть примерно так.



4. В меню **Дата** выберите строку **Время**. В поле надписи появится текущее время.

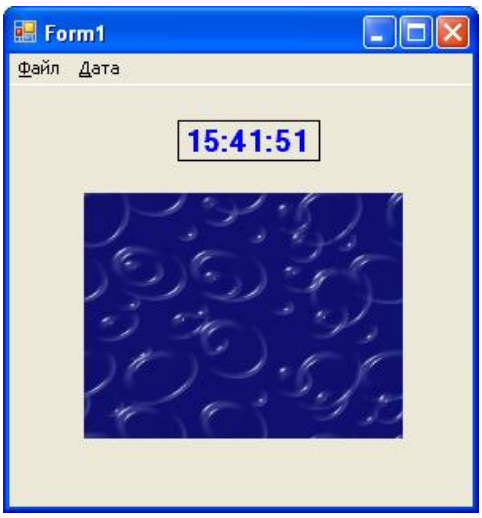
5. В меню **Дата** выберите команду **Цвет текста**. Появится диалоговое окно Color (Цвет), показанное ниже.



Диалоговое окно Color (Цвет) содержит элементы, которые позволяют изменить цвет надписи в программе. По умолчанию в этом окне выбран текущий цвет - черный.

6. Щелкните на синем поле, а затем на кнопке **OK**.

Диалоговое окно Color (Цвет) закроется, а цвет текста в надписи изменится на синий.



7. В меню **Дата** щелкните на команде **День**.
8. Откройте меню **Файл**.

Обратите внимание, что команда **Закрыть** включена. (Мы включили ее с помощью свойства Enabled = True.)

9. Нажмите **З** (клавишу доступа для **Закрыть**, при необходимости переключите клавиатуру в русский регистр), чтобы закрыть изображение.

Файл закроется, и точечное изображение Windows исчезнет (это сработало ключевое слово Nothing.)

10. Откройте меню **Файл**. Теперь команда **Закрыть** отключена, так как в области вывода изображений картинки нет.
11. Выберите команду **Выход**. Программа закроется, и появится среда разработки Visual Studio.

Следующий шаг: привязка сочетаний клавиш к пунктам меню

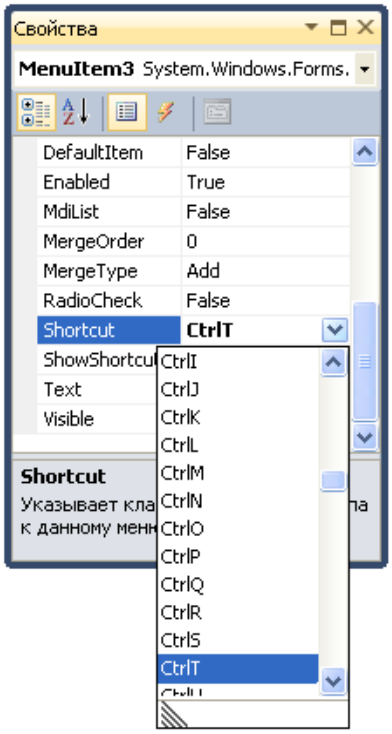
Определение сочетания клавиш для меню Дата

1. В меню **Дата** выберите команду **Время**.

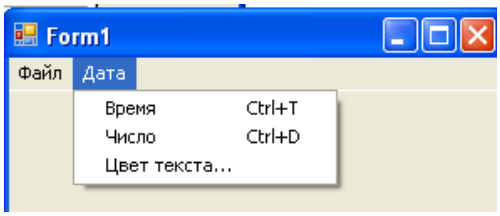
Сочетание клавиш определяется в свойстве Shortcut выбранной команды, в окне Свойства.

2. Откройте окно Свойства, выберите свойство Shortcut, затем нажмите стрелку раскрывающегося списка в столбце справа, прокрутите этот список и выберите CtrlT. Окно Свойства будет выглядеть так, как показано на рисунке справа.

Совет. Обычно Visual Basic при выполнении программы показывает в меню сочетания клавиш, чтобы подсказать пользователям, какие клавиши следует нажимать. Чтобы скрыть от пользователя эти комбинации клавиш (например, если для них не хватает места), задайте для свойства ShowShortcut значение False. Сочетания клавиш по-прежнему будут работать, но подсказок в меню пользователи не увидят.



3. Выберите команду **Число**, а затем измените ее свойство Shortcut на значение CtrlD. Теперь давайте запустим программу и попробуем использовать сочетания клавиш.
4. На стандартной панели инструментов нажмите кнопку Начать.
5. Нажмите (Ctrl)+(T), чтобы выполнить команду **Время**. В программе появится текущее время.
6. Нажмите (Ctrl)+(D), чтобы выполнить команду **Число**. В программе появится текущая дата.



7. Щелкните на меню **Дата**. Рядом с командами **Время** и **Число** будут показаны сочетания клавиш. Visual Basic дописывает эти комбинации клавиш, когда они определены с помощью свойства Shortcut.

Контрольные вопросы:

1. Назовите элемент управления для добавления меню.
2. Рекомендации по созданию меню.
3. Добавление клавиш доступа. Как визуально отличаются пункты меню, для которых назначена клавиша доступа.
4. Назначение сочетаний клавиш для пунктов меню, свойство.
5. Перечислить элементы управления для создания диалоговых окон.
6. Метод, используемый для отображения диалогового окна.

Разработка функциональной схемы работы приложения

Лабораторная работа № 33-34

Тема: «Разработка оконного приложения с несколькими формами»

Цель:*научиться создавать оконное приложение с несколькими формами.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Добавление в программу новых форм

Во многих случаях для взаимодействия с пользователем одной формы вполне достаточно. Но если вам требуется обмениваться с пользователем большим объемом информации, Visual Basic позволяет добавить в программу дополнительные формы. Каждая новая формы рассматривается как объект, который наследует свои свойства от класса System.Windows.Forms.Form. Первая форма программы называется Form1.vb. Последующие формы называются Form2.vb, Form3.vb и т.д. Следующая таблица содержит список способов практического применения дополнительных форм в вашей программе.

Каждая новая форма имеет уникальное имя и свой собственный набор объектов, свойств, методов и процедур событий.

Форма	Описание
"Титульная" форма	Форма, которая при запуске программы отображает сообщение приветствия, графику или информацию об авторских правах.
Инструкции программы	Форма, которая отображает информацию и подсказки о том, как работать с программой.
Диалоговые окна	Дополнительные диалоговые окна, которые принимают данные и отображают результаты работы программы.
Содержимое документов	Форма, которая отображает содержимое одного или более файлов и изображений, используемых в программе.

Как используются формы

Visual Basic позволяет пользоваться формами достаточно гибко. Вы можете сделать все формы программы видимыми одновременно, а можете загружать и выгружать формы по мере их необходимости. Если вы выводите сразу несколько форм, то можете разрешить пользователю переключаться между ними, а можете управлять порядком, в котором эти формы используются. Форма, которой при ее отображении на экране передается фокус ввода, называется **диалоговым окном.** Диалоговые окна (называемые в Visual Basic 6 **модальными формами**) сохраняют фокус ввода до тех пор, пока пользователь не нажмет на **OK**, на Cancel (**Отмена**) или не закроет ее другим способом. Чтобы в Visual Basic .NET отобразить существующую форму как диалоговое окно, откройте ее с помощью метода `ShowDialog`.

Если вы хотите отобразить форму, на которую пользователь сможет переключиться, используйте вместо метода `ShowDialog` метод `Show`. В Visual Basic 6 формы, которые могли терять фокус ввода, назывались немодальными формами, и вы по-прежнему можете услышать такое их название. Большинство приложений для Microsoft Windows использует для отображения информации обычные немодальные формы, так как они дают

пользователю большую гибкость, и при создании новой формы в Microsoft Visual Studio этот стиль является стилем по умолчанию. Так как формы - это просто члены класса `System.Windows.Forms.Form`, вы также можете создать и отобразить формы с помощью кода программы.

Чтобы узнать, как в приложении на Visual Basic определяется форма по умолчанию, изучите код в разделе, сгенерированном Windows Form Designer, расположенном в верхней части каждой новой формы.

Использование свойства DialogResult в вызывающей форме

Вы можете эффективно использовать в программе на Visual Basic свойство `DialogResult`, которое присваивается в диалоговом окне. Более сложное диалоговое окно может предлагать пользователю дополнительные кнопки - Cancel (**Отмена**), Yes (Да), No (Нет), Abort (Прервать) и так далее. Каждая кнопка диалогового окна может быть ассоциирована со своим типом действия в главной программе. А в каждой процедуре события кнопки диалогового окна можно присваивать свойству `DialogResult` формы значение, соответствующее названию кнопки, как это сделано в следующем операторе программы:

☞Задание на лабораторную работу.

Создать приложение для Windows "Семерка" (Lucky Seven). Эта программа имитирует игровой автомат со "счастливыми" Для отображения справочной информации о программе «Семерка» использовать вторую форму. Добавьте вторую форму с помощью команды Вторая форма должна отображать файл Readme.txt, который должен содержать справочную информацию.

Порядок выполнения работы

9. Графический интерфейс "Семерки" имеет две кнопки, три поля для показа "счастливых" чисел, цифровую фотографию, символизирующую выигрыши, и текст "Счастливая семерка". Чтобы сконструировать этот интерфейс, нужно создать семь объектов на форме Семерка, а затем изменить некоторые свойства в каждом из них. Затем нужно добавить код программы для кнопок Крутить и Выход, который обрабатывает нажатия пользователя на этих кнопках и генерирует случайные числа. Чтобы создать эту программу с самого начала, необходимо выполнить в Visual Basic три основных шага по разработке программы: создать интерфейс пользователя, настроить свойства и написать код программы. В таблице описан этот процесс для "Семерки".

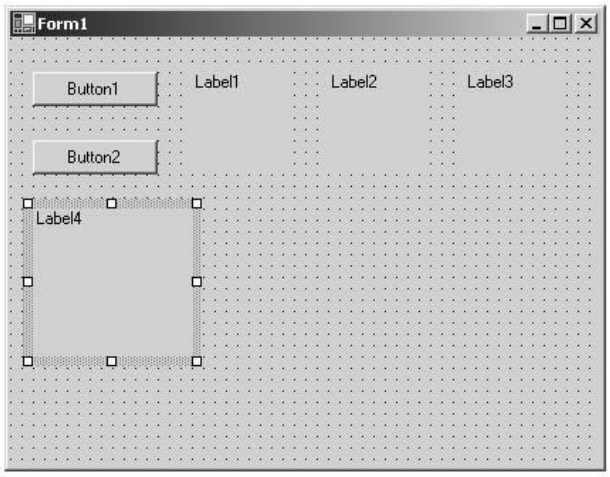
Таблица 2.1.

Шаг программирования	**Число элементов**
Создать графический интерфейс пользователя	7 объектов
Настроить свойства	12 свойств
Написать код программы	2 объекта

Создание пользовательского интерфейса

1. Создайте новый проект командой Создать проект (NewProject) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).
2. Выберите элемент Приложение WindowsForms и нажмите кнопку ОК.
3. В области элементов выберите элемент управления Button и добавьте на форму две кнопки.
4. Добавьте надписи, которые используются в программе для показа чисел. В области элементов выберите элемент управления Label. И создайте на форме четыре надписи с именами Label1, Label2, Label3 и Label4.

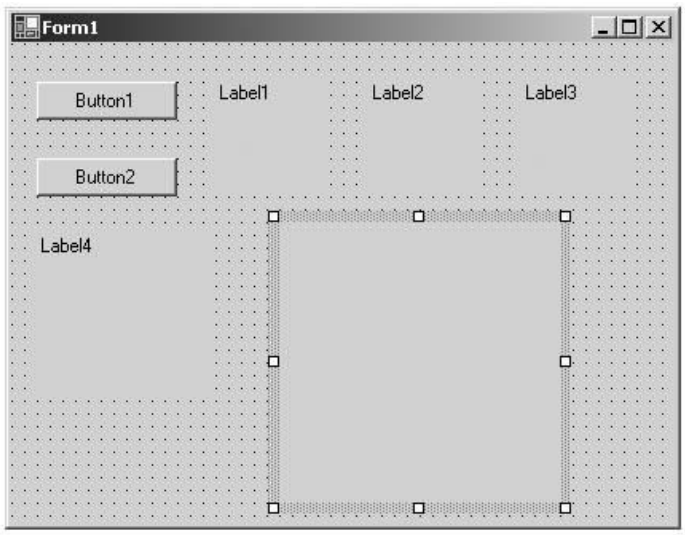
Ваши надписи будут выглядеть примерно так, как показано на иллюстрации ниже. (Если надписи выглядят не вполне правильно, их размер можно изменить.)



Добавление изображения

1. Теперь добавьте на форму рамку для рисунка, чтобы показать графический приз, когда пользователь выбросит "7".В области элементов выберите элемент управления PictureBox.

2. Нарисуйте большой прямоугольник прямо под тремя надписями для цифр. В результате поле рисунка должна выглядеть так.



Этот объект в программе будет называться PictureBox1, и это имя в дальнейшем используется при написании кода программы.

Теперь нужно настроить интерфейс, задав несколько свойств.

Настройка свойств кнопки

1. Щелкните на первой кнопке Button1.Текущее значение свойства Textизмените на слово Крутить и нажмите клавишу (Enter).

2. Измените свойство Text для второй кнопки так, чтобы там было написано "Выход".

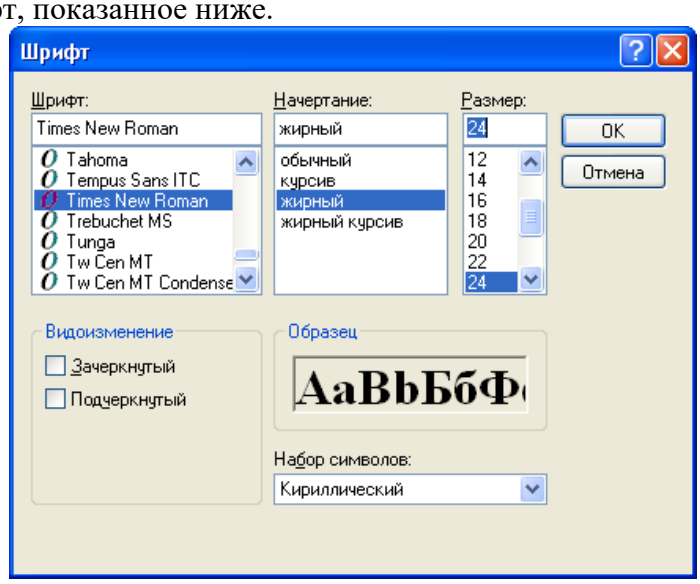
Настройка свойств надписей для чисел

1. Выберите первую надпись, предназначенную для чисел (Label1), а затем, удерживая нажатой клавишу (Shift), щелкните на второй и третьей надписях. Вокруг каждой из надписей, по которой вы щелкаете, появляются прямоугольники выделения и маркеры изменения размера. Выделите все три надписи и отпустите клавишу (Shift). Теперь нужно изменить свойства TextAlign, BorderStyle и Font так, чтобы числа, которые будут появляться в этих надписях, находились в центре, были обведены рамкой и показывались одним и тем же шрифтом одинакового размера. Все эти свойства перечислены в окне Свойства в категории Внешний вид.

2. В окне Свойства выберите TextAlign, а затем нажмите справа на стрелку раскрывающегося списка. Появляется схема, показывающая варианты центровки текста в пределах поля надписи. Выберите вариант форматирования по центру MiddleCenter.

3. Теперь измените свойство BorderStyle. Выберите его в списке и нажмите справу стрелку раскрывающегося списка. Выберите FixedSingle, чтобы добавить вокруг каждой надписи тонкую рамку.

4. Теперь измените шрифт для надписей, изменив параметры свойства Font. Выберите его в окне Свойства, а затем нажмите справа кнопку многоточия. Появится диалоговое окно Шрифт, показанное ниже.



5. Измените шрифт на Times New Roman, стиль шрифта - на жирный, а его размер - на 24, а затем нажмите OK. Шрифт, стиль и размер текста на надписях изменятся на новые.

6. Теперь удалите текст для этих трех надписей, чтобы при запуске программы эти поля были пустыми. Выбор шрифта для надписей сохранится, так как он является отдельным свойством. Чтобы это сделать, необходимо выбрать каждую из надписей по отдельности и с помощью клавиши Delete очистить свойство Text.

Установка свойств описательной надписи

1. Выберите объект четвертой надписи Label4.
2. В окне Свойства измените свойство Textна "Счастливая семерка".
3. Щелкните на свойстве Font, а затем на кнопке многоточия.
4. В диалоговом окне Шрифт измените шрифт на Arial, стиль шрифта - на жирный и размер - на 18. Нажмите OK, при этом шрифт надписи обновится.
5. Теперь измените цвет отображаемого текста. В окне Свойства выберите ForeColor и во втором столбце нажмите стрелку раскрывающегося списка. Выберите пурпурный цвет на закладке Польз.(Custom). Цвет текста в поле надписи изменится на пурпурный.

Свойства поля для изображения

Объект для показа изображения будет содержать картинку с человеком, платящим игроку деньги при выигрыше (то есть когда в числовых надписях формы есть хотя бы одна семерка). Чтобы задать размеры картинки, нужно изменить свойство SizeMode, а в свойстве Image следует указать имя JPEG-файла, который будет загружен в поле изображения. Следует также задать свойство Visible, которое определяет состояние изображения в начале работы программы.

Установка свойств поля изображения

1. На поле формы щелкните на объекте поля изображения.
2. В окне Свойства выберите свойство SizeMode, которое находится в категории Поведение. Нажмите на стрелке раскрывающегося списка и выберите StretchImage. Если

для SizeMode выбрано значение StretchImage, то прежде чем показать файл изображения, Visual Studio изменит его размер так, чтобы оно в точности совпадало с размерами поля.

3. В окне Свойства выберите свойство Image, а затем щелкните на кнопке многоточия во втором столбце. Появится диалоговое окно Открыть и выберите файл jpg с нужным изображением.

4. В окне Properties Свойства в категории Behavior (Поведение) выберите свойство Visible. Чтобы сделать изображение невидимым при запуске программы, выберите False.

Готовая форма должна выглядеть примерно так, как показано на рисунке ниже.

5. Дважды щелкните мышью на строке заголовка окна Properties (Свойства) , чтобы вернуть его в его закрепленное положение.

Написание кода

Таблицы значений свойств

Таблица 2.2.

Объект	**Свойство**	**Значение**
Button1	Text	"Крутить"
Button2	Text	"Выход"
Label1, Label2, Label3	BorderStyle	Fixed Single
	Font	Times New Roman, жирный, 24 пункта
	Text	(пусто)
	TextAlign	MiddleCenter
Label4	Text	"Счастливая семерка"
	Font	Arial, жирный, 18 пунктов
	ForeColor	Purple
PictureBox1	Image	"абсолютный адрес файла с изображением"
	SizeMode	StretchImage
	Visible	False

Работа в Редакторе кода

1. В поле формы дважды щелкните мышью на кнопке Выход. В центральном окне среды разработки Visual Studio появится редактор кодаМежду строками Private Sub и End Sub введите `End`

1. Дважды щелкните мышью на кнопке Крутить. Между строками Private Sub и End Sub введите следующие ниже строки программы:

```
PictureBox1.Visible = False      ' скрыть изображение
Label1.Text = CStr(Int(Rnd() * 10))      ' получить число
Label2.Text = CStr(Int(Rnd() * 10))
Label3.Text = CStr(Int(Rnd() * 10))
' если какой-либо из заголовков - это 7, то отобразить картинку и подать
' звуковой сигнал
If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text = "7") Then
    PictureBox1.Visible = True
    Beep()
End If
```

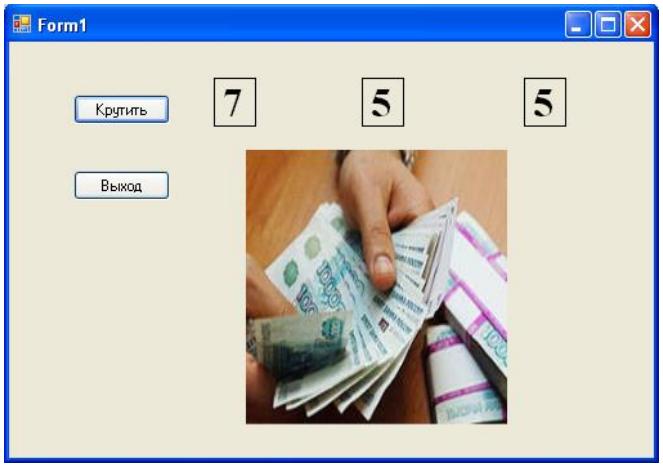
Функция Rnd в каждой строке возвращает случайное число в интервале между 0 и 1 (число с десятичной запятой и несколькими знаками после нее), а функция Int возвращает целую часть произведения этого случайного числа на 10. Это вычисление дает нам случайные числа в интервале между 0 и 9 - именно то, что нам требуется для нашего приложения игрового автомата.

Затем необходимо выполнить в нашем коде одно обязательное действие - скопировать эти случайные числа в три поля надписей на форме, но прежде, чем это сделать, эти числа должны быть преобразованы в текст с помощью функции CStr (convert to string - преобразовать в текст). Обратите внимание, как CStr, Int и Rnd объединены вместе в одно выражение в программе - они работают совместно и дают результат аналогично математической формуле. После вычисления и преобразования эти значения присваиваются свойствам Text первых трех надписей формы.

Последняя группа операторов программы проверяет, не равняется ли одно из случайных чисел "7". Если одно или несколько из этих чисел равны "7", то программа показывает средневековое изображение выплаты денег, а звуковой сигнал провозглашает о победе.

Запуск программы "Семерка"

1. Нажмите кнопкуНачать отладку(Start) на стандартной панели инструментов. Программа "Семерка" будет скомпилирована и запущена в среде программирования. Появится созданный вами графический интерфейс.
2. Нажмите кнопку Крутить. Программа найдет три случайных числа и покажет их в надписях на форме.



Так как в первом поле надписи выпала "7", появляется цифровое фото, изображающее выплату денег, и компьютер издает звуковой сигнал. Вы выиграли! Звук, который вы услышите, зависит от установок в окне Sounds and Devises (Звуки и аудиоустройства) в панели управления Windows - чтобы сделать звук в этой игре интереснее, измените звук, установленный по умолчанию, на что-нибудь более динамичное.

3. Нажмите кнопку Крутить еще несколько раз, наблюдая за результатом "вращений" в числовых полях.

Почти в половине ваших экспериментов вы будете выигрывать (реальные шансы - 2,8 раза из 10; просто вам поначалу везет). В дальнейшем можно будет усложнить игру, выводя цифровое фото только тогда, кода появятся две или три семерки, или подсчитывая текущую сумму выигрышей.

4. Когда вы закончите экспериментировать с вашим новым детищем, нажмите кнопку Выход. Программа остановится, и на экране снова появится среда программирования.

Совет. Если вы снова запустите программу, то заметите, что "Семерка" показывает ту же самую последовательность случайных чисел. Здесь нет никакой ошибки - функция Rnd в Visual Basic спроектирована так, чтобы сначала отображать повторяющуюся последовательность чисел, чтобы программа выдавала одни и те же предсказуемые результаты в целях отладки. Чтобы создать истинно "случайные" числа, нужно использовать функцию Randomize.

Следующий шаг: внесение дополнений в программу

Добавьте в программу "Семерка" специальный оператор с именем Randomize.

1. Теперь вы должны добавить в процедуру Form_Load - специальную процедуру, которая ассоциирована с формой и исполняется каждый раз при запуске программы - оператор Randomize.

2. Чтобы перейти к тексту процедуры Form_Load, дважды щелкните мышью на форме (но не на одном из ее объектов). В редакторе кода появится процедура Form_Load.

3. Введите `Randomize()` , а затем нажмите на клавишу со стрелкой вниз. Оператор Randomize добавлен в программу и будет исполняться каждый раз при ее запуске. Чтобы создать истинно случайную начальную точку для функции Rnd в процедуре Button1_Click, в Randomize используются системные часы. При использовании Randomize программа будет каждый раз выдавать новую последовательность, и числа не будут повторяться.

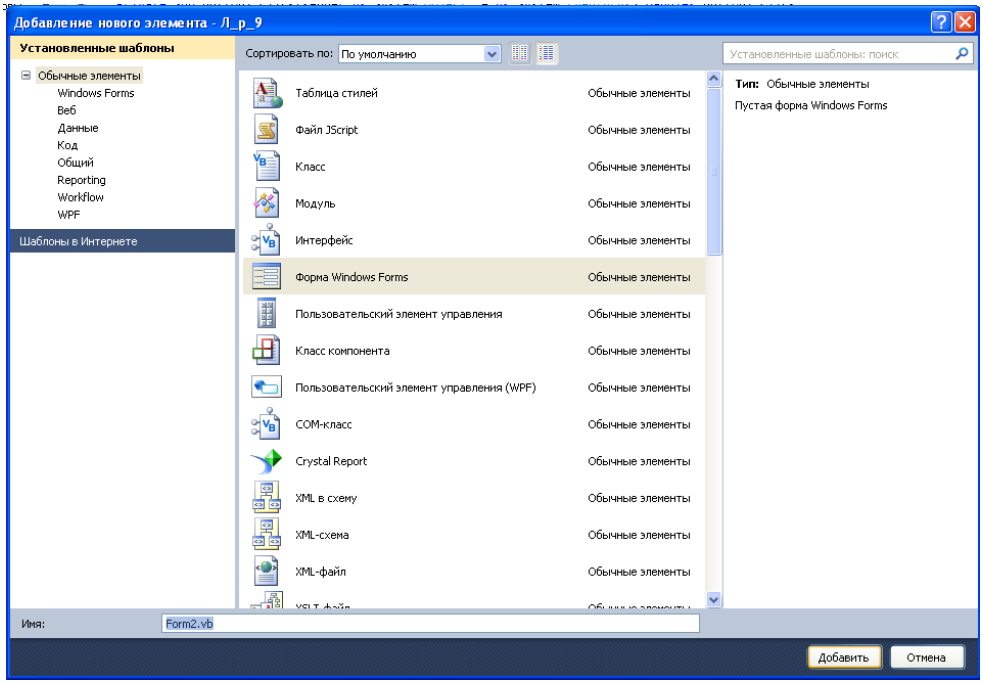
4. Запустите новую версию "Семерки".

Работа с несколькими формами

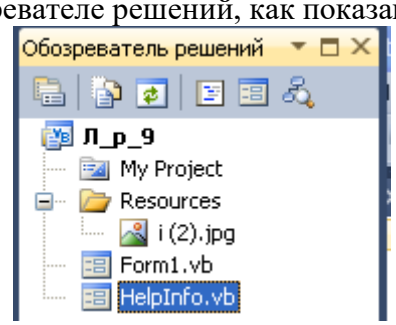
Использование второй формы для отображения справочной информации о программе «Семерка». Добавьте вторую форму с помощью команды Добавить форму Windows из меню Проект, а отображать эту форму на экране будете в коде программы с помощью метода `ShowDialog`. Вторая форма будет отображать файл Readme.txt, который должен содержать справочную информацию.

Добавление второй формы

1. Чтобы добавить в проект вторую форму, щелкните на команде Добавить форму Windows в меню Проект. Вы увидите такое диалоговое окно.



2. Введите в текстовом поле Имя (Name) имя файла HelpInfo.vb, а затем щелкните на Добавить (Open). В проект Семерка будет добавлена вторая форма с именем HelpInfo.vb. Она появится в Обозревателе решений, как показано здесь.



Теперь необходимо добавить в форму HelpInfo.vb несколько элементов управления.

3. Используйте элемент управления Label и нарисуйте в верхней части формы HelpInfo.vb метку. Сделайте длину этой метки равной ширине формы так, чтобы в ней уместился длинный текст.

4. Используйте элемент управления TextBox и создайте объект текстового поля.

5. Установите свойство Multiline этого объекта текстового поля на значение True, чтобы можно было изменить размер этого объекта.

6. Измените размер объекта текстового поля так, чтобы он заполнял почти все пространство формы.

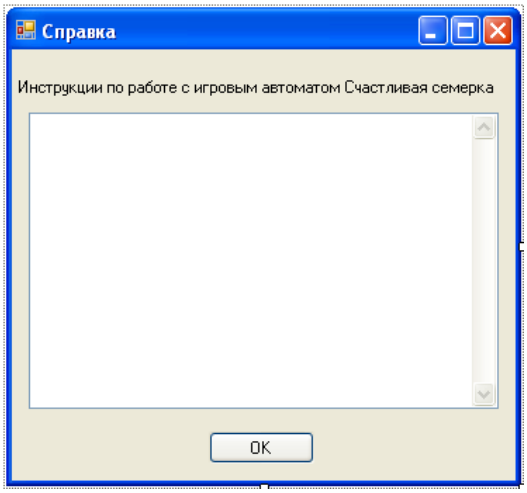
7. Используйте элемент управления Button и создайте в нижней части формы объект кнопки.

8. Установите для объектов формы HelpInfo.vb следующие свойства:

Объект	Свойство	Установка
Label1	Text	"Инструкции по работе с игровым автоматом Счастливая семерка"
TextBox1	Scrollbars	Vertical
	Text	пустой (empty)
Button1	Text	"OK"

HelpInfo.vb	Text	"Справка"

9. Форма HelpInfo.vb выглядит так.



10. Теперь необходимо ввести строку кода программы в процедуру события `Button1_Click` формы HelpInfo.vb.

11. Чтобы отобразить в Редакторе кода процедуру события Button1_Click, сделайте двойной щелчок мышью на кнопке **OK**.

12. Введите следующий оператор программы:

```
Me.DialogResult = DialogResult.OK
```

Форма HelpInfo.vb действует в этом проекте как диалоговое окно, так как она открывается в процедуре события `Form1` с помощью метода `ShowDialog`. После того, как пользователь изучил справочную информацию в этом диалоговом окне, кнопка **OK** закрывает форму, устанавливая свойство `DialogResult` текущей формы (Me) на значение `DialogResult.OK`. Это значение является константой Visual Basic, указывающей, что диалоговое окно закрыто, при этом вызывающую процедуру должно вернуться значение "OK". Более сложное диалоговое окно может с помощью параллельных процедур событий кнопок возвращать и другие значения, такие, как `DialogResult.Cancel`, `DialogResult.No`, `DialogResult.Yes` и `DialogResult.Abort`. Однако когда устанавливается свойство `DialogResult`, форма автоматически закрывается.

13. Прокрутите код программы в Редакторе кода в начало. Введите следующий оператор Imports:

```
Imports System.IO
```

Этот оператор создает в проекте ссылку на библиотеку классов, содержащую класс `StreamReader`. Класс `StreamReader` не связан непосредственно с определением или использованием дополнительных форм - просто используем его как быстрый способ добавления в новую используемую форму текстовой информации.

14. Снова отобразите форму HelpInfo.vb, а затем сделайте двойной щелчок мышью на фоне формы. В Редакторе кода появится процедура события HelpInfo_Load.

15. Введите следующие операторы программы:

```
Dim StreamToDisplay As StreamReader
StreamToDisplay = _
New StreamReader("Z:\Основы_программирования\ЛР_VB.Net\readme.txt")
TextBox1.Text = StreamToDisplay.ReadToEnd
StreamToDisplay.Close()
TextBox1.Select(0, 0)
```

Вместо того чтобы вводить содержимое справочного файла в свойство `Text` объекта текстового поля (что может занять много времени), для открытия, чтения и отображения в объекте текстового поля файла Readme.txt используйте класс `StreamReader`. Файл Readme.txt является документом поддержки продукта, содержащим информацию об использовании программы Счастливая семерка. Он содержит инструкции по работе, информацию по деинсталляции и контактную информацию. К сожалению, выводить русские символы с помощью этого метода затруднительно. Поэтому мы выведем содержимое англоязычного файла Readme.txt, который вы самостоятельно создадите**.**

`StreamReader` - это альтернативный по отношению к использованию функции Visual Basic FileOpen способ открывать текстовый файл, предоставляемый .NET Framework. Чтобы использовать `StreamReader`, необходимо включить в начало кода вашей формы библиотеку классов `System.IO`. Затем объявите переменную (`StreamToDisplay`) типа `StreamReader`, которая будет хранить содержимое текстового файла, и откройте этот текстовый файл, указав путь к нему. Наконец, с помощью метода `ReadToEnd`, который получает из файла весь текст, начиная с текущей позиции (начало текстового фала) и до конца этого файла, вы считываете содержимое текстового файла в переменную `StreamToDisplay`, и присваиваете ее свойству `Text` текстового поля. Оператор `StreamReader.Close` закрывает текстовый файл, а метод Select удаляет выделение из текста, размещенного в объекте текстового поля.

Вы закончили создание формы HelpInfo.vb. Теперь вы добавите объект кнопки и некий код в первую форму.

Отображение второй формы с помощью процедуры события

1. Щелкните в Обозревателе решений на Form1.vb, а затем щелкните на кнопке Просмотреть конструктор. В среде разработки появится форма Семерка. Теперь необходимо добавить в нижний правый угол формы кнопку **Справка**.
2. Используйте элемент управления Button и нарисуйте в нижнем правом углу формы небольшую кнопку.
3. Используйте окно Свойства, чтобы установить свойство `Text` кнопки на значение **Справка**. Ваша форма должна выглядеть примерно так.
4. Чтобы отобразить в Редакторе кода процедуру события Button3_Click, сделайте двойной щелчок мышью на кнопке **Справка**.
5. Введите следующие операторы программы:

```
Dim frmHelpDialog As New HelpInfo()
frmHelpDialog.ShowDialog()
```

Эти операторы демонстрируют, как объявлять и отображать в коде программы вторую форму. Visual Basic .NET требует, чтобы перед использованием второй формы явно объявили переменную типа формы. Добавив в проект форму HelpInfo.vb, вы создали класс с именем HelpInfo; теперь вы используете его для объявления с помощью оператора `Dim` переменной с именем `frmHelpDialog`.

Второй оператор программы использует переменную `frmHelpDialog`, открывая форму HelpInfo.vb как диалоговое окно с помощью метода `ShowDialog`. Также можно использовать для открытия этой формы метод `Show`, но в этом случае Visual Basic не рассматривал бы HelpInfo.vb как диалоговое окно; форма была бы немодальной, и пользователь мог бы переключаться между этой и главной формами по мере необходимости. В дополнение к этому свойство `DialogResult` в процедуре события `Button1_Click` формы HelpInfo.vb не закрывало бы форму HelpInfo.vb - вместо этого потребовался бы оператор программы Me.Close

При создании собственных проектов помните о различиях между модальными и немодальными формами. Между этими типами форм существуют заметные различия, и вы обнаружите, что каждый из них дает пользователю определенные выгоды.

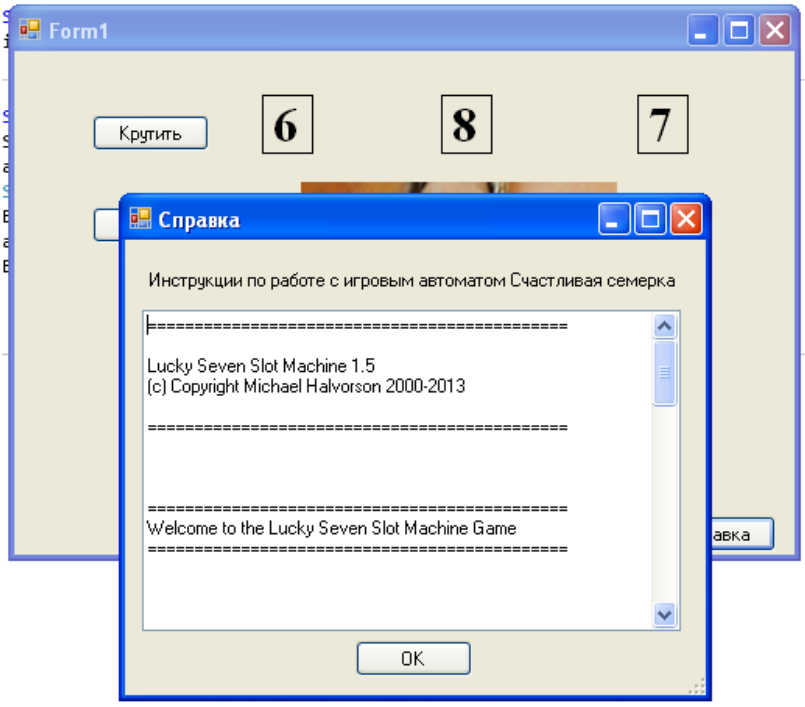
Теперь запустите программу, чтобы увидеть, как работает приложение, содержащее несколько форм.

Запуск программы

1. Щелкните на кнопке Начать отладку стандартной панели инструментов.

Появится главная форма Lucky Seven.

2. Чтобы немного поиграть в игру, щелкните семь или восемь раз на кнопке **Крутить**.
3. Щелкните на кнопке **Справка** первой формы. Visual Basic откроет вторую форму проекта (HelpInfo.vb) и отобразит в объекте текстового поля файл Readme.txt. Форма будет выглядеть примерно так.



4. Чтобы просмотреть весь файл Readme, используйте вертикальную полосу прокрутки.
5. Чтобы закрыть форму HelpInfo.vb, щелкните на кнопке **OK**. Форма закроется, и снова станет активной первая форма.
6. Еще несколько раз щелкните на кнопке **Крутить**, а затем снова щелкните на кнопке **Справка**. Снова появится форма HelpInfo.vb, работающая точно так же, как и в первый раз. Обратите внимание, что пока активна вторая форма, вы не можете

активизировать первую. Это происходит из-за того, что вторая форма является диалоговым окном или модальной формой, и прежде, чем вы сможете продолжить работу с программой, необходимо закрыть ее.

7. Щелкните на кнопке **OK**, а затем на кнопке **Закрыть** первой формы. Программа остановится, и вернется среда разработки.

Использование свойства DialogResult в вызывающей форме

В вызывающей процедуре события - другими словами, в процедуре события `Button3_Click` формы `Form1` - вы напишите дополнительный код программы, который будет определять, на какой кнопке диалогового окна щелкнул пользователь. Эта информация передается обратно в вызывающую процедуру через свойство `DialogResult`, которое содержится в имени переменной, использованном для объявления и создания экземпляра второй формы. Например, следующий код в `Form1` может быть использован для того, чтобы проверить, щелкнул ли пользователь на кнопке **OK**, Cancel или какой-либо еще кнопке диалогового окна. (Первые две строки у вас есть, они показывают имя переменной, которое необходимо использовать.)

```
Dim frmHelpDialog As New HelpInfo()
frmHelpDialog.ShowDialog()

If frmHelpDialog.DialogResult = DialogResult.OK Then
    MsgBox("Пользовательнажал OK")
ElseIf frmHelpDialog.DialogResult = DialogResult.Cancel Then
    MsgBox("пользователь нажал Отмена")
Else
    MsgBox("Была нажата другая кнопка")
End If
```

Подобные процедуры событий, которые объявляют переменную, открывают диалоговое окно и обрабатывают выбор, сделанный в этом диалоговом окне, позволяют добавлять в программы любое количество форм и создавать интерфейс пользователя, который выглядит профессионально, является гибким и дружественным к пользователю.

Контрольные вопросы:

1. Назвать класс , свойства которого наследует каждая форма.
2. Характеристики модальных и немодальных окон.
3. Назначение функций `CStr, Int и Rnd`.
4. Назначение и различия методов `Show и ShowDialog.`
5. Назначение оператора `Imports System.IO` в программе.
6. Что нам дает использование класса `StreamReader.`
7. Использование свойства DialogResult в вызывающей форме

Разработка игрового приложения

Создание процедур обработки событий. Компиляция и запуск приложения (4 часа)

Лабораторная работа № 38-39

*Тема: «*Разработка интерфейса приложения*»*

Цель: *научиться создавать многооконные приложения (MDI-приложения).*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Что такое MDI?

MDI-форма, которую также называют родительской, представляет собой контейнер для дочерних форм приложения. MDI-форма имеет ряд характеристик:

- Создаваемое приложение может иметь только одну MDI-форму.
- На панели задач Windows представляется лишь исходное MDI-окно, а все остальные дочерние окна на панели задач не представляются.
- При минимизации MDI-окна, минимизируются и все дочерние окна. При восстановлении - восстанавливаются.

MDI расшифровывается как **multiple document interface** (многодокументный интерфейс). В приложениях с MDI, в основном (родительском) окне можно окрыть более одного дочернего окна. Данная возможность обычно используется в электронных таблицах или текстовых редакторах.

В прежних версиях VB при программировании приложений с интерфейсом **MDI** (**Multiple Document Interface**) родительская форма MDI выбиралась на стадии конструирования. В .NET эта задача решается иначе – свойству **IsMdiContainer** формы задается значение True. Программист создает дочерние формы MDI на стадии конструирования или выполнения, а затем заносит в их свойство **Mdi Parent** ссылку на форму со свойством **IsMdiContainer**, равным True.

Таким образом, в программах VB.NET можно сделать то, что было практически нереально в предыдущих версиях VB, – изменять связи MDI во время работы программы. Кроме того, приложение может содержать несколько родительских форм MDI; в VB6 такая возможность не поддерживалась.

Рассмотрим пример. Создайте приложение со следующей процедурой **Forml_Load**:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Me.Text = "I'm an MDI Parent"

    Me.IsMdiContainer = True

    Dim MyChild As New System.Windows.Forms.Form()

    MyChild.MdiParent = Me

    MyChild.Show()

    MyChild.Text = "MDI Child"
End Sub
```

Конечно, это весьма жалкое подобие приложения MDI. В приложениях MDI обычно присутствует меню **Window**, позволяющее расположить открытые дочерние окна "черепицей" или "мозаикой", а также активизировать любое дочернее окно.

☞**Задание на лабораторную работу.**

Создать программу, позволяющую просматривать графические файлы, таким образом, чтобы каждый выбранный графический файл открывался в новом дочернем окне.

Для этого нам потребуются следующие компоненты:

- компонент MainMenu, который мы будем использовать для открытия диалоговой панели для выбора файлов;
- компонент OpenFileDialog, который будет использоваться для выбора графического файла;
- компонент PictureBox, в котором мы будем отображать содержимое графического файла.

Создание пользовательского интерфейса

5. Создайте новый проект командой Создать проект (NewProject) из меню Файл (File) (порядок создания нового проекта подробно описан в лабораторной работе № 1).
6. Выберите элемент Приложение WindowsForms и нажмите кнопку OK.
7. Добавьте на форму компоненты MainMenu, OpenFileDialog, PictureBox.

Настройка свойств

1. Задайте для формы свойство Text равным Graphics View.
2. Далее значение свойства StartPosition измените на CenterScreen — в результате наше приложение всегда будет отображаться в середине экрана.
3. Если вы хотите, чтобы размер окна нельзя было изменить, установите значение FormBorderStyle равным FixedSingle или Fixed3D. В противном случае при изменении размера окна нарушится расположение элементов внутри него.
4. Для элемента PictureBox установите трехмерную рамку (BorderStyle = Fixed3D), а также измените свойство Anchor — с его помощью вы укажете, как должны изменяться размеры элемента в зависимости от размеров окна.
5. С помощью компонента MainMenu создайте элемент File и два подэлемента — Open и Exit. Для этого воспользуйтесь редактором меню, предоставляемым Visual Studio .NET и вызываемым при щелчке мышью по компоненту MainMenu.

Написание кода

После того как меню готово, необходимо написать код, который будет выполняться при выборе той или иной команды. Для этого дважды щелкните мышью по соответствующей команде и попадете в редактор кода.

1. Для команды File | Open напишите следующий код:

```
Dim Bmp As Bitmap

If OpenFileDialog1.ShowDialog() = DialogResult.OK Then

    PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    Bmp = New Bitmap(OpenFileDialog1.FileName)
    PictureBox1.Image = CType(Bmp, Image)

End If
```

В первой строке мы создаем переменную типа Bitmap, в которую будем загружать содержимое выбранного нами графического файла. Затем мы вызываем метод ShowDialog объекта OpenFileDialog, чтобы отобразить на экране панель выбора файлов. Если файл выбран и нажата кнопка Open, мы выполняем следующие действия:

- задаем режим отображения графики в компоненте PictureBox — в нашем примере графика должна быть «растянута» во весь размер компонента;
- создаем новый экземпляр объекта Bitmap, в который загружаем содержимое выбранного нами файла;
- отображаем содержимое объекта Bitmap в компоненте PictureBox.

2. Для команды File | Exit мы напишите следующий код:

```
Application.Exit()
```

Выполнение этого кода приведет к завершению приложения и к закрытию его окна.

Создание MDI-приложений

После того как вы освоили создание меню, измените функциональность приложения таким образом, чтобы каждый выбранный графический файл открывался в новом дочернем окне, то есть превратите однооконное приложение в Multiple Document Interface (MDI)-приложение.

1. Создание MDI-приложения начинается с того, что изменяется свойство IsMDIContainer формы и присваивается ему значение True.
2. После этого мы используйте команду Проект/Добавить форму (Project | Add Windows Form) для добавления к проекту еще одной формы, которая будет служить в качестве дочерней формы. В диалоговой панели Добавить новый элемент (Add New Item) выберите элемент Windows Form и нажимите кнопку Открыть (Open).
3. Перенесите компонент PictureBox из главной формы приложения в дочернюю форму. Для этого следует выбрать компонент PictureBox, нажать клавишу Ctrl-X, затем перейти на дочернюю форму и нажать клавишу Ctrl-V.
4. Ваше MDI-приложение практически готово — осталось лишь изменить код, выполняющийся при выборе команды File|Open:

```
Dim Bmp As Bitmap
Dim Child As New Form2()

Child.MdiParent = Me

If OpenFileDialog1.ShowDialog() = DialogResult.OK Then

    Child.PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    Bmp = New Bitmap(OpenFileDialog1.FileName)
    Child.PictureBox1.Image = CType(Bmp, Image)
    Child.Show()

End If
```

Сначала мы создаем переменную Child, которая имеет тип Form2 и будет использоваться для создания дочерних форм. Затем указываем родительское окно дочерней формы — ключевое слово Me используется в Visual Basic для указания текущего класса. Поскольку компонент PictureBox теперь располагается в дочернем окне, мы изменяем

способ обращения к нему. После того как графика загружена, мы показываем нашу дочернюю форму на экране.

5. Для того чтобы сделать приложение более удобным, давайте показывать в заголовке дочернего окна полное имя отображаемого в нем файла. Для этого добавьте в код обработчика события следующую строку:

```
Child.Text = OpenFileDialog1.FileName
```

6. Если же мы вы хотите отобразить только имя файла, нам надо написать такой код:

```
Dim FI As FileInfo
FI = New FileInfo(OpenFileDialog1.FileName)
Child.Text = FI.Name
```

и добавить ссылку на пространство имен System.IO в самом начале кода нашей программы:

```
Imports System.IO
```

7. Теперь добавьте код, который позволит управлять дочерними окнами. Для этого создайте еще один пункт меню — Childs и добавьте в него следующие элементы: Tile Horizontal; Tile Vertical; Cascade; Arrange Icons.

8. Затем напишите код для каждой из команд:

Команда TileHorizontal

```
Me.LayoutMdi(MdiLayout.TileHorizontal)
```

Команда TileVertical

```
Me.LayoutMdi(MdiLayout.TileVertical)
```

Команда Cascade

```
Me.LayoutMdi(MdiLayout.Cascade)
```

Команда ArrangeIcons

```
Me.LayoutMdi(MdiLayout.ArrangeIcons)
```

9. Включите еще одну возможность — меню для переключения между окнами. Добавьте к главному меню элемент Window и измените значение его свойства MdiList на True.

10. Теперь используйте еще один компонент — ContextMenu, который позволит изменять способ отображения графики в компоненте PictureBox. Добавьте компонент ContextMenu к дочерней форме и создайте в нем следующие элементы: Normal; Center; Stretch; Autosize, которые повторяют все возможные значения свойства SizeMode компонента PictureBox.

11. Установитесвойство ContextMenu компонента PictureBox в ContextMenu1. Затем напишите код, который будет выполняться при вызове каждой из команд контекстного меню:

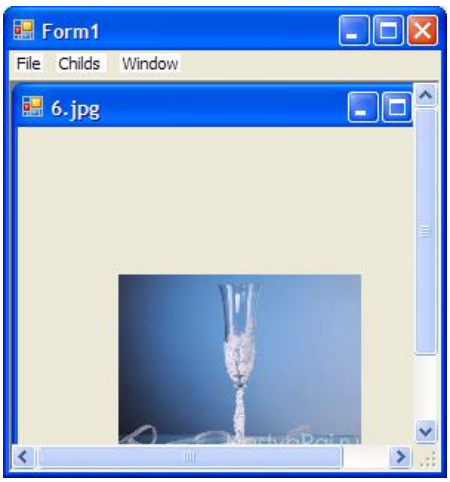
```
PictureBox1.SizeMode = PictureBoxSizeMode.Normal
PictureBox1.SizeMode = PictureBoxSizeMode.CenterImage
PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
PictureBox1.SizeMode = PictureBoxSizeMode.AutoSize
```

После этого надо удалить строку:

```
Child.PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
```

из обработчика события выбора команды File | Open в главном меню нашего приложения.

12. Запустите и протестируйте MDI-приложение



Контрольные вопросы:

1. Опишите форму MDI.
2. Какое свойство необходимо изменить для родительской формы MDI.
3. Назначение оператора `Imports System.IO` в программе.
4. Поясните следующий код

```
Dim Bmp As Bitmap
Dim Child As New Form2()

Child.MdiParent = Me

If OpenFileDialog1.ShowDialog() = DialogResult.OK Then

    Child.PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    Bmp = New Bitmap(OpenFileDialog1.FileName)
    Child.PictureBox1.Image = CType(Bmp, Image)
    Child.Show()

End If
```

5. .
6. Что дает использование строки кода :

```
Child.Text = OpenFileDialog1.FileName
```

Тестирование, отладка приложения

Лабораторная работа № 41

Тема: «Классы ООП: виды, назначение, свойства, методы, события. Объявления класса»

Цель: *научиться объявлять классы, создавать экземпляры классов в среде Visual Studio.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задание, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Процесс создания объектов (экземпляров класса) в Visual Basic .NET можно разделить на три этапа:

1. Создание файла классов.
2. Написание кодов в файле классов.
3. Объявление в программе переменных, которые представляют собой файлы классов.

Если переменная представляет файл классов, она называется объектом — термином, взятым из концепции объектно-ориентированного программирования.

Файл классов является, по сути, шаблоном, который определяет, как должны вести себя объекты. После создания файла классов вам еще нужно объявить переменную — чтобы создать таким образом сам объект.

Определение объекта

После того как файл классов будет создан, приступайте к написанию кодов, определяющих, что из себя представляет новый объект. Обычно такие коды (они также называются модулем класса) состоят из трех частей:

1. объявление переменных;
2. объявление свойств (данных);
3. методы, являющиеся процедурами BASIC, которые манипулируют переменными и свойствами.

```
Public Class Classl
Объявление переменных
Объявление свойств
Методы
End Class
```

Каждый объект инкапсулирует (отделяет) свои данные (свойства) от остальной части программы. Эти данные могут быть изменены лишь командами (методами), сохраненными в том же файле классов. Таким образом, другие команды программы никогда непосредственно не обращаются к данным, принадлежащим какому-то объекту.

Если возникает необходимость изменить способ обработки каких-либо данных, просматривать всю программу уже не нужно — достаточно изменить коды только одного объекта.

Таким образом, объектно-ориентированное программирование помогает изолировать команды, которые имеют доступ к определенным данным, что значительно снижает вероятность возникновения ошибок при внесении изменений в коды программы.

Объявление переменных

Объявление переменных в начале модуля класса является хорошей практикой (к тому же обязательной), поскольку позволяет в любой момент видеть, какие данные этим

классом используются. Если нужно объявить переменную, доступ к которой могут получить команды только этого класса, объявите ее как локальную переменную:

```
Private Число As Integer
```

Чтобы объявить глобальную переменную, замените слово Private ключевым словом Public:

```
Public Число As Integer
```

Объявление свойств объектов

Свойства объектов представляют те данные, доступ к которым имеют все коды вашей программы. Любая часть программы может передать свойствам объектов новые значения и извлечь хранимые ими данные. Коды объявления свойств выглядят приблизительно так:

```
Dim. ИмяПеременной As ТипДанных
Public Property НазваниеСвойства() As ТипДанных
Get
НазваниеСвойства = ИмяПеременной
End Get
Set  (ByVal Значение As ТипДанных)
ИмяПеременной = Value
End Set
End Property
```

Чтобы объявить свойство объекта, нужно создать локальную переменную (ИмяПеременной), в которой будет храниться значение свойства. Вместо имени ИмяПеременной можно присвоить любое другое имя. Тип этой переменной (ТипДанных) должен совпадать с типом данных, объявленным для свойства. Например, если объявлено, что свойство будет содержать значения типа String, переменная также должна иметь тип String. И. наконец, необходимо указать имя, посредством которого другие коды программы смогут иметь доступ к значениям свойств объектов (НазваниеСвойства). Например, если нужно объявить свойство объекта, именуемое словом Вектор и содержащее значения типа Integer, это можно сделать так:

```
Dim ЗначВек As .Integer
Public Property Вектор () As Integer
Get
Вектор = ЗначВек
End Get
Set
ЗначВек = Value
End Set
End Property
```

Вот как эти коды будут восприняты Visual Basic .NET.

В первой строке объявляется о создании переменной ЗначЗек, которая может содержать значения типа Integer.

2. Вторая строка говорит: "Создай свойство Вектор, которое будет представлять данные типа Integer".

3. Третья, четвертая и пятая строки дают указание: "Когда другая часть программы хочет извлечь (Get) данные, представляемые свойством Вектор, передай ей значение переменной ЗначВек".

4. Шестая, седьмая и восьмая строки говорят: "Когда другая часть программы хочет передать (Set) новое значение свойству Вектор, сохрани его как значение локальной переменной ЗначВек".

5. А девятая строка сообщает: "На этом объявление свойства завершается"'.

Создание объектов

Модуль класса— это еще не объект. Он является лишь средством, с помощью которого объекты создаются. Согласно терминологии объектно-ориентированного программирования, создание объектов называется также созданием экземпляров классов.

Чтобы создать экземпляр класса, нужно создать объект, который будет представлять модуль класса. Для этого используется ключевое слово New:

```
Dim ИмяОбъекта As New НазваниеКласса
```

Вот что эта строка означает для Visual Basic .NET.

1. Слово Dim говорит: "Сейчас будет объявлено о создании объекта".
2. Словом ИмяООъекта обозначается имя объекта.
3. Слово New дает указание: "Создай новый объект, который будет представлять модуль класса, именуемый НазваниеКласса".

НазваниеКласса — это то имя, которое вы дали файлу класса в момент его создания. Если вы сами не укажете это имя, Visual Basic .NET присвоит ему автоматически генерируемое имя наподобие C l a s s l .

Использование объектов

После того как объект создан, остается только использовать его для:

- сохранения значений в свойствах объектов;
- извлечения данных через свойства объектов;
- манипулирования данными объектов с помощью методов объектов.

Чтобы передать объекту данные, необходимо написать такой код:

```
ИмяОбъекта.Свойство = Значение
```

Извлечь информацию, хранящуюся в объекте, вам поможет код:

```
ИмяПеременной = ИмяОбъекта.Свойство
```

А чтобы запустить метод объекта, наберите следующее:

```
ИмяОбъекта.Метод
```

☞ *Задание. Следуя указаниям, создайте программу, которая запрашивает у нового сотрудника имя, фамилию и дату рождения. Вы будете хранить эту информацию в свойствах нового класса с именем* Person*, и создадите метод класса, который будет вычислять текущий возраст нового сотрудника. Этот проект научит вас создавать собственные классы, экземпляры классов (объекты) а также как использовать эти классы в процедурах событий вашей программы.*

Добавление в ваш проект нового класса

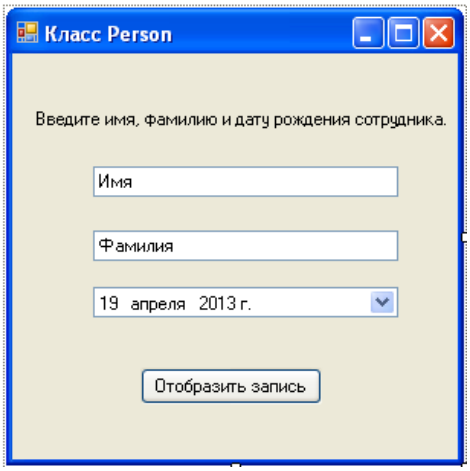
Класс, определенный пользователем, позволяет определить в программе ваши собственные объекты, которые имеют свойства, методы и события, точно так же, как объекты, создаваемые на формах Windows с помощью элементов управления из Области элементов. Чтобы добавить в ваш проект новый класс, щелкните в меню Проект (Project) на команде Добавить класс (Add Class), а затем определите этот класс с помощью кода программы и нескольких новых ключевых слов Visual Basic.

Создание проекта Person Class

1. Запустите *Microsoft Visual Studio 2010,* затем создайте в своей папке новый проект с именем **My Person Class**.
2. Используйте элемент управления Label и добавьте в верхней части формы Form1 длинную метку.
3. Используйте элемент управления TextBox и нарисуйте под меткой два широких текстового поля.
4. Используйте элемент управления DateTimePicker и нарисуйте под текстовыми полями объект выбора даты и времени.
5. Используйте элемент управления Button и нарисуйте под объектом выбора даты и времени кнопку.
6. Установите для объектов формы следующие свойства:

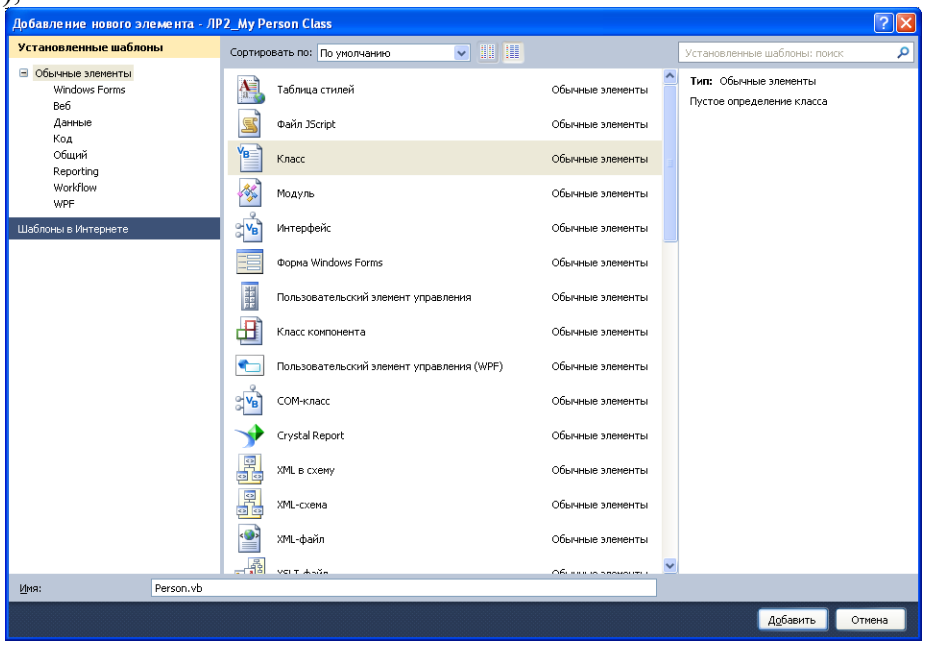
Объект	Свойство	Установка
Label1	Text	Введите имя, фамилию и дату рождения сотрудника.
TextBox1	Text	Имя
TextBox2	Text	Фамилия
Button1	Text	Отобразить запись
Form1	Text	Класс Person

7. Ваша форма должна выглядеть примерно так.



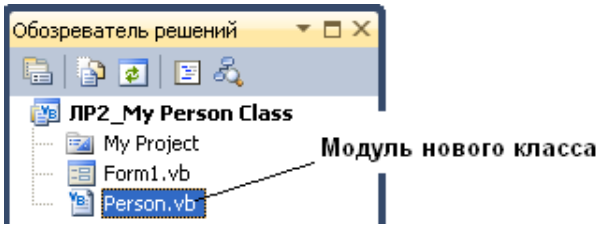
Это базовый интерфейс пользователя для формы, которая определяет запись нового сотрудника фирмы. (Эта форма не подключена к базе данных, так что храниться может только одна запись.) Теперь вы должны добавить в проект класс для хранения информации из этой записи.

8. Щелкните на команде Добавить класс (Add Class) в меню Проект (Project). Visual Studio откроет диалоговое окно Добавление нового элемента (Add New Item), показанное ниже.



Диалоговое окно Добавление нового элемента дает возможность задать имя вашего класса. Когда вы присвоите имя, обратите внимание, что вы можете сохранить в новом модуле класса несколько классов и указать имя, которое будет для них общим.

9. Введите в текстовом поле Имя (Name) имя Person.vb, а затем щелкните Добавить. Visual Studio откроет в Редакторе кода пустой модуль класса и добавит имя файла Person.vb в ваш проект в Обозревателе решений, как показано на рисунке.



Объявление переменных класса

• Под оператором программы `Public Class Person` введите следующие объявления переменных:

```
Private Name1 As String
Private Name2 As String
```

Здесь вы объявляете две переменные, которые будут использованы исключительно в модуле класса для хранения значений двух строковых свойств. Переменные объявлены с помощью ключевого слова `Private`, так как по соглашению Visual Basic программисты должны держать внутренние переменные класса закрытыми - другими словами, недоступными для просмотра извне самого модуля класса.

Создание свойств

1. Под объявлением переменных введите следующий оператор программы и нажмите клавишу (Enter):

```
Public Property FirstName() As String
```

Этот оператор создает свойство вашего класса с именем `FirstName`, которое имеет тип `String`. Когда вы нажмете (Enter), Visual Studio немедленно создаст структуру кода для остальных элементов объявления свойства. Требуемыми элементами являются: блок `Get`, который определяет, что программисты увидят, когда будут проверять свойство `FirstName`, блок `Set`, который определяет, что произойдет, когда свойство `FirstName` будет установлено или изменено, и оператор `End Property`, который отмечает конец процедуры свойства.

Примечание. В Visual Basic 6 процедуры свойств содержали блоки кода Property Get, Property Let и Property Set. Этот синтаксис больше не поддерживается.

2. Заполните структуру процедуры свойства так, чтобы она выглядела, как показано ниже.

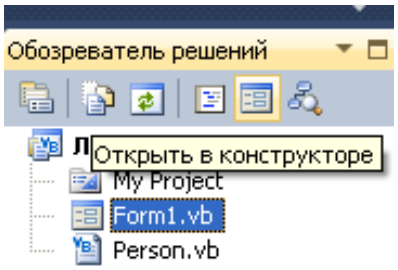
```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal Value As String)
        Name1 = Value
    End Set
End Property
```

Ключевое слово `Return` указывает, что при обращении к свойству `FirstName` будет возвращена строковая переменная `Name1`. При установке значения свойства блок `Set` присваивает переменной `Name1` строковое значение. Обратите особое внимание на переменную `Value`, используемую в процедурах свойств для обозначения значения, которое присваивается свойству класса при его установке. Хотя этот синтаксис может выглядеть странно, просто поверьте мне - именно так создаются свойства в элементах управления, хотя более сложные свойства будут иметь здесь дополнительную программную логику, которая будет проверять значения и производить вычисления.

3. Под оператором `End Property` введите для свойства `LastName` вашего класса вторую процедуру свойства. Она должна выглядеть так, как показано ниже.

```
Public Property LastName() As String
    Get
        Return Name2
    End Get
    Set(ByVal Value As String)
        Name2 = Value
    End Set
End Property
```

Эта процедура свойства аналогична первой, за исключением того, что она использует вторую строковую переменную (`Name2`), которую вы объявили в верхней части кода класса. Вы закончили определять два свойства вашего класса. Теперь перейдем к методу с именем `Age`, который будет определять текущий возраст нового сотрудника на основе даты рождения.



Создание метода

• Под процедурой свойства `LastName` введите следующее определение функции:

```
Public Function Age(ByVal Birthday As Date) As Integer
    Return Int(Now.Subtract(Birthday).Days / 365.25)
End Function
```

Чтобы создать метод класса, который выполняет некое действие, добавьте в ваш класс процедуру Sub. Хотя многие методы не требуют для выполнения своей работы аргументов, метод `Age`, определенный мной, требует для своих вычислений аргумент Birthday типа Date. Это метод использует для вычитания даты рождения нового сотрудника из текущей системной даты метод Subtract, и возвращает значение, выраженное в днях, деленных на 365.25 - примерную длину одного года в днях. Функция Int преобразует это значение в целое, и это число с помощью оператора `Return` возвращается в вызывающую процедуру - как и в случае с обычной функцией.

Определение класса закончено! Вернитесь к форме `Form1` и используйте новый класс в процедуре события.

Совет. Хотя в данном примере это и не делалось, в реальном проекте полезно добавить в модуль класса логику для проверки типов данных. Это делается для того, чтобы неправильное использование свойств или методов, не приводило к возникновению ошибок времени исполнения, из-за которых выполнение программы может прерваться.

Создание объекта с помощью нового класса

1. Щелкните в Обозревателе решений на значке Form1.vb, а затем на кнопке Открыть в конструкторе. Появится интерфейс пользователя Form1.
2. Чтобы открыть в Редакторе кода процедуру события Button1_Click, сделайте двойной щелчок мышью на кнопке **Отобразить запись**.
3. Введите следующие операторы программы:

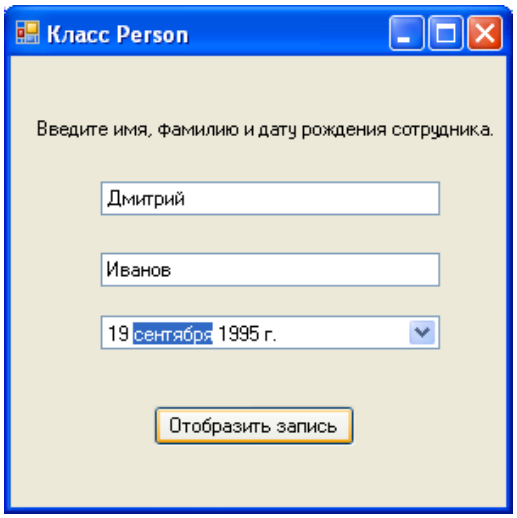
```
Dim Employee As New Person
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " в возрасте " & Employee.Age(DOB) & "лет.")
```

Эта процедура сохраняет в объекте с именем `Employee`, который имеет тип `Person`, значения, введенные пользователем. Ключевое слово `New` указывает, что вы хотите немедленно создать новый экземпляр объекта `Employee`. Теперь нужно объявить переменную с помощью класса, созданного вами самими! Затем процедура объявляет переменную с именем DOB типа Date. Она будет хранить дату, введенную пользователем, и устанавливает свойства `FirstName` и `LastName` объекта `Employee` равными имени и фамилии, введенным в два объекта текстовых полей формы. Значение, возвращаемое объектом выбора даты и времени, сохраняется в переменной DOB, а последний оператор программы отображает окно сообщения, содержащее свойства `FirstName` и `LastName`, а также возраст нового сотрудника, определенный методом `Age`, который при передаче в него переменной DOB возвращает целое значение. Как только вы определили класс в модуле класса, его легко можно использовать в процедуре события.

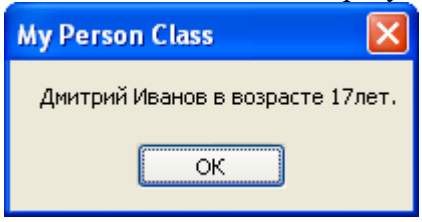
4. Чтобы запустить программу, щелкните на кнопке Начать отладку (F5). В среде разработки появится интерфейс пользователя, готовый к приему ваших данных.
5. Введите в текстовое поле First Name ваше имя, а в текстовое поле Last Name - фамилию.
6. Щелкните на раскрывающемся списке объекта выбора даты и времени, и прокрутите его до вашей даты рождения.

Совет. Вы можете быстро прокрутить список, щелкнув в открытом диалоговом окне объекта выбора даты на поле года. Появятся небольшие стрелки прокрутки, и вы сможете переходить сразу на год вперед или назад. Также можно быстро перейти на нужный вам месяц, щелкнув на поле месяца, а затем на месяце в появившемся меню.

Ваша форма будет выглядеть примерно так.



7. Щелкните на кнопке **Отобразить запись**. Ваша программа сохраняет значения имени и фамилии в свойствах и использует метод `Age` для вычисления текущего возраста нового сотрудника. Появится диалоговое окно с результатом.



8. Чтобы закрыть это окно сообщения, щелкните на **OK**, а затем поэкспериментируйте с несколькими различными значениями дат, щелкая на **Отобразить запись** каждый раз, когда вы меняете значение поля даты рождения.

Контрольные вопросы:

10. Определите понятия класс, экземпляр класса, объект.
11. На какие этапы можно разбить процесс создания экземпляров класса.
12. Как создать базовый файл класса, какой код он содержит.
13. Синтаксис объявления переменной класса.
14. Создание в классе нового свойства.
15. Создание в классе нового метода.
16. Объявление переменной объекта для использования в классе.
17. Обращение к свойствам и методам объекта

Лабораторная работа №42

Тема: «Создание наследованного класса»

Цель: *научиться создавать наследованные классы, реализовать наследование форм в среде Visual Studio.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Оператор **Inherits** используется для объявления нового класса, называемого *производным классом*, который основан на существующем классе, называемом *базовым классом*. Производные классы наследуют и могут расширять свойства, методы и события, поля и константы, определенные в базовом классе.

- Все классы могут наследоваться по умолчанию, если только они не помечены зарезервированным словом **NotInheritable**. Классы могут наследовать из других классов проекта или из классов других сборок, на которые ссылается проект.
- В отличие от языков, которые позволяют множественное наследование, Visual Basic позволяет только единичное наследование в классах; то есть производные классы могут иметь только один базовый класс. Хотя в классах не поддерживается множественное наследование, они все же могут реализовывать множественные интерфейсы, которые способны эффективно выполнять те же самые задачи.
- Чтобы предотвратить предоставление элементов с ограничениями в базовом классе, тип доступа к производному классу должен быть таким же, как и тип доступа к базовому классу, или более строгим. Например, **Public** класс не может наследовать **Friend** или **Private** класс, а **Friend** класс не может наследовать **Private** класс.

Модификаторы наследования

Для поддержки наследования Visual Basic вводит следующие инструкции на уровне класса и модификаторы:

- Инструкция **Inherits** определяет базовый класс.
- Модификатор **NotInheritable** не позволяет использовать класс в качестве базового класса.
- Модификатор **MustInherit** определяет, что класс предназначен только для использования в качестве базового класса.Невозможно создать напрямую экземпляры классов **MustInherit**; их можно создать только как экземпляры базового класса в производном классе

Наследование форм с помощью инструмента Выбор наследования

В терминологии объектно-ориентированного программирования наследование означает, что один класс получает объекты, свойства, методы и другие атрибуты другого класса. Visual Basic всегда использует это при создании в среде разработки новой формы. Первая форма проекта (`Form1`) определена на основе класса `System.Windows.Forms.Form` и получает от него свои значения по умолчанию. На самом деле каждый раз, когда с помощью команды Добавить форму Windows (Add Windows Form) из меню Проект (Project) создается новая форма, этот класс указывается в верхней части кода каждой формы с использованием ключевого слова `Inherits`, как показано ниже:

Inherits

System.Windows.Forms.Form

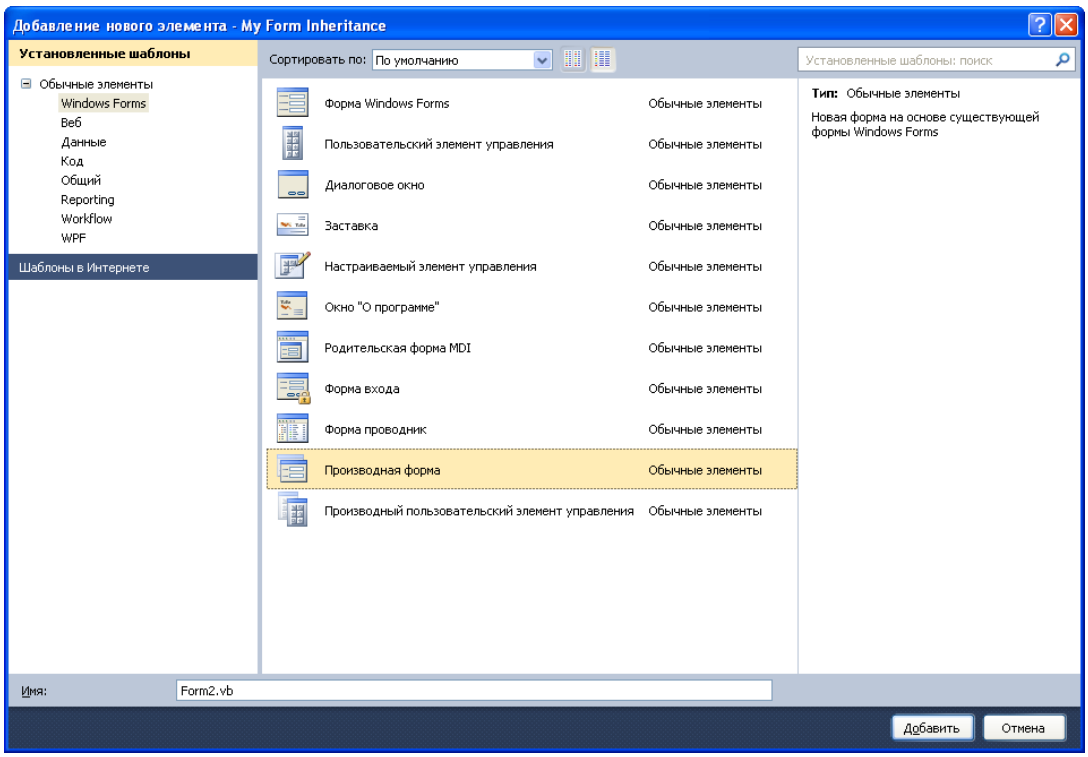
Вы неосознанно всегда использовали наследование для определения форм Windows, которые использовали при создании приложений Visual Basic. Хотя существующие формы могут наследоваться с помощью кода программы, разработчики Visual Studio .NET сочли эту задачу важной и разработали специальный инструмент среды разработки, облегчающий этот процесс. Этот инструмент называется Выбор наследования (Inheritance Picker). Он доступен через команду Добавить производную форму (Add Inherited Form) в меню Проект (Project). В следующем упражнении вы будете использовать Выбор наследования (Inheritance Picker) для создания второй копии диалогового окна проекта.

Задание 1. Наследование простого диалогового окна

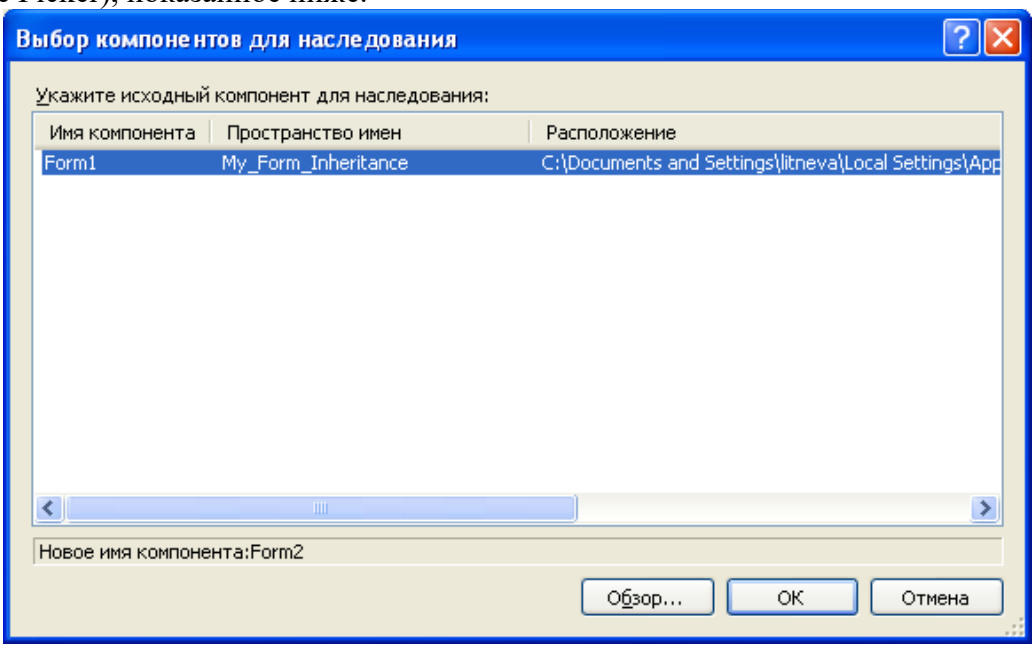
1. Запустите Visual Studio и создайте новый проект с именем **My Form Inheritance**.
2. Отобразите форму проекта и используйте элемент управления Button для добавления в нижнюю часть формы двух расположенных рядом объектов кнопок.
3. Измените свойства Text кнопок Button1 и Button2 на "OK" и "Отмена" соответственно.
4. Чтобы отобразить в Редакторе кода процедуру события Button1_Click, сделайте двойной щелчок мышью на кнопке **OK**.
5. Введите следующий оператор программы: `MsgBox("Вы нажали OK")`
6. Снова отобразите форму, сделайте двойной щелчок мышью на кнопке **Отмена**, а затем введите в процедуре события `Button2_Click` следующий оператор программы: `MsgBox("Вы нажали Отмена")`
7. Снова отобразите форму, а затем установите свойство `Text` формы на значение "Диалоговое окно". Теперь у вас есть простая форма, которую можно использовать как основу для диалогового окна программы. С помощью некоторых настроек вы можете использовать эту базовую форму для выполнения нескольких задач - просто нужно добавить на нее элементы управления, которые потребуются вашему приложению.

Попрактикуйтесь в наследовании форм. Первым шагом в этом процессе является сборка - или **компиляция** - проекта, так как наследовать можно только от тех форм, которые скомпилированы в виде файлов .exe или .dll. Каждый раз, когда компилируется базовая форма, изменения, сделанные в этой базовой форме, передаются в производную (наследованную) форму.

8. Щелкните на команде Построить решение (Build Solution) в меню Построение (Build). Visual Basic скомпилирует ваш проект и создаст .exe-файл.
9. Щелкните на команде Добавить производную форму (Add Inherited Form) в меню Проект (Project). Вы увидите диалоговое окно, показанное ниже. Как обычно, Visual Studio приводит список всех возможных шаблонов, которые вы можете включить в проект. На панели Установленные шаблоны выберите Windows Forms, а справа -Производная форма (Inherited Form). Текстовое поле Имя (Name) в нижней части диалогового окна позволяет присвоить вашей производной форме имя; это имя, которое появится в Обозревателе решений (Solution Explorer) и в имени файла формы на диске.



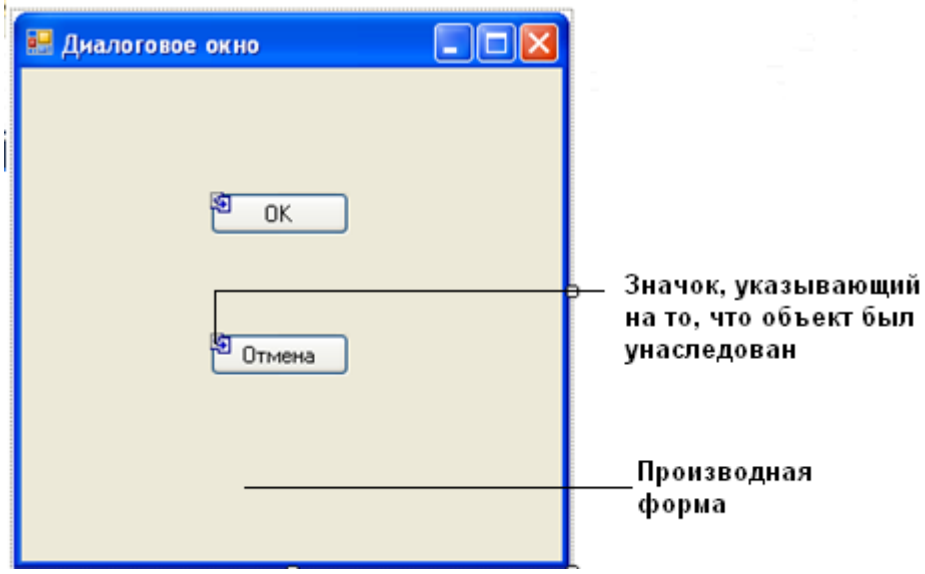
10. Щелкните на Добавить (Add), чтобы принять для новой формы установки по умолчанию. Visual Studio отобразит диалоговое окно Выбор компонентов для наследования (Inheritance Picker), показанное ниже.



Это диалоговое окно содержит перечень всех форм текущего проекта, от которых можно наследовать. Если вы хотите просмотреть другие скомпилированные формы, щелкните на кнопке Обзор (Browse) и найдите на вашем жестком диске требуемый .dll-файл. (Если вы хотите наследовать от формы, которая не является компонентой текущего проекта, форма должна быть скомпилирована в .dll-файл.)

11. Щелкните в диалоговом окне Выбор наследования (Inheritance Picker) на `Form1`, а затем на **OK**. Visual Studio создаст в Обозревателе решений (Solution Explorer) элемент Form2.vb и отобразит в Конструкторе Windows Forms (Windows Forms Designer) производную форму. На следующем рисунке обратите внимание, что форма выглядит

идентично окну `Form1`, созданному ранее, за исключением того, что две кнопки содержат маленькие значки, которые указывают, что объекты получены из наследуемого источника.

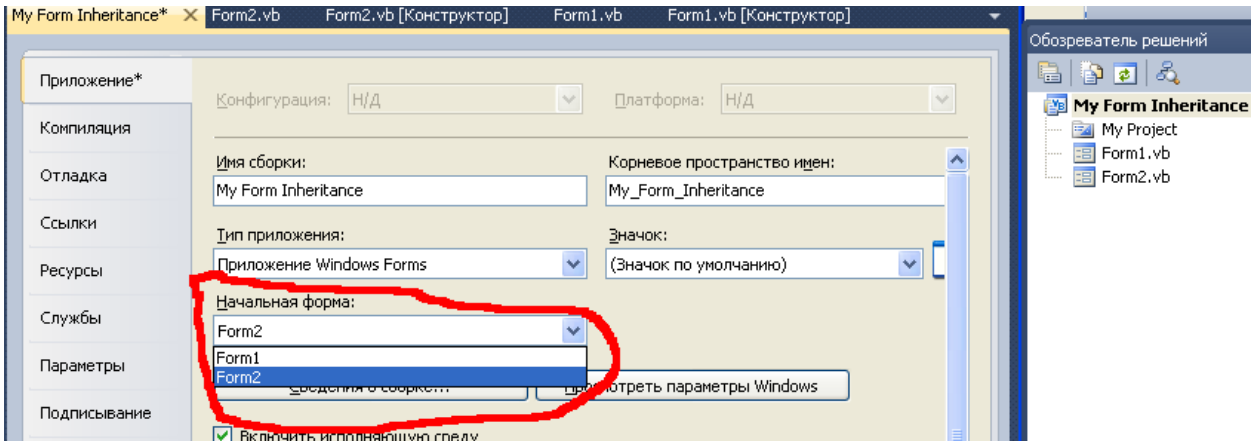


Иногда сложно отличить производную форму от базовой (маленькие значки наследования не так очевидны), так что используйте для различения этих форм Обозреватель решений (Solution Explorer) и закладки окон среды разработки.

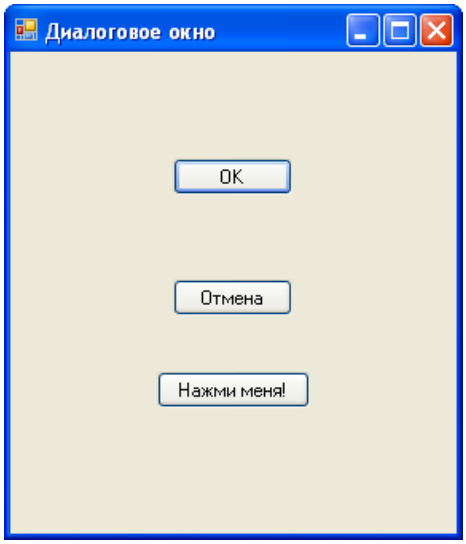
Теперь вы должны добавить к производной форме несколько новых элементов.

Настройка производной формы

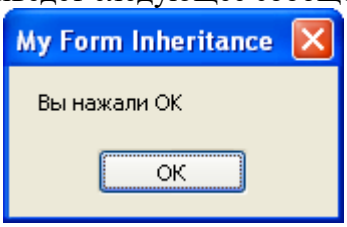
1. Используйте элемент управления Button и добавьте на Form2 (производная форма) третью кнопку.
2. Установите свойство Text этой кнопки равным "Нажми меня!".
3. Сделайте двойной щелчок мышью на кнопке **Нажми меня**!
4. В процедуре события `Button3_Click` введите следующий оператор программы: `MsgBox("Это подчиненная форма!")`
5. Снова отобразите форму Form2, а затем попробуйте сделать двойной щелчок мышью на кнопках **OK** и **Отмена**. Вы не можете отобразить или отредактировать процедуры событий для этих унаследованных объектов без дополнительных действий, которые не обсуждаются здесь. Однако вы можете добавить в форму новые объекты и настроить их.
6. Увеличьте форму. Вы также можете изменить другие характеристики формы, такие, как ее размер и расположение.
7. Щелкните в Обозревателе решений (Solution Explorer) на значке проекта My Form Inheritance, а затем на команде Свойства (Properties) в меню Проект (Project). Появится диалоговое окно.
8. Щелкните на раскрывающемся списке Начальная форма, далее на Form2, а затем на **OK**. Запустите новый проект.



9. Щелкните на кнопке Начать (Start). Откроется производная форма, показанная ниже.



10. Щелкните на кнопке **OK**. Производная форма выполнит процедуру события наследуемой формы `Form1`, и она выведет следующее сообщение.



11. Щелкните на **Отмена**, а затем на **Нажми меня**!. Form2 отобразит наследуемое сообщение формы. Производная форма настроена так, чтобы включить новый объект, а также два наследуемых объекта кнопки. Вы сделали первые шаги в освоении механизма наследования с помощью диалогового окна Inheritance Picker (Выбор наследования).

12. Чтобы закрыть окно сообщения, щелкните на **OK**, а затем на **Закрыть** формы, чтобы завершить выполнение программы. Программа остановится, и вернется среда разработки.

☝Задание 2. Наследование базового класса

Точно так же, как при наследовании классов форм, вы можете наследовать обычные классы, которые вы сами определяете с помощью команды **Добавить класс (AddClass)** и модуля класса. Механизм наследования базового (родительского) класса состоит в использовании оператора **`Inherits`**, который включает ранее определенный класс в новый класс. Затем вы можете добавить в производный (дочерний) класс новые свойства или методы, которые будут отличать его от базового класса.

Задание: Изменить проект MyPersonClass (проект из Л_Р_№2), добавив в модуль класса `Person` второй класс, определенный пользователем. Этот новый класс с именем `Teacher` будет наследовать от класса `Person` свойство `FirstName`, свойство `LastName` и метод `Age`, и будет добавлять свойство с именем Grade, в которое будет записываться уровень, на котором обучает новый учитель.

Использование ключевого слова Inherits

1. Щелкните в Обозревателе решений на классе Person.vb, а затем щелкните на кнопке Просмотреть код (ViewCode).
2. Прокрутите Редактор кода вниз так, что текстовый курсор окажется стоящим после оператора End Class. Как упоминалось выше, вы можете включить в модуль класса более одного класса, при условии, что каждый класс отделен от остальных операторами Public Class и End Class. Вы создадите в этом модуле класса новый класс с именем Teacher, а для встраивания в него методов и свойств, определенных вами в классе Person, будете использовать ключевое слово Inherits.
3. Введите в Редактор кода следующее определение класса.

```
Public Class Teacher
    Inherits Person
    Private Level As Short

    Public Property Grade() As Short
        Get
            Return Level
        End Get
        Set(ByVal Value As Short)
            Level = Value
        End Set
    End Property
End Class
```

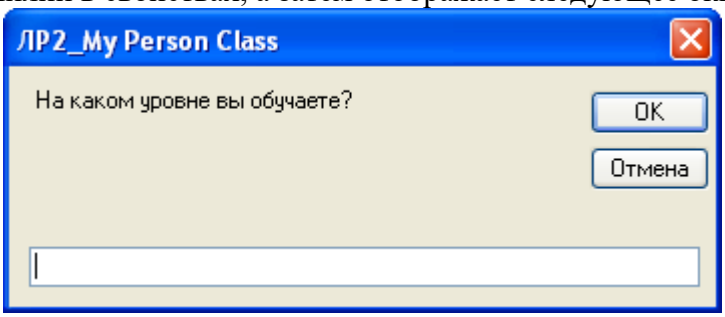
Оператор `Inherits` связывает класс `Person` с этим новым классом, встраивая в него все свои переменные, свойства и методы. Если бы класс `Person` находился в отдельном модуле или проекте, вы могли бы указать его расположение, задав пространство имен, точно так же, как вы указываете классы библиотеки .NET Framework с помощью оператора Imports. По существу класс `Teacher определен` как специальный тип класса `Person` - в дополнение к свойствам `FirstName` и `LastName` класс `Teacher` имеет свойство Grade, которое хранит уровень студента, обучаемого учителем. Теперь вы будете использовать этот новый класс в процедуре события `Button1_Click`.

4. Отобразите процедуру события `Button1_Click` формы `Form1`. Вместо создания новой переменной для хранения класса `Teacher`, просто используйте имеющуюся переменную `Employee` - единственное различие будет в том, что теперь вы можете установить свойство Grade нового сотрудника.
5. Измените процедуру события `Button1_Click` в соответствии со следующим кодом.

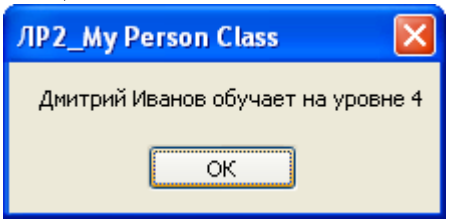
```
Dim Employee As New Teacher
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
Employee.Grade = InputBox("На каком уровне вы обучаете?")
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " обучает на уровне " & Employee.Grade)
```

В этом примере мы удалили вычисление текущего возраста - метод `Age` не используется, но сделали это только для того, чтобы сократить до минимума информацию, выводимую в окне сообщения. Когда вы определяете свойства и методы класса, вам не требуется использовать их в коде программы.

6. Чтобы запустить программу, щелкните на кнопке Начать отладку. На экране появится форма для ввода данных о новом сотруднике.
7. Введите в текстовое поле **Имя** ваше имя, а в текстовое поле **Фамилия** - фамилию.
8. Щелкните на объекте выбора даты и прокрутите его до вашего дня рождения.
9. Щелкните на кнопке **Отобразить запись**. Ваша программа сохраняет значения имени и фамилии в свойствах, а затем отображает следующее окно ввода.



10. Введите 4, а затем щелкните на **OK**, чтобы закрыть окно ввода. Приложение сохраняет число 4 в новом свойстве Grade и использует свойства `FirstName`, `LastName` и Grade для отображения информации о новом сотруднике в подтверждающем окне сообщения. Вы увидите такое сообщение.



11. Если хотите, поэкспериментируйте еще с несколькими значениями, а затем щелкните на кнопке **Закрыть** формы. Программа остановится, и вернется среда разработки.

Контрольные вопросы:

1. Что означает наследование в терминологии ООП?
2. Какой оператор необходимо использовать для объявления производного класса?
3. На основе какого класса определена первая форма проекта?
4. Опишите действия по добавлению производной формы.
5. Назовите визуальные отличия производной формы от базовой.

Программирование приложений (4 часа)

Лабораторная работа №45

Тема: «Перегрузка методов»

Цель: *научиться выполнять перегрузку методов в среде Visual Studio.*

Ход работы:

1. Изучить теоретическую часть.
2. Выполнить задания, следуя указаниям.
3. Ответить на контрольные вопросы (в устной форме).
4. Предъявить преподавателю результаты работы программы и исходные коды.
5. Оформить отчет в соответствии с ходом работы (тема, цель, условие задачи, программный код, результаты тестирования программы, выводы).

🕮Теоретическая часть

Переопределение свойств и методов в производных классах

По умолчанию производный класс наследует свойства и методы от своего базового класса. Если наследуемое свойство или метод в производном классе должен работать другим образом, то оно может быть *переопределено*. То есть, в производном классе можно определить новую реализацию метода. Следующие модификаторы используются для управления переопределением свойств и методов.

- **Overridable** позволяет переопределение свойства или метода в производном классе.
- **Overrides** переопределяет **Overridable** свойство или метод, определенное в базовом классе.
- **NotOverridable** не позволяет переопределять свойство или метод в наследующем классе.По умолчанию **Public** методы являются **NotOverridable**.
- **MustOverride** требует обязательного переопределения свойства или метода производным классом.При использовании ключевого слова **MustOverride** определение метода состоит только из инструкций **Sub**, **Function** или **Property**.Другие инструкции не допускаются, и в частности отсутствуют инструкции **End Sub** и **End Function**.Методы **MustOverride** должны объявляться в классах **MustInherit**.

Ключевое слово MyBase

Ключевое слово **MyBase** ведет себя подобно объектной переменной, ссылающейся на базовый класс текущего экземпляра. Ключевое слово **MyBase** часто используется для обращения к членам базового класса, которые переопределены или скрыты в производном классе. В частности, можно воспользоваться вызовом MyBase.New для явного вызова конструктора базового класса из конструктора производного класса.

Ключевое слово MyClass

Ключевое слово **MyClass** ведет себя подобно объектной переменной, ссылающейся текущий экземпляр класса, который был изначально реализован. Ключевое слово **MyClass** напоминает слово **Me**, однако при каждом вызове метода или свойства **MyClass** предполагается, что метод или свойство имеет атрибут NotOverridable (Visual Basic). Следовательно, на метод или свойство не оказывает никакого влияния его переопределение в производном классе.

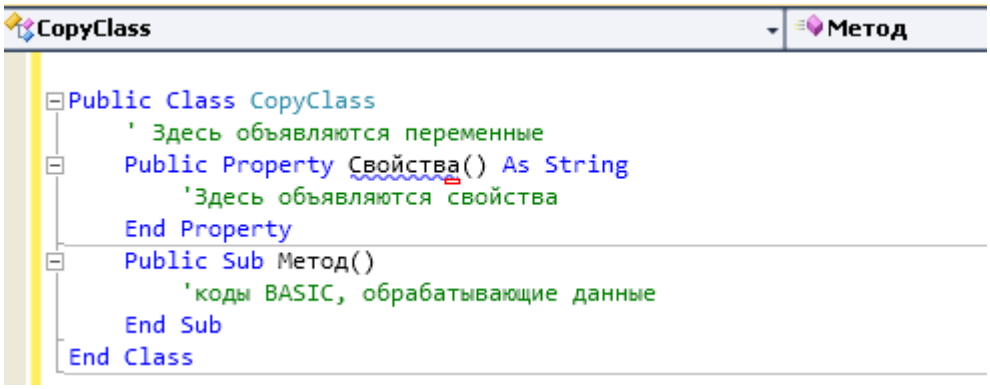
Чтобы перегрузить какой-либо метод или свойство необходимо выполнить такие действия:

1. В окне Обозреватель решений (Solution Explorer) щелкните на названии файла классов, коды которого должны быть наследованы для создания новых классов.

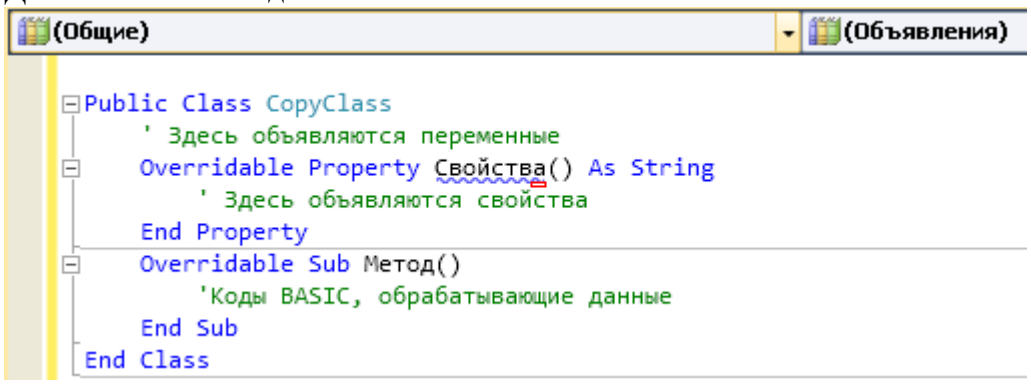
2. Нажмите клавишу <F7>, Visual Basic .NET отобразит на экране коды BASIC выбранного файла.

3. Для каждого свойства или метода, коды которого могут быть перегружены, замените слово **Public** ключевым словом **Overridable** .

Допустим, коды выбранного файла выглядят так:



Для свойств и методов замените слово Public словом Overridable:



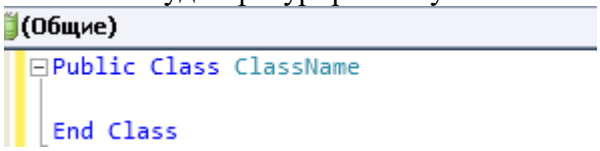
Коды тех свойств и методов, для которых вы оставите слово Public, в новых объектах не могут быть изменены.

4. Выберите команду Проект/Добавить класс (Project/Add Class).

Откроется диалоговое окно Добавить новый элемент (Add New Item).

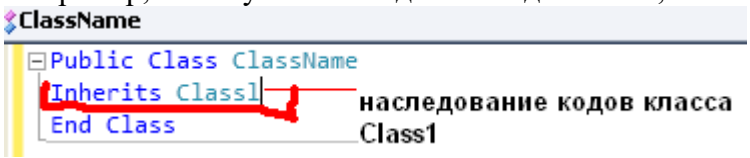
5. Наберите имя нового файла классов в поле Имя (Name) и щелкните на кнопке Добавить (Add).

Visual Basic .NET отобразит показанные ниже коды пустого класса, где вместо слова ClassName будет фигурировать указанное вами имя:



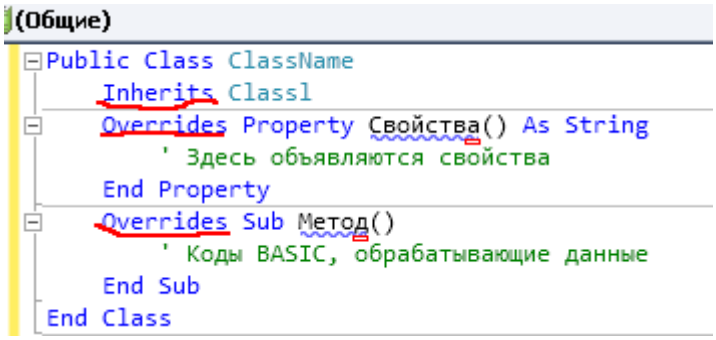
6. Наберите слово **Inherits** , а затем укажите название класса, коды которого должны быть наследованы.

Например, если нужно наследовать коды класса, названного именем Classl, наберите



7. Наберите коды свойств и методов, которые должны быть перегружены, но вместо слова Public используйте слово Overrides.

Слово Overrides используется при написании свойств и методов, коды которых будут отличаться от кодов свойств и методов исходного файла с теми же именами:



Таким образом, при создании нового класса вам нужно будет написать коды только тех свойств и методов, которые должны отличаться от свойств и методов исходного класса. Если приемы наследования и перегрузки кодов кажутся вам не очень понятными, не спешите сразу же отказываться от их применения. Помните, что они действительно могут помочь вам быстрее и более простым способом создавать корректно работающие программы, позволяя повторно использовать либо коды свойств и методов других объектов, либо только имена этих свойств и методов.

Задание. Чтобы реализовать перегрузку методов, прежде необходимо разработать приложение, в котором создаются файлы классов для представления местоположения улыбающегося и грустного лиц. Каждый раз, когда пользователь щелкает на кнопке Переместить улыбающееся лицо или Переместить грустное лицо, эти картинки начинают перемещаться. Воспользуйтесь графическим редактором для создания подобных рисунков.

Ниже в таблице приведен список объектов пользовательского интерфейса, которые нужно создать для этой программы, и список значений, которые нужно присвоить их свойствам.

Объект	Свойство	Значение
PictureBox1	Image Name SizeMode	указать URL рисунка picSmile StretchImage
PictureBox2	Image Name SizeMode	указать URL рисунка picFrown StretchImage
Button1	Text Name	Переместить улыбающееся лицо btnSmile
Button2	Text Name	Выход btnExit
Button3	Text Name	Переместить грустное лицо btnFrown

1. Запустите Visual Studio и создайте новый проект.
2. Отобразите форму проекта и используйте элемент управления PictureBox и Button для добавления объектов боксов для рисунков и кнопок.
3. Измените свойства Image, Name, SizeMode элементов PictureBox1 и PictureBox2 на значения, указанные в таблице выше.
4. Измените свойства кнопок Button1, Button2 и Button3 на значения, указанные в таблице выше.
5. Дважды щелкните на кнопке формы **Переместить улыбающееся лицо** и создайте такую процедуру обработки события:

```
Dim Smile As New Classl()
Smile.xspot = picSmile.Location.X
Smile.Move()
picSmile.Left = Smile.xspot
```

6. Отобразите форму, сделайте двойной щелчок мышью на кнопке **Переместить грустное лицо** и создайте такую процедуру обработки события:

```
Dim Frown As New Classl()
Frown.yspot = picFrown.Location.Y
Frown.Move()
picFrown.Top = Frown.yspot
```

7. Снова отобразите форму, сделайте двойной щелчок мышью на кнопке **Выход**, а затем введите в процедуре события следующий оператор программы: `Me.Close()`
8. Создайте отдельный файл класса (Проект/Добавить класс) и наберите следующее:

```
Public Class Classl
    Dim x, y As Integer
    Const War = 10
    Public Property xspot() As Integer
        Get
            xspot = x
        End Get
        Set(ByVal Value As Integer)
            x = Value
        End Set
    End Property
    Public Property yspot() As Integer
        Get
            yspot = y
        End Get
        Set(ByVal Value As Integer)
            y = Value
        End Set
    End Property
    Public Sub Move()
        x = x + War
        y = y + War
    End Sub
End Class
```

9. Проект готов, запустите его и протестируйте.

Действия по перегрузке методов

10. Измените коды класса Classl. vb, заменив для метода слово Public словом Overridabie:

```
            y = Value
        End Set
    End Property
    Overridable Sub Move()
        x = x + War
        y = y + War
    End Sub
End Class
```

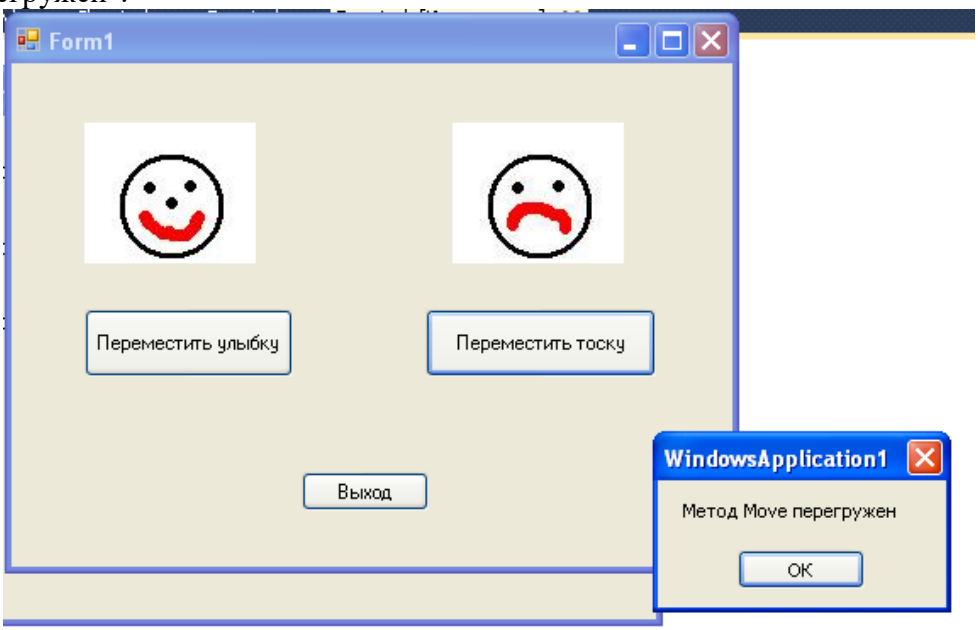
11. Затем создайте новый файл классов (Проект/Добавить класс), назовите его CopyClass и наберите для него код:

```
Public Class CopyClass
    Inherits Class1
    Overrides Sub Move()
        MsgBox("Метод Move перегружен")
    End Sub
End Class
```

12. Измените коды в файле формы таким образом, чтобы был использован как оригинальный файл классов (Class1), так и наследованный (CopyClass):

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Sub BtnSmile_Click(sender As System.Object, e As System.EventArgs) Handles BtnSmile.Click
        Dim Smile As New Class1()
        Smile.xspot = picSmile.Location.X
        Smile.Move()
        picSmile.Left = Smile.xspot
    End Sub

    Private Sub BtnExit_Click(sender As System.Object, e As System.EventArgs) Handles BtnExit.Click
        Me.Close()
    End Sub

    Private Sub BtnFrown_Click(sender As System.Object, e As System.EventArgs) Handles BtnFrown.Click
        Dim Frown As New CopyClass()
        Frown.yspot = picFrown.Location.Y
        Frown.Move()
        picFrown.Top = Frown.yspot
    End Sub
End Class
```

Если теперь вы щелкнете на кнопке формы **Переместить улыбающееся лицо**, улыбающееся лицо, как и прежде, будет перемещаться по экрану. Если же вы щелкнете на кнопке **Переместить грустное лицо**, Visual Basic.NET обратится к наследованному классу CopyClass, коды которого были перегружены, и на экране появится сообщение "Метод Move перегружен".



Контрольные вопросы:

1. Что такое перегрузка методов.
2. Перечислить и охарактеризовать модификаторы, которые используются для управления переопределением свойств и методов.
3. Охарактеризуйте ключевые слова MyBase и MyClass.
4. Опишите действия, необходимые для перегрузки.