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

Рассмотрено
цикловой комиссией
Протокол заседания № 1 .
от «31» августа 2020 г.
Председатель цикловой комиссии
_____ Чобану Л.А.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине

ОП.05 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРММИРОВАНИЯ

по специальности
10.02.04 «Обеспечение информационной безопасности телекоммуникационных систем»

Квалификация техник по защите информации

Разработчик:
Преподаватель
Белгородский индустриальный колледж
Чобану Л.А.

Белгород 2020 г.

Содержание

	Стр.
1. Пояснительная записка	3
1.1. Краткая характеристика дисциплины, ее цели и задачи. Место лабораторных работ в курсе дисциплины	3
1.2. Организация и порядок проведения лабораторных работ	3
1.3. Общие указания по выполнению лабораторных работ	3
1.4. Критерии оценки результатов выполнения лабораторных работ	3
2. Тематическое планирование лабораторных работ	5
3. Содержание лабораторных работ	7
Лабораторная работа №1 «Создание блок-схем линейной и разветвляющейся структуры»	7
Лабораторная работа №2 «Создание блок-схемы циклической структуры»	9
Лабораторная работа №3 «Простые программы. Типичные ошибки. Хороший стиль программирования»	11
Лабораторная работа №4 «Составление программ линейной структуры. Составление программ разветвляющейся структуры»	14
Лабораторная работа №5 «Составление программ разветвляющейся усложненной структуры»	16
Лабораторная работа №6 «Составление программ циклической структуры»	18
Лабораторная работа №7 «Составление программ усложненной структуры»	20
Лабораторная работа №8 «Вложенные циклы»	21
Лабораторная работа №9 «Применение множеств»	23
Лабораторная работа №10 «Решение задач с использованием строк»	25
Лабораторная работа №11 «Решение задач с использованием строк, срезов строк»	30
Лабораторная работа №12 «Списки»	32
Лабораторная работа №13 «Решение задач с использованием кортежей»	33
Лабораторная работа №14, 15 «Методы списков и строк.»	35
Лабораторная работа №16, 17 «Словари в Python»	39
Лабораторная работа №18, 19 «Организация функций»	42
Лабораторная работа №20 «Применение рекурсивных функций»	46
Лабораторная работа №21, 22 «Применение библиотек Python»	50
Лабораторная работа №23 «Описание свойств структуры и действия над объектами структурного типа»	56
Лабораторная работа №24, 25 «Создание класса, объявление объектов»	62
Лабораторная работа №26, 27, 28 «Создание наследованного класса»	70
4. Информационное обеспечение обучения	78

1. Пояснительная записка

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

Дисциплина ОП.05 Основы алгоритмизации и программирования является частью рабочей основной образовательной программы в соответствии с ФГОС по специальности СПО 10.02.04 «Обеспечение информационной безопасности телекоммуникационных систем».

Дисциплина изучается в III-IV семестрах. В целом рабочей программой предусмотрено 56 часов на выполнение лабораторных работ, что составляет 52 % от обязательной аудиторной нагрузки, которая составляет 108 часов, при этом максимальная нагрузка составляет 124 часа, из них 4 часа приходится на самостоятельную работу обучающихся.

Цель настоящих методических рекомендаций: оказание помощи обучающимся в выполнении лабораторных работ по дисциплине ОП.05 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.05 Основы алгоритмизации и программирования предусматривает проведение 56 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 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ПК 1.1. Производить монтаж, настройку, проверку функционирования и конфигурирование оборудования информационно-телекоммуникационных систем и сетей.

ПК 1.4. Осуществлять контроль функционирования информационно-телекоммуникационных систем и сетей.

– обоснованность и четкость изложения материала;

– уровень оформления работы.

– анализ результатов.

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.

2. Тематическое планирование лабораторных работ

	Наименование тем	**Вид и название работы студента**	**Количество часов на выполнение работы**
Раздел 1	**Основные принципы программирования**		**4**
1.2	Базовые конструкции структурного программирования	**Лабораторная работа №1 «**Создание блок-схем линейной и разветвляющейся структуры»	2
		Лабораторная работа №2 «Создание блок-схемы циклической структуры»	2
Раздел 2	**Программирование на алгоритмическом языке**		**40**
2.1	Базовые средства языка Python	**Лабораторная работа №3** «Простые программы. Типичные ошибки. Хороший стиль программирования»	2
2.2	Операторы языка программирования	**Лабораторная работа №4** «Составление программ линейной структуры. Составление программ разветвляющейся структуры»	2
		Лабораторная работа №5 «Составление программ разветвляющейся усложненной структуры»	2
		Лабораторная работа №6 «Составление программ циклической структуры»	2
		Лабораторная работа №7 «Составление программ усложненной структуры»	2
		Лабораторная работа №8 «Вложенные циклы»	2
2.3	Коллекции в Python	**Лабораторная работа №9** «Применение множеств»	2
		Лабораторная работа №10 «Решение задач с использованием строк»	2
		Лабораторная работа №11 «Решение задач с использованием строк, срезов строк»	2
		Лабораторная работа №12 «Списки»	2
		Лабораторная работа №13 «Решение задач с использованием кортежей»	2
		Лабораторная работа №14, 15 «Методы списков и строк.»	4
		Лабораторная работа №16, 17 «Словари в Python»	4

2.4	Функции в Python	**Лабораторная работа №18, 19** «Организация функций»	4
		Лабораторная работа №20 «Применение рекурсивных функций»	2
2.5	Библиотеки Python	**Лабораторная работа №21, 22** «Применение библиотек Python»	4
Раздел 3	**Объектно-ориентированное программирование**		**12**
3.2	Структуры	**Лабораторная работа №23** «Описание свойств структуры и действия над объектами структурного типа»	2
3.4	Наследование	**Лабораторная работа №24, 25** «Создание класса, объявление объектов»	4
		Лабораторная работа №26, 27, 28 «Создание наследованного класса»	6
		Итого:	**56**

3. Содержание лабораторных работ

Лабораторная работа №1 «Создание блок-схем линейной и разветвляющейся структуры»

Цель работы: приобретение навыков работы в построении алгоритмов.

Теоретические сведения:

Этапы решения задачи на ЭВМ. Работа по решению любой задачи с использованием компьютера делится на следующие этапы:

1. Постановка задачи.
2. Формализация задачи.
3. Построение алгоритма.
4. Составление программы на языке программирования.
5. Отладка и тестирование программы.
6. Проведение расчетов и анализ полученных результатов.

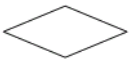
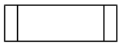
Блок-схема алгоритма — графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и блоков — графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

Линейный алгоритм –это такой алгоритм, в котором все операции выполняются последовательно одна за другой.

Алгоритмы разветвленной структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие.

Алгоритмы циклической структуры. Циклические алгоритмы подразделяют на алгоритмы с предусловием, постусловием и алгоритмы с конечным числом повторов. В алгоритмах с предусловием сначала выполняется проверка условия окончания цикла и затем, в зависимости от результата проверки, выполняется (или не выполняется) так называемое тело цикла.

В таблице приведены наиболее часто употребляемые символы.

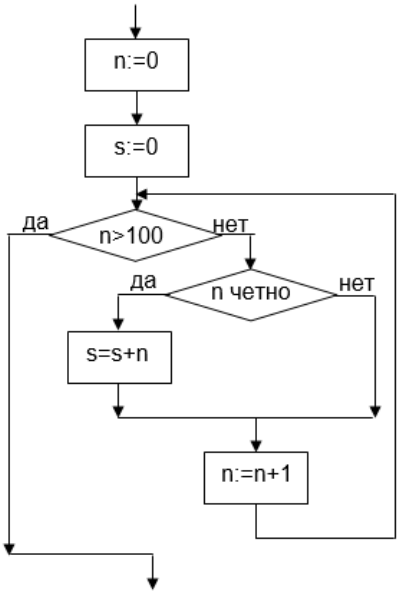
Название	Символ (рисунок)	Выполняемая функция (пояснение)
1. Блок вычислений		Выполняет вычислительное действие или группу действий
2. Логический блок		Выбор направления выполнения алгоритма в зависимости от условия
3. Блоки ввода/вывода		Ввод или вывод данных вне зависимости от физического носителя
	Вывод данных на печатающее устройство	
4. Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
5. Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
6. Блок модификации		Выполнение действий, изменяющих пункты алгоритма
7. Соединитель		Указание связи между прерванными линиями в пределах одной страницы
8. Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Правила построения блок-схем:

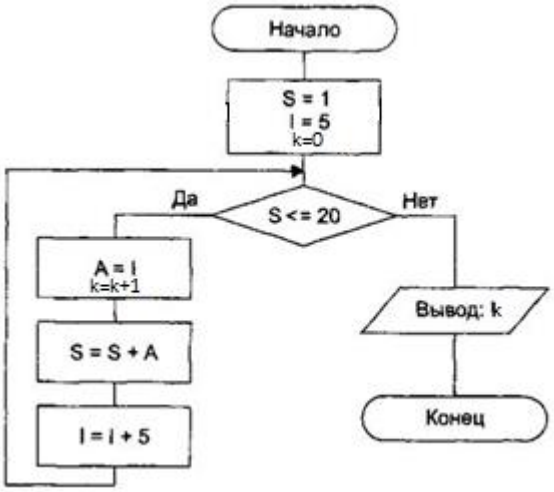
1. Блок-схема выстраивается в одном направлении либо сверху вниз, либо слева направо
2. Все повороты соединительных линий выполняются под углом 90 градусов

Практическая часть:

Задание 1. Определите значение переменной s после выполнения фрагмента алгоритма



Задание 2. Чему будет равно k ?



Задание 3. Составить блок-схему для решения задачи согласно варианту:

1.Составить алгоритм для решения квадратного уравнения ax2+bx+c=0.

2.Определить, можно ли из отрезков с длинами x,y и z построить треугольник.

3.Ввести два числа a и b. Большее число заменить утроенным произведением, меньшее–полусуммой.

4.Найти квадрат наибольшего из двух чисел a и b. Вывести на экран число 1, если наибольшим является число a , число 2–если наибольшим числом является b.

5. Написать алгоритм решения задачи, которая решает уравнение ax+b=0 относительно x для любых чисел a и b, введенных с клавиатуры.

Оформите отчет согласно требованиям. Напишите вывод о проделанной работе.

Лабораторная работа №2 «Создание блок-схемы циклической структуры»

Цель: Изучить фундаментальные алгоритмы поиска минимума и максимума.

Теоретические сведения:

Алгоритм нахождения максимального элемента массива выполняется следующим образом:

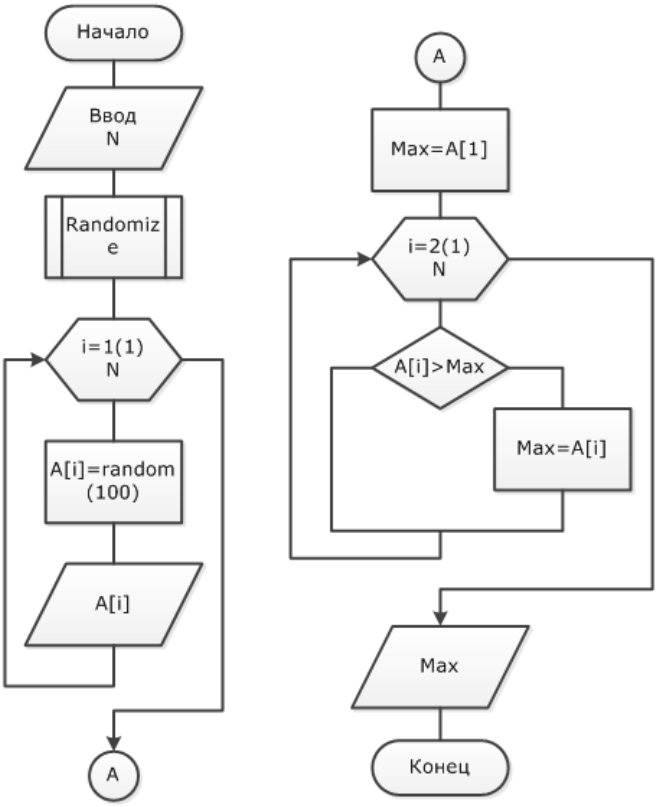
Сначала указываем, что первый элемент массива считается максимальным, иначе говоря – *Max = A[i]*.

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: Составить программу для решения следующей задачи

Вариант 1

Вычислить значения величины *y* при заданных условиях

$$y = \begin{cases} a^2 - a, при \;\; a > 10 \\ \dfrac{4{,}2}{a-7}, при \;\; 0 \le a \le 10 \\ 8a + 1, при \;\; a < 0 \end{cases}$$

Вариант 2

Вычислить значения величины *M* при заданных условиях

$$M = \begin{cases} 12x, при \;\; x > 10 \\ 3x^2, при \;\; 0 < x \le 10 \\ -x + 2, при \;\; x \le 0 \end{cases}$$

Вариант 3

Вычислить значения величины *z* при заданных условиях

$$z = \begin{cases} 2x - 2, если \;\; x > 80 \\ \sqrt{x+4}, если \;\; -10 \le x \le 80 \\ x^2 + 10, если \;\; x < -10 \end{cases}$$

Вариант 4

Вычислить значения величины *p* при заданных условиях

$$p = \begin{cases} 1{,}4e^2 + x, если \;\; x > 15 \\ \dfrac{10}{x-3}, если \;\; 0 \le x \le 15 \\ 8\sin x, если \;\; x < 0 \end{cases}$$

Вариант 5

Составить алгоритм, который по трем введенным числам определит, могут ли они являться сторонами треугольника

Вариант 6

Дано натуральное число. Определить: а) является ли оно четным; б) оканчивается ли оно цифрой 7.

Сделать выводы о проделанной работе.

Лабораторная работа №3 «Простые программы. Типичные ошибки. Хороший стиль программирования»

Цель: Изучить основные схемы простых программ. Научиться применять хороший сталь программирования при решении задач**.**

Теоретические сведения:

При построении рекурсивных функций принят традиционный в теории алгоритмов конструктивный подход: задается «базис», т.е. несколько простейших, очевидным образом вычислимых функций и способ построения из них остальных функций с помощью специальных операторов.

В качестве простейших функций в теории рекурсивных функций приняты следующие :

1. $O(x) = 0$ – константа «ноль».
2. $S(x) = x + 1$ – « последователь »
3. $I_m^n(x_1, x_2, \ldots, x_n) = x_m \; ; (m \le n)$ – функция тождества или выбора аргумента.

Эти функции можно считать простейшими, т.к. для любых значений аргументов из натурального ряда мы немедленно определяем значение функции.

Для построения примитивно-рекурсивных функций используются операторы суперпозиции и примитивной рекурсии.

Оператором суперпозиции S_m^n называется подстановка в функцию от *m* переменных *m* функций от *n* переменных, что дает новую функцию от *n* переменных. Суперпозицией функций *g* и $f_1, f_2, \ldots, f_m$ называют функцию

$$h(x_1, x_2, \ldots, x_n) = g\left[f_1(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n)\right]$$

Пример 1. Пусть $f_1(x_1, x_2) = x_1 - 2 \cdot x_2$; $f_2(x_1, x_2) = x_1^2 + x_2$;

$$f_3(x_1, x_2) = \frac{3 \cdot x_1 + x_2}{x_2}; \quad g(y_1, y_2, y_3) = C \cdot y_1 + \frac{y_2}{y_3}.$$

$$h(y_1, y_2, y_3) = C \cdot (x_1 - 2 \cdot x_2) + \frac{(x_1^2 + x_2)}{3 \cdot x_1 + x_2}.$$

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.

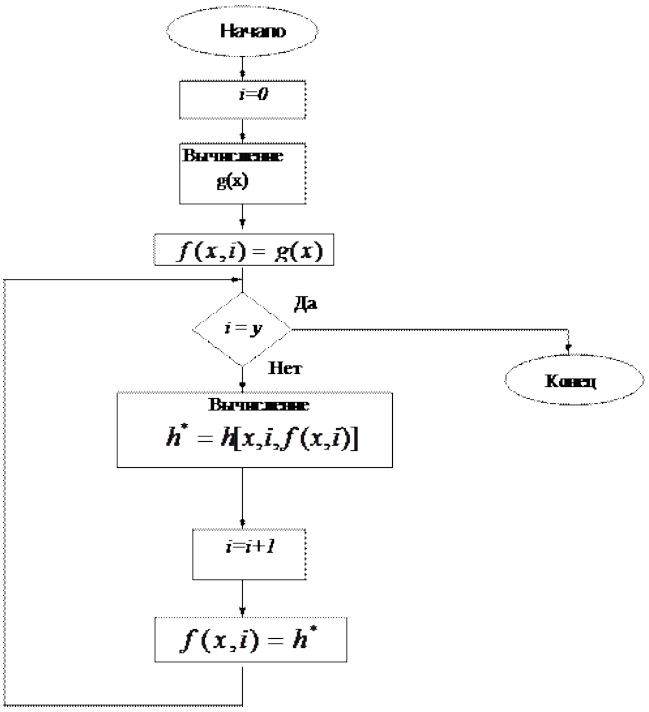


Рисунок - Алгоритм вычисления $f(x,y)$ по схеме примитивной рекурсии.

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.

Определение примитивно-рекурсивной функции

Функция называется *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.

Для того, чтобы показать, что какая-либо функция является примитивно-рекурсивной, достаточно построить ее согласно данному определению. Однако такое построение получается слишком сложным и громоздким. Поэтому в большинстве случаев данную функцию пытаются выразить с помощью суперпозиции и примитивной рекурсии через другие функции, примитивная рекурсивность которых доказана ранее.

Приведем примеры доказательства примитивной рекурсивности некоторых простых арифметических функций.

Пример 3. Константа *a* получается путем суперпозиции функций

$S(x)$ и $O(x)$:

$$a = \underbrace{S(\ldots S(S(0))\ldots)}_{a\ раз}.$$

Пример 4. Операция сложения $f_+(x,y) = x + y$ может быть определена с помощью оператора примитивной рекурсии :

$$f_+(x,0) = x = I_1^1(x,y)$$

$$f_+(x,y+1) = f_+(x,y) + 1 = S(f_+(x,y)).$$

В качестве функции $g(x)$ записана функция тождества, функция *h* во втором равенстве – это $S(x)$. Таким образом, функция $f_+(x,y)$ получена из простейших $I_1^1(x,y)$ и $S(x)$ путем применения оператора примитивной рекурсии, что соответствует определению примитивно-рекурсивной функции.

Пример 5. Примитивная рекурсивность операции умножения доказывается с использованием сложения :

$$f_*(x,0) = 0;$$

$$f_*(x,y+1) = f_*(x,y) + x = f_+(x, f_*(x,y))$$

Операция вычитания не является примитивно-рекурсивной, т.к. она не всюду определена : результат операции *a-b* при $b > a$ не определен в области натуральных чисел. Однако примитивно-рекурсивной является так называемое арифметическое вычитание.

Пример 6. Арифметическое вычитание:

$$f_-(x,y) = x \dot{-} y = \begin{cases} x - y, & \text{если } x > y \\ 0, & \text{если } x \le y. \end{cases}$$

Для доказательства примитивной рекурсивности $f_-(x,y)$ вначале рассмотрим операцию $x \dot{-} 1$:

$$f_{-1}(0) = 0 \dot{-} 1 = 0;$$

$$f_{-1}(x+1) = x;$$

т.е. операция $x \dot{-} 1$ примитивно–рекурсивна.

$$f_-(x,0) = x;$$

$$f_-(x,y+1) = x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1 = f_-(x,y) = 1.$$

Тогда :

следовательно арифметическое вычитание примитивно–рекурсивно.

Пример 7. Функция $sg(x)$ - аналог функции $sign(x)$ для натуральных чисел.

$$sg(x) = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x \ne 0. \end{cases}$$

Функция $sg(x)$ примитивно–рекурсивна :

$$sg(0) = 0,$$

$$sg(x+1) = 1.$$

При доказательстве примитивной рекурсивности не обязательно явным образом использовать оператор примитивной рекурсии.

Лабораторная работа №4 «Составление программ линейной структуры. Составление программ разветвляющейся структуры»

Цель: *научиться составлять простейшие программы линейной структуры на языке программирования. Научиться работать с условным оператором If...Then...Else.*

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 $\frac{|x|-|y|}{1+|xy|}$.

Теоретические сведения

Алгоритмическая структура ветвления программируется в TP 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 №5 «Составление программ разветвляющейся усложненной структуры»

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

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 (цикл с предусловием), 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 №6 «Составление программ циклической структуры»

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

ЗАДАНИЕ: Решить задачу, используя цикл while. Составить блок-схему, написать вывод о проделанной работе.

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 и k. Вынести на экран НОД(m, n, k). (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? В какой день расстояние, которое пробежал спортсмен превысит 15 км?
6. Дано натуральное число n? Найти первую цифру числа n. (Использовать только возможности цикла while, целочисленное деление и вычисление остатка от деления)
7. Дано натуральное число n? Сколько цифр в числе n? (функцию len() использовать нельзя)
8. Найти сумму всех n-значных чисел, кратных к (к = 1, 2, 3).
9. Напишите программу, которая имитирует проверку пароля, придуманного пользователем. Пользователь вводит пароль, потом ещё раз его же, для подтверждения. (если пароль, который ввёл пользователь (в первый раз) короче 8 символов, программа выводит "Короткий!" и завершает свою работу; если пароль достаточно длинный, но введённый во второй раз пароль не совпадает с первым, программа выводит "Различаются."; если же и эта проверка пройдена успешно, программа выводит "OK" (латинскими буквами).
10. Дано натуральное число n? Чему равна сумма его цифр? (Использовать только возможности цикла while, целочисленное деление и вычисление остатка от деления)
11. 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).
12. 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...).

ЗАДАНИЕ: Решить задачу, используя цикл for. Составить блок-схему, написать вывод о проделанной работе.

1. Найдите все трёхзначные и четырёхзначные числа Армстронга. (Число Армстронга. Числом Армстронга считается натуральное число, сумма цифр которого, возведенных в n-ную степень (N – количество цифр в числе) равна самому числу. Например, 153 = 13 + 53 + 33.)
2. Дано натуральное число n. Вычислить: s = $1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\ldots-\frac{1}{2}$;
3. Дано натуральное число n. Вычислить: s= $\frac{2}{1}+\frac{3}{2}+\frac{4}{3}+\ldots+\frac{m+1}{m}$.
4. Дано натуральное число n. Вычислить: s= 1!+2!+3!+…+n! (n>1).

5. Напишите программу, которая считывает одно натуральное число и выводит на первой строке все делители этого числа в порядке возрастания, разделённые пробелами, а на второй — «ПРОСТОЕ» или «НЕТ» в зависимости от того, простым было введённое число или нет.

6. С клавиатуры вводится натуральное число n > 0, потом n чисел, каждое на новой строке. Вычислите и напечатайте знакочередующуюся сумму ряда (первое число прибавить, второе вычесть, третье прибавить и т.д.)

Задача*

Напишите программу, которая отгадывает загаданное целое число от 1 до 1000 (пользователь загадывает число в уме и не сообщает программе). Угадать число нужно не более чем за 10 попыток. На каждую попытку пользователь отвечает, что названное число больше загаданного (вводит символ “>”), меньше загаданного (“<”) или угадано правильно (“=”).

Используйте бинарный поиск. Программа должна каждый раз называть число, находящееся посередине исследуемого диапазона — в результате станет ясно, в какой половине диапазона находится искомое число.

Задача**

Мы находимся на острове, на котором закопан клад. Мы находимся в точке с координатами (0, 0) и смотрим на север. Нам известно, где закопан клад, но этого мало: остров полон опасностей, и нужно перемещаться строго по указаниям карты, которая, к счастью, тоже имеется в нашем распоряжении. Мы хотим найти клад как можно скорее.

Известны координаты клада и указания, которым нужно следовать, чтобы его найти. Каждое указание карты состоит из одного слова и, возможно, одного натурального числа. Слово — одно из набора: «вперёд», «налево», «направо», «разворот» или «стоп».

После слова «вперёд» следует количество шагов, которое следует пройти в том направлении, куда мы в данный момент смотрим. Слова «налево» или «направо» означают, что нужно изменить направление взгляда под прямым углом, «разворот» — что прямо на обратное. Команда «стоп» означает остановку.

Найдите минимальное количество указаний карты, которое нужно выполнить, чтобы прийти к кладу.

Формат ввода

Сначала вводятся два числа на отдельных строчках: координаты клада по оси икс (запад-восток) и игрек (юг-север). Затем следует некоторое количество указаний карты. Каждое указание карты состоит из одного слова и, возможно, одного натурального числа на отдельной строке. Слово — одно из набора: «вперёд», «налево», «направо», «разворот» или «стоп».

Формат вывода

Программа выводит на отдельных строках минимальное количество указаний карты, которое нужно выполнить, чтобы прийти к кладу, и направление взгляда в этот момент (одно из: «север», «юг», «запад», «восток»). Гарантируется, что карта приводит к кладу.

Лабораторная работа №7 «Составление программ усложненной структуры»

Цели:• *Содействовать формированию у студентов навыков работы с циклами;*

•Обеспечить закрепление способов работы с операторами циклов FOR.

Задание:

Составить программы с использованием циклов For (цикл с параметром) и Repeat (цикл с постусловием) для задачи, решенной вами в лабораторной работе № 6.

Лабораторная работа №8 «Вложенные циклы»

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

Задание: Написать программу для решения предложенной задачи своего варианта. Составить блок-схему, сделать выводы о проделанной работе

1. Дано число n, найти наибольшее по сумме цифр число. Вывести на экран это число и сумму его цифр.
2. Введен промежуток от n до m. Найти числа количество делителей у которых не меньше введенного значения. Для найденных чисел вывести на экран количество делителей и все делители.
3. Дано число n и последовательность чисел, введенных пользователем. Сколько раз встречается заданная цифра в введенной последовательности чисел. Количество вводимых чисел и цифра, которую необходимо посчитать, задаются вводом с клавиатуры.
4. Дано число n и последовательность чисел больших 2. Посчитать, сколько среди них простых чисел.

Дано число n, найти наибольшее по сумме цифр число. Вывести на экран это число и сумму его цифр.

```
n = int(input())
max_s = 0
max_m = 0
while n != 0:
    m = n
    s = 0
    while n>0:
        s += n%10
        n //= 10
    if s > max_s:
        max_s = s
        max_m = m
    n = int(input())
print('Число',max_m,'имеет максимальную сумму цифр:', max_s)
```

Введен промежуток от n до m. Найти числа количество делителей у которых не меньше введенного значения. Для найденных чисел вывести на экран количество делителей и все делители.

```
a = int(input("Минимум: "))
b = int(input("Максимум: "))
n = int(input("Минимальное количество делителей: "))
```

```
while a <= b:
    m = 0
    for i in range(1,a+1):
        if a%i == 0:
            m += 1
    if m >= n:
        print(a,'-',m,end=' - ')
        for i in range(1,a+1):
            if a%i == 0:
                print(i,end=' ')
        print()
    a += 1
```

Дано число n и последовательность чисел, введенных пользователем. Сколько раз встречается заданная цифра в введенной последовательности чисел. Количество вводимых чисел и цифра, которую необходимо посчитать, задаются вводом с клавиатуры.

```
n = int(input("Сколько будет чисел? "))
d = int(input("Какую цифру считать? "))
count = 0
for i in range(1,n+1):
    m = int(input("Число " + str(i) + ": "))
    while m > 0:
        if m%10 == d:
            count += 1
        m = m // 10

print("Было введено %d цифр %d" % (count, d))
```

Дано число n и последовательность чисел больших 2. Посчитать, сколько среди них простых чисел.

```
from math import sqrt

count = 10
for i in range(10):
    n = int(input())
    for j in range(2, int(sqrt(n))+1):
        if n%j == 0:
            count -= 1
            break

print(count)
```

Лабораторная работа №9 «Применение множеств»

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

- *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 №10 «Решение задач с использованием строк»

Цель работы: в ходе выполнения лабораторной работы выработать навыки работы со структурой данных.

Теоретические сведения

Тип String (строка) в Турбо Паскале широко используется для обработки текстов. Этот тип является стандартным и во многом похож на одномерный массив символов Array [0..N] of Char. Значение N соответствует количеству символов в строке и может меняться от 0 до 255. Символы, входящие в строку, занимают позиции с 1 до N. Начальный байт строки с индексом 0 содержит информацию о ее длине, т.е. это символ с кодом, равным длине строки.

Можно, также описывать переменные типа String[K], где K - целое число не больше 255. Так определяются строки с длиной не больше K. Этот тип уже не является стандартным. С символами строки можно работать как с элементами массива из символов, но в отличие от массивов, строки можно вводить целиком, сравнивать друг с другом и сцеплять операцией "+".

Сравнение строк выполняется посимвольно в соответствии с их кодами до первого несовпадения. Если одна из строк закончилась до первого несовпадения, то она считается меньшей. Пустая строка меньше любой строки

Переменная строкового типа (String) может рассматриваться как массив элементов символьного типа (Char). Например, если в программе определены переменные S: string; C: char; и задано S:='Москва', то S[1]='M', S[2]='o' и т. д. и возможно присвоение, например: C:= S[1];

Существует ряд стандартных функций и процедур для работы со строками.

- Функция Length(s) выдает длину строки s.
- Функция Concat(s1,s2,..,sn) возращает строку s1+s2+..+sn.
- Функция Copy(s,p,k) возвращает фрагмент строки s, который начинается в позиции p и имеет длину k.
- Функция Pos(s1,s) ищет первое вхождение подстроки s1 в строку s и возвращает номер первого символа s1 в строке s или 0 если не нашли.
- Процедура Delete(s,p,k) удаляет из строки s фрагмент, который начинается в позиции p и имеет длину k.
- Процедура Insert(s,s1,p) вставляет в строку s1 подстроку s, начиная с заданной позиции p.

Турбо паскаль позволяет производить преобразования числовых значений в строковые и наоборот. Для этого используются процедуры Str(X:n:d,S) и Val(S,X,e). Первая получает их числа X строку S с изображением этого числа, в которой не менее n символов и из них d знаков после запятой. Параметры n и d необязательные. Вторая процедура получает из строки S число X. При успешном результате e=0.

Теоретические сведения

Символьный тип данных

Значениями символьного типа **Char** являются элементы конечного и упорядоченного множества символов. Чаще всего используются символы американского стандарта ASCII. Значения типа Char записываются одним символом (например, ‘*’, ‘S’, ‘2’). В рамках этого типа десятичные цифры упорядочены в соответствии с их числовыми значениями (например, ‘5’<’6’). Буквы упорядочены в алфавитном порядке (например, ‘B’<’C’).

Стандартные функции работы с символьным типом

Ord(W)	Определение порядкового номера символа W
Chr(N)	Определение символа по его порядковому номеру N

Символы с кодами от 0 до 127 составляют так называемую основную таблицу кодов ASCII. Эта часть идентична на всех IBM- совместимых компьютерах. Цифрам от 0 до 9 соответствуют коды от 48 до 57, буквам латинского алфавита от A до Z - коды от 65 до 90, буквам от a до z – от 97 до 122, буквам русского алфавита от А до Я – коды от 128 до 159.

Строковый тип данных

Тип данных **String** предназначен для хранения последовательности символов. Строка должна быть заключена в апострофы. Максимальная длина строки указывается в квадратных скобках. Если она не указана, максимальная длина полагается 255. Важной особенностью является то, что к каждому символу строки возможен доступ по его номеру.

Стандартные функции и процедуры работы со строковым типом

ПРОЦЕДУРЫ	
Delete(S, P, N)	Удаление N символов из строки S, начиная с позиции P
Insert (W, S, P)	В строку S , начиная с позиции P, вставляется строка W. Если результат превысит 255 символов, строка обрывается
Str(V, S)	Число V преобразуется в строку, результат в S
Val (S, V, W)	Если строка состоит только из цифр, то они преобразуются в числовое значение переменной V, значение W равно0. в противном случае, когда строка состоит не только из цифр, - преобразование не выполнится, W <>0 признак ошибки
ФУНКЦИИ	
Copy(S, P, N)	Из строки S, начиная с позиции P , выбирается N символов
Length(S)	Определяется длина строки S, т.е. число символов из которых она состоит
Pos(W, S)	В строке S отыскивается первое вхождение строки W (номер позиции). Если вхождение нет, то выдается 0
Concat(S1, S2, …Sn) или S1+S2+… +Sn	Строки S1, S2, …Sn записываются одна за другой . Если результат превысит 255 символов, строка обрывается.
Trim(S)	Удаление начальных и конечных пробелов строки

Пример выполнения задания:

Условие Организовать цикл ввода символов и вывода его кода. Ввод закончить если введена точка **VAR** B: Char; **BEGIN** **Rereat** Readln(b); Writeln(Ord(b)); **Until** b=’.’; **END**.	**Условие**: Ввести строку. Вывести первые два символа и два последних символа через тире **VAR** S, s1,s2, w: String; K: Integer; **BEGIN** Readln(S); S1:=copy(S,1,2); K:=Length(S); S2:=copy(S,k-1,2); W:=S1+’-‘+S2; Writeln(W); **END**.

Контрольные вопросы

- В чем разница между символьной и строковой величинами?
- Какова максимально возможная длина строки?
- Перечислить функции, которые используются для работы с символьными и со строковыми данными.

Практические задания

1	1. Ввести слово. Вывести “да”, если первый и последний символ совпадает, в противном случае вывести “нет”. 2. Ввести строку. Определить количество цифр, которые она содержит.

	3. Ввести строку. Удалить из нее все буквы А.
2	1. Ввести слово. Первую и последнюю букву поменять местами. 2. Ввести строку. Определить каких букв больше А или О. 3. Ввести строку и слово. Удалить слово из строки, если она в ней содержится.
3	1. Ввести слово. Определить является ли оно десятичной записью целого числа. 2. Определить сколько раз в строке встречается сочетание “ht”. 3. Ввести строку, содержащую формулу. Определить правильно ли в ней расставлены скобки (т.е. находится ли справа от каждой открывающейся скобки соответствующая ей закрывающаяся скобка).
4	1. Вывести первое слово в исходной строке. 2. Ввести строку. Подсчитать , сколько в ней букв R, K, L. 3. Ввести строку. Изменить ее так, чтобы все латинские буквы стали заглавные.
5	1. Ввести слово .Вывести его без первой и последней буквы. 2. Ввести строку. Заменить в ней каждую точку на троеточие. 3. Ввести строку. Определить сколько раз в строке встречается каждый символ (например, дано kooaaooa, результат а- 3 раза, о -4 раза, к – 1 раз).
6	1. Удалить среднюю букву при нечетной длине строки и две средние буквы при четной длине строки. 2. Вывести второе слово в исходной строке. 3. Ввести строку. Найти длину самого короткого слова.
7	1. Ввести строку. Вывести вторую половину строки. 2. Определить сколько раз встречается “о” в первом слове строки. 3. Ввести строку. Слова, следующие за точкой и первое слово должны начинаться с заглавной буквы. Исправить строку, если это не так.
8	1. Ввести строку. Определить содержится ли в первой половине слова + 2. Ввести строку. Вывести на экран слова, содержащие три буквы. 3.Ввести строку, содержащую латинские буквы. Определить, каких букв больше заглавных или строчных
9	1. Ввести строку, которая содержит символ +. Определить, сколько символов следует после него. 2. Ввести строку. «Задвоить» каждый символ строки (например, дано ABCD, результат AABBCCDD). 3. Ввести строку. Удалить из нее все буквы А и О.
10	1. Ввести слово, в котором содержится буква А, причем не на последнем месте. Вывести символ, следующий за А. 2. Определить количество слов в строке, которые начинаются и заканчиваются одним и тем же символом. 3. Ввести строку из нескольких слов. Разделитель слов пробел. Удалить лишние пробелы, оставив только один пробел между словами.

11	1. Ввести слово. Вывести слово, полученное путем перестановки местами половинок введенного слова. 2. Ввести строку. Подсчитать , сколько в ней символов * D F H. 3.Ввести строку. Удалить из нее все цифры
12	1. В строке есть одна точка с запятой; определить количество символов до нее и после. 2. Ввести строку. Определить сколько раз встречается сочетание abc. 3. Ввести строку. Изменить ее так, чтобы все латинские буквы стали строчными.
13	1. Ввести число N и один символ сформировать строку, в которой символ повторяется N_раз . 2. Ввести строку. Заменить в ней строчные буквы на заглавные. 3. Ввести два слова. Определить можно ли из букв первого слова составить второе слово.
14	1. Ввести строку. Удалить из нее все буквы О. 2. Ввести строку. Определить количество заглавных букв. 3. Ввести строку. Заменить в ней сочетание NO на YES.

Лабораторная работа №11 «Решение задач с использованием строк, срезов строк»

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

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.

Задачи повышенной сложности

1. Зашифровать введенную с клавиатуры строку, поменяв местами первый символ со вторым, третий с четвертым и т. д. Затем провести дополнительную шифровку результата смещением кода. Провести дешифровку.
2. Составить программу, организующую перемещение текстового окна 8x8 по экрану. См. задачу 2. Движение начинается по нажатию клавиши и заканчивается либо по нажатию клавиши, либо при достижении окном края экрана. Варианты движения: а) из левого верхнего угла в правый нижний угол. При неточном "попадании" в нижний угол смещать окно по одной из сторон до точной остановки в углу. б) из левого нижнего угла в правый верхний. в) из центра экрана к одной из боковых сторон. При достижении края размер окна

Лабораторная работа №12 «Списки»

Цель: *научиться использовать списки для решения задач на языке программирования*

Задания:

1. Дана строка, в которой слова разделены одним пробелом. Замените первые буквы всех слов на заглавные (если слово начинается с заглавной буквы, оставьте без изменения).
2. Дана строка, в которой слова разделены одним пробелом. Подсчитайте, сколько букв 'w' встречается в каждом слове.
3. Дана строка, в которой слова разделены одним пробелом. Подсчитайте, сколько в каждом слове букв, совпадающих с его первой буквой.
4. Дана строка. Преобразуйте ее так, чтобы сначала следовали цифровые символы, а затем все остальные. Порядок следования символов между собой не изменять.
5. Выведите на экран самое короткое слово из введенной строки.
6. Напишите программу, которая вводит строку и выводит ее, сокращая каждый раз на 1 символ до тех пор, пока в строке не останется 1 символ.
7. Введите два целых числа. Преобразуйте числа в две строки, объедините их в одну строку и выведите результат на экран.
8. Выясните, какая из букв - первая или последняя - встречается в строке чаще.
9. Вычислите суммы позиций, на которых в слове стоят буквы ‘п’ и ‘т’.
10. Посчитайте число различных символов в слове.
11. Слово состоит из двух частей одинаковой длины и соединительной гласной. Найдите обе части этого слова.
12. Выведите на печать подстроку, заключенную между двумя запятыми.
13. Даны две строки (одинаковой длины). Создайте третью строку из символов, которые на одинаковых позициях совпадают.
14. В строке, состоящей из латинских букв, подсчитайте количество гласных букв.
15. Задано существительное первого склонения, оканчивающееся на ‘а’. Напечатайте это слово во всех падежах.

Контрольные вопросы

1. Что будет являться результатом работы функции побайтового копирования строк, если длина строки-источника превосходит допустимый размер строки-приемника?
2. Что будет являться результатом работы функции побайтового копирования строк, если длина строки-источника меньше размера строки-приемника?
3. Почему при сравнении строк важен регистр символов?
4. Как сравниваются строки разной длины?
5. Какие возможны последствия при обращении к неинициализированной строке?
6. Почему функция изменения регистра символов строки может некорректно работать с кириллицей?
7. Вычеркните из заданного слова все буквы, совпадающие с его последней буквой

Лабораторная работа №13 «Решение задач с использованием кортежей»

Цель работы: *научиться решать задачи с использованием кортежей.*

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 №14, 15 «Методы списков и строк.»

Цель: *Научиться использовать стандартные методы для работы со строками при решении задач на языке программирования*

Теоретические сведения:

Операция	**Описание**	**Пример**
`s2 in s`	Проверка, что подстрока `s2` содержится в `s`	`'m' in 'team'`
`s2 not in s`	Проверка, что подстрока `s2` не содержится в `s` то же, что `not (s2 in s)`	`'I' not in 'team'`
`s + s2`	Конкатенация (склейка) строк, то есть строка, в которой сначала идут все символы из `s`, а затем все символы из `s2`	`'tea' + 'm' == 'team'`
`s * k`	Строка `s`, повторенная `k` раз	`'ha' * 3 == 'hahaha'`
`s[n]`	n-й элемент строки, отрицательные `n` — для отсчета с конца	`'team'[2] == 'a'` `'team'[-1] == 'm'`
`s[start:stop:step]`	Срез строки	`'mama'[:2] == 'ma'`
`len(s)`	Длина строки	`len('abracadabra') == 11`

`s.find(s2)`
`s.rfind(s2)`
	Индекс начала первого или последнего вхождения подстроки `s2` в `s` (вернет -1, если `s2 not in s`)	`s = 'abracadabra'`
		`s.find('ab') == 0`
		`s.rfind('ab') == 7`
		`s.find('x') == -1`

`s.count(s2)`	Количество неперекрывающихся вхождений `s2` в `s`	`'abracadabra'.count('a') == 5`
`s.startswith(s2)` `s.endswith(s2)`	Проверка, что `s` начинается с `s2` или оканчивается на `s2`	`'abracadabra'.startswith('abra')`
`s += s2` `s *= k`	Заменить содержимое строки на `s + s2` и `s * k`соответственно	

`s.isdigit()`
`s.isalpha()`
`s.isalnum()`
	Проверка, что в строке `s` все символы — цифры, буквы (включая кириллические), цифры или буквы соответственно	`'100'.isdigit()`
		`'abc'.isalpha()`
		`'E315'.isalnum()`

`s.islower()` `s.isupper()`	Проверка, что в строке s не встречаются большие буквы, маленькие буквы. Обратите внимание, что для обеих этих функций знаки препинания и цифры дают True	`'hello!'.islower()` `'123PYTHON'.isupper()`
`s.lower()` `s.upper()`	Строка s, в которой все буквы (включая кириллические) приведены к верхнему или нижнему регистру, т. е. заменены на строчные (маленькие) или заглавные (большие)	`'Привет!'.lower() == 'привет!'` `'Привет!'.upper() == 'ПРИВЕТ!'`
`s.capitalize()`	Строка s, в которой первая буква — заглавная	`'привет'.capitalize() == 'Привет'`

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Строка s, у которой удалены символы пустого пространства (пробелы, табуляции) в начале,
	в конце или с обеих сторон
		`' Привет! '.strip() == 'Привет!'`

`s.ljust(k, c)` `s.rjust(k, c)`	Добавляет справа или слева нужное количество символов c, чтобы длина s достигла k	`'Привет'.ljust(8, '!') == 'Привет!!'`
`s.join(a)`	Склеивает строки из списка a через символ s	`'+'.join(['Вася', 'Маша']) == 'Вася+Маша'`
`s.split(s2)`	Список всех слов строки s (подстрок, разделенных строками s2)	`'Раз два три!'.split('а') == ['Р', 'з дв', ' три!']`
`s.replace(s2, s3)`	Строка s, в которой все неперекрывающиеся вхождения s2 заменены на s3 Есть необязательный третий параметр, с помощью которого можно указать, сколько раз производить замену	`'Раз два три!'.replace('а', 'я') =='Ряз двя три!'` `'Раз два три!'.replace('а', 'я', 1) == 'Ряз два три!'`
`list(s)`	Список символов из строки строки s	`list('Привет') == ['П', 'р', 'и', 'в', 'е', 'т']`

`bool(s)`	Проверка, что строка не пустая (возвращает `True`, если не пустая, и `False` в противном случае)	
`int(s)` `float(s)`	Если в строке `s` записано целое (дробное) число, получить это число, иначе — ошибка	`int('25') == 25`
`str(x)`	Представить любой объект `x` в виде строки	`str(25) == '25'`

Задания:

16. Дана строка, в которой слова разделены одним пробелом. Замените первые буквы всех слов на заглавные (если слово начинается с заглавной буквы, оставьте без изменения).
17. Дана строка, в которой слова разделены одним пробелом. Подсчитайте, сколько букв 'w' встречается в каждом слове.
18. Дана строка, в которой слова разделены одним пробелом. Подсчитайте, сколько в каждом слове букв, совпадающих с его первой буквой.
19. Дана строка. Преобразуйте ее так, чтобы сначала следовали цифровые символы, а затем все остальные. Порядок следования символов между собой не изменять.
20. Выведите на экран самое короткое слово из введенной строки.
21. Напишите программу, которая вводит строку и выводит ее, сокращая каждый раз на 1 символ до тех пор, пока в строке не останется 1 символ.
22. Введите два целых числа. Преобразуйте числа в две строки, объедините их в одну строку и выведите результат на экран.
23. Выясните, какая из букв - первая или последняя - встречается в строке чаще.
24. Вычислите суммы позиций, на которых в слове стоят буквы ‘п’ и ‘т’.
25. Посчитайте число различных символов в слове.
26. Слово состоит из двух частей одинаковой длины и соединительной гласной. Найдите обе части этого слова.
27. Выведите на печать подстроку, заключенную между двумя запятыми.
28. Даны две строки (одинаковой длины). Создайте третью строку из символов, которые на одинаковых позициях совпадают.
29. В строке, состоящей из латинских букв, подсчитайте количество гласных букв.
30. Задано существительное первого склонения, оканчивающееся на ‘а’. Напечатайте это слово во всех падежах.

Контрольные вопросы

8. Что будет являться результатом работы функции побайтового копирования строк, если длина строки-источника превосходит допустимый размер строки-приемника?
9. Что будет являться результатом работы функции побайтового копирования строк, если длина строки-источника меньше размера строки-приемника?

10. Почему при сравнении строк важен регистр символов?
11. Как сравниваются строки разной длины?
12. Какие возможны последствия при обращении к неинициализированной строке?
13. Почему функция изменения регистра символов строки может некорректно работать с кириллицей?
14. Вычеркните из заданного слова все буквы, совпадающие с его последней буквой

Лабораторная работа №16, 17 «Словари в Python»

Словарь (в Python он называется `dict`) — тип данных, позволяющий, как и список, хранить много данных. В отличие от списка, в словаре для каждого элемента можно самому определить «индекс», по которому он будет доступен. Этот индекс называется **ключом**.

Создание словаря

Вот пример создания словаря для энциклопедии об актерах мирового кино:

```
actors = {
 'Джонни Депп': 'Джон Кристофер Депп Второй родился 9 июня 1963 года '
                'в Овенсборо, Кентукки...',
 'Сильвестр Сталлоне': 'Сильвестр Гарденцио Сталлоне родился в Нью-Йорке. '
                       'Его отец, парикмахер Фрэнк Сталлоне — иммигрант из Сицилии...',
 'Эмма Уотсон': 'Эмма Шарлотта Дуерр Уотсон родилась в семье английских адвокатов. '
                'В пять лет переехала вместе с семьей из Парижа в Англию...',
 # ...
 }
```

Создание словаря

Элементы словаря перечисляются в фигурных скобках (как и элементы множества!) и разделяются запятой. До двоеточия указывается ключ, а после двоеточия — значение, доступное в словаре по этому ключу.

Пустой словарь можно создать двумя способами:

```
d = dict()

# или так

d = {}
```

Вспомните, что создать пустое множество можно, только используя функцию `set()`. Теперь понятно, почему это так — пустые фигурные скобки зарезервированы для создания словаря.

Обращение к элементу словаря

После инициализации словаря мы можем быстро получать статью про конкретного актера:

```
print(actors['Эмма Уотсон'])
```

Важно!

Обращение к элементу словаря выглядит как обращение к элементу списка, только вместо целочисленного индекса используется ключ. В качестве ключа можно указать выражение: Python вычислит его значение, прежде чем обратится к искомому элементу.

```
first_name = 'Сильвестр'

last_name = 'Сталлоне'

print(actors[first_name + ' ' + last_name])
```

Если ключа в словаре нет, возникнет ошибка:

```
print(actors['Несуществующий ключ'])
```

```
KeyError: 'Несуществующий ключ'
```

Добавление и удаление элементов

Важная особенность словаря — его динамичность. Мы можем добавлять новые элементы, изменять их или удалять. Изменяются элементы точно так же, как в списках, только вместо целочисленного индекса в квадратных скобках указывается ключ:

```
actors['Эмма Уотсон'] = 'Новый текст статьи об Эмме Уотсон'
```

Также в словари можно добавлять новые элементы и удалять существующие.

Добавление элемента

Добавление синтаксически выглядит так же, как и изменение:

```
actors['Брэд Питт'] = 'Уильям Брэдли Питт, более известный как Брэд
Питт — ' \

                      'американский актёр и продюсер. ' \

                      'Лауреат премии «Золотой глобус» за 1995 год,
...'
```

Удаление элемента

Для удаления можно использовать инструкцию `del` (как и в списках):

```
del actors['Джонни Депп']

# больше в словаре нет ни ключа 'Джонни Депп',

# ни соответствующего ему значения
```

```
print(actors['Джонни Депп'])
```

```
KeyError: 'Джонни Депп'
```

Удаление элемента

Удалять элемент можно и по-другому:

```
actors.pop('Джонни Депп')
```

Единственное отличие этого способа от вызова `del` — он возвращает удаленное значение. Можно написать так:

```
deleted_value = actors.pop('Джонни Депп')
```

В переменную `deleted_value` положится значение, которое хранилось в словаре по ключу 'Джонни Депп'. В остальном этот способ идентичен оператору `del`. В частности, если ключа 'Джонни Депп' в словаре нет, возникнет ошибка KeyError.

Важно!

Чтобы ошибка не появлялась, этому методу можно передать второй аргумент. Он будет возвращен, если указанного ключа в словаре нет. Это позволяет реализовать безопасное удаление элемента из словаря:

```
deleted_value = actors.pop('Джонни Депп', None)
```

Если ключа 'Джонни Депп' в словаре нет, в `deleted_value` попадет None.

Проверка наличия элемента в словаре

Оператор `in` позволяет проверить, есть ли ключ в словаре:

```
if 'Джонни Депп' in actors:
    print('У нас есть статья про Джонни Деппа')
```

Проверить, что ключа нет, можно с помощью аналогичного оператора `not in`:

```
if 'Сергей Безруков' not in actors:
    print('У нас нет статьи о Сергее Безрукове')
```

Лабораторная работа №18, 19 «Организация функций»

***Цель**: Научиться оформлять внешние функции.*

Ход работы:

Теоретические сведения:

Набор встроенных функций в языке программирования достаточно широк (ABS, SQR, TRUNC и т.д.). Если в программу включается новая, нестандартная функция, то ее необходимо описать в тексте программы, после чего можно обращаться к ней из программы. Обращение к функции осуществляется в правой части оператора присваивания, с указанием имени функции и фактических параметров. Функция может иметь собственные локальные константы, типы, переменные, процедуры и функции. Описание функций в Паскале аналогично описанию процедур.

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.

Процедуры и функции состоят из операторов. Локальных данных и внутренних процедур и функций

О*бмен* аргументами и результатами между основной программой и процедурой производится через *параметры* (*формальные* и *фактические*).

Правила соответствия между формальными и фактическими параметрами: соответствие *по количеству*, соответствие *по последовательности* и соответствие *по типам*.

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

Отличительные особенности функций: - результат выполнения - одно значение, которое присваивается имени функции и передается в основную программу; - имя функции может входить в выражение как операнд.

Пример 1. Написать подпрограмму-функцию степени ax, где a, x – любые числа. Воспользуемся формулой: ax = ex ln a

```
program p2;
var  f,  b, s, t, c, d : real; { глобальные параметры}
       function  stp (a, x : real) : real;
       var   y : real;  { локальные параметры}
          begin
             y := exp (x * ln ( a)) ;
             stp:= y;{присвоение имени функции результата вычислений подпр-мы}
     end;  { описание функции закончено }
 begin            {начало основной программы }
  d:= stp  (2.4, 5); {вычисление степеней  разных  чисел и переменных }
  writein (d,  stp (5,3.5));
       read (f, b, s, t);    c := stp  (f, s)+stp  (b, t);
  writeln (c);
 end.
```

Пример 2. Псевдослучайные числа. Функция, возвращающая значение и меняющая параметр

```
var s, i: integer;

function next(var seed: integer): integer;
    const
        multiplier = 37;
        increment = 3;
        cycle = 64;
    begin
        next := seed;
        seed := (multiplier * seed + increment) mod cycle
    end;

begin
    s := 16;
    writeln(next(s));
    writeln(next(s));
    writeln(next(s));
    writeln(next(s));

    s := 16;
    for i := 1 to 64 do
        write(next(s):3);

readln
end.
```

Использование var в заголовке – необычный прием для функции.

Будучи вызвана с параметром s, содержащим число 16, эта функция возвращает число 16. Вместе с тем, возвращая 16, функция изменяет значение, хранящееся в переменной seed (или s), на 19. Если функцию вызвать снова, с полученным значением s, то она возвратит число 19 и изменит значение s на 2. Продолжая вызывать функциюnext, получим определенную последовательность целых чисел, начинающуюся с исходного значения s (цикл for).

Замечательным свойством этой последовательности является то, что каждое значение от 0 до 63 встречается в ней один раз. Более того, шестьдесят пятый вызов функции next дает число 16 и начинается новый цикл. Другими словами, начиная с любого желаемого целого, функция генерирует фиксированную перестановку чисел от 0 до 63.

Такой прием используется преимущественно для генерации "случайных" чисел (правильнее их называть псевдослучайными – чтобы подчеркнуть их предсказуемость).

Цикл из 64 чисел, конечно, слишком мал; Грогоно предлагает константы для генерации перестановки чисел от 0 до 65 535:

const multiplier = 25173; increment = 13849; cycle = 65536;

Подбор констант с нужными свойствами – нетривиальная задача.

Приведенная выше функция возвращает значение и изменяет значение параметра. Такой способ действий применяется нечасто. В большинстве функций нет необходимости изменять параметры, и, следовательно, нет смысла использовать слово var в заголовке.

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 №20 «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.

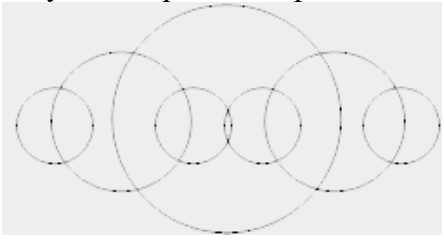
Математически конечность последовательности независимо от начального i не доказана, но на практике последовательность останавливается всегда.

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:



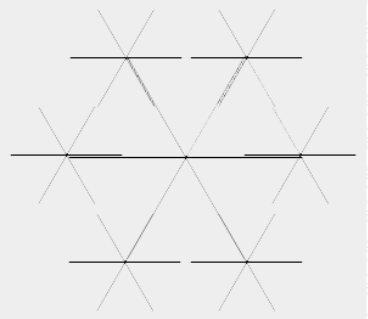
Изображение строится по следующему правилу: строится окружность с заданным радиусом r. Затем на диаметрально противоположных точках окружности (x- r и x+

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:

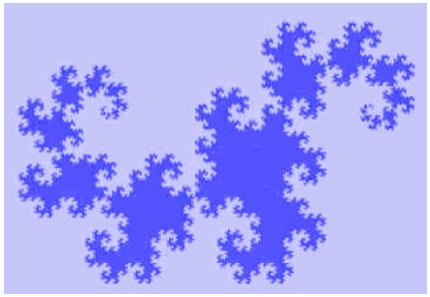
Рекурсивная программа построения снежинки



```
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 №21, 22 «Применение библиотек Python»

Цель: *Способствовать формированию навыков создания и использования библиотеки подпрограмм на примере модуля, формированию навыков применения процедур создания графических примитивов.*

Ход работы:

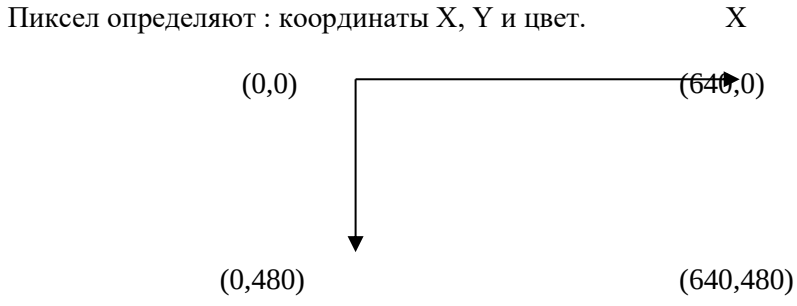
1. ***Изучить теоретическую часть***
2. ***Выполнить задания, опираясь на примеры из теоретической части***
3. ***Ответить на контрольные вопросы***
4. ***Предъявить преподавателю результаты работы (результаты работы программ из примеров и индивидуальной части) и исходные коды.***
5. ***Оформить отчет в соответствии с ходом работы***

🕮 Теоретическая часть

В *графическом режиме* экран рассматривается как последовательность точек (пикселов), из которых строится изображение.

Количество пикселов в строке и количество строк на экране характеризуют его *разрешающую способность.*

640x480 - в строке 640 пикселов , а строк всего 480.



Для работы в графическом режиме разработана библиотека GRAPH, содержащая множество графических процедур и набор драйверов.

Модуль Graph представляет собой библиотеку подпрограмм Turbo Pascal, обеспечивающих полное управление графическими режимами

различных мониторов - CGA, EGA, VGA, SVGA. Библиотека содержит более 50 графических процедур и функций, как базовых (рисование точек, линий, окружностей и т.п.), так и расширяющих возможностей базовых (многоугольники, закрашивание фигур, вывод текста и др.).Для запуска программ, использующих модуль Graph, должен быть доступен один (или несколько) графических драйверов (BGI файлов), например, egavga.bgi. Для компилирования программы, использующей модуль Graph, нужно указать путь к файлу GRAPH.TPU (в строке ввода **Options|Directories|Unit Directories**) и иметь доступ к стандартным модулям в файле TURBO.TPL

Основные директивы модуля GRAPH

Uses GRAPH - подключение модуля GRAPH.

InitGraph(gd,gm, Путь к драйверу) - инициирование графического режима,

gd- переменная типа Integer, определяет тип графического драйвера,

gm- переменная того же типа, задающая режим работы графического адаптера;

Путь к драйверу -выражение типа String, содержащее путь к файлу драйверу (egavga.bgi)

Для задания двух первых параметров в модуле Graph определены специальные константы, но проще всего использовать для указания типа драйвера константу с именем ***Detect***, что позволяет не указывать режим.

Путь к драйверу представляет собой последовательность, состоящую из имени диска и списка имен каталогов и подкаталогов, разделенных символом "\" и заключенную в апострофы, в которых находится файл драйвера. Например, 'P:\BP\BGI'.

Если скопировать файл egavga.bgi в текущий каталог (из которого вы вошли в среду Турбо Паскаль), то путь к файлу состоит из двух апострофов ('').

Если в программе есть вызов процедуры ***InitGraph***, то должен присутствовать соответствующий вызов процедуры ***CloseGraph***. Эта

процедура очищает экран и переводит дисплей в текстовый режим. В одной программе можно несколько раз выполнять инициализацию графического режима и его закрытие.

Для очистки экрана в графическом режиме используется процедура ***ClearDevice***.

Установка цвета и стиля заполнения

SetColor(< *константа определяющая цвет*>***:word);*** **-** установка цвета графического изображения.

SetBkColor(<*константа определяющая цвет фона*>***:word);*** **-** установка цвета фона;

SetFillStyle(<константа стиля заполнения>***:word;*** <константа цвета заполнения>***:word);*** **-** установка способа закраски.

Таблица цветов

Константа		**Цвет**
Имя	**Значение**	
Black	0	Черный
Blue	1	Синий
Green	2	Зеленый
Cyan	3	Бирюзовый
Red	4	красный
Magenta	5	малиновый
Brown	6	коричневый
LightGray	7	светло-серый
DarkGray	8	темно-серый
LightBlue	9	ярко-голубой
LightGreen	10	ярко-зеленый
LightCyan	11	ярко-бирюзовый
LightRed	12	ярко-красный
LightMagenta	13	ярко-малиновый
Yellow	14	желтый
White	15	белый

Таблица констант для стандартных стилей заполнения.

Константа		**Стиль заполнения**
Имя	**Значение**	
EmptyFill	0	заполнение цветом фона
SolidFill	1	заполнение текущим цветом

LineFill	2	Заполнение символами ---
LtslashFill	3	заполнение символами // нормальной толщ.
SlashFill	4	заполнение символами // удвоенной толщ.
BkslashFill	5	заполнение символами \\ удвоенной толщ.
LtbkSlashFill	6	заполнение символами \\ нормальной толщ.
HatchFill	7	заполнение вертикально-горизонтальной штриховкой тонкими линиями
XhatchFill	8	заполнение штриховкой крест-накрест по диагонали «редкими» тонкими линиями
InterLeaveFil	9	заполнение штриховкой крест-накрест по диагонали «частыми» тонкими линиями
WideDotFill	10	заполнение «редкими» точками
CloseDotFill	11	заполнение «частыми» точками

Процедуры создания графических примитивов

1. Текущий указатель.

При построении изображения иногда надо указать точку начала вывода. В текстовом режимах эту точку указывает курсор, который присутствует на экране (его можно убрать). В графическом режимах видимого курсора нет, но есть невидимый текущий указатель. Для перемещения текущего указателя по экрану дисплея служит процедура ***MoveTo(x,y)*** - перемешает указатель в точку с координатами (x,y).

MoveRel(dx, dy) **-** перемещает указатель на dx точек по горизонтали и dy точек по вертикали от предыдущей позиции.

Примеры

```
MoveTo(200,100);
 MoveRel(5,10); {указатель переместится в точку (205,110)}
```

Чтобы определить максимальное значение координат X Y для установленного видеорежима, используют функции

GetMaxX : integer; максимум по X

GetMaxy : integer; максимум по Y

Установить указатель в центр экрана.

```
var  Xcentr, Ycentr : integer;
begin
  Xcentr:=GetMaxX  div  2;
  Ycentr:=GetMaxy  div  2;
  MoveTo(Xcentr, Ycentr);
```

2. Вывод точки

PutPixel(x,y : integer; *<цвет точки>****);***

где x,y координаты точки.

3. Вывод отрезка

Line(x1,y1,x2,y2);

(x1,y1) - координаты начала отрезка

(x2,y2) - координаты конца отрезка

!!! Обратите внимание на то, что в процедуре не задается цвет. В этом и аналогичных случаях цвет определяется процедурой SetColor().

LineTo(x,y) - строит отрезок из точки текущего положения указателя в точку с координатами (x,y).

LineRel(dx,dy) - строит отрезок из точки текущего положения указателя в точку с координатами (x+dx, y+dy)

4. Построение прямоугольника

Rectangle(x1,y1,x2,y2:integer);

Bar(x1,y1,x2,y2:integer) - рисует прямоугольник и закрашивает его цветом и стилем, определенным в процедуре SetFillStyle().

5. Построение дуг, окружностей, эллипсов.

Circle(x,y,<радиус> : word); - окружность указанного радиуса

Ellipse(x,y:integer; <нач_угол>,<кон_угол> ***:word; xR,yR : word) -*** построение эллиптических дуг.

X,Y - координаты центра,

xR, yR - длина горизонтальной и вертикальной полуосей в пикселах.

Угол отсчитывается против часовой стрелки и указывается в градусах. Дуга эллипса вычерчивается от заданного начального угла до конечного угла. Если значение начального угла 0, а конечного 360 - будет построен полный эллипс.

6. Построение закрашенного эллипса:

FillEllipse(x,y:integer; xR,yR);

X,Y - координаты центра,

xR, yR - длина горизонтальной и вертикальной полуосей в пикселах.

Стиль заполнения области внутри эллипса устанавливается процедурой SetFillStyle(), а самого эллипса - SetColor().

7. Заполнение внутренней или внешней области замкнутой фигуры.

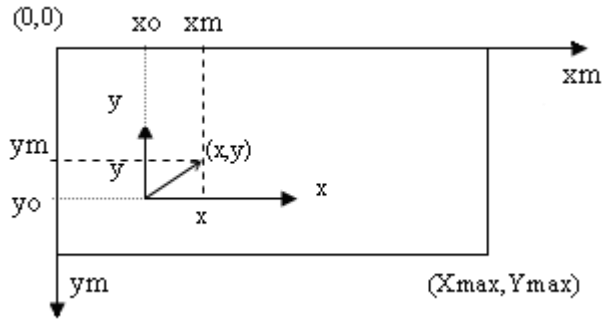
FloodFill(x,y:integer; <*цвет границы области*>***);***

Стиль задан SetFillStyle()

X,Y - координаты точки внутри (или вне) замкнутой области.

Машинные координаты

Начало машинной системы координат, направление осей, а также максимальные значения координат монитора показаны на рисунке



На рисунке приведена также машинная (x_m, y_m) и физическая (x,y) системы координат. Для изображения на экране точки с физическими координатами (x,y) необходимо определить ее машинные координаты (x_m, y_m). Расчетные формулы имеют следующий вид (попробуйте самостоятельно получить эти формулы):

$x_m=x_0+x*M_x$,

$y_m=y_0-y*M_y$,

где M_x, M_y-масштабы соответственно по осям x и y, которые показывают число пикселей в одной физической единице,

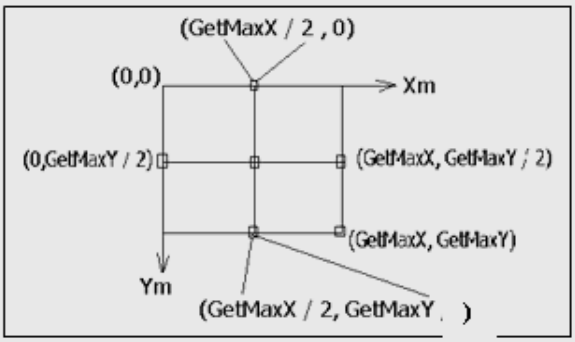
x, y- физические координаты точки,

x_m, y_m- машинные координаты точки,

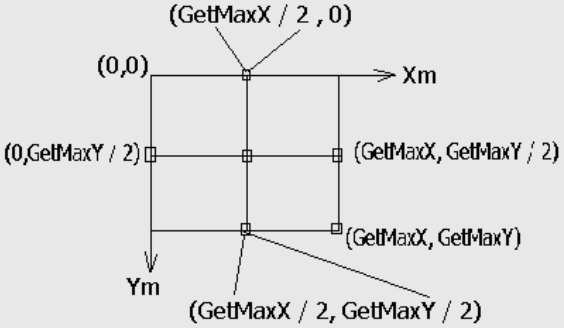
x_0, y_0- машинные координаты начала физической системы координат.

Пример 1. Нарисовать через весь экран горизонтальную и вертикальную линии, пересекающиеся в центре монитора.

Этапы разработки программы сведены в таблицу.

N	*Этапы программирования*	*Выполнение*
1.	*Постановка задачи*	Нарисовать через весь экран горизонтальную и вертикальную линии, пересекающиеся в центре монитора. *Изобразим вид экрана с указанием координат требуемых линий*
2.	*Математическое описание*	
3.	*Разработка структограммы*	Описание gd ,gm : integer Инициализация графики Рисование линии Закрытие графики
4.	*Написание программы*	**Program P5;** **Uses graph;** {подключение граф.модуля} **Var gd,gm:integer;** **Begin** **gd:=detect;**{определение граф. драйвера**}** **InitGraph(gd,gm,' ');**{инициализация графики**}** **Line(0, round(GetMaxY/2), GetMaxX,** **Round(GetMaxY/2));**{гориз. лин.} **Line(round(GetMaxX/2), 0,** **Round(GetMaxX/2), GetMaxY);** {вертик. лин.} **Readln;** {пустой ввод**}** **CloseGraph;** {закрытие графики} **End.**
5.	*Отладка и получение результатов*	*Выполнить самостоятельно*

Пример 2. Написать программу построения графика функции $y=x^2$ для $x\in[-1;1]$.

N	*Этапы программирования*	*Выполнение*
1.	*Постановка задачи*	Построить график функции $y=x^2$ для $x\in[-1;1]$. *Изобразим вид экрана, который мы хотели бы получить после выполнения программы.*
2.	*Математическое описание*	

3.	*Разработка структограммы*	Описание gd,gm, x,y,xm,ym,x0,y0,Mx,My
		Ввод x0,y0,Mx,My
		Инициализация графики
		For xm:=20 to 620 do
		x:=(xm-x0)/Mx;
		y:=sqr(x);
		ym:=y0-y*My;
		PutPixel(xm,ym,1);
		Закрытие графики

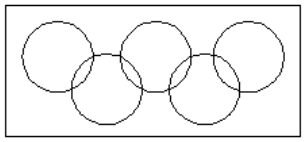
4.	*Написание программы*	**Program P6;**
		Uses graph;
		Var gd,gm:integer;
		x,y:real;
		x0,y0,xm,ym,Mx,My:integer;
		begin
		gd:=detect;
		InitGraph(gd,gm,' ');
		Mx:=300; x0:=320; My:=440; y0:=460;
		{выбраны для монитора 640×480 пикс.**}**
		For xm:=20 to 620 do
		Begin
		x:=(xm-x0)/Mx;
		y:=sqr(x);
		ym:=round(y0-y*My);
		PutPixel(xm,ym,1);
		End;
		Readln;
		CloseGraph;
		End.

5.	*Отладка и получение результатов*	*Выполнить самостоятельно*

Пример 3. Построить в центре экрана синий прямоугольник, закрасив его линиями вида \\ темно-серого цвета. Фон экрана сделать белым.

```
Program graph1;
uses Graph;
var Driver, Mode : integer;
begin
Driver:=Detect;                  {инициализация графического}
InitGraph(Driver, Mode,'')       {    режима}
SetBkColor(15);                  { установка цвета фона - белый}
SetColor(1);                     {установка текущего цвета - синего}
Cleardevice;              {очистка экрана установленным цветом фона}
SetFillStyle(5,8);        {установка стиля заполнения}
Rectangle(290,290, GetMaxX-290, GetMaxy-290);   {прямоугольник}
FloodFill(301,230,1); {заполнение прямоугольника выбранным   стилем}
ReadLn;
CloseGraph;
end.
```

Пример 4: Написать программу, которая выводит флаг Олимпийских игр. Изображение флага приведено ниже (одной клетке соответствует пять пикселов).

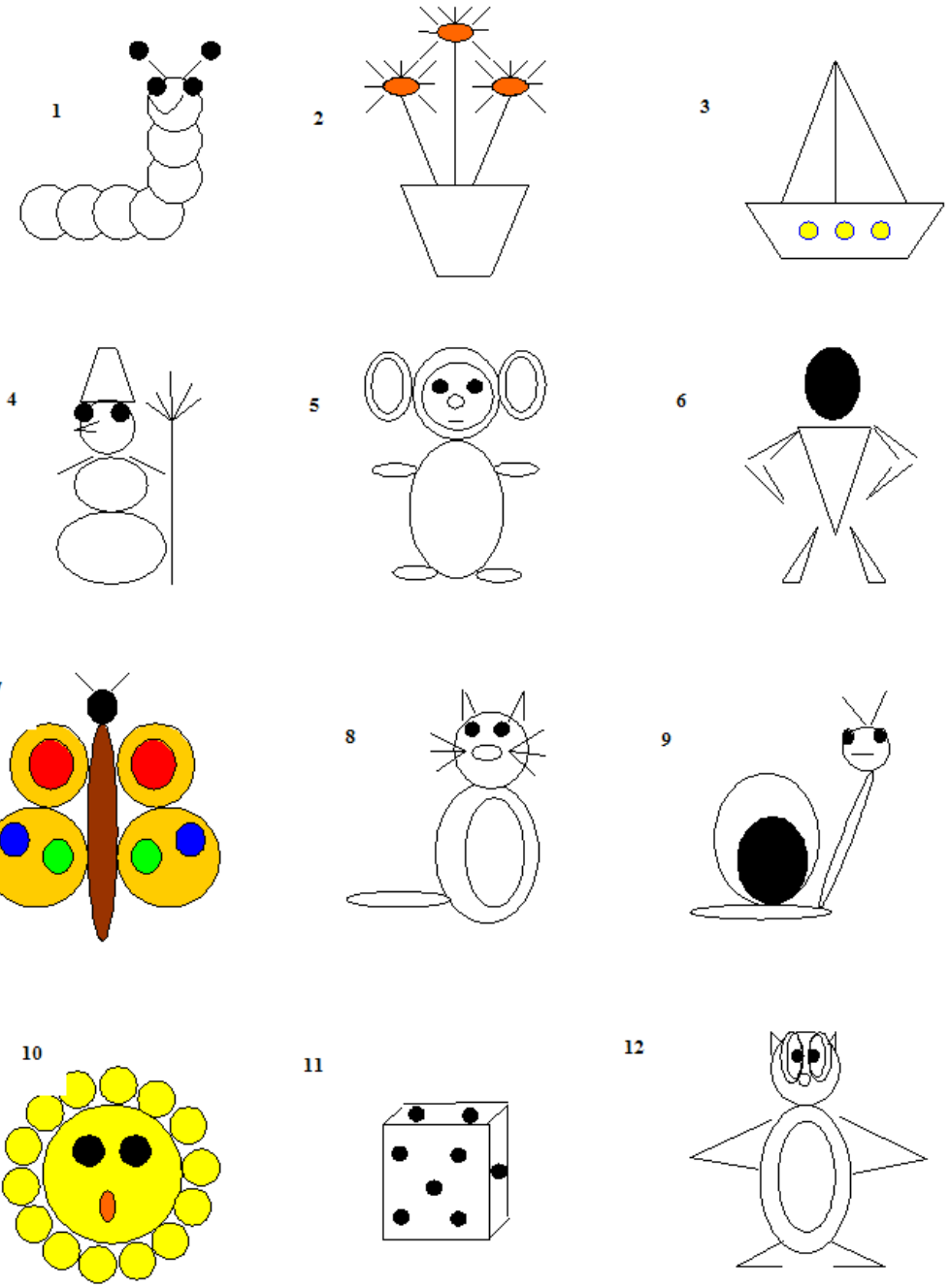


```
program Flag;
uses Graph;
var Driver: integer;
    Regim: integer;
    Path:string;
ErrCode:integer;
Begin
Driver:=VGA;
Regim:=VGAHi;
Path:='e:\tp\bgi';
InitGraph(Driver,Regim.Path);
ErrCode:=GraphResult;
If ErrCode<>grOk then
    Begin
        Writeln ('Ошибка инициализации графического режима.');
        Writeln('Для завершения работы программы нажмите <Enter>');
        Readln;
        Halt(1);
    end;
    SetFillStyle(SolidFill,LightGray);
    Bar(80,80,200,135);
    SetColor(Green);
    Circle(100,100,15);
    SetColor(Black);
    Circle(140,100,15);
    SetColor(Red);
    Circle(180,100,15);
    SetColor(Yellow);
    Circle(120,110,15);
    SetColor(Blue);
    Circle(160,110,15);
    Readln;
    CloseGraph;
End.
```

Задание:

1. Изучить теоретическую часть лабораторной работы.

2. Выполнить отладку программ и получить результаты (**Пример 1**, **Пример 2, Пример 3**).
3. Выполнить задания по вариантам, используя модуль GRAPH. Использовать для ориентира **Пример 4.**



Контрольные вопросы:

1. В каком модуле языка Паскаль хранятся подпрограммы по работе с графикой? Как подключить его к основной программе?

2. Как инициализировать графический режим и как из него выйти?
3. Как инициализировать графический режим, если неизвестно, какой графический драйвер используется на данной машине?
4. Всегда ли необходимо явно задавать все параметры процедуры InitGraph?
5. Как направлены оси координат в графическом режиме языка Паскаль? Где расположено начало координат?
6. Что такое пиксел и каковы его характеристики?
7. Пусть Ваш экран имеет стандартную разрешающую способность 640x480 ед. Какие координаты будет иметь при этом левый верхний угол экрана, левый нижний угол, правый верхний угол, правый нижний угол?
8. Перечислить процедуры создания графических примитивов, назовите их формат.
9. Какими процедурами можно нарисовать окружность?
10. Сколько основных цветов могут использоваться языком Паскаль 7.0? Приведите примеры обозначения цветов.
11. Какие типы штриховки Вам известны?
12. К каким результатам приводит обращение к процедуре FloodFill?
13. Как на языке Паскаль установить режим рисования фиолетовым цветом по розовому фону?
14. С помощью каких известных Вам операторов языка Паскаль можно нарисовать треугольник?

Лабораторная работа №24, 25 «Создание класса, объявление объектов»

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.

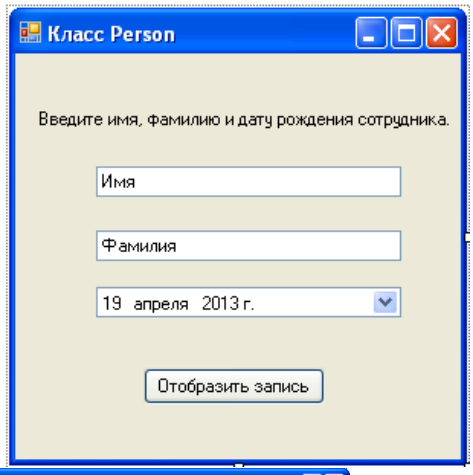
Создание проекта Person Class

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:

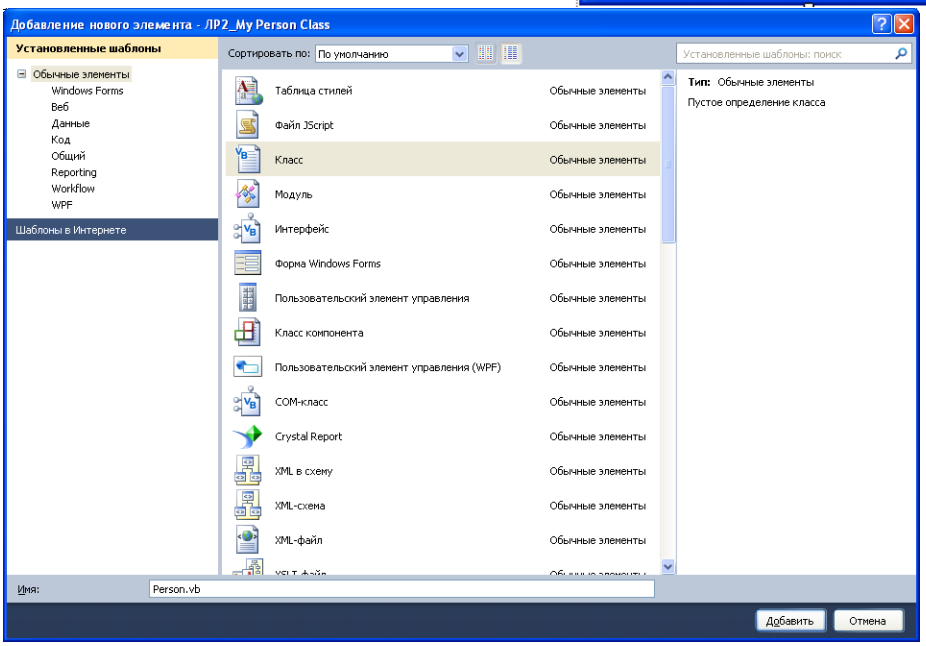
Объект	Свойство	Установка
Label1	Text	Введите имя, фамилию и дату рождения сотрудника.
TextBox1	Text	Имя
TextBox2	Text	Фамилия
Button1	Text	Отобразить запись
Form1	Text	Класс Person

7. Ваша форма должна выглядеть примерно так.



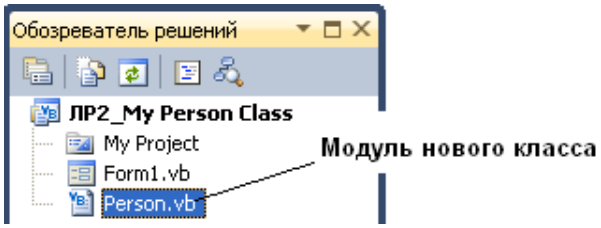
Это базовый интерфейс пользователя для формы, которая определяет запись нового сотрудника фирмы. (Эта форма не подключена к базе данных, так что храниться может только одна запись.) Теперь вы должны добавить в проект класс для хранения информации из этой записи.

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.

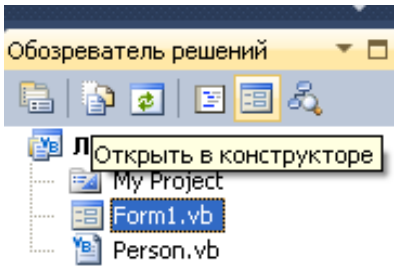
```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal Value As String)
        Name1 = Value
    End Set
End Property
```

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:

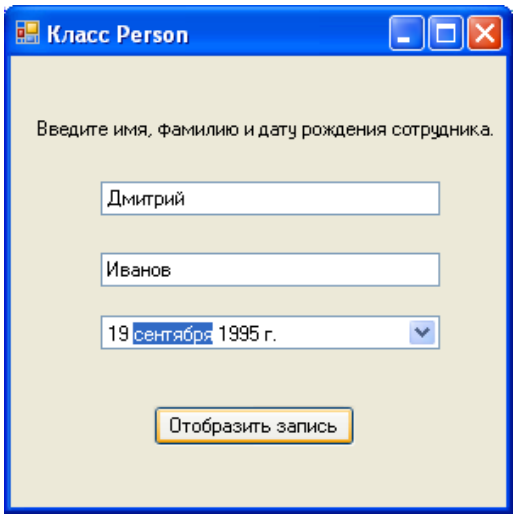
```
Dim Employee As New Person
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " в возрасте " & Employee.Age(DOB) & "лет.")
```

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.

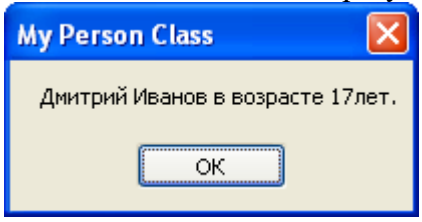
4. Чтобы запустить программу, щелкните на кнопке Начать отладку (F5). В среде разработки появится интерфейс пользователя, готовый к приему ваших данных.
5. Введите в текстовое поле First Name ваше имя, а в текстовое поле Last Name - фамилию.
6. Щелкните на раскрывающемся списке объекта выбора даты и 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:

1. Определите понятия класс, экземпляр класса, объект.
2. На какие этапы можно разбить процесс создания экземпляров класса.
3. Как создать базовый файл класса, какой код он содержит.
4. Синтаксис объявления переменной класса.
5. Создание в классе нового свойства.
6. Создание в классе нового метода.
7. Объявление переменной объекта для использования в классе.
8. Обращение к свойствам и методам объекта

Лабораторная работа №26, 27, 28 «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

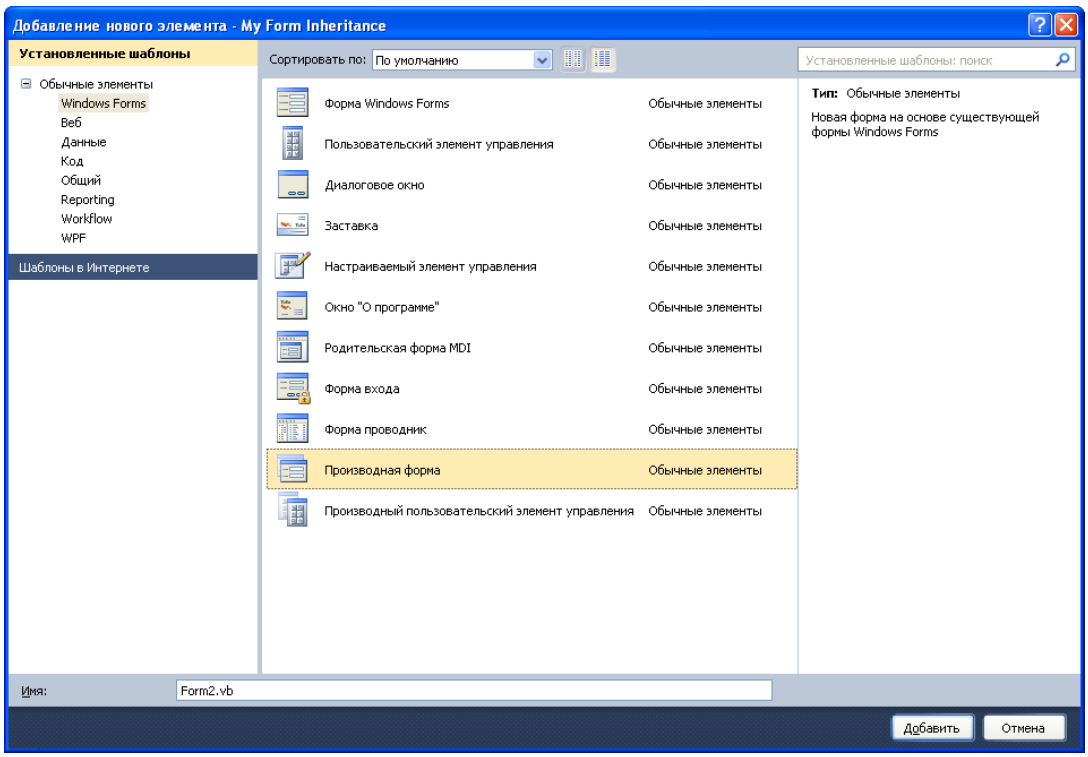
Вы неосознанно всегда использовали наследование для определения форм Windows, которые использовали при создании приложений Visual Basic. Хотя существующие формы могут наследоваться с помощью кода программы, разработчики Visual Studio .NET сочли эту задачу важной и разработали специальный инструмент среды 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

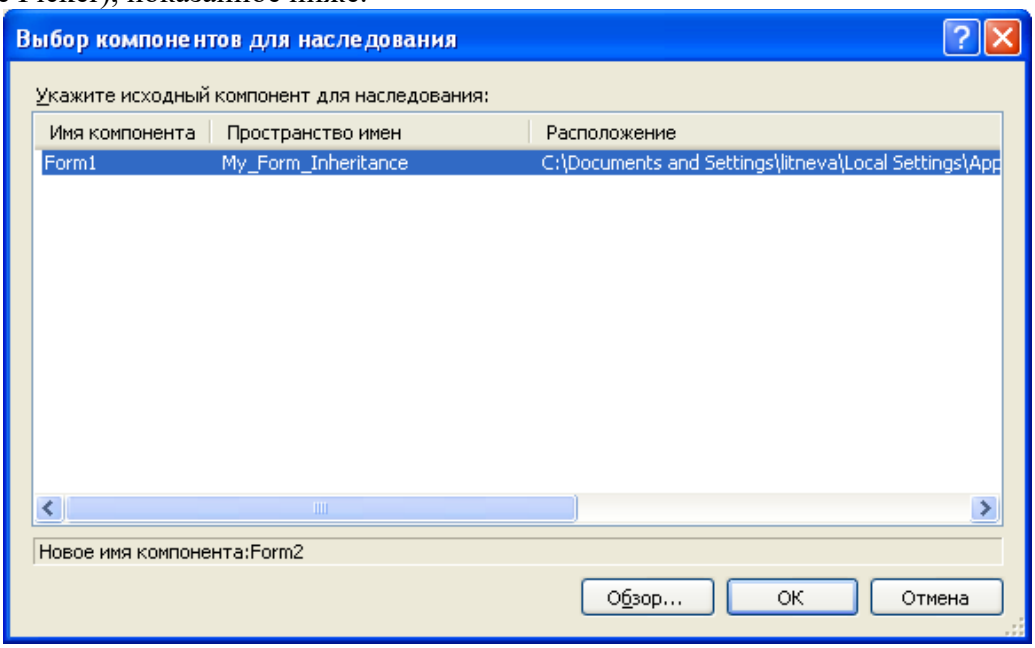
1. Запустите Visual Studio и создайте новый проект с именем **My Form Inheritance**.
2. Отобразите форму проекта и используйте элемент управления Button для добавления в нижнюю часть формы двух расположенных рядом объектов кнопок.
3. Измените свойства Text кнопок Button1 и Button2 на "OK" и "Отмена" 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.



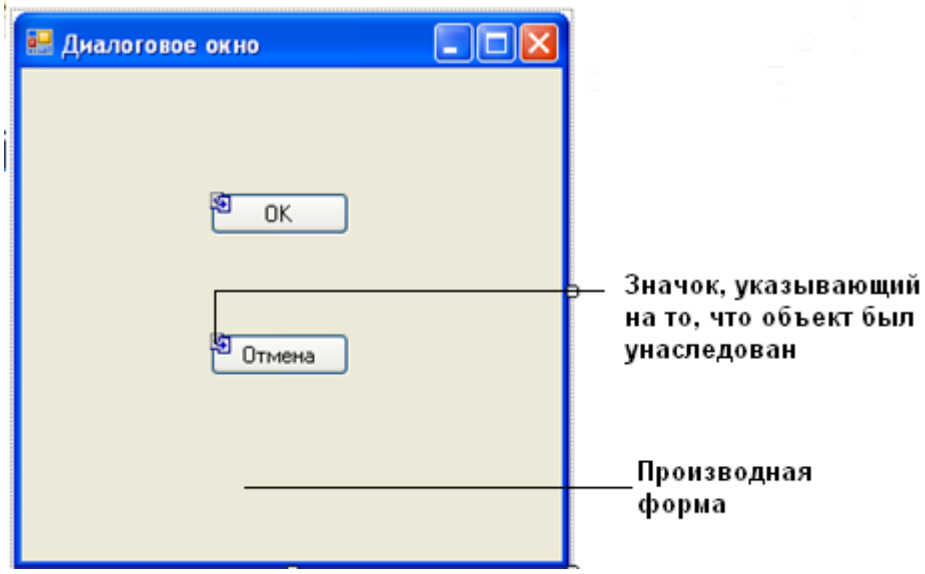
10. Щелкните на Добавить (Add), чтобы принять для новой формы установки по умолчанию. Visual Studio отобразит диалоговое окно Выбор компонентов для наследования (Inheritance Picker), показанное ниже.



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.

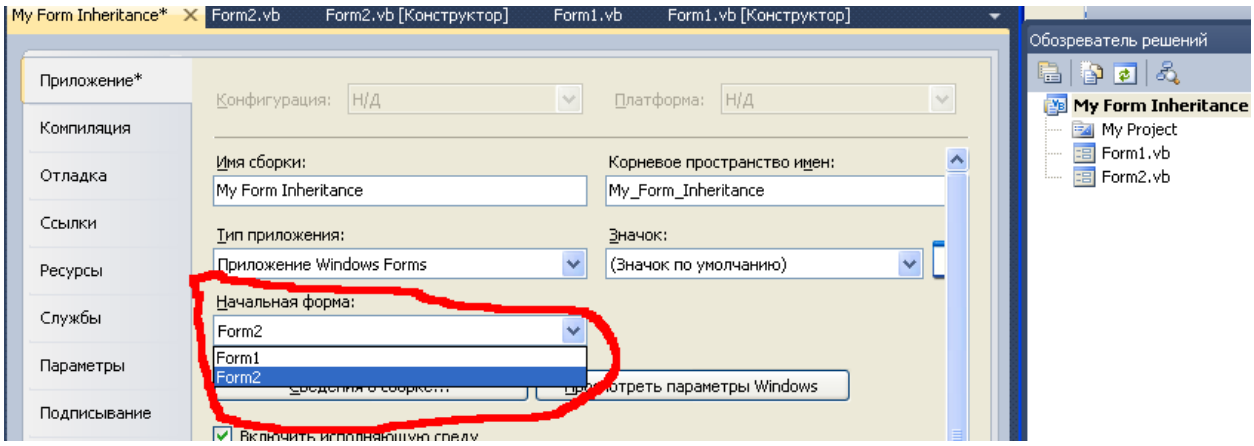


Иногда сложно отличить производную форму от базовой (маленькие значки наследования не так очевидны), так что используйте для различения этих форм Обозреватель решений (Solution Explorer) и закладки окон среды разработки.

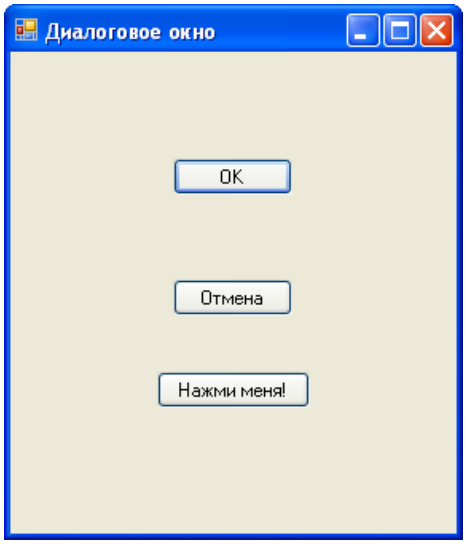
Теперь вы должны добавить к производной форме несколько новых элементов.

Настройка производной формы

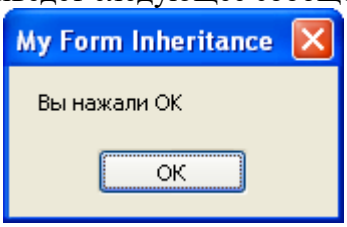
1. Используйте элемент управления Button и добавьте на Form2 (производная форма) третью кнопку.
2. Установите свойство Text этой кнопки равным "Нажми меня!".
3. Сделайте двойной щелчок мышью на кнопке **Нажми меня**!
4. В процедуре события `Button3_Click` введите следующий оператор программы: `MsgBox("Это 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
```

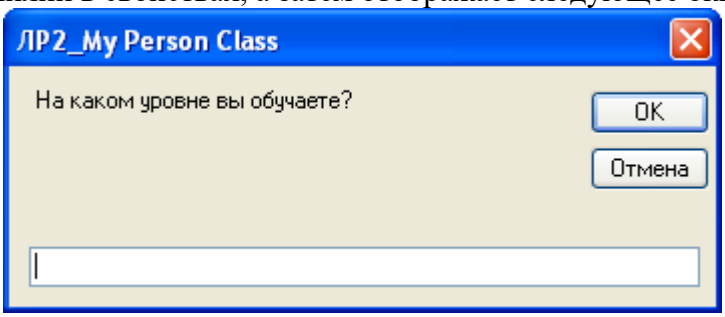
Оператор `Inherits` связывает класс `Person` с этим новым классом, встраивая в него все свои переменные, свойства и методы. Если бы класс `Person` находился в отдельном модуле или проекте, вы могли бы указать его расположение, задав пространство имен, точно так же, как вы указываете классы библиотеки .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.

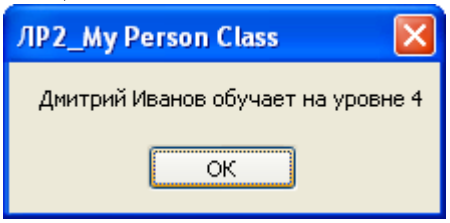
```
Dim Employee As New Teacher
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
Employee.Grade = InputBox("На каком уровне вы обучаете?")
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " обучает на уровне " & Employee.Grade)
```

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.

4. Информационное обеспечение обучения

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

Основные источники:

1. Васильев А.Н. Python на примерах [Электронный ресурс]: практический курс по программированию/ Васильев А.Н.— Электрон. текстовые данные.— Санкт-Петербург: Наука и Техника, 2017.— 432 c.

2. Коврижных А.Ю. Основы алгоритмизации и программирования. Часть 1. Задачи и упражнения. Практикум [Электронный ресурс]: учебно-методическое пособие/ Коврижных А.Ю., Кончук Е.А., Лузина Г.Е.— Электрон. текстовые данные.— Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2016.— 52 c.

3. Коврижных А.Ю. Основы алгоритмизации и программирования. Часть 2. Расчетные работы. Практикум [Электронный ресурс]: учебно-методическое пособие/ Коврижных А.Ю., Кончук Е.А., Лузина Г.Е.— Электрон. текстовые данные.— Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2016.— 44 c.

4. Лубашева Т.В. Основы алгоритмизации и программирования [Электронный ресурс]: учебное пособие/ Лубашева Т.В., Железко Б.А.— Электрон. текстовые данные.— Минск: Республиканский институт профессионального образования (РИПО), 2016.— 379 c.

5. Семакин И.Г. Основы программирования и баз данных [Текст]: учеб. для студентов учреждений среднего проф. образования / И. Г. Семакин. - М.: Академия, 2017. - 224 с. - (Профессиональное образование. Информатика и вычислительная техника)

6. Семакин И.Г. Основы алгоритмизации и программирования [Текст]: учеб. / И. Г. Семакин, А. П. Шестаков. - 4-е изд., стер.;. - М.: Академия, 2017. - 304 с. - (Профессиональное образование. Информатика и вычислительная техника)

7. Сузи Р.А. Язык программирования Python [Электронный ресурс]/ Сузи Р.А.— Электрон. текстовые данные.— Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 350 c.

8. Шелудько В.М. Основы программирования на языке высокого уровня Python [Электронный ресурс]: учебное пособие/ Шелудько В.М.— Электрон. текстовые данные.— Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017.— 146 c.

9. Шелудько В.М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули [Электронный ресурс]: учебное пособие/ Шелудько В.М.— Электрон. текстовые данные.— Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017.— 107 c.

Дополнительные источники:

1. Семакин И.Г., Шестаков А.П. Основы алгоритмизации и программирования. Практикум –М.: ОИЦ «Академия», 2016

2. Разумавская Е.А. Алгоритмизация и программирование [Электронный ресурс]: практическое пособие/ Разумавская Е.А.— Электрон. текстовые данные.— Санкт-Петербург: Санкт-Петербургский юридический институт (филиал) Академии Генеральной прокуратуры РФ, 2015.— 49 c.

3. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум для СПО/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Саратов: Профобразование, 2019.— 155 c.