АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ СОЦИАЛЬНЫЙ ИНСТИТУТ»

Утверждаю
Декан факультета
_____Ж.В. Игнатенко
«15» сентября 2025 г.

**Методические указания**
к семинарам и по выполнению самостоятельной работы
ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Специальность: 09.02.09 Веб-разработка

Квалификация: разработчик веб-приложений

Направленность: Разработка веб-приложения на стороне клиента

Форма обучения очная

Разработано
Преподаватель
_____ Зеденская Э.В.

Рекомендована
на заседании кафедры ПИМ
от «15» сентября 2025 г.
протокол № 2
Зав. кафедрой _____ Д.Г.Ловянников

Подписан цифровой подписью:
Федоровский Александр Петрович
Дата: 2025.11.14 14:32:21+03'00'

Ставрополь, 2025г.

## СОДЕРЖАНИЕ

**ВВЕДЕНИЕ** ..... 3
Практическая работа №1. Знакомство со средой программирования. ..... 4
Составление программ линейной структуры. ..... 4
Практическая работа № 2. Составление программ циклической структуры. Обработка одномерных массивов. Обработка двумерных массивов.. ..... 10
Практическая работа №3. Работа со строками. Работа с данными типа множество. Файлы последовательного доступа. ..... 28
Практическая работа №4. Типизированные файлы. Нетипизированные файлы. ..... 41
Практическая работа №5. Организация процедур. Организация функций. ..... 44
Практическая работа №6. Применение рекурсивных функций. ..... 48
Практическая работа №7. Создание библиотеки подпрограмм. Использование указателей для организации связанных списков. Изучение интегрированной среды разработчика. ..... 54
Практическая работа №8. Создание проекта с использованием компонентов для работы с текстом. Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени. ..... 60
Практическая работа №9. События компонентов (элементов управления), их сущность и назначение. Создание процедур на основе событий. Создание проекта с использованием кнопочных компонентов. Создание проекта с использованием компонентов стандартных диалогов и системы меню. ..... 63
Практическая работа №10. " Разработка функциональной схемы работы приложения. Разработка оконного приложения с несколькими формами. Разработка игрового приложения." ..... 100
Практическая работа №11. Создание процедур обработки событий. Компиляция и запуск приложения. Разработка интерфейса приложения. Тестирование, отладка приложения. ..... 101
Практическая работа № 12-13. " Классы ООП: виды, назначение, свойства, методы, события. Объявления класса. Создание наследованного класса. Перегрузка методов " ..... 103
**2. Методические рекомендации обучающимся по выполнению внеаудиторнойОшибка! Закладка не определена.**
**(самостоятельной) работы** ..... **Ошибка! Закладка не определена.**
**2.1. Подготовка к лекциям** ..... **Ошибка! Закладка не определена.**
**2.2. Изучение специальной методической литературыОшибка! Закладка не определена.**
**2.3. Подготовка компьютерной презентации** ..... **Ошибка! Закладка не определена.**
**2.4 Рекомендации по написанию реферата** ..... **Ошибка! Закладка не определена.**
**2.5 Методические рекомендации по подготовке докладов и тематических выступлений** ..... **Ошибка! Закладка не определена.**
**2.6. Case-study, ролевая игра** ..... **Ошибка! Закладка не определена.**
**2.7. Групповая дискуссия** ..... **Ошибка! Закладка не определена.**
**3.УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ** ..... 110

# *Введение*

Методические указания по учебной дисциплине **Основы алгоритмизации и программирования** для выполнения практических работ созданы Вам в помощь для работы на занятиях, подготовки к практическим работам, правильного составления отчетов.

Приступая к выполнению практической работы, Вы должны внимательно прочитать цель и задачи занятия, ознакомиться с требованиями к уровню Вашей подготовки в соответствии с федеральными государственными стандартами, краткими теоретическими и учебно-методическими материалами по теме практической работы, ответить на вопросы для закрепления теоретического материала.

Все задания к практической работе Вы должны выполнять в соответствии с инструкцией, анализировать полученные в ходе занятия результаты по приведенной методике.

Отчет о практической работе Вы должны выполнить по приведенному алгоритму, опираясь на образец.

Наличие положительной оценки по практическим работам необходимо для допуска к экзамену по УД, поэтому в случае отсутствия на уроке по любой причине или получения неудовлетворительной оценки за практическую работу Вы должны найти время для ее выполнения или пересдачи.

**Внимание!** Если в процессе подготовки к практическим работам или при решении задач у Вас возникают вопросы, разрешить которые самостоятельно не удается, необходимо обратиться к преподавателю для получения разъяснений или указаний в дни проведения дополнительных занятий.

# *Практическая работа №1. Знакомство со средой программирования.*

## *Составление программ линейной структуры.*

**Цель:**

- Познакомиться с интерфейсом среды Visual Studio .NET
- Изучить основные элементы управления, их свойства и методы
- Познакомиться с пространством имен библиотеки классов .NETFramework
- Разработать проект с использованием основных элементов управ-ления и меню

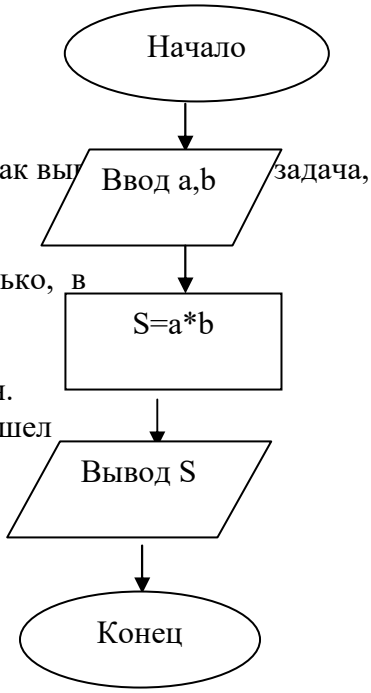
### Краткие теоретические сведения

**<u>Линейные алгоритмы</u>** ***- это такие алгоритмы, в которых действия совершаются одно за другим, в строго определенном порядке***

Рассмотрим пример составления линейного вычислительного алгоритма.

**Пример 1:** Вычислить площадь прямоугольника по заданной длине и ширине:

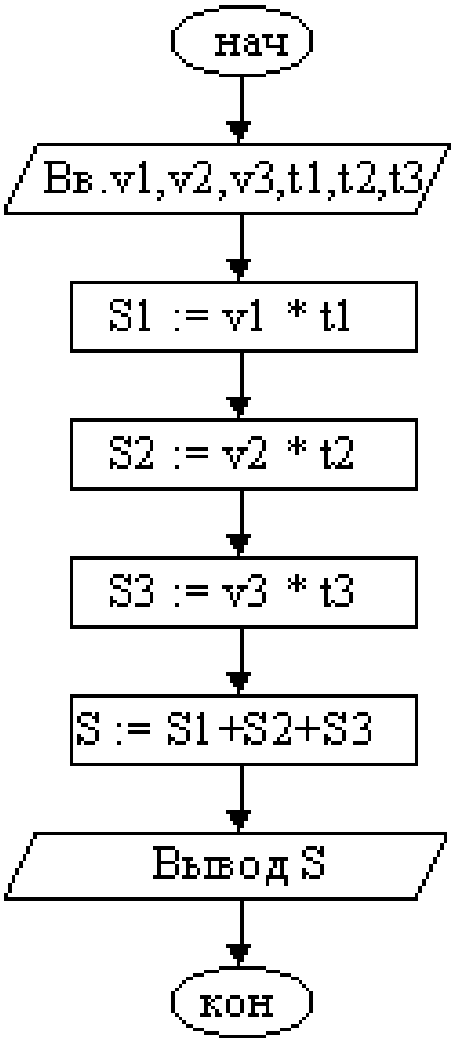
1. Ввести a, b
2. Вычислить площадь S по формуле a*b
3. Вывести полученный результат на экран.
4. Закончить выполнение алгоритма.




Это словесно – формульная форма записи алгоритма. Посмотрим, как вы задача, записанная с помощью графических блоков.

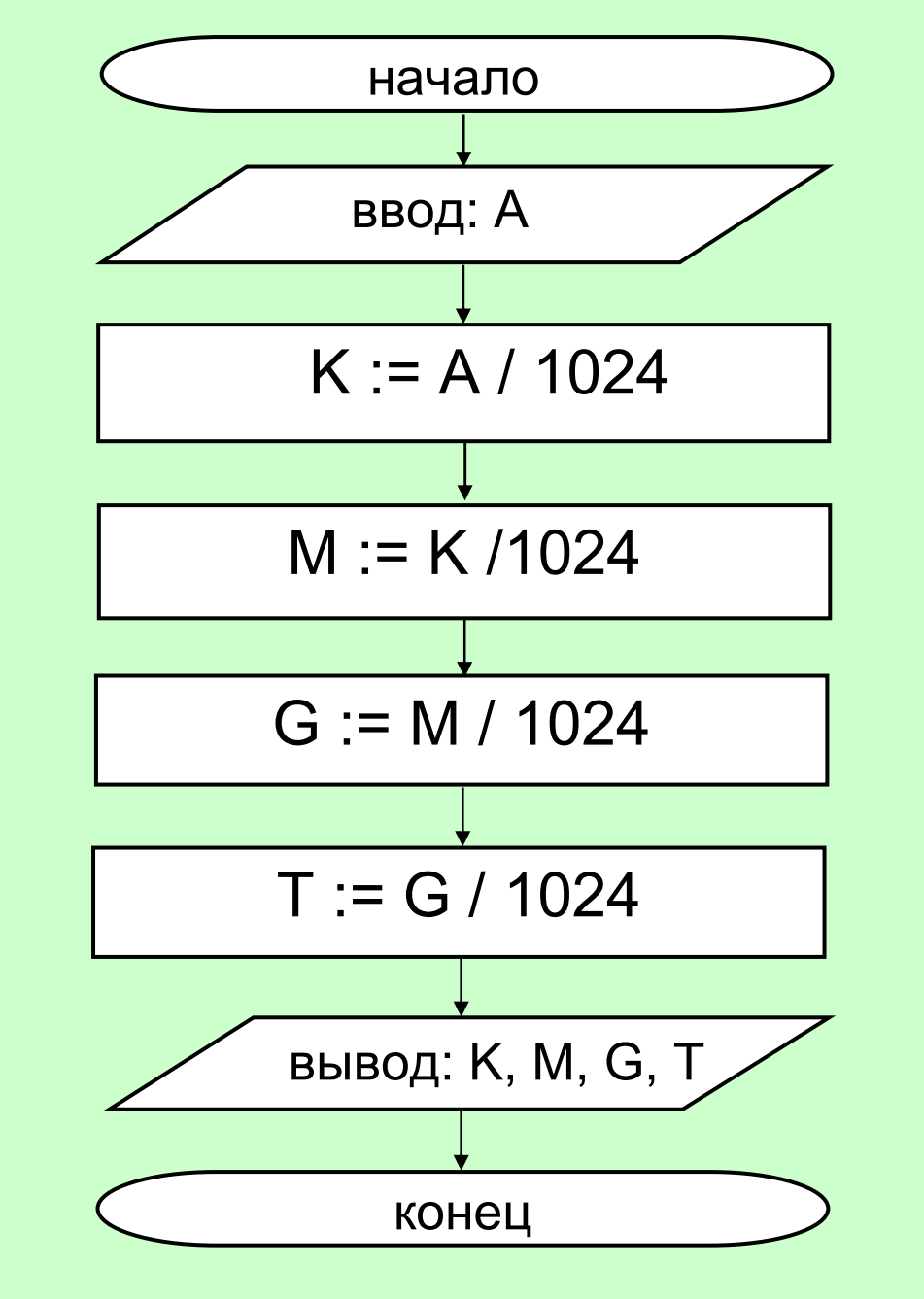
Вычислительных действий в блок-схеме может быть и несколько, в зависимости от условия задачи и от оформления.

**Пример 2.** Пешеход шел по пересеченной местности. Его скорость движения по равнине v1 км/ч, в гору — v2 км/ч и под гору — v3 км/ч. Время движения соответственно t1, t2 и t3 ч. Какой полный путь прошел пешеход?




1. Ввести v1, v2, v3, t1, t2, t3.
2. S1 := v1 * t1.
3. S2 := v2 * t2.
4. S3 := v3 * t3.
5. S := S1 + S2 + S3.
6. Вывести значение S.
7. Конец.

**Пример 3.** Дана величина А, выражающая объем информации в байтах. Перевести А в более крупные единицы измерения информации( Килобайты – К, Мегабайты – М, Гигабайты – Г). Составьте блок-схему алгоритма решения поставленной задачи.




ИНТЕРФЕЙС ОКНА ПРИЛОЖЕНИЯ Visio

Существует несколько способов запуска этой программы. Обычно при инсталляции программа установки автоматически добавляет Visio в меню Пуск, в папку Office. Откроем меню Пуск, пункт Программы, Office, и в открывшемся списке щелкнем по пункту MS Office Visio 2007. Другой способ запуска программы - двойной щелчок по ярлыку MS Visio на рабочем столе.

При открытии MS Visio по умолчанию в главном окне два дополнительных. Справа

– это окно Область задач и слева – окно Библиотеки шаблонов. С помощью шаблонов создаются всевозможные документы деловой графики – рисунки, схемы и диаграммы.

Синяя полоса в верхней части окна называется Строка заголовка. В ней написано имя открытого приложения, под ней – Панель меню.

СОЗДАНИЕ НОВОГО ДОКУМЕНТА С ПОМОЩЬЮ ШАБЛОНОВ

Для того, чтобы начать работу в программе MS Visio нужно либо создать новый документ, либо открыть уже созданный документ и продолжить работу в нем. Документ в MS Visio - это файл, который содержит листы с созданными пользователем изображениями. Рассмотрим сначала какие существуют способы создания нового документа. В меню Файл, выбираем подменю Новый, затем необходимо выбрать необходимый шаблон.

НАСТРОЙКА СТРАНИЦЫ Размер бумаги

Для выбора предложены размеры бумаги, которые поддерживаются текущим принтером, а также стандартные размеры. Если установить флажок Как в принтере на вкладке Размер страницы, изменения размера бумаги будут применяться также к странице документа.

В меню Файл выберите команду Предварительный просмотр, чтобы проверить соответствие размеров страницы документа размерам бумаги для печати.

Ориентация бумаги

Установите один из этих флажков, чтобы задать книжную или альбомную ориентацию страницы документа. Если на вкладке Размер страницы установлен флажок Как в принтере, изменения ориентации бумаги будут также применены к странице документа.

В меню Файл выберите команду Предварительный просмотр, чтобы проверить соответствие размеров страницы документа размерам бумаги для печати.

Настройка

При нажатии этой кнопки открывается диалоговое окно Настройка печати, в котором можно выбрать такие параметры печати, как поля, центрирование, принтер и источник бумаги.

Сетка

Установите этот флажок, чтобы вывести на печать линии сетки, отображаемые в окне документа.

Просмотр

В этой части окна отображается эскиз страницы, позволяющий проверить соотношение размеров текущей страницы документа и бумаги для печати.

Можно задать одинаковые размеры страницы документа и бумаги для печати. Для этого на вкладке Размер страницы установите флажок Как в принтере.

ДОБАВЛЕНИЕ НОВОЙ СТРАНИЦЫ

1.В левом нижнем углу окна документа щелкните правой кнопкой мыши вкладку страницы Page-1, а затем выберите в контекстном меню команду Вставить страницу.

2.Перейдите на вкладку Свойства страницы и введите имя для страницы или воспользуйтесь именем по умолчанию.

3.Для изменения масштаба или размера новой страницы перейдите на вкладку Масштаб документа или Размер страницы.

СОЗДАНИЕ ПРОСТОЙ БЛОК-СХЕМЫ

С помощью блок-схем можно документировать процедуры, анализировать процессы, обозначать рабочий или информационный процессы, затраты на отслеживание, эффективность и т. д.

1.В меню Файл последовательно выберите команды Создать, Бизнес или Блоксхема, а затем — команду Простая блок-схема.

2.Для каждого шага документируемого процесса перетащите в документ фигуру блок-схемы.

3.Соедините фигуры блок-схемы.

a.Перетащите фигуру из набора элементов на страницу документа и расположите ее вблизи другой фигуры.

b.Не отпуская кнопку мыши, переместите указатель на один из светло-синихтреугольных маркеров. Цвет маркера изменится натемно-синий.

c.Отпустите кнопку мыши. Фигура будет размещена на странице, и к обеим фигурам будет добавлена и приклеена соединительная линия.

4.Для добавления текста в фигуру выделите ее, а затем введите текст. Закончив ввод, щелкните за пределами текстового блока.

5.Для отображения последовательности шагов процесса фигуры в блок-схememожно пронумеровать.

a.В блок-схеме выделите фигуры, которым нужно присвоить номера.

В меню Сервис последовательно выберите команды Надстройки, Дополнительные решения Visio, а затем — команду Нумерация фигур.

c. На вкладке Общие в группе Операция установите переключатель в положение Автонумерация.

d.В группе Применить к установите переключатель в положение Выбранные фигуры, а затем нажмите кнопку ОК Совет. Для автоматической нумерации новых фигуры блок-схемы по мере их перетаскивания на страницу, в диалоговом окне Нумерация фигур установите флажок Продолжать нумерацию фигур при перетаскивании на страницу

**Задания для практического занятия:**

1. Рассмотреть примеры 1-3.
2. Разработать словесно -формульный алгоритм по индивидуальному заданию.
3. Разработать блок-схему алгоритма решения задачи средствами MS Visio.
4. Оформить отчет в MS Word.

**Индивидуальные задания**

Вариант 1.

1. Дана сторона квадрата. Найти его периметр.
2. Дан радиус окружности. Найти длину окружности и площадь круга.
3. Известны количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.
4. Даны катеты прямоугольного треугольника. Найти его периметр.
5. Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.
6. Дано двузначное число. Найти:

а) число десятков в нем;

б) число единиц в нем;

в) сумму его цифр;

г) произведение его цифр.

Вариант 2

1. Дан радиус окружности. Найти ее диаметр.
2. Даны два целых числа. Найти их среднее арифметическое.
3. Даны катеты прямоугольного треугольника. Найти его гипотенузу.
4. Даны два числа. Найти их сумму, разность, произведение, а также частное от деления первого числа на второе.
5. Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.
6. Дано трехзначное число. Найти:

а) число единиц в нем;

б) число десятков в нем;

в) сумму его цифр;

г) произведение его цифр.

Вариант 3

1. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.
2. Даны два целых числа. Найти их среднее геометрическое.
3. Найти площадь кольца по заданным внешнему и внутреннему радиусам.
4. Даны два числа. Найти среднее арифметическое и среднее геометрическое их модулей.
5. Даны длины сторон прямоугольника. Найти его периметр и длину диагонали.
6. Дано двузначное число. Получить число, образованное при перестановке цифр заданного числа.

**Методика анализа результатов, полученных в ходе практической работы**

1. В результате выполнения заданий у Вас должны быть алгоритмы описанные словесно-формульным способом и в виде блок-схемы, разработанной в MS Visio по индивидуальным заданиям.

.

**Порядок выполнения отчета по практической работе**

1. Разработать программу по индивидуальному заданию.
2. Предоставить преподавателю отчет в MS Word.

**Пример отчета**

**Практическая работа №1. Знакомство со средой программирования. Составление программ линейной структуры. Составление программ разветвляющейся структуры.**

ЦЕЛЬ РАБОТЫ: Формирование представлений о линейных алгоритмах, отработка навыков записи алгоритмов с помощью блок-схем и умений устанавливать соответствие между командами алгоритма, записанного словесно, и элементами блок-схемы

**Индивидуальные задания**

Вариант 1.

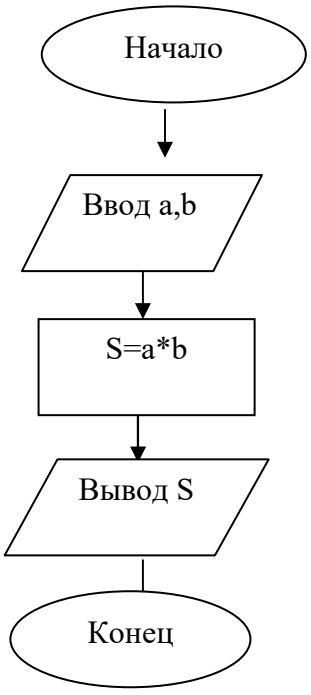
1. Вычислить площадь прямоугольника по заданной длине и ширине:
2. Дан радиус окружности. Найти длину окружности и площадь круга.
3. Известны количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.
4. Даны катеты прямоугольного треугольника. Найти его периметр.
5. Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.
6. Дано двузначное число. Найти:
   а) число десятков в нем;
   б) число единиц в нем;
   в) сумму его цифр;
   г) произведение его цифр.

**Задание 1:** Вычислить площадь прямоугольника по заданной длине и ширине:

Словесно -формульный алгоритм

| |
|---|
| Ввести a, b<br>Вычислить площадь S по формуле a*b<br>Вывести полученный результат на экран.<br>Закончить выполнение алгоритма |

Блок-схема:



# *Практическая работа № 2. Составление программ циклической структуры. Обработка одномерных массивов. Обработка двумерных массивов..*

**Учебная цель:** Отработать практические навыки разработки и программирования вычислительного процесса циклической структуры, получение дальнейших навыков по отладке и тестированию программы.

**Учебные задачи:**

1. Закрепить теоретические знания о реализации циклического процесса в среде ABC Pascal
2. Отработать практические навыки реализации циклических алгоритмов с условиями и с параметром.
3. Отработать практические навыки использования операторов fopr, while, repeat .
4. Закрепить практические навыки по составлению тестов и отладке программы

**Образовательные результаты, заявленные во ФГОС третьего поколения:**

Студент должен

уметь:

- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.

знать:

- этапы решения задачи на компьютере

**Задачи практической работы**:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы циклической структуры
4. Оформить отчет.

**Краткие теоретические и учебно-методические материалы по теме практической работы**

Отметим **отличия и особенности хорошего стиля работы** с рассмотренными циклическими операторами.

| **Цикл с предусловием While (пока условие истинно)** | **Цикл с постусловием Repeat (до истинности условия)** |
|---|---|
| До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл | |
| В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций завершился | |
| Цикл работает пока условие истинно (пока True) | Цикл работает пока условие ложно (пока False) |
| Цикл завершается, когда условие становится ложным (до False) | Цикл завершается, когда условие становится истинным (до True) |
| Цикл может не выполниться ни разу, если исходное значение условия при входе в цикл False | Цикл обязательно выполнится как минимум один раз |
| Если в теле цикла требуется выполнить более одного оператора, то необходимо использовать составной оператор | Независимо от количества операторов в теле цикла, использование составного оператора не требуется |
| **Цикл со счетчиком (с параметром) For** | |
| Начальная установка переменной счетчика цикла до заголовка не требуется | |
| Изменение в теле цикла значений переменных, стоящих в заголовке не допускается | |

| |
|---|
| Количество итераций цикла неизменно и точно определяется значениями нижней и верхней границ и шага приращения |
| Нормальный ход работы цикла может быть нарушен оператором goto или процедурами Break и Continue |
| Цикл может не выполниться ни разу, если шаг цикла будет изменять значение счетчика от нижней границы в направлении, противоположном верхней границе |

**Вопросы для закрепления теоретического материала к практическому занятию:**

1. Виды операторов цикла?
2. Запишите конструкцию цикла с предусловием. В каких случаях применяется цикл с предусловием?
3. Будет ли выполняться циклическая часть программы, если логическое выражение является ложным с самого начала в операторе While?
4. Назовите отличия итерационных циклов и цикла с параметром.

**Задания для практического занятия:**

1. Написать программы.
2. Составить отчет

**Инструкция по выполнению практической работы**

1. Выполните задания для самостоятельной работы
2. Ответьте на вопросы для повторения письменно в тетради.

**Индивидуальные задания**

**Задание 1.** Решить задачу в соответствии с вариантом и нарисовать блок-схему.

1. Составить алгоритм вычисления куба суммы всех нечетных чисел от 1 до 99.
2. Отпечатать значения x, $x^2$, $x^3$, если x меняется от 1 до 100 с шагом 0,5.
3. Посчитать количество чисел кратных m (m вводится с клавиатуры) в промежутке от X до Y.
4. Написать программу для нахождения суммы M чисел, вводимых с клавиатуры.
5. Вывести на экран в строку все числа первой сотни, оканчивающиеся на 5.
6. Напишите программу, выводящую на экран степени числа k (k вводится с клавиатуры) от 2 до 10.
7. Написать программу для нахождения суммы пяти произвольных чисел, вводимых с клавиатуры.
8. Составить алгоритм вычисления суммы кубов всех четных чисел от 100 до 300.
9. Посчитать сумму всех чисел, делящихся на 13 в интервале [1,1000]
10. Написать программу вычисления среднего роста десяти человек, данные о которых вносятся с клавиатуры.
11. Найти сумму всех четных целых чисел от 100 до 500.
12. Составьте блок-схему алгоритма и программу вычисления суммы всех трехзначных чисел, кратных 4.

**Задание 2.** Для данных N или m составить алгоритм вычисления значения выражения:

| | |
|---|---|
| Вариант №1 | $a = \sum_{i=1}^{50} \frac{1}{i^5}$ |
| Вариант №2 | $f = \prod_{x=1}^{5} \frac{x^3 - 35}{x - \lvert tgx \rvert}$ |

| | |
|---|---|
| Вариант №3 | $sum = \sum_{i=1}^{N} (-1)^i (10+\pi)^{2-i}$ |
| Вариант №4 | PS=$1^2$+$2^2$+$3^2$+$4^2$+ ….+$10^2$ |
| Вариант №5 | $c = \sum_{k=1}^{N} (-1)^k \frac{1}{2+k}$ |
| Вариант №6 | Q=Sin 1 + Sin 2+ Sin 3 + Sin 4 +…+ Sin N. |
| Вариант №7 | $y = \prod_{r=1}^{M} r^2 - \frac{r}{0.1}$ |
| Вариант №8 | $g = \sum_{j=1}^{30} \frac{(-1)^i}{i^6}$ |
| Вариант №9 | R=Cos 1 + Cos 3 + Cos 5 + …+ Cos ( 2N-1). |
| Вариант №10 | $s = \sum_{i=3}^{N} i - \sqrt{\frac{i-5}{\sin i}}$ |
| Вариант №11 | $p = \prod_{x=m}^{N} \frac{1-x}{1+x^2}, (m < N)$ |
| Вариант №12 | T=Sin 2 * Sin 3 * Sin 4 *…* Sin N. |

**Задание 3:**

В соответствии с вариантом написать два вида программ: используя оператор с постусловием и оператор с предусловием.

1. Дано натуральное число n (n<9999). Посчитать количество цифр в числе n?
2. Дано натуральное число n (n<9999). Посчитать чему равна сумма его цифр?
3. Дано натуральное число n (n<9999). Найти первую цифру числа.
4. Дано натуральное число n (n<9999). Выяснить, входит ли цифра 3 в запись числа n.
5. Вычислить сумму квадратов всех целых чисел, меньших заданного числа A.
6. Найти самую большую цифру целого числа.
7. Дано число n и цифра D. Сколько раз данная цифра встречается в целом числе?
8. Дано натуральное число n. Найти количество четных цифр целого положительного числа.
9. Дано натуральное число n. Найти сумму цифр целого числа, больших 5.
10. Дано натуральное число n.. Верно ли, что число начинается и заканчивается одной и той же цифрой.
11. Дано натуральное число. Найти произведение цифр этого числа.
12. Дано натуральное число. Верно ли, что сумма цифр данного числа равна A (A вводится с клавиатуры).

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий 1-3 у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

**Порядок выполнения отчета по практической работе**

1. Составить блок-схему решения индивидуальных задач из заданий 1-3 по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

## Составление программ усложненной структуры.

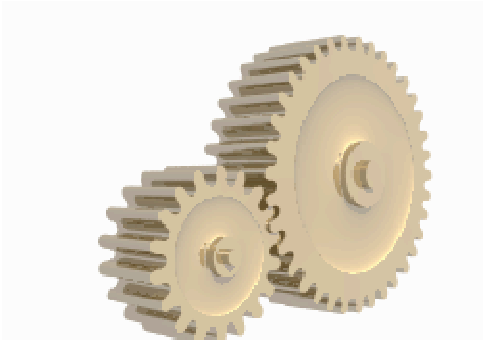
### Краткие теоретические и учебно-методические материалы по теме практической работы

Основная идея использования вложенных циклов состоит в том, что даже когда некий процесс, требует цикла -- повтора действий (т.н. "внешний цикл"), то даже внутри отдельного действия можно запустить свой цикл (т.н. "внутренний" или "вложенный цикл") для решения какой-то "местной" задачи.

**То есть:** внутри витка внешнего цикла, можно запустить цикл внутренний, тогда на один виток внешнего цикла, внутренний цикл будет каждый раз выполнять все свои витки.

Графическое представление вложенных циклов

Работу циклов также можно сравнить с вращением связанных шестерёнок разного размера:



-- внешний цикл это как бы большая шестерёнка, за один свой оборот (виток цикла), внешний цикл заставляет вращаться вложенный цикл (меньшую шестерёнку) несколько раз.

Такая иллюстрация точна в случае, если число повторов вложенного цикла не зависит от того какой именно (1-ый, n-ый или иной) виток делает внешний цикл, а так бывает не всегда. Почему выясним, рассматривая примеры ниже.

**Примеры кода решений задач с вложенными циклами**

**Пример №1.1: Repeat/until + For: работа с пользователем до его указания на завершение программы**

Предположим, что вы работаете с пользователем раз за разом выполняя похожие действия до тех пор, пока пользователь не введёт какую-то команду, показывающую, что пора заканчивать работу.

В этом случае ожидание очередной команды от пользователя, а также реакцию на неё имеет смысл поместить в тело цикла. Пусть при этом в ответ на команду нам тоже нужно делать что-то, что решается с помощью цикла -- вот мы и получили первый пример схемы, где без вложенных циклов не обойтись.

В качестве конкретного пример рассмотрим решение задачи:

Пользователь вводит целые положительные числа, большие 55. Пока он не введёт число 2222 в ответ на каждое введённое число выводите все целые числа от 11 до этого числа, если же пользователь ввёл ноль, то объявите о завершении работы программы.

Решение:

```
var   a,i:integer;
begin
  repeat   // внешний цикл
    writeln('vvedite chislo >5:');
    readln(a);

    for i:=1 to a do // (вложенный цикл) выводим все числа до a
      write(i, ' ');
```

```
    writeln(); // перенос строки
  until (a = 22);  // конец тела внешнего цикла
  writeln('zaversheno!');
  readln();
end.
```

Прокомментируем это решение:

В качестве внешнего цикла мы выбрали repeat/until, чтобы проверять условие уже после ввода значения пользователем.

В качестве внутреннего цикла мы выбрали for -- ведь каждый раз будет известно число, до которого надо выводить меньшие числа. Можно было бы использовать и любой другой цикл, но for в таких случаях использовать грамотнее и красивее.

минусом выбора repeat/until внешним циклом является то, что эта программа, в случае если пользователь введёт число 2222, все равно выведет ряд чисел, а только потом завершится.

Последний пункт вызывает желание (да-да, программирование должно вас увлекать ;) переписать код так, чтобы в случае, если пользователь ввёл 2222 ряд чисел не выводился.

**Пример №1.2 (продолжение): While + For: работа с пользователем до его указания на завершение программы**

Это пример является продолжением предыдущего и одновременной иллюстрацией ситуации, где цикл For вложен в While:

```
var   a,i:integer;
begin

  writeln('vvedite chislo >5:');
  readln(a);

  while (a <> 22) do
  begin   //  начало тела внешнего цикла

    for i:=1 to a do // (вложенный цикл) выводим все числа до a
      write(i, ' ');

    writeln(); // перенос строки

    writeln('vvedite chislo >5:'); // очередной раз запрашиваем число в цикле
    readln(a);

  end;  // конец тела внешнего цикла

  writeln('zaversheno!');
  readln();
end.
```

9

0

1

2

Как работает эта программа:

1. Сначала, ещё до цикла мы просим пользователя ввести число первый раз, если это число = 22, то цикл вообще не начнётся и программа будет завершена без вывода ряда.

2. Если пользователь вводит число не равное 22, то цикл начнётся, так как число уже известно, то в витке цикла мы сначала выведем значения до введённого числа, а только потом в конце витка запросим очередное число.

1. Пример №2 -- вывод таблицы умножения

Вывод всевозможных таблиц -- классический пример задач, где требуются вложенные циклы. **Основная идея** в подобных задачах состоит в том, что:

2. есть какой-то главный (внешний) цикл, тело которого должно решить задачу вывода на экран очередной строки (ну и расчета значений, которые нужно выводить)

3. но задача в вывода очередной строки в теле внешнего цикла, решается размещением в этом теле ещё одного вложенного цикла, который, например, формирует эту очередную строку из неких фрагментов, например символов или групп символов.

Таким образом получается, что на один виток внешнего цикла приходятся все витки внутреннего цикла, на второй виток внешнего внутренний цикл снова работает несколько раз до своего очередного завершения и т.д.

Рассмотрим решение задачи:

Вывести на экран таблицу умножения чисел от 1 до 9.

**Решение (**for в for**):**

```
var i, j: integer;
begin
  for i := 1 to 9 do // цикл по строкам таблицы, счетчик как левый множитель
    begin
      for j := 1 to 9 do // выводим равенства очередной строки, счётчик как правый
множитель
        write(i, '*', j, '=', i*j, ' ');
    writeln(); // переносим строку
   end;

0
   readln();
end.
1
```

Конечно, в качестве внешнего цикла можно было бы использовать любую другую из оставшихся двух конструкцию, например, давайте перепишем это решение используя вложение **for в while**:

```
var i, j: integer;
begin
  i := 1; // начальное значение для счетчика внешнего цикла
  while (i <= 9) do // цикл по строкам таблицы, счетчик как левый множитель
    begin

      for j := 1 to 9 do // выводим равенства очередной строки, счётчик как правый
множитель
        write(i, '*', j, '=', i*j, ' ');
    writeln(); // переносим строку
0
    i:=i+1; // увеличиваем значение счетчика внешнего цикла
1   end;
```

```
    readln();
  end.
```

Или даже while в repeat-until**:**

```
var i, j: integer;
begin
  i := 1; // начальное значение для счетчика внешнего цикла

  repeat // начало тела внешнего цикла

    j := 1; //  сбрасываем значение счетчика внутреннего цикла в единицу (чтобы он повторился как и предыдущий раз), или если речь идёт о первом витке, то это действие можно назвать заданием начального значения счетчика
    while (j<=9) do // выводим равенства очередной строки, счётчик как правый множитель
    begin
      write(i, '*', j, '=', i*j, ' ');
      j:=j+1; // увеличиваем значение счетчика внутреннего цикла
    end;

    writeln(); // переносим строку
    i:=i+1; // увеличиваем значение счетчика внешнего цикла

  until (i > 9);  // проверка условия выхода из внешнего цикла и конец его тела
  readln();
end.
```

**Вопросы для закрепления теоретического материала к практическому занятию:**

1. Виды операторов цикла?
2. Запишите конструкцию цикла с предусловием. В каких случаях применяется цикл с предусловием?
3. Будет ли выполняться циклическая часть программы, если логическое выражение является ложным с самого начала в операторе While?
4. Назовите отличия итерационных циклов и цикла с параметром.

**Задания для практического занятия:**

1. Написать программы.
2. Составить отчет

**Инструкция по выполнению практической работы**

1. Выполните задания для самостоятельной работы
2. Ответьте на вопросы для повторения письменно в тетради.

**Задачи для самостоятельного решения**

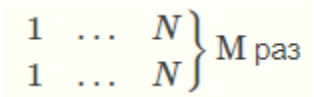
1. Выведите на экран таблицу умножения используя только циклы вида repeat/until.
2. Выведите на экран таблицу умножения используя только циклы вида while.
3. Выведите на экран таблицу умножения используя один цикл while и один repeat-until .
4. Пользователь вводит числа до тех пор пока не введёт число меньшее 11. В ответ на каждое введённое им число выводите на экран все нечетные числа от 1 до это числа, при этом делящиеся на 5. Если же пользователь ввел число меньшее 11, то завершите программу.
5. Пользователь вводит первое целое число-ограничитель mm. А затем начинает вводить целые числа по одному, пока не введёт число большее числа-ограничителя. Если очередное целое число больше 11, то в ответ на каждое такое число программа должна выводить все целые числа от единицы до этого числа.

**Примечание:** это задача на вложенные циклы, в качестве внешнего надо использовать while, а в качестве внутреннего можно использовать или for или while.

6. Пользователь вводит целое положительное число, если оно не соответствует критериям (то есть не является целым и положительным), выведете сообщение об ошибке, в противном случае выведете на экран все числа от 1 до введённого пользователем.
7. Модифицируйте предыдущую задачу так, чтобы в случае, если число удовлетворяет требованиям (целое, положительное), то на экран выводились четные числа.
8. Выведете на экран числа от 1 до 5 два раза с помощью вложенных циклов. Так чтобы в консоли было:

```
1 2 3 4 5
1 2 3 4 5
```

9. **M** раз выведете на экран числа от **1** до **N** с помощью вложенных циклов. Так чтобы в консоли было:




10. Модифицируйте предыдущую задачу так, чтобы в каждой чётной (той, у которой номер чётный) строке выводилось **N** символов, а в каждой нечетной **N/2** символов (сделайте проверку того, что N/2N/2 больше нуля)
11. Пользователь вводит числа до тех пор пока им не будет передан ноль. В ответ на каждое число программа должна сообщать чётное оно или нет.
12. Пользователь вводит четное целое число (если нечетное сообщите об ошибке). Делите это число в цикле на 2 до тех пор пока оно делится, выводя каждый промежуточный результат, например для 12 в консоли получим:

```
6
3
```

А для **8**:

```
4
2
1
```

13. Пользователь два целых числа M и N целое число, если M четное, делайте то же, что и в предыдущей задачи, а если нечётное, то умножайте M в цикле на 33 до тех пор пока результат не станет больше N (и выводите ), например для:

```
M := 2;
N := 15;
```

Получим:

```
6
18
```

14. С помощью вложенных циклов выведите на экран таблицу умножения числе от 1 до 9, начнётся она как-то так:

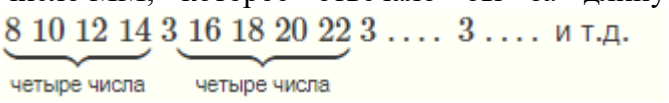
1x1 = 1
1x2 = 2
.....

15. С помощью вложенных циклов выведите на экран таблицу деления чисел от 1 до 9.
16. Пользователь вводит целое положительное число N, если оно не соответствует критериям (то есть не является целым и положительным), выведете сообщение об ошибке, в противном случае выведите на экран все числа последовательности, не большие NN, сформированной следующим образом:

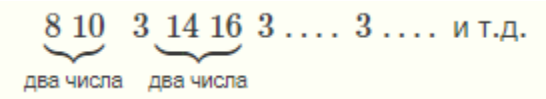
8 10 3 12 14 3 16 18 3 20 22 3 и т.д.

-- то есть всё начинается с восьмерки, затем число увеличивается на 2, затем выводит тройка и ещё пара увеличенных на 2 чисел и т.д.

17. Модифицируйте решение предыдущей задачи. так чтобы пользователь вводил второе число MM, которое отвечало бы за длину возрастающего фрагмента, например для M=4:




Заметьте. что в предыдущей задаче MM было зафиксировано =2:




18. Пользователь передает целое положительное число N, выведете на экран последовательность от 11 до NN "ёлочкой", например для N=17:

```
1
2 3
4 5 6
7 5 9 10
12 13 15 16
17
```

19. Модифицируйте предыдущий вывод "ёлочкой" так, чтобы в каждой нечетной строке выводились только четные числа, а в каждой четной только нечетные.

20. Пользователь передает целые положительные число N и M, выведете на экран последовательность от 11 до N, так чтобы ширина "ёлочки" увеличивалась до M чисел, то уменьшалась до 11. Например, для M=3 и N=25 получим:

```
1
2 3
4 5 6          --максимум три числа
7 5
9
10 17
18 19 20       --снова три числа
21 22
23
24 25 . . . . .
```

21. Пользователь передает целые положительные число N, выведете на экран последовательность от 11 до N, так чтобы ширина "ёлочки" росла волнами. Например, для M=49 получим:

```
1
2 3          --сначала до двух
4
5 6
7 8 9        --потом до трёх
10 11
12           --возвращаемся к одному
13 14
15 16 17
18 19 20 21          --тут уже четыре
22 23 24
25 26          --снова убывает
27
28 29
30 31 32
33 34 35 36
37 38 39 40 41
42 43 44 45
46 47 48
49
```

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

**Порядок выполнения отчета по практической работе**

1. Составить блок-схему решения индивидуальных задач из заданий 1-3 по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

## Обработка одномерных массивов.

**Типовые примеры на одномерные массивы.**

**Пример 1:** Дан одномерный массив из 30 элементов. Найти сумму элементов этого массива.

```
Program MylO_lm;
Const n=30;
Type MyArray=Array[I..n] Of Integer;
Var A: MyArray;
s, i: Integer;
Begin
{заполнение массива с клавиатуры}
WriteLn('Введите ',n, ' чисел');
For i:=l To n Do
ReadLn(A[i]);
{обработка массива}
s:=0;
For i:=1 To n Do
s:=s+A[i] ;
{вывод результата}
WriteLn('Их сумма равна ',s);
ReadLn;
End.
```

**Пример 2:** Дан одномерный массив из n элементов. Найти max элемент этого массива и его номер.

```
Program MylO_2m;
Const n=20;
Type MyArray=Array[1..n] Of Integer;
Var A: MyArray;
 i,max, x: Integer;
Begin
{заполнение массива случайными числами}
Randomize;
For i:=l To n Do
begin
A[i]=random(30)-10;
Write (A[i]:4);
End;
{обработка массива}
max:= A[1];
For i:=1 To n Do
If A[i]>max then begin max:= A[i] ; x:=i; end;
{вывод результата}
WriteLn('max= ',max, ‘x=’,x);
ReadLn;
End.
```

**Пример 3:** Дан одномерный массив из n элементов. Все четные элементы заменить на -1.

```
Program MylO_3m;
```

```
Const n=20;
Type MyArray=Array[1..n] Of Integer;
Var A: MyArray;
 i, x: Integer;
Begin
{заполнение массива случайными  числами}
Randomize;
For i:=l To n Do
begin
A[i]=random(30)-5;
Write (A[i]:4);
End;
Writeln;
{обработка массива}
For i:=1 To n Do
If A[i] mod 2 = 0  then A[i]:= -1;
{вывод результата}
Writeln (‘новый массив’);
For i:=l To n Do
Write (A[i]:4);
ReadLn;
End.
```

**Задания для практического занятия:**

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

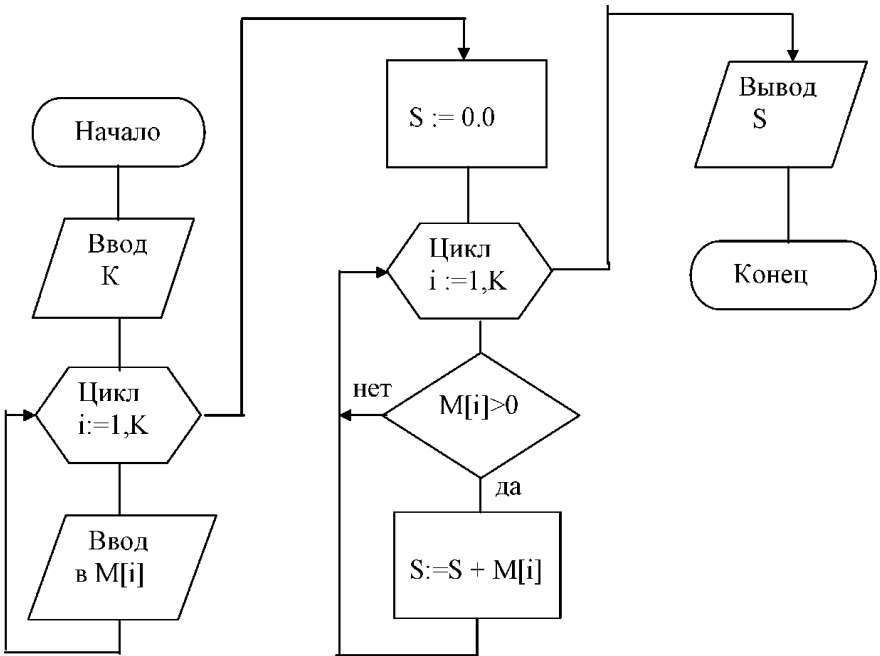
**Инструкция по выполнению практической работы**

1. Рассмотрите примеры из теоретической части.
2. Рассмотрите пример выполнения практической работы
3. Выполните задания для самостоятельной работы

**Пример выполнения практической работы**

Составить схему алгоритма и программу определения суммы S всех положительных элементов одномерного массива M, содержащего K вещественных чисел (K<20). Числа в массив M ввести с экрана дисплея, результат суммирования также вывести на экран.

Схема алгоритма




**Текст программы**

```
Program pr1;
Uses crt; {Подключение стандартного модуля TP}
Var m:array[1..20] of real;
```

```
{Описание массива вещественных чисел}
    S:real; {описание переменной для хранения суммы}
    i,K:byte; {описание переменных для счетчика и количества чисел}
    Begin
      Clrscr; {Очистка экрана}
      Write(' Введите число элементов в массиве K ='); Readln(K);
      Writeln(' Введите ',K,' вещественных чисел через пробел');
      For i:=1 to K do read(M[i]); { Ввод вещественных чисел}
      writeln; { Переход на новую строку на экране}
      S:=0.0; { Присвоение начального значения суммы}
      For i:=1 to K do
      If M[i]>0 then S:=S+M[i]; { Суммирование положительных чисел}
      Writeln(' S = ',S); { Вывод полученной суммы}
    End.
```

**Индивидуальные задания**

**1. Выполните задания в соответствии с вариантом.**

**Вариант 1**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 2**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.

2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

**Вариант 3**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 4**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.

2. Дано n чисел. Найдите сумму чисел, равных заданному числу a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 5**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 6**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.

2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

**Вариант 7**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 8**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.

2. Дано n чисел. Найдите сумму чисел, равных заданному числу a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 9**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 10**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.

2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

**Вариант 11**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.

2. Дано n чисел. Найдите сумму чисел, больших заданного числа a.

3. В заданном массиве замените нулем наибольший элемент.

4. Найдите полупроизведение всех положительных элементов массива.

5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Вариант 12**

1.В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.

2. Дано n чисел. Найдите сумму чисел, равных заданному числу a.

3. В заданном массиве замените нулем наименьший элемент.

4. Найдите полупроизведение всех отрицательных элементов массива.

5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.

6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.

7. Дано n чисел. Замените все отрицательные числа их модулями.

8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

**Порядок выполнения отчета по практической работе**

1. Составить блок-схему решения индивидуальных задач из заданий по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

## Обработка двухмерных массивов.

**Краткие теоретические материалы**

Двумерный массив - это структурированный тип данных в виде прямоугольной таблицы. Положение элементов в двумерных массивах описывается двумя индексами.

Например, двумерный массив размером 3*3, то есть в нем будет три строки, а в каждой строке по три элемента:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Каждый элемент имеет свой номер, как у одномерных массивов, но сейчас номер уже состоит из двух чисел – номера строки, в которой находится элемент, и номера столбца. Таким образом, номер элемента определяется пересечением строки и столбца.

A[2,1] - это элемент, стоящий во второй строке и в первом столбце.

Описание двумерного массива (матрицы):

Var A: array[1..100,1..100] of integer;

**Ввод двумерного массива**

Для последовательного ввода элементов двумерного массива используется цикл for, в котором нужно последовательно изменять номер строки с 1-й до последней и в каждой строке перебирать элементы столбцов с 1-го до последнего.

Рассмотрим пример ввода двумерного массива Паскаля с клавиатуры:

```
var
   a, : array [1..5, 1..10] of integer;
   i, j: integer; { индексы массива }
 begin
   for i :=1 to 5 do {цикл для перебора всех строк}
     for j :=1 to 10 do {перебор всех элементов строки по столбцам}
       readln ( a [ i , j ]); {ввод с клавиатуры элемента, стоящего в i -й строке и j -м столбце}
```

Двумерный массив можно заполнить случайным образом, т.е. использовать функцию random (N), а также присвоить каждому элементу матрицы значение некоторого выражения. Способ заполнения двумерного массива выбирается в зависимости от поставленной задачи, но в любом случае должен быть определен каждый элемент в каждой строке и каждом столбце.

**Вывод двумерного массива Паскаля на экран**

Вывод элементов двумерного массива также осуществляется последовательно, необходимо напечатать элементы каждой строки и каждого столбца. При этом нужно, чтобы элементы, стоящие в одной строке, печатались рядом, т.е. в строку, а элементы столбца располагались один под другим. Для этого необходимо выполнить следующую последовательность действий (рассмотрим фрагмент программы для массива, описанного в предыдущем примере):

```
for i :=1 to 5 do {цикл для перебора всех строк}
 begin
   for j :=1 to 10 do {перебор всех элементов строки по столбцам}
```

write ( a [ i , j ]:4); {печать элементов, стоящих в i -й строке матрицы в одной экранной строке, при этом для вывода каждого элемента отводится 4 позиции}
writeln ; {прежде, чем сменить номер строки в матрице, нужно перевести курсор на начало новой экранной строки}
end ;

**Типовые примеры на двумерные массивы.**

**Пример 1.** Чтобы заполнить двумерный массив (матрицу) случайными числами, введите программу:

```
Const n=6; m=7;
Var I,j: integer;
    a: array[1..n, 1..m] of integer;
begin
// заполнение массива
Randomize;
for i:=1 to n do begin
for j:=1 to m do begin
a[i,j]:=random(10);
write(a[i,j]:5);
end;
wrieln;
end;
end.
```

**Пример 2.** Решите задачу поиска max числа в массиве и его положения.

```
Const n=6; m=7;
Var I,j: integer;
    a: array[1..n, 1..m] of integer;
begin
// заполнение массива
Randomize;
for i:=1 to n do begin
for j:=1 to m do begin
a[i,j]:=random(10);
write(a[i,j]:5);
end;
wrieln;
end;
// нахождение max
Max:=a[1,1];
for i:=1 to n do
for j:=1 to m do
if a[i,j]>max then  begin max:=a[I,j]; x:=I; y:=j; end;
// вывод результатов
Writeln ('max=', max);
Writeln (' строка=' , x, 'столбец=', y);
End.
```

**Пример 3.** В двумерном массиве заменить все отрицательные элементы на ноль.

```
Const n=10; m=10;
Var I,j: integer;
 a: array[1..n, 1..m] of integer;
begin
// заполнение массива
Randomize;
for i:=1 to n do begin
for j:=1 to m do begin
```

```
a[i,j]:=random(20)-10;
write(a[i,j]:5)-5;
end;
wrieln;
end;
// нахождение отрицательных и замена
for i:=1 to n do
for j:=1 to m do
if a[i,j]<0 then  a[I,j]:=0;
// вывод результатов
for i:=1 to m do begin
for j:=1 to n do begin
write(a[i,j]:5);
end;
wrieln; end; end.
```

**Задания для практического занятия:**

4. Рассмотрите примеры и проверьте их работу на ПК.
5. Решите самостоятельно задачу поиска mix числа в массиве и его положения.
6. Решите самостоятельно задачу  замены  элементов больших 5 на единицы.
7. Написать программы для решения задач в соответствии с вариантом.
8. Составить отчет

**Инструкция по выполнению практической работы**

4. Рассмотрите примеры из теоретической части.
5. Выполнить примеры.
6. Выполните задания для самостоятельной работы.

**Индивидуальные задания**

**Вариант 1**

**1.** Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые превосходят заданное число.

2. Составьте программу для ввода с клавиатуры элементов матрицы 3*3 и вычисления суммы элементов каждого столбца.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех положительных элементов и распечатки их на экране ЭВМ.

4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней арифметической.

**Вариант 2**

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые  не превосходят заданное число.

2. Составьте программу для ввода с клавиатуры элементов матрицы 4*4 и вычисления суммы элементов каждой строки.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех отрицательных элементов и распечатки их на экране ЭВМ.

4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней геометрической.

**Вариант 3**

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые равны заданному числу.

2. Составьте программу для ввода с клавиатуры элементов матрицы 3*3 и вычисления суммы элементов каждой строки.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех положительных элементов и распечатки их на экране ЭВМ.

4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней арифметической.

**Вариант 4**

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые превосходят заданное число.

2. Составьте программу для ввода с клавиатуры элементов матрицы 4*4 и вычисления суммы элементов каждого столбца.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех отрицательных элементов и распечатки их на экране ЭВМ.

4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней геометрической.

**Вариант 5**

**1.** Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые не превосходят заданное число.

2. Составьте программу для ввода с клавиатуры элементов матрицы 3*3 и вычисления суммы элементов каждой строки.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех нулевых элементов и распечатки на экране ЭВМ их номеров.

4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива суммы.

## Методика анализа результатов, полученных в ходе практической работы

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

## Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

# *Практическая работа №3. Работа со строками. Работа с данными типа множество. Файлы последовательного доступа.*

**Учебная цель**: Отработать навыки работы со строковыми переменными, с данными типа множество. Изучить файлы последовательного доступа

**Учебные задачи:**

1. Закрепить теоретические знания о строковом типе данных
2. Овладение навыками применения процедур и функций для работы со строками.
3. Научиться использовать возможности языка программирования для обработки строк.
4. Закрепить практические навыки по составлению тестов и отладке программы.

**Образовательные результаты, заявленные во ФГОС третьего поколения:**

Студент должен

уметь:

- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.

знать:

- этапы решения задачи на компьютере

**Задачи практической работы**:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки строк
4. Оформить отчет.

**Краткие теоретические и учебно-методические материалы по теме практической работы**

**Строка** представляет собой особую форму одномерного массива символов, которая имеет существенное отличие. Массив символов имеет фиксированную длину (количество элементов), которая определяется при описании. Строка имеет две разновидности длины:

- общая длина строки, которая характеризует размер памяти, выделяемый строке при описании;
- текущая длина строки (всегда меньше или равна общей длине), которая показывает количество смысловых символов строки в каждый конкретный момент времени.

Строка в Паскале – упорядоченная последовательность символов. Количество символов в строке называется ее длиной. Длина строки в Паскале может лежать в диапазоне от 0 до 255. Каждый символ строковой величины занимает 1 байт памяти и имеет числовой код в соответствии с таблицей кодов ASCII.

Код ASCII (American Code for Information Interchange – Американский стандартный код для обмена информацией) имеет основной стандарт и его расширение. Основной стандарт использует шестнадцатеричные коды 00-7F, расширение стандарта – 80-FF. Основной стандарт является международным и используется для кодирования управляющих символов, цифр и букв латинского алфавита; в расширении стандарта используются символы псевдографики и буквы национальных алфавитов.

Встроенная функция **Length** позволяет определить фактическую длину текстовой строки, хранящейся в указанной переменной (а не величину предельного размера строки, установленную при декларации).

Строковая константа Паскаля – последовательность символов, заключенная в апострофы. Например, ‘строковая константа’, ‘243’. Два следующих друг за другом апострофа (‘’) обозначают пустую строку, т.е. строку с нулевой длиной.

**Описание строковой переменной Паскаля**

Для описания строковых переменных в Паскале существует предопределенный тип **string**.

В общем виде описание строковой переменной будет выглядеть следующим образом:

Пример описания строковой переменной в Паскале:

Var <имя_переменной>: string[<максимальная длина строки>]

Например:

```
Var
        s1: string[10];
        s2: string[20];
        smax: string;
```

В приведенном выше описании строковая переменная s1 может содержать не более 10 символов, переменная s2 – не более 20 символов. Если же при описании строки ее максимальная длина не указывается, то по умолчанию принимается максимально допустимая длина, равная 255 символам (переменная smax)..

Символы в строке упорядочены, каждый из них имеет порядковый номер, начиная с первого. Имеется возможность обратиться к любому элементу строки, указав его номер, так же как это делается в одномерных массивах. Например, s1[2] позволяет обратиться ко второму символу в строке s1, при этом мы можем поменять это значение, выполнив оператор присваивания s1[2]:= ‘r’, можем вывести на экран это значение или присвоить его другой переменной.

**Действия со строками в Паскале**

Операция **слияния** (сцепления, конкатенации) применяется для соединения нескольких строк в одну, обозначается знаком «+». Операция слияния применима для любых строковых выражений, как констант, так и переменных. Функция **Concat** выполняет конкатенацию (или сцепление) строк Str1, Str2, ..., StrN в том порядке, в каком они указаны в списке параметров. Общее количество символов всех сцепленных строк не должно превышать 255.

Операции **отношения** позволяют сравнивать строки на отношение равенства (=), неравенства (<>), больше (>), меньше (<), больше или равно (>=), меньше или равно (<=). В результате сравнения двух строк получается логическое значение (true или false). Сравнение строк производится слева направо посимвольно до первого несовпадающего символа, большей считается та строка, в которой первый несовпадающий символ имеет больший код в таблице кодировки. Если строки имеют различную длину, но в общей части символы совпадают, считается, что короткая строка меньше. Строки равны, если они имеют равную длину и соответствующие символы совпадают.

Пример действий со строками в Паскале:

‘строка’<>’строки’ (верно, т.к. не совпадают последние символы);

‘Abc’<’abc’ (отношение истинно, т.к. код символа ‘A’ равен 65 в десятичной системе счисления, а код символа ‘a’ – 97);

‘год’>’век’ (отношение верно, т.к. буква ‘г’ в алфавите стоит после буквы ‘в’, а, следовательно, имеет больший код).

**Стандартные функции для работы со строками в Паскале**

**Copy (S, poz, n)** выделяет из строки S, начиная с позиции poz, подстроку из n символов. Здесь S – любое строковое выражение, poz, n – целочисленные выражения.

| Значение S | Выражение | Результат |
|---|---|---|
| ‘строка символов’ | Copy(S,3,3) | рок |

Функция **Copy** позволяет копировать фрагмент некоторой строки из одной переменной в другую. Вызывая эту функцию, нужно указать следующие параметры:

- имя строки, из которой должен извлекаться копируемый фрагмент,
- позицию в строке, начиная с которой будет копироваться фрагмент,
- число копируемых символов.

**Concat (s1, s2,...,sn)** выполняет слияние строк s1, s2,...,sn в одну строку.

| Выражение | Результат |
|---|---|
| Concat(‘язык’, ‘’, ‘Pascal’) | ‘язык Pascal’ |

**Length(S)** определяет текущую длину строкового выражения S. Результат – значение целого типа.

| Значение S | Выражение | Результат |
|---|---|---|
| ‘(a+в)*c’ | Length(s) | 7 |

**Pos(subS, S)** определяет позицию первого вхождения подстроки subS в строку S. Результат – целое число, равное номеру позиции, где находится первый символ искомой подстроки. Если

вхождение подстроки не обнаружено, то результат функции будет равен 0. С помощью функции **Pos** можно осуществить поиск некоторого фрагмента в строке. Если заданный фрагмент в строке присутствует, то функция возвращает номер позиции, с которой он начинается. Если фрагмент не найден, то функция возвращает нуль.

| Значение S | Выражение | Результат |
|---|---|---|
| ‘предложение’ | Pos(‘e’, S) | 3 |
| ‘предложение’ | Pos(‘a’, S) | 0 |

**Стандартные процедуры для работы со строками в Паскале**

**Delete (S, poz, n)** удаляет из строки S, начиная с позиции poz, подстроку из n символов. Здесь S – строковая переменная (в данном случае нельзя записать никакое другое строковое выражение, кроме имени строковой переменной, т.к. только с именем переменной связана область памяти, куда будет помещен результат выполнения процедуры); poz, n – любые целочисленные выражения.

| Исходное значение S | Оператор процедуры | Конечное значение S |
|---|---|---|
| ‘abcdefg’ | Delete(s, 2, 3) | ‘aefg’ |

**Insert(subS, S, poz)** вставляет в строку S, начиная с позиции poz, подстроку subS. Здесь subS – любое строковое выражение, S – строковая переменная (именно ей будет присвоен результат выполнения процедуры), poz – целочисленное выражение.

| Исходное значение S | Оператор процедуры | Конечное значение S |
|---|---|---|
| ‘рис. 2’ | Insert(‘№’, S, 6) | ‘рис. №2’ |

**Процедуры преобразования типов в Паскале**

**Str(x, S)** преобразует число x в строковый формат. Здесь x – любое числовое выражение, S – строковая переменная. В процедуре есть возможность задавать формат числа x. Например, str(x: 8: 3, S), где 8 – общее число знаков в числе x, а 3 – число знаков после запятой.

| Оператор процедуры | Значение S |
|---|---|
| Str (sin(1):6:4, S) | ‘0.0175’ |
| Str (3456, S) | ‘3456’ |

**Val(S, x, kod)** преобразует строку символов S в число x. Здесь S – строковое выражение, x – числовая переменная (именно туда будет помещен результат), kod – целочисленная переменная (типа integer), которая равна номеру позиции в строке S, начиная с которой произошла ошибка преобразования, если преобразование прошло без ошибок, то переменная kod равна 0.

| Тип X | Оператор процедуры | Значение X | Значение kod |
|---|---|---|---|
| Real | Val(’12.34’, x, kod) | 12.34 | 0 |
| Integer | Val(’12.34’, x, kod) | 12 | 3 |

**Функция Upcase** позволяет преобразовывать символ любой литеры из строчного в прописной. Эта функция рассчитана на обработку отдельного символа, поэтому для обработки строки символов с помощью этой функции приходится организовывать цикл.

**Практическая часть**

**Пример 1.** Определение длины строки.

```
Program DemoFunctionLength;
Var
Word : string;
Begin
write ('Введите слово :');
readln(Word);
writeln('Это слово состоит из ',Length (Word),' букв');
End.
```

**Пример 2.** Преобразование строчных букв в прописные.

```
Program DemoFunctionUpcase;
    Var
    Word : string;
    i : Byte;
Begin
        Word := 'фирма Microsoft';
        for i := 1 to Length (Word) do
        Word[i] := UpCase (Word[i]);
```

```
        writeln(Word); {выводится текст 'фирма MICROSOFT'}
    End.
```

**Пример 3.** Копирование фрагмента текста.

```
    Program DemoFunctionCopy;
    Var
        Word : string;
        Word1 : string[20];
    Begin
        Word := 'фирма Microsoft';
        writeln(Word); {выводится текст 'фирма Microsoft'}
        Word1 := Copy (Word,1,5);
        writeln(Word1); {выводится текст 'фирма'}
    End.
```

**Пример 4.** Определение позиции символа, входящего в строку.

```
    Program DemoFunctionPos;
    Var
        Word : string;
        SearchWord : string[20];
        Position : Byte;
    Begin
        Word := 'фирма Microsoft';
        writeln(Word); {выводится текст 'фирма Microsoft'}
        writeln ('Введите искомый текст ');
        readln (SearchWord);
        Position := Pos(SearchWord, Word);
        if Position <> 0 then
    begin
write ('Фрагмент <',SearchWord,'> содержится в строке <',Word);
writeln ('>, начиная с позиции ',Position );
    end
    else
writeln('Фрагмент <',SearchWord,'> не содержится в строке <',Word, '>');
    End.
```

**Пример 5.** Объединение двух строк.

```
    Program DemoFunctionConcat;
    Var
       Word : string;
       Word1, Word2 : string[20];
    Begin
        Word1 := 'фирмы ';
        Word2 := 'Microsoft';
        Word := Concat('Компьютеры ',Word1,Word2);
        writeln(Word); {выводится текст 'Компьютеры фирмы Microsoft'}
    End.
```

**Пример:** Из набора 10 слов, длина каждого из которых 5 символов, напечатать все слова, отличные от слова "hello".

```
program r2;
typesl=array [1..5] of char;
msl=array [1. .10] of sl;
var
s:msl;
i:integer;
begin
    writeln ('введите 10 слов по 5 символов ');
    for i:=l to 10 do
        read (s[i]);
    writeln;
```

```
    for i:=l to 10 do
        if s[i]<>o 'hello' then write (s[i],'  ');
    writeln;
end.
```

**Тип данных STRING**

В Турбо Паскаль предусмотрен тип данных STRING.Переменная типа STRING, в отличие от обычного массива литер, может принимать значения переменной длины. Максимально возможная длина переменной 255 символов. Например:

L:STRING[200]; cit:STRING[30]; ouw:STRING[10]:

В скобках указывается максимальная длина для данной переменной. Для ввода значений типа STRING необходимо использовать READLN, а не READ.За один раз может быть введена только одна строка. Две строки можно сравнивать, используя операции отношения (сначала сравниваются самые левые символы, если они равны, то сравниваются следующие и т.д.). Пример: Распечатать по алфавиту k слов.

```
program In1;
const k=5;
var:
c: integer;
s: 1..k;
ln: array [1..kl of string[20]:
begin
    for c:=l to k do readln (ln[c]);
    writeln:
    for c:=l to k do begin
        s:=l;
    forc:=2 to k do
    If (ln[c]<ln[s]) and (ln[c]<>'  ') then s:=c;
    Writeln (ln[s]);
    Ln[s]:='  ';
    End;
end.
```

**Инструкция по выполнению практической работы**

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

**Индивидуальные задания**

| № варианта | Задание |
|---|---|
| 1. | 1. Подсчитать, сколько раз в данной строке встречается буква Ch, вводимая с клавиатуры.<br>2. Подсчитать длину первого слова в строке.<br>3. Раздвинуть заданный текст, вставив заданную последовательность после i–го символа каждого слова. |
| 2. | 1. Из заданной строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы.<br>2. Из заданной строки получить новую, заменив в ней все «7» на символ «о». Распечатать полученную строку.<br>**3.** В заданном тексте слова разделены пробелами или запятыми. Напечатать список слов, начинающихся с символа, введенного с клавиатуры. |
| 3. | 1. Заменить в заданной строке все буквы Ch1 на Ch2 (их значения вводить с клавиатуры).<br>2. Проверить, является ли заданное слово палиндромом (Например, слова палиндромы: КАЗАК, ШАЛАШ, МАДАМ). |

| | |
|---|---|
| | 3. Определить наличие слов в заданном тексте содержащих сочетание символов, задаваемое с клавиатуры. |
| 4. | 1. Заменить все вхождения подстроки Str1 на подстроку Str2, которые вводятся с клавиатуры.<br>2. В строке заменить все фразы "что и требовалось доказать " на "ч. т. д.", тем самым сжав ее.<br>3. Из заданной последовательности слов удалить слова, содержащие числа. |
| 5. | 1. В заданной строке после каждой буквы Ch вставить строку Str1.<br>2. В строке заменить все пробелы, стоящие на четных позициях, на двойные.<br>3. Подсчитать количество слов, разделенных пробелами, содержащих k гласных букв (k задается с клавиатуры). |
| 6. | 1. В заданной строке удвоить каждое вхождение буквы Ch.<br>2. Подсчитать, сколько слов в заданной строке заканчивается буквой «я», если известно, что после слова обязательно стоит либо точка, либо пробел, либо запятая.<br>3. Выбрать из заданного текста все слова, которые имеют в своем составе по 2 и более одинаковых символа. |
| 7. | 1. В заданной строке удвоить каждое вхождение буквы Ch.<br>2. Дана строка символов. Удалить из нее слова, начинающиеся с заданной буквы.<br>3. Вывести на печать список слов, имеющих приставку (несколько букв), задаваемую с клавиатуры. |
| 8. | 1. Дана последовательность слов. Напечатать все слова, отличные от слова «hello”.<br>2. Дана строка символов. Найти слово, которое имеет четную длину и начинается с заданного символа.<br>3. Каждое слово текста преобразовать таким образом, чтобы оно читалось слева направо. |
| 9. | 1. Дана последовательность слов. Напечатать все слова в алфавитном порядке.<br>2. Из заданной строки получить новую, заменив в ней все группы букв «абсд» на «абс».<br>3. Список фамилий в произвольном порядке через запятую задается с экрана. Упорядочить фамилии по алфавиту. |
| 10. | 1. Дана последовательность слов. Напечатать все слова последовательности, которые встречаются в ней по одному разу.<br>2. Определить, сколько слов в заданной строке начинается на букву "a".<br>3. Распечатать столбцами слова из текста заканчивающиеся на согласную букву и на гласную букву. |
| 11. | 1. Дано предложение. Напечатать все различные слова.<br>2. Какой процент слов в строке начинается на букву " k ".<br>3. Из заданного текста распечатать столбцами отдельно гласные и согласные буквы каждого слова. |
| 12. | 1. Дана последовательность слов. Напечатать все слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова все предыдущие вхождения последней буквы.<br>2. Вводится строка. Определить, сколько слов начинается с той же буквы, с которой начинается первое слово.<br>3. Из произвольного текста распечатать слова, повторяющиеся в порядке убывания частоты повторения. |

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

## Работа с данными типа множество.

### Краткие теоретические материалы

Наряду с числом множество является фундаментальным математическим понятием. Паскаль - один из немногих алгоритмических языков, который имеет встроенные средства для работы с множествами.

В математике под множеством понимается некоторый набор элементов. Например, множество фигур на плоскости (прямоугольник, круг, ромб, квадрат). В математике рассматриваются конечные и бесконечные множества, состоящие из произвольных элементов. В Паскале множества всегда конечные, причем состоят из небольшого числа элементов (в Турбо Паскале - до 255).

Постоянные множества и в математике и в Паскале задаются перечнем их элементов.

| Математика | Паскаль |
|---|---|
| {1,2,3} | [1,2,3] |
| {'A','K','B','L'} | ['A','K','B','L'] |
| ∅ | пустое множество [] |
| {1,2,...,N} | [1..N] |

В квадратных скобках могут находиться не только константы, но любые выражения типа элементов множества, например, [2++x,8-3].

К множествам применимы следующие операции.

объединение C=A∪B

пересечение C=A∩B

разность C=A\B

Под множеством в языке паскаль понимают ограниченный, неупорядоченный набор различных элементов одинакового типа. Можно, например говорить о множестве радиодеталей, транспортных средств, станков и т.д. всему множеству в целом дается имя. Каждый объект множества называется элементом множества. Тип элементов, входящих в множество, называется базовым. В качестве базового типа можно использовать простые типы: стандартный (кроме действительного), перечисляемый и ограниченный.

Базовый тип задается диапазоном или перечислением. Область значений типа множество - набор всевозможных подмножеств, составленных из элементов базового типа. Если базовый тип принимает n значений, то тип множество для него будет иметь $2^n$ вариантов значений. В выражениях на языке Pascal значения элементов множества указываются в квадратных скобках: [1, 2, 3, 4], ['a', 'b', 'c', 'd'], ['a'...'z'].

Множества должны быть объявлены в var или type.

var <имя_ множества>: set of <базовый элемент>;

**Например:**var god: set of 1880..2000; c: set of char;

type <имя типа>= set of <базовый элемент>;

var <имя множества>: <имя типа>;

set of 'a'..'z' - множество прописных английских букв;

set of 1..100 - множество целых чисел от 1 до 100;

set of (winter, spring, summer, outumn) - множество времен года

set of char - множество всех символов.

В языке паскаль имеются следующие операции над множествами:

\+ - объединение множеств;

* - пересечение множеств;

\- - вычитание множеств;

=,<> - проверка множеств на равенство, неравенство;

<=,>= - проверка множеств на включение;

in - проверка на принадлежность элемента множеству ( c in a).

**Операция “равно” (=).** Два множества *A* и *B* считаются равными, если они состоят из одних и тех же элементов. Порядок следования элементов в сравниваемых множествах значения не имеет.

| Значение A | Значение B | Выражение | Результат |
|---|---|---|---|
| *[1, 2, 3, 4]* | *[1, 2, 3, 4]* | *A = B* | *True* |
| *['a'..'2']* | *['b'..'2']* | *A = B* | *False* |
| *['a'..'2']* | *['2'..'a']* | *A = B* | *True* |

**Операция “не равно” (<>).** Два множества *A* и *B* считаются не равными, если они отличаются по мощности или по значению хотя бы одного элемента.

| **Значение A** | **Значение B** | **Выражение** | **Результат** |
|---|---|---|---|
| *[1, 2, 3,]* | *[3, 1, 2, 4]* | *A <> B* | *True* |
| *['a'..'2']* | *['6'..'2']* | *A <> B* | *True* |

**Операция “больше или равно” (>=).** Используется для определения принадлежности множества. Результат операции *A >= B* равен *True*, если все элементы множества *B* содержатся во множестве *A*. В противном случае результат равен *False*.

| **Значение A** | **Значение B** | **Выражение** | **Результат** |
|---|---|---|---|
| *[1, 2, 3, 4]* | *[2, 3, 4]* | *A >= B* | *True* |
| *['a'..'2']* | *['b'..'t']* | *A >= B* | *True* |
| *['z','x','c']* | *['c','x']* | *A >= B* | *True* |

**Операция “меньше или равно” (<=).** Используется аналогично предыдущей операции. Результат выражения *A <= B* равен *True*, если все элементы множества *A* содержатся во множестве *B*. В противном случае результат равен *False*.

| **Значение A** | **Значение B** | **Выражение** | **Результат** |
|---|---|---|---|
| *[1, 2, 3,]* | *[1, 2, 3, 4]* | *A <= B* | *True* |
| *['a'..'h']* | *['2'..'a']* | *A <= B* | *True* |
| *['a'..'v']* | *['a','n','v']* | *A <= B* | *True* |

**Операция *In*.** Используется для проверки принадлежности какого-либо значения указанному множеству. Обычно применяется в условных операциях.

| **Значение A** | **Выражение** | **Результат** |
|---|---|---|
| *2* | *if A in [1, 2, 3] then...* | *True* |
| *'v'* | *if A in ['a'..'n'] then...* | *False* |
| *x1* | *if A in [x0, x1, x2, x3] then...* | *True* |

Все значения множества представляются в памяти последовательностями битов одинаковой длины. За каждое значение базового типа "отвечает" один бит. Если множество содержит некоторый элемент, в "ответственном" за него бите хранится 1, если не содержит - хранится 0.

**Задача №1**. Имеются три множества символьного типа, которые заданы своими конструкторами: Y1=['A','B','D','R','M']; Y2=['R','A','H','D']; Y3=['A','R']; Сформировать новое множество: X=(Y1*Y2)+(Y1\Y2). Вывести на печать полученное множество x, проверить, включено ли y3 во множество x. (файл MNOG1.PAS)

**Решение.**

```
program mnog1;
var y1,y2,y3,x: set of char;
c:char;
begin
y1:=['a','b','d','r','m'];
y2:=['r','a','h','d'];
y3:=['a','r'];
x:=(y1*y2)+(y1-y2);
write('множество x=');
for c:='a' to 'r' do
if c in x then write(c);
writeln;
if y3<=x then write('y3 включено в x')
else write('y3 не включено в x');
end.
```

**Задача №2.** Из множества целых чисел 1..20 выделить: множество чисел, делящихся без остатка на 6; множество чисел, делящихся без остатка или на 2, или на 3. (файл MNOG2.PAS).

**Решение.**

```
program mnog2;
const n=20;
var n2,n3,n6,n23: SET OF 1..N;
k:1..N;
begin
n2:=[ ]; n3:=[ ];
for k:=2 to n do
begin
if (k mod 2)=0 then n2:=n2+[k];
if (k mod 3)=0 then n3:=n3+[k];
end;
n6:=n2*n3;
n23:=n2+n3;
writeln('на 6 делятся числа:');
for k:=1 to n do
if k in n6 then write(k:3);
writeln;
writeln('на 2 или на 3 делятся числа:');
for k:=1 to n do
if k in n23 then write(k:3);
end.
```

**Задача №3.** Дан текст (например, 5a7233111bge2257cde.) Признаком конца текста является точка. Вывести на экран цифры, которые встречаются в тексте. (файл MNOG3.PAS).

**Решение.**

```
program mnog3;
var z: SET OF 0..9;
k,i:integer;
si:char;
begin
readln;
write('=>');
z:=[ ];
repeat
read(si);
k:=ord(si)-ord('0');
if k in [0..9] then z:=z+[k]
until si='.';
if z=[ ] then writeln ('В тексте нет цифр')
else begin
```

```
write('В тексте имеются следующие цифры:');
for i:=1 to 9 do
if i in z then write (i:2);
writeln;
end;
end.
```

**Задания для практического занятия:**

1. Найдите ответ на вопрос "Какие структурированные типы данных существуют в Паскале?".
2. Найдите ответ на вопрос "Что понимается под множеством в Паскале?"
3. Найдите ответ на вопрос "Как объявляются множества?".
4. Какие типы данных используются в качестве базовых при построении множественных типов.
5. Найдите ответ на вопрос "Как в Паскале обозначаются операции над множествами?".
6. Какие операции определены над переменными множественного типа и каков их приоритет?
7. Чем похожи и чем отличаются множества и массивы,
8. Как представляются множества в памяти.

a) Вычислить следующие выражения:

a) [5]<=[1..5]

b) ['a'..'d', 'k'..'m'] + ['d'..'k'];

c) [Москва, Минск, Сочи]*[Сочи]

d) [7,1,3..6]=[1..7]

e) [',','9','0','.']-[',','.']

f) 15 in [1..10]

g) ['A'..'D', 'K'..'M'] + ['D'..'K'];

h) [Jan, Feb, Mar]*[Mar];

i) [2, 1, 3..6] = [1..7];

j) 15 In [1..10];

k) [',', '(', ')', '.'] - [',', '.'].

9. Дано описание переменной множественного типа: Var Pm: Set of (Red, Grey, Blue, Black). Выписать все допустимые значения этой переменной.
10. Будут ли равны множества:

a. ['A'..'D'] и ['A','B','C','D'];

b. [White, Black] и [Black, White].

11. Вычислить выражение: [1..14]*[5, 12..60] + [4..7] - [2*16]*[6].
12. Упростить данное выражение множественного типа:

[11..17]*[2] + [7, 17..40]*[2..17] - [2..8];

13. Познакомиться с условием задачи № 1.
14. Вызвать с диска файл (mnog1.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ примера № 1 записать в тетрадь.
15. Найдите ответ на вопрос "Как организована в программе проверка на включение множества Y3 в множество X?".
16. Познакомиться с условием задачи № 2.
17. Вызвать с диска файл (mnog2.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ записать в тетрадь. Ответить на вопрос "Как в программе сформировано множество чисел, делящихся без остатка на 6; множество чисел, делящихся без остатка или на 2, или на 3?"
18. Познакомиться с условием задачи № 3.
19. Вызвать с диска файл (mnog3.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ записать в тетрадь. Разобраться как работает данная программа.

**Инструкция по выполнению практической работы**

1. Рассмотрите примеры из теоретической части.

2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны отчет по каждому заданию.

## Файлы последовательного доступа.

### Параметры-значения

В окне редактора наберите программу и сохраните файл под именем T051.PAS:

```
Var X,Y : Integer;                                  1
    Procedure Reverse ( A, B : Integer );           2
    Begin                                           3
      A := A * 10; B := B + 5;                      4
        WriteLn ( B : 3, A : 3)                     5
    End;                                            6
Begin                                               7
    X := 1; Y := 100;                               8
Reverse  (X, Y);                                    9
 Reverse  (4, 5);                                   10
 Reverse  (4 * X, 5 * Y);                           11
 End.                                               12
```

Начертите в тетради таблицу:

| | Фактические параметры | | Формальные параметры | | Результат вывода на экран |
|---|---|---|---|---|---|
| | 1-ый параметр | 2-ой параметр | 1-ый параметр | 1-ый параметр | |
| | 1 | 2 | 3 | 4 | 5 |
| | | | | | |
| | | | | | |
| | | | | | |

Выполните программу. Обратите внимание на полученный результат.

Выполните программу еще раз, но в пошаговом режиме (клавиша F7). Задайте в окне отладчика просмотр значений переменных X, Y и A, B. Для удобства просмотра измените размер и положение окон (программы и Watches) так, чтобы они не накладывались друг на друга. Обратите внимание, как меняются значения этих переменных.

Какое значение имеют переменные A и B перед обращением к процедуре Reverse и после? Ответ запишите в тетрадь, заполнив графы с 1 по 5.

### Параметры-переменные

В окне редактора наберите программу и сохраните файл под именем T052.PAS:

```
Var I : Integer;                                    1
Procedure Count ( X : Integer );                    2
Begin                                               3
   X := X + 1 ;                                     4
  Write ( ' X = ' , X );                            5
End;                                                6
Begin                                               7
    I := 5;                                         8
    Write ( ' I 1 = ' , I );                        9
    Count ( I );                                    10
    Write ( ' I 2 = ' , I );                        11
End.                                                12
```

Начертите в тетради таблицу:

| | Номер строки программы | Результат вывода на экран для процедуры | |
|---|---|---|---|
| | | Count (X : Integer ); | Count (Var X: Integer); |
| | 1 | 2 | 3 |

| | | | |
|---|---|---|---|
| 1 | 9-я строка | | |
| 2 | 5-я строка | | |
| 3 | 11-я строка | | |

Выполните программу. Результат выполнения программы (какие сообщения выдаются на экран монитора) запишите во второй столбец таблицы

Внесите изменения в программу, заменив строку 2 на следующий текст:

**Procedure Count ( Var X : Integer );**

Выполните программу. Результат выполнения программы (какие сообщения выдаются на экран монитора) запишите в третий столбец таблицы.

Какая разница между полученными результатами во втором и в третьем столбцах? Чем это можно объяснить? Запишите ответ в тетрадь.

Выполните программу еще раз, но в пошаговом режиме с прослеживанием работы процедуры Count (клавиша F7). Задайте для просмотра в окне отладчика переменные I и X. Обратите внимание, как меняются значения этих переменных. Почему получены такие результаты, и как это объяснить?

Локальные и глобальные переменные

В окне редактора наберите программу и сохраните файл под именем T053.PAS:

```
Var A, B : Integer;                                  1
Procedure ZB1;                                       2
Var A: Integer;                                      3
Begin                                                4
A := 10; B:= 15;                                     5
WriteLn ('A2 = ', A, 'B2 = ', B)                     6
End;                                                 7
Begin                                                8
A:= 18; B:=37;                                       9
WriteLn ('A1 = ',A, 'B1 = ',B);                      10
ZB1;                                                 11
WriteLn ('A3 = ',A, 'B3 = ',B)                       12
End.                                                 13
```

**Начертите в тетради таблицу:**

| | Номер строки программы | Результат вывода на экран |
|---|---|---|
| | 1 | 2 |
| 1 | 6-я строка | |
| 2 | 10-я строка | |
| 3 | 12-я строка | |

Выполните программу в пошаговом режиме с прослеживанием работы процедуры ZB1 (клавиша F7).

Задайте для просмотра в окне отладчика переменные A и B. Обратите внимание, как меняются значения этих переменных.

1. Запишите в тетрадь результаты выполнения программы, заполнив таблицу. Почему получены такие результаты, и как это объяснить?

Вопросы для повторения

2. Что такое подпрограмма? Для чего нужна подпрограмма?
3. Какие виды подпрограмм есть в Turbo Pascal? Чем они отличаются друг от друга?
4. Что такое процедура? Ее общий вид.
5. Что называется параметром, и каково его назначение?
6. Правила записи параметров процедуры.
7. Формальные и фактические параметры. Каково отличие параметров-значений от параметров-переменных? Что может выступать в качестве параметров-значений и параметров-переменных?

8. Как вызвать процедуру в программе?

9. Опишите последовательность событий при вызове процедуры с параметрами-значениями.

10. Опишите последовательность событий при вызове процедуры с параметрами-переменными.

11. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры.

12. Чем отличаются локальные переменные от глобальных переменных?

13. Напишите, что выведет на экран следующая программа. Где в программе используется параметр-переменная, а где параметр-значение?

```
Var A,B:Integer;

Procedure TR54 ( Var X,Y:Integer );
Begin  Y:=Y+5; X:=2*Y;     End;

Procedure TR55(Y:Integer; Var X:Integer);
Begin  Y :=Y + 5; X := 2 * Y;          End;

Begin
A:=7;B:=-8;
TR54 (A,B); WriteLn( 'A = ', A, 'B = ', B);
TR55 (A,B); WriteLn(' A = ', A, 'B = ', B);
End.
```

14. Напишите, что выведет на экран следующая программа:

```
Var A,B:Integer;

Procedure TR56 (X:Integer; Var Y:Integer);
Begin
If X < 0 Then Y:= – X + 3
          Else Y := X * 2
End;

Begin
A := – 2 ; B := 3;
TR56 (A + 3, B);      WriteLn ( 'A = ', A, 'B = ', B);
TR56 (1 - A, B);      WriteLn (' A = ', A, 'B = ', B);
TR56 (7, B);          WriteLn (' A = ', A, 'B = ', B) End.
```

15. Установите, что будет выведено на экран монитора в результате работы первого и второго вариантов программы. Сравните результаты и сделайте вывод.

```
Var X : Real;
    Procedure TR57;
    Var X : Real;
    Begin
        WriteLn ( X )
    End;
Begin
    X := Pi;
    TR57
End.
```

```
Var X : Real;
    Procedure TR58;

    Begin
        WriteLn ( X )
    End;
Begin
    X := Pi;
    TR58
End.
```

| | |
|---|---|

Задачи для решения (на занятии или дома)

Напишите программы с использованием процедур.

1. Используя процедуру вычисления степени числа, вычислить значение выражения: $y=a_1*x^4+a_2*x^3+a_3*x^2+a_4*x+a_5$
2. Используя процедуру обмена местами двух значений, упорядочить значения трех переменных в порядке их возрастания.
3. Дано натуральное число. Найти все его делители. Посчитать их количество.
4. Даны два натуральных числа. Определить, является ли первое число перевертышем второго.

## *Практическая работа №4. Типизированные файлы. Нетипизированные файлы.*

**Цель работы:** научиться описывать в программе типизированные файлы, обрабатывать типизированные и нетипизированные файлы.

Оборудование: ПК, ИСР Pascal ABC

### Теоретические материалы

Типизированные файлы

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (т.е. доступ к компоненту по его порядковому номеру).

Перед первым обращением к процедурам ввода-вывода указатель файла стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода-вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

**Процедура READ**

Обеспечивает чтение очередных компонентов типизированного файла.Формат обращения: READ (<ф.п.>,<сп.ввода>)

Здесь <сп.ввода> - список ввода, содержащий одну или более переменных такого же типа, что икомпоненты файла.

Файловая переменная <ф.п.> должна быть объявлена предложением FILE OF... и связана с именем файла процедурой ASSIGN. Файл необходимо открыть процедурой RESET. Если файл исчерпан, обращение к READ вызовет ошибку ввода-вывода.

**Процедура WRITE**

Используется для записи данных в типизированный файл.Формат обращения: WRITE (<ф.п.>,<сп.вывода>)

Здесь <сп.вывода> - список вывода, содержащий одно или более выражений того же типа, что икомпоненты файла.

**Процедура SEEK**

Смещает указатель файла к требуемому компоненту.

Формат обращения: SEEK (<ф.п.>,<N компонента>)

Здесь <N компонента> - выражение типа LONGINT, указывающее номер компонента файла.Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

**Функция FILEPOS**

Возвращает значение типа LONGINT, содержащее порядковый номер компонента файла, который будет обрабатываться следующей операцией ввода-вывода.

Формат обращения: FILEPOS (<ф.п.>)

Функцию нельзя использовать для текстовых файлов. Первый компонент файла имее тпорядковый номер 0.

**2. Нетипизированные файлы**

Нетипизированные файлы объявляются как файловые переменные типа FILE и отличаютс я тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы, с одной стороны, совместимыми с любыми другими файлами, а с другой -

позволяет организовать высокоскоростной обмен данными между диском и памятью.

Чтение файла осуществляется с помощью стандартной процедуры RESET:

RESET (<ф.п.>);

Здесь <ф.п.> - файловая переменная,

связанная ранее процедурой ASSIGN с ужесуществующим файлом

При выполнении этой процедуры дисковый файл подготавливается к чтению информации . В результате специальная переменная-указатель, связанная с этим файлом, будет указывать на начало файла, т.е. на компонент с порядковым номером 0.

Стандартная процедура REWRITE (<ф.п.>).

инициирует запись информации в файл, связанный ранее с файловой переменной <ф.п.>. Процедурой REWRITE нельзя инициировать запись информации в ранее существовавший диско вый файл: при выполнении этой процедуры старый файл уничтожается и никаких сообщений об этом в программу не передается. Новый файл подготавливается к приему информации и его указ атель принимает значение 0.

При инициации нетипизированного файла процедурами RESET или REWRITE можно указать длину записи нетипизированного файла в байтах.

Например, так:

```
var
f
:
file;beg
in
.....
assign(f,
'myfile.dat')reset(f,512
);
.....
end.
```

Длина записи нетипизированного файла указывается вторым параметром при обращениик процедурам RESET или REWRITE, в качестве которого может использоваться выражение типа WORD. Если длина записи не указана, она принимается равной 128 байтам.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением READ и WRITE

Работа с файлам записей

Пример1. Сформировать файл Fm.dat, содержащий экзаменационную ведомость одной ст уденческой группы. Записи файла состоят из следующих полей: фамилия, имя, отчество, номер зачетной книжки, оценка.

```
Program
examen;Uses
crt;
Type
  stud=recordfi
  o:string[30];
  Nz:string[6];
  mark:integer
  end;
Var Fstud
 :file of stud;
S:stud;
N,I
:byte;Begi
n clrscr;


Assign(Fstud,
  'Fm.dat'); Write('kolv
  o
  studentov');Readln(n)
  ;
For
  I:=1  To  n
  doBegin
    Write('vvedite fio:');
    readln(s.fio); Write('vvedite nomer
    zachetki:'); readln(s.Nz);Write('vvedite
    ocenka:');
    readln(s.mark); Write(Fstud,s)
  end;
  Writeln('Formirovanie zave
  rsheno');Close(Fstud)
End.
```

Прямой доступ к записям файла

В стандарте языка Паскаль допустим только последовательный доступ к элементам файла. Одной из дополнительных возможностей, реализованный в Паскале, является прямой доступ к записям файла

Задав номер элемента файла, можно непосредственно установить указатель на него. После этого можно читать или перезаписывать данный элемент. Установка указателя на нужный элемент файла производится п роцедурой:

Процедура SEEK

Смещает указатель файла к требуемому компоненту. Формат обращения:SEEK (<ф.п.>,<N компонента>)

Здесь <N компонента> - выражение типа LONGINT, указывающее номер компонента файла.Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

**ЗАДАНИЕ**

1. Записать в файл последовательного доступа N действительных чисел. Выч ислитьпроизведение компонентов файла.
2. В типизированном файле, компонентами которого являются целые числа, подсчитатьколичество отрицательных чисел.
3. В типизированном файле, компонентами которого являются вещественные числа,вычислить произведение положительных чисел.

## *Практическая работа №5. Организация процедур. Организация функций.*

**Цель работы:**

- овладение практическими навыками организации работы в среде ABC PASCAL
- научиться разрабатывать программы с использованием процедур и функций
- приобрести практические навыки по организации подпрограмм

**Оборудование:**

- программная часть - интегрированная среда ABC Pascal
- задание на выполнение работы

**Задания:**

**1.** Используя подпрограммы-функции и написать программу вычисления:

| | |
|---|---|
| 1. | Даны действительные числа **a**, **b**, **c**. Получить: $\frac{\max(a,a+b)+\max(a,b+c)}{1+\max(a+bc,b,15)}$ |
| **2.** | Даны действительные числа **a**, **b**. Получить **u = min(a, b-a)**, **y = min(ab, a+b)**, **k = min(u+v$^2$, 3.14)**. |
| **3.** | Даны действительные числа **s**, **t**. Получить: **g(1.2,s)+g(t,s)-g(2s-1.5t)**, **\|g(ln(s,t+1))-g(t,s)\|**, где $g(a,b)=\frac{a^2+b^2}{a^2+2ab+3b^2+4}$ |
| **4.** | Даны действительные числа **x**, **y**. Получить: **f(x,-2y,1.17)+f(2.2,x,x-y)**, **tg(f(x+y,xy,y-x)+f(3.1,1.4,y-sinx))**, где $f(a,b,c)=\frac{2a-b-\sin c}{5+\lvert c\rvert}$. |
| **5.** | Даны неотрицательные целые числа **a**, **b**. Найти **F(a, b)**, где $F(m,n)=\frac{n!-m!}{(n+m)!}$. (Определить вспомогательную функцию, вычисляющую факториал). |
| **6.** | Даны два натуральных числа **a**, **b**. Вычислить $\frac{a!!-ab}{a!!+ab}$. Функция **x!!** Определяется следующим образом: |

| | |
|---|---|
| | **x!! = 1*3*5*...*x**, если **x** нечетно,<br>**x!! = 2*4*6*...*x**, если **x** четно. |
| **7.** | Даны действительные числа **a**, **b**, **c**. Получить<br>$\frac{\min(a, a+b)+\min(a, b+c)}{1+\min(a+bc, b)}$. |
| **8.** | Даны действительные числа **a**, **b**. Получить **r = max (a, b + a)**, **d = max (ab, a + b)**, **s = max (r + d$^2$, 3.14)**. |
| **9.** | Вычислить $A_n^k = \frac{n!}{(n-k)!}$ и $P_n = n!$. Нахождение факториала числа оформите в виде функции. |
| **10.** | Определить количество счастливых билетов на отрезке от N до M (билет счастливый, если сумма первых трех его цифр равна сумме последних трех). |
| **11.** | Дано простое число. Составить функцию, которая будет находить следующее за ним простое число. |
| **12.** | Написать программу для вычисления суммы факториалов всех нечетных чисел от 1 до m. |

**1. Написать программу используя процедуры:**

| | |
|---|---|
| Вариант 1. | Задана окружность **(x-a)$^2$ + (y-b)$^2$ = R$^2$** и точки **P(p1, p2)**, **F(f1, f2)**, **L(l1, l2)**. Выяснить и напечатать, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры. |
| **2.** | Используя процедуру вычисления степени числа, вычислить значение выражения:<br>y=a1*x4+a2*x3+a3*x2+a4*x+a5 |
| **3.** | Используя процедуру обмена местами двух значений, упорядочить значения трех переменных в порядке их возрастания. |
| **4.** | Дано натуральное число. Найти все его делители. Посчитать их количество используя процедуру. |
| **5.** | Даны два натуральных числа. Определить, является ли первое число перевертышем второго используя процедуру. |
| **6.** | Даны действительные числа S и T. Получить G(1.2, s) + G(t,s) + G(2*s-l,s*t), где G(A,B) = (A2 + b2)/(A2 + 2*A*b + B2 + 3*B2 + 4). |
| **7.** | Даны действительные числа S,T. Получить H(s,t) + max(H(s-t,s*t)2,H(s-t,s+t)4) + H(l,l), где H(A,B) = A/(1+A2) + B/(1+B2)-(A2-B2). |
| **8.** | Треугольник задан координатами своих вершин. Составить программу для вычисления его площади с использованием подпрограммы |
| **9.** | Четыре точки заданы своими координатами X(x1, x2), Y(y1, y2), Z(z1, z2), P(p1, p2). Выяснить, какие из них находятся на максимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры. |
| **10.** | Задана окружность (x-a)2 + (y-b)2 = R2 и точки P(p1, p2), F(f1, f2), L(l1, l2). Выяснить и напечатать, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры. |
| **11.** | Четыре точки заданы своими координатами X(x1, x2, x3), Y(y1, y2, y3), Z(z1, z2, z3), T(t1, t2, t3).Выяснить, какие из них находятся на минимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры. |
| **12.** | Число является автоморфным, то есть квадрат этого числа заканчивается этим же числом, например, число 6, так как его квадрат – 36 заканчивается на 6 или число 25 – его полный квадрат 625. Найти все автоморфные числа из промежутка от A до B. |

**Контрольные вопросы:**

1) В чем разница между процедурами и функциями?
2) Описание процедур и функций в программе.

3) Обращения к процедурам и функциям.
4) Локальные и глобальные переменные языка Паскаль.
5) Формальные и фактические параметры.

## Организация функций. Организация процедур

1. Приступая к разработке программы с функциями, повторите ОК.

ФУНКЦИИ В ПАСКАЛЕ

Повторяющиеся фрагменты программы оформляются в языке программирования в подпрограммы специальным образом. Подпрограмме дается имя, по которому к ней можно обращаться (вызывать подпрограмму). Использование подпрограмм не только улучшает структуру и внешний вид программы, но и уменьшает вероятность ошибок и облегчает отладку.

В Паскале имеется два вида подпрограмм — процедуры и функции. Их структура очень похожа на структуру основной программы.

Описание ФУНКЦИИ

Заголовок функции состоит из слова Function, за которым указывается имя функции, затем в круглых скобках записывается список формальных параметров, далее ставится двоеточие и указывается тип результата функции.

В теле функции обязательно должен быть хотя бы один оператор присваивания, в левой части которого стоит имя функции, а в правой — ее значение. Иначе значение функции не будет определено.

Общий вид описания функции следующий: Function Имя[(Список формальных параметров)]: Тип результата;

**Begin**

Тело функции, в котором обязательно должно быть присваивание
Значение:= Имя функции

**End;**

2. Разобрать и проанализировать пример использования функции в программе.

**Задача 1**

Составить программу, подсчитывающую число сочетаний без повторения из n элементов по k.

Число сочетаний без повторения вычисляется по формуле:

Решение

Обозначим через n и k переменные для хранения введенных чисел; C — переменную для хранения результата.

Чтобы подсчитать количество сочетаний без повторения, необходимо вычислить n!, (n—k)!, k!

**I этап**. Опишем функцию для вычисления факториала числа n (n!= 1* 2*... * n).

```
Function factorial(n:Integer):Longint; {заголовок функции}
   Var  {описательная часть}
 i: Integer;
        rez: Longint;
 Begin   {тело функции}
rez:=l;
For i:=l To n Do rez:=rez*i;
factorial:=rez;   {присваивание значения имени функции}
End;
```

Первая строчка в описании функции — это ее заголовок. Служебное слово Function (функция) указывает на то, что именем factorial названа функция. В скобках записан список формальных параметров функции, состоящий из одной переменной целого типа. Далее в заголовке указан тип значения функции. В данном примере результат функции factorial — длинное целое число.

За заголовком функции следует описательная часть функции, которая, как и у программы, может состоять из разделов описания переменных, констант, типов. В данном примере имеется

только раздел описания переменных. В нем описаны переменные / (счетчик цикла) и rez (для накопления значения факториала).

Далее идет раздел операторов (тело функции). Результат присваивается имени функции, таким образом, функция получает свое значение.

**II этап.** Составим программу нашего примера с вызовом функции.

В тексте программы описания функций всегда следуют за разделом описания переменных и до начала основной части, как и описания процедур. После того как функция описана, ее можно использовать в программе.

```
Program Example;
Var n,k: Integer;
 al, a2, a3, c: Longint;
 Function factorial(n:Integer):Longint;
Var i: Integer;
rez: Longint;
 Begin rez:=l;
For i:=l To n Do rez:=rez*i;
 factorial :=rez;
 End;
 Begin
writeLn ('Ввод n и k:');
{ вычисление n!}
{вычисление k!}
{вычисление (n-k)!}
{результат}
{печать результата}
ReadLn(n,k);
 al:=factorial(n); a2:=factorial(k); a3:=factorial(n-k); c:=al div (a2*a3); Writeln(c);
 ReadLn;
End.
```

```
    Program Имя;

Описательная часть программы
Function Имя [(Список формальных параметров)]:
Тип результата;
Описательная часть функции
Begin
Тело функции
 End;
Begin
Тело программы (функция вызывается по имени factorial для вычисления три раза)
End.
```

Пусть n=5, k=3. Когда в программе встречается оператор al: =factorial (n), выполняются следующие действия:

выделяется память для переменных, описанных в функции factorial;

формальному параметру присваивается значение фактического: n : =п (п=5);

выполняется функция, вычисляется факториал числа 5;

значение функции передается в место обращения к этой функции, то есть присваивается переменной al.

В операторах a2:=factorial(k) и a3:=factorial(n-k) еще дважды вызывается функция factorial с параметрами k=3 и n-k=2. Всего в программе имеется 3 обращения к функции factorial, столько же раз выполняются и описанные выше действия.

Еще раз подчеркнем, что функция — это самостоятельная часть программы, имеющая собственные переменные, которым отводится отдельное место в памяти ЭВМ. Этим объясняется тот факт, что переменные с одинаковыми именами, используемые в функции и в основной программе, являются разными (в рассмотренном примере - переменная n основной программы и параметр n

функции). При выполнении программы машина «не путает» имена этих переменных, так как области их действия не совпадают.

1. Проверить работу программы.
2. Ответить на контрольные вопросы:
1. Что называется подпрограммой в Турбо Паскале?
2. В каком разделе программы описываются Функции в программе?
3. В каком случае в программе используется подпрограмма-функция?

## *Практическая работа №6.* ***Применение рекурсивных функций.***

**Цель:** отработать навыки использования функций

1. Для каждой задачи описать назначение каждой переменной и работу каждой строки программы (коротко).
2. Исполнить программу задачи № 1 для:

| №№ | X | Y | X1 | Y1 | X2 | Y2 | X3 | Y3 | Ответ |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 21 | 7 | 10 | 11 | 20 | 3 | 30 | 16 | |
| **2** | 187 | 93 | 162 | 102 | 213 | 76 | 296 | 157 | |
| **3** | 200 | 100 | 10 | 170 | 160 | 10 | 300 | 190 | |

3. Исполнить программу задачи № 2 для:

| №№ | A | B | Ответ |
|---|---|---|---|
| **1** | 2.3 | 21.3 | |
| **2** | -0,6 | 31.2 | |
| **3** | 0.15 | -2.1 | |

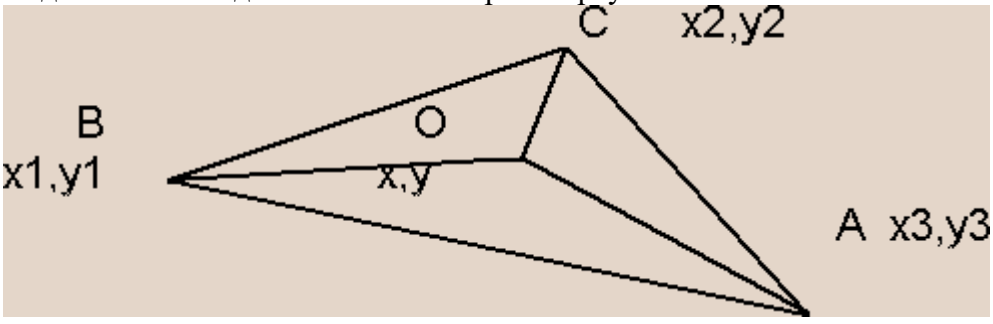
4. Исполнить программу задачи № 3 для:

| №№ | n | k | C |
|---|---|---|---|
| 1 | 10 | 23 | |
| 2 | 25 | 6 | |
| 3 | 4 | 11 | |
| 4 | 15 | 7 | |

5. Исполнить программу задачи № 4 исправив ее так, чтобы выводилось количество цифр каждого числа для:

| №№ | N1 | K1 | N2 | K2 | Результат |
|---|---|---|---|---|---|
| 1 | 12345 | | 456 | | |
| 2 | 23 | | 345678 | | |
| 3 | 11119 | | 89 | | |
| 4 | 11123 | | 30567 | | |

**Задача № 1.** Даны координаты вершин треугольника и точки внутри него. Найти расстояние от данной точки до ближайшей стороны треугольника.




**Решение.**

1. Нахождение длин сторон (OB, OC,OA,BA,CA,BC).
2. Нахождение площади трех треугольников S1(ABO), S2 (OBC), S3 (OCA).
3. Нахождение высот треугольников.
4. Сравнение высот (нахождение до ближайшей стороны).

```
program tr;
var x,y,x1,y1,x2,y2,x3,y3:integer;
ob,oc,oa,bc,ca,ba,s1,s2,s3,h1,h2,h3,n1,n2: real;

procedure dlin(r,t,c,d:real; var l1:real);
begin
l1:=sqrt(sqr(r-c)+sqr(t-d));
end;

procedure pl(f,g,j:real; var s:real);
var p:real;
begin
p:=(f+g+j)/2;
s:=sqrt(p*(p-f)*(p-g)*(p-j));
end;

procedure wis(a,s:real; var h:real);
begin
h:=2*s/a;
end;

procedure min(e,p:real; var n:real);
begin
if e<p then n:=e else n:=p;
end;

begin
read (x,y,x1,y1,x2,y2,x3,y3);
dlin(x,y,x1,y1,ob);
dlin(x,y,x2,y2,oc);
dlin(x,y,x3,y3,oa);
dlin(x1,y1,x3,y3,ba);
dlin(x2,y2,x3,y3,ca);
dlin(x1,y1,x2,y2,bc);
pl(ba,ob,oa,s1);
pl(ob,bc,oc,s2);
pl(oc,ca,oa,s3);
wis(ba,s1,h1);
wis(bc,s2,h2);
wis(ca,s3,h3);
min(h1,h2,n1);
min(n1,h3,n2);
writeln('расстояние до ближайшей стороны =',n2:5:2);
end.
```

**Задача № 2.** Вычислить: z=f(sin a, b) + f(cos a, b) + f (sin2 a, b-1) + f(sin a - cos a, b2 - 1) + f(sin2 a-1, cos a+b), где

$$f(u,t)=\begin{cases} u+\sin t, & \text{если } u>0 \\ u+t, & \text{если } u<=0 \end{cases}$$

```
program pr2;
var a,b,z:real;
```

```
function fun(u,t:real):real;
begin
if u>0 then fun:=u+sin(t) else fun:=u+t;
end;

begin
read (a,b);
z:=fun(sin(a),b)+fun(cos(a),b)+fun(sqr(sin(a)),b-1)+fun(sin(a)-cos(a),b*b-1)+fun(sqr(sin(a))-1,cos(a)+b);
writeln (z:8:2);
end.
```

**Задача №3**. Составить программу вычисления функции $C_n^k = \frac{n!}{k!(n-k)!}$

```
Program Example _ 30;
Var n, k: Integer;
        a1, a2, a3, c: Longint;

Function factorial (n: Integer): Longint;
Var i: Integer;
       rez: Longint;
Begin
     rez: = 1;
    For i: = 1 To n Do rez: = rez*i;
    factorial: = rez;
End;

Begin
  Writeln ('Введите n и k для подсчета числа C_n^k:');
  Readln (n, k);
 a 1: = factorial (n);                    {вычисление n!}
 a 2: = factorial (k);                    {вычисление k!}
 a 3: = factorial (n-k);                  {вычисление (n-k)!}
 c:=a1 Div (a2 * a3);                     {результат}
 Writeln (c);
 Readin;
End.
```

**Задача №4.** Написать функцию, подсчитывающую количество цифр числа. Используя её, определить, в какой их двух данных чисел больше цифр.

```
Program E_2;
Var n1,n2:Longint;
K1,k2:Byte;

Function quantity (x: Longint); Byte;
Var k: Byte;
Begin
     K:=0;
     While x<>0 Do
Begin
        Inc(k);
        X:=x Div 10;
     End;
     Quantity:=k;
```

```
End;

Begin
    Writeln('Введите два числа');
    Readln(n1,n2);
    K1=Quantity(n1);
    K2:=Quantity(n2)
    If k1=k2 Then Writeln('Одинаковое количество цифр');
    If k1<k2 Then Writeln('Во втором числе цифр больше');
    If k1>k2 Then Writeln('в первом числе цифр больше');
    Readln;
End.
```

6.Используя процедуры или функции, решите следующие задачи:

1. Найти сумму цифр числа.
2. Найти первую цифру числа.
3. Найти количество и сумму делителей числа
4. Определить, является ли число совершенным. То есть равно ли оно сумме своих делителей, кроме самого себя (используя преобразованную функцию из предыдущего примера).
5. Определить, является ли число простым.
6. Составить программу, проверяющую, является ли число палиндромом (например, число 12721 - палиндром)
7. Дано четыре числа. Вывести на экран наибольшую из первых цифр заданных чисел. (например, если даны числа 25, 730, 127, 1995, то должна напечататься цифра 7).

## Программирование модуля.

### Краткие теоретические материалы

**Задача.** Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида P/Q (P — целое, Q — натуральное): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) сокращение дроби; 6) возведение дроби в степень N (N — натуральное); 7) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

Дробь представить следующим типом:

```
Type Frac = Record
              P : Integer;
              Q : 1.. High(LongInt)
            End;
```

Используя этот модуль, решить задачи:

1. Дан массив A — массив обыкновенных дробей. Найти сумму всех дробей, ответ представить в виде несократимой дроби. Вычислить среднее арифметическое всех дробей, ответ представить в виде несократимой дроби.

2. Дан массив A — массив обыкновенных дробей. Отсортировать его в порядке возрастания.

```
Unit Droby;
Interface
  Type
    Natur = 1..High(LongInt);
    Frac = Record
             P : LongInt; {Числитель дроби}
             Q : Natur {Знаменатель дроби}
           End;
Procedure Sokr(Var A : Frac);
Procedure Summa(A, B : Frac; Var C : Frac);
Procedure Raznost(A, B : Frac; Var C : Frac);
Procedure Proizvedenie(A, B : Frac; Var C : Frac);
```

```
Procedure Chastnoe(A, B : Frac; Var C : Frac);
Procedure Stepen(A : Frac; N : Natur; Var C : Frac);
Function Menshe(A, B : Frac) : Boolean;
Function Bolshe(A, B : Frac) : Boolean;
Function Ravno(A, B : Frac) : Boolean;
Function MensheRavno(A, B : Frac) : Boolean;
Function BolsheRavno(A, B : Frac) : Boolean;
Function NeRavno(A, B : Frac) : Boolean;
{Раздел реализации модуля}
Implementation
{Наибольший общий делитель двух чисел - вспомогательная функция, ранее не объявленная}
Function NodEvklid(A, B : Natur) : Natur;
Begin
  While A <> B Do
    If A > B Then
          If A Mod B <> 0 Then A := A Mod B Else A := B
         Else
          If B Mod A <> 0 Then B := B Mod A Else B := A;
    NodEvklid := A
End;
Procedure Sokr; {Сокращение дроби}
Var M, N : Natur;
Begin
  If A.P <> 0 Then
  Begin
    If A.P < 0 Then M := Abs(A.P)
                Else M := A.P; {Совмещение типов, т.к. A.P - LongInt}
    N := NodEvklid(M, A.Q);    A.P := A.P Div N; A.Q := A.Q Div N
  End
End;
Procedure Summa; {Сумма дробей}
Begin
  {Знаменатель дроби}  C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q);
  {Числитель дроби}      C.P := A.P * C.Q Div A.Q + B.P * C.Q Div B.Q;
  Sokr(C)
End;
Procedure Raznost; {Разность дробей}
Begin
  {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q);
  {Числитель дроби}     C.P := A.P * C.Q Div A.Q - B.P * C.Q Div B.Q;
  Sokr(C)
End;
Procedure Proizvedenie;
Begin
  {Знаменатель дроби} C.Q := A.Q * B.Q;
  {Числитель дроби} C.P := A.P * B.P;
  Sokr(C)
End;
Procedure Chastnoe;
Begin
  {Знаменатель дроби}  C.Q := A.Q * B.P;
  {Числитель дроби}      C.P := A.P * B.Q;
  Sokr(C)
End;
Procedure Stepen; {Степень}
Var I : Natur;
Begin
```

```
    C.Q := 1; C.P := 1; Sokr(A);
    For I := 1 To N Do Proizvedenie(A, C, C)
End;
Function Menshe;
Begin Menshe := A.P * B.Q < A.Q * B.P End;
Function Bolshe;
Begin  Bolshe := A.P * B.Q > A.Q * B.P End;
Function Ravno;
Begin Ravno := A.P * B.Q = A.Q * B.P End;
Function BolsheRavno;
Begin BolsheRavno := Bolshe(A, B) Or Ravno(A, B) End;
Function MensheRavno;
Begin MensheRavno := Menshe(A, B) Or Ravno(A, B) End;
Function NeRavno;
Begin  NeRavno := Not Ravno(A, B) End;
{Раздел инициализации модуля}
Begin
End.
```

Дадим некоторые рекомендации по разработке модулей:

1) спроектировать модуль, т.е. выделить основные и вспомогательные подпрограммы, другие ресурсы;

2) каждую подпрограмму целесообразно отладить отдельно, после чего «вклеить» в текст модуля.

Сохраним текст разработанной программы в файле DROBY.PAS и откомпилируем наш модуль. Для этого можно воспользоваться внешним компилятором, поставляемым вместе с Turbo Pascal. Команда будет выглядеть так: TPC DROBY.PAS. Если в тексте нет синтаксических ошибок, получим файл DROBY.TPU, иначе будет соответствующее сообщение с указанием строки, содержащей ошибку. Другой способ компиляции модуля — в среде программирования Turbo Pascal выбрать в пункте меню Run подпункты Make или Build (при этом должна быть включена компиляция на диск).

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

Для примера решим задачу суммирования массива дробей.

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

**Задания для практического занятия:**

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

**Инструкция по выполнению практической работы**

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

**Порядок выполнения отчета по практической работе**

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

## *Практическая работа №7. Создание библиотеки подпрограмм. Использование указателей для организации связанных списков. Изучение интегрированной среды разработчика.*

**Учебная цель**: Отработать навыки работы создания библиотеки подпрограмм..

**Учебные задачи:**

1. Закрепить теоретические знания о понятии «библиотеки подпрограмм»
2. Закрепить навыки работы с библиотекой подпрограмм в языке программирования Pascal.
3. Овладеть навыками решения задач с использованием модулей
4. Закрепить практические навыки по составлению тестов и отладке программы.

**Образовательные результаты, заявленные во ФГОС третьего поколения:**

Студент должен

уметь:

- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.

знать:

- этапы решения задачи на компьютере

**Задачи практической работы**:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

**Краткие теоретические и учебно-методические материалы по теме практической работы**

Стандартный язык Pascal не располагает средствами разработки и поддержки библиотек программиста (в отличие, скажем, от языка Fortran и других языков программирования высокого уровня), которые компилируются отдельно и в дальнейшем могут быть использованы как самим разработчиком, так и другими. Если программист имеет достаточно большие наработки, и те или

иные подпрограммы могут быть использованы при написании новых приложений, то приходится эти подпрограммы целиком включать в новый текст. В Pascal это ограничение преодолевается за счет, во-первых, введения внешних процедур, во-вторых, разработки и использования модулей. В настоящей публикации на примерах рассмотрим работу с теми и другими программными единицами. Начнем с внешних подпрограмм. Такой механизм предусматривает, что исходный текст каждой процедуры или функции хранится в отдельном файле и при необходимости с помощью специальной директивы компилятора включается в текст создаваемой программы.

Покажем это на примере задач целочисленной арифметики, где аргументы, результаты и промежуточные величины являются целыми (Integer, Word, LongInt и т.д.). Вот несколько таких задач.

1. Дано натуральное число n. Найти сумму первой и последней цифры этого числа.

2. Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.

3. Дано натуральное число n. Дописать к нему цифру k в конец и в начало (если это возможно, т.е. результат не выйдет за диапазон допустимых значений), или сообщить о невозможности выполнения операции.

4. Найти наибольшую цифру в записи данного натурального числа.

5. Дано натуральное число n. Переставить его цифры так, чтобы образовалось максимальное число, записанное теми же цифрами.

При решении каждой из этих задач может быть использована функция, возвращающая количество цифр в записи натурального числа. Вот возможный вариант такой функции:

```
Function Digits(N : LongInt) : Byte;
Var Kol : Byte;
Begin Kol := 0;
While N <> 0 Do
 Begin Kol := Kol + 1;
N := N Div 10 End;
Digits := Kol
End;
```

Сохраним этот текст в файле с расширением .inc (это расширение внешних подпрограмм в Pascal), например, digits.inc.

Еще необходима функция возведения натурального числа в натуральную степень.

```
Function Power(A, N : LongInt) : LongInt; {файл power.inc}
Var I, St : LongInt;
Begin St := 1;
For I := 1 To N Do St := St * A;
Power := St End;
```

Попробуем использовать функции при решении задачи номер один.

```
Program Example1;
Var N, S : LongInt; {$I digits.inc} {подключаем внешнюю функцию digits.inc, возвращающую количество цифр в записи числа}
{$I power.inc} {внешняя функция, выполняющая возведение числа A в степень N}
Begin
Write('Введите натуральное число: ');
ReadLn(N); {для определения последней цифры числа N берем остаток от деления этого числа на 10, а для определения первой делим N на 10 в степени на единицу меньшую, чем количество цифр в записи числа (нумерация разрядов начинается с 0)}
S := N Mod 10 + N Div Power(10, Digits(N) - 1);
WriteLn('Искомая сумма: ', S)
End.
```

Внешние процедуры создаются и внедряются в использующие их программы аналогично функциям, и мы не будем подробно на этом останавливаться. Далее речь пойдет о модулях: их структуре, разработке, компиляции и использовании.

Модуль - это набор ресурсов (функций, процедур, констант, переменных, типов и т.д.), разрабатываемых и хранимых независимо от использующих их программ. В отличие от внешних подпрограмм модуль может содержать достаточно большой набор процедур и функций, а также других ресурсов для разработки программ. Обычно каждый модуль содержит логически связанные между собой программные ресурсы.

В основе идеи модульности лежат принципы структурного программирования. Существуют стандартные модули Pascal, которые обычно описываются в литературе по данному языку.

Модуль имеет следующую структуру:

Unit <имя модуля>; {заголовок модуля} Interface {интерфейсная часть}

Implementation {раздел реализации}

Begin {раздел инициализации модуля}

End.

После служебного слова Unit записывается имя модуля, которое (для удобства дальнейших действий) должно совпадать с именем файла, содержащего данный модуль. Поэтому (как принято в MS DOS) имя не должно содержать более 8 символов.

В разделе Interface объявляются все ресурсы, которые будут в дальнейшем доступны программисту при подключении модуля.

Для подпрограмм здесь указывается лишь полный заголовок. В разделе Implementation реализуются все подпрограммы, которые были ранее объявлены. Кроме того, здесь могут содержаться свои константы, переменные, типы, подпрограммы и т.д., которые носят вспомогательный характер и используются для написания основных подпрограмм. В отличие от ресурсов, объявленных в разделе Interface, все, что дополнительно объявляется в Implementation, уже не будет доступно при подключении модуля. При написании основных подпрограмм достаточно указать их имя (т.е. не нужно полностью переписывать весь заголовок), а затем записать тело подпрограммы.

Наконец, раздел инициализации (который часто отсутствует) содержит операторы, которые должны быть выполнены сразу же после запуска программы, использующей модуль. Приведем пример разработки и использования модуля.

Поскольку рассмотренная ниже задача достаточно элементарна, ограничимся листингом программы с подробными комментариями.

Задача. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида P/Q (P - целое, Q - натуральное): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) сокращение дроби; 6) возведение дроби в степень N (N - натуральное); 7) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

Дробь представить следующим типом:

Type Frac = Record P : Integer; Q : 1.. High(LongInt) End;

Используя этот модуль, решить задачи: 1. Дан массив A - массив обыкновенных дробей. Найти сумму всех дробей, ответ представить в виде несократимой дроби. Вычислить среднее арифметическое всех дробей, ответ представить в виде несократимой дроби. 2. Дан массив A - массив обыкновенных дробей. Отсортировать его в порядке возрастания.

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

```
Begin If A.P <> 0 Then Begin If A.P < 0 Then M := Abs(A.P) Else M := A.P; {Совмещение типов, т.к. A.P - LongInt}
 N := NodEvklid(M, A.Q); A.P := A.P Div N; A.Q := A.Q Div N End End;
Procedure Summa; {Сумма дробей}
Begin {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q); {Числитель дроби} C.P := A.P * C.Q Div A.Q + B.P * C.Q Div B.Q; Sokr(C) End;
Procedure Raznost; {Разность дробей} Begin {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q); {Числитель дроби} C.P := A.P * C.Q Div A.Q - B.P * C.Q Div B.Q; Sokr(C) End;
Procedure Proizvedenie; Begin {Знаменатель дроби} C.Q := A.Q * B.Q; {Числитель дроби} C.P := A.P * B.P; Sokr(C) End;
Procedure Chastnoe; Begin {Знаменатель дроби} C.Q := A.Q * B.P; {Числитель дроби} C.P := A.P * B.Q; Sokr(C) End;
Procedure Stepen; {Степень} Var I : Natur; Begin C.Q := 1; C.P := 1; Sokr(A); For I := 1 To N Do Proizvedenie(A, C, C) End; Function Menshe; Begin Menshe := A.P * B.Q < A.Q * B.P End;
Function Bolshe; Begin Bolshe := A.P * B.Q > A.Q * B.P End; Function Ravno; Begin Ravno := A.P * B.Q = A.Q * B.P End;
Function BolsheRavno; Begin BolsheRavno := Bolshe(A, B) Or Ravno(A, B) End;
 Function MensheRavno; Begin MensheRavno := Menshe(A, B) Or Ravno(A, B) End;
 Function NeRavno; Begin NeRavno := Not Ravno(A, B) End;
{Раздел инициализации модуля} Begin End.
```

Дадим некоторые рекомендации по разработке модулей: 1) спроектировать модуль, т.е. выделить основные и вспомогательные подпрограммы, другие ресурсы; 2) каждую подпрограмму целесообразно отладить отдельно, после чего «вклеить» в текст модуля.

Сохраним текст разработанной программы в файле DROBY.PAS и откомпилируем наш модуль. Для этого можно воспользоваться внешним компилятором, поставляемым вместе с Pascal. Команда будет выглядеть так: TPC DROBY.PAS. Если в тексте нет синтаксических ошибок, получим файл DROBY.TPU, иначе будет соответствующее сообщение с указанием строки, содержащей ошибку. Другой способ компиляции модуля - в среде программирования Pascal выбрать в пункте меню Run подпункты Make или Build (при этом должна быть включена компиляция на диск).

Теперь можно подключить модуль к программе, где планируется его использование. Для примера решим задачу суммирования массива дробей.

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

Вторую задачу предлагаем решить читателю самостоятельно. Как видно из примера, для подключения модуля используется служебное слово USES, после чего указывается имя модуля и происходит это сразу же после заголовка программы. Если необходимо подключить несколько модулей, они перечисляются через запятую. При использовании ресурсов модуля совсем не нужно знать, как работают его подпрограммы. Достаточно обладать информацией, как выглядят их заголовки и какое действие эти подпрограммы выполняют. По такому принципу осуществляется работа со всеми стандартными модулями. Поэтому, если программист разрабатывает модули не только для личного пользования, ему необходимо сделать полное описание всех доступных при подключении ресурсов. В таком случае возможна полноценная работа с таким продуктом.

Ещё несколько слов о видимости объектов модуля. Если в программе, использующей модуль, имеются идентификаторы, совпадающие с точностью до символа с идентификаторами модуля, то они «перекрывают» соответствующие ресурсы модуля. Тем не менее, даже в такой

ситуации доступ к этим ресурсам модуля может быть получен таким образом: <имя модуля>.<имя ресурса>.

**Индивидуальные задания**

I. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над комплексными числами: 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) вычисление модуля комплексного числа; 6) возведение комплексного числа в степень n (n - натуральное). Комплексное число представить следующим типом: Type Complex = Record R, M : Real; {действительная и мнимая часть числа} End; Используя этот модуль, решить задачи: 1. Дан массив A - массив комплексных чисел. Получить массив C, элементами которого будут модули сумм рядом стоящих комплексных чисел.

2. Дан массив A[M] - массив комплексных чисел. Получить матрицу B[N, M], каждая строка которой получается возведением в степень, равную номеру этой строки, соответствующих элементов данного массива A. II. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций с квадратными матрицами: 1) сложение двух матриц; 2) умножение одной матрицы на другую; 3) нахождение транспонированной матрицы; 4) вычисление определителя матрицы. Матрицу описать следующим образом: Const NMax = 10; Type Matrica = Array [1..NMax, 1..Nmax] Of Real; Используя этот модуль, решить следующие задачи: 1. Решить систему линейных уравнений N-го порядка (2<=N<=10) методом Крамера. 2. Задан массив величин типа Matrica. Отсортировать этот массив в порядке возрастания значений определителей матриц.

III. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над векторами на плоскости: 1) сложение; 2) вычитание; 3) скалярное умножение векторов; 4) умножение вектора на число; 5) длина вектора. Вектор представить следующим типом: Type Vector = Record X, Y : Real End; Используя этот модуль, решить задачи: 1. Дан массив A - массив векторов. Отсортировать его в порядке убывания длин векторов. 2. С помощью датчика случайных чисел сгенерировать 2N целых чисел. N пар этих чисел задают N точек координатной плоскости. Вывести номера тройки точек, которые являются координатами вершин треугольника с наибольшим углом.

IV. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над натуральными числами в P-ичной системе счисления (2<=P<=9): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) перевод из десятичной системы счисления в P-ичную; 6) перевод из P-ичной системы счисления в десятичную; 7) логическая функция проверки правильности записи числа в P-ичной системе счисления; 8) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). P-ичное число представить следующим типом: Type Chislo = Array [1..64] Of 0..8; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени записаны в P-ичной системе счисления). Ответ выдать в P-ичной и десятичной системах счисления. 2. Дан массив A - массив чисел, записанных в P-ичной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в P-ичной и десятичной системах счисления.

V. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над натуральными числами в шестнадцатеричной системе счисления: 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) перевод из двоичной системы счисления в шестнадцатеричную; 6) перевод из шестнадцатеричной системы счисления в десятичную; 7) функция проверки правильности записи числа в шестнадцатеричной системе счисления; 8) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени записаны в шестнадцатеричной системе счисления). Ответ выдать в шестнадцатеричной и десятичной системах счисления. 2. Дан массив A - массив чисел, записанных в шестнадцатеричной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в шестнадцатеричной и десятичной системах счисления.

VI. Определим граф как набор точек, некоторые из которых соединены отрезками, подграф - граф, подмножество данного графа. Реализовать в виде модуля набор подпрограмм, определяющих: 1) число точек в графе; 2) число отрезков в графе; 3) число изолированных подграфов в графе (подграфов, не соединенных отрезками); 4) диаметр графа - длину максимальной незамкнутой линии в графе (длина каждого звена - единица); 5) граф - объединение двух графов; 6) подграф - пересечение двух графов; 7) подграф - дополнение данного графа до полного (графа с тем же количеством вершин, что и в заданном, и с линиями между любыми двумя вершинами); 8) число отрезков, выходящих из каждой вершины графа; 9) при запуске должны инициализироваться

переменные: Full_Graph - полный граф с числом вершин NumberOfVertix, Null_Graph - граф без отрезков с числом вершин NumberOfVertix. Граф представить как объект Const NumberOfVertix = 50; Type Graph = Array[1..NumberOfVertix, 1..NumberOfVertix] Of Boolean; Используя модуль, решить задачу: найти все правильные графы из N вершин (граф правилен, если из всех вершин выходит равное количество отрезков).

VII. Реализовать в виде модуля набор подпрограмм для работы с длинными целыми числами (числами, выходящими за диапазон допустимых значений любого целого типа): 1) сложение; 2) вычитание; 3) умножение; 4) нахождение частного и остатка от деления одного числа на другое; 5) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). Длинное число представить следующим типом: Type Tsifra = 0..9; Chislo = Array [1..1000] Of Tsifra; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени - длинные числа). 2. Дан массив длинных чисел. Упорядочить этот массив в порядке убывания.

VIII. Реализовать в виде модуля набор подпрограмм для выполнения операций с многочленами от одной переменной (первый многочлен степени m, второй - степени n): 1) сложение; 2) вычитание; 3) умножение; 4) деление с остатком; 5) операции отношения (равно, не равно); 6) возведение в натуральную степень k одного из многочленов; 7) вычисление производной от многочлена; 8) вычисление значения в точке x0. Многочлен представить следующим типом: Type Mnogochlen = Array [1..500] Of Integer; Используя этот модуль, решить задачи: 1. Найти наибольший общий делитель многочленов P(x) и Q(x). 2. Вычислить: Ps(x)-Qr(x) (s, r - натуральные).

IX*. Реализовать в виде модуля набор подпрограмм для работы с длинными действительными числами (числами, выходящими за диапазон допустимых значений любого действительных типа или не представленных в памяти ЭВМ): 1) сложение; 2) вычитание; 3) умножение; 4) нахождение частного от деления одного числа на другое с заданным количеством знаков после запятой; 5) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше); 6) тригонометрические функции, где аргументом и значениями являются длинные действительные числа (указание: использовать разложение соответствующей функции в ряд). Длинное действительное число представить следующим типом: Type Tsifra = 0..9; Chislo = Array [1..1000] Of Tsifra; LongReal = Record Znak : 0..1; {0 - "плюс", 1 - "минус"} Ts, Dr : Chislo {целая и дробная части} End; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание - длинное действительное, показатель степени - длинное целое число). 2. Дан массив длинных действительных чисел. Упорядочить этот массив в порядке возрастания

**Задания для практического занятия:**

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

## Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

## Методика анализа результатов, полученных в ходе практической работы

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

## Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

# *Практическая работа №8. Создание проекта с использованием компонентов для работы с текстом. Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.*

**Учебная цель**: отработать навыки применения стандартных процедур и функций для обработки строк.

**Учебные задачи:**

1. Закрепить теоретические знания о понятии «строка»
2. Овладение навыками использования стандартных функций и процедур для работы со строками
3. Закрепить практические навыки по составлению тестов и отладке программы.

**Образовательные результаты, заявленные во ФГОС третьего поколения:**

Студент должен

уметь:

- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.

знать:

- этапы решения задачи на компьютере

**Задачи практической работы**:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки строк с использованием стандартных функций и процедур
4. Оформить отчет.

**Краткие теоретические и учебно-методические материалы по теме практической работы**

Паскалю и означает текстовую переменную. Ее запись: VAR A:STRING:[N]; где N - максимальная длина текстовой переменной.

Ввод текстовых переменных:

READ (A);

A='ТЕКСТ'

Операции и функции:

1. Две стринговые переменные можно складывать:

```
program ttt;
var a,b,c: srting[40];
begin
a:='программирование';
b:=' на Паскале';
c:=a+b;
write (c);
end.
```

2. LENGTH(переменная), определяет длину символьной переменной k:=length(a);
3. COPY(A,K,L) выделяет из строки заданную подстроку, где А - стринговая переменная;

K - начальная позиция с которой начинается подстрока;

L - количество рассматриваемых позиций.

4. INSERT(X,A,L) вставка подстроки в строку X - вставляемый фрагмент, А - стринговая переменная, куда вставляется фрагмент, L - позиция вставки
5. STR(X,A) преобразует число X в строку А

6. POS(X,A) находит в строке A подстроку X (позицию вхождения). Результат выполнения функции - целое число.

7. DELETE(X,A,L) удаление L символов из строки X, где X - стринговая переменная; A - позиция удаления; L - количество удаляемых символов.

8. CONCAT(S1,S2,...,SN) сцепление строк S1,S2,...,SN

9. VAL(A,X,C) преобразование строки A в число переменной X. Если параметр C содержит 0, значит преобразование прошло успешно, в противном случае C содержит номер позиции в строке A, где обнаружен неверный символ.

10. UPCASE(B) функция типа char, преобразует латинские строчные буквы в заглавные.

**Задания для практического занятия:**

1. Рассмотреть пример.
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

**Инструкция по выполнению практической работы**

**1. Запишите программу в тетрадь.**

```
program p1;
    uses crt;
    var x:real; y:integer; c,z:char;
    s,s1,s2,s3,s4,s5:string;
    begin
        clrscr;
        s2:='круговорот';
        writeln('длина слова круговорот', length(s2));
        s:=concat('12','345');
        writeln(s);
        s1:=copy(s,3,2);
        writeln(s1);
        y:=pos('рот',s2);
        writeln(y);
        z:=upcase(c);
        writeln(z);
        delete(s2,3,6);
        writeln(s2);
        insert('abc',s2,3);
        writeln(s2);
    end.
```

**2. Выполните задания.**

1. Проанализируйте устно, что должно быть на экране после выполнения программы (пример 1)? Ответы запишите карандашом против каждой строки вывода на экран.
2. Загрузите Паскаль.
3. Выполните программу. Сравните полученные вами ответы с ответами на ЭВМ.
4. Выполните задания для самостоятельной работы.

**Вариант-1**

1. Дана строка символов да точки. Группы символов в ней между пробелами считаются словами. Определить количество слов.

2. Дана строка символов до точки. Определить, является ли она записью десятичного числа кратного двум.

3. Дана строка символов до точки. Вывести ее на экран задом наперед.

4. Составить программу, которая заменяет все окончания “ый” на”ая”.

5.Подсчитать, сколько раз в данной строке встречается буква, вводимая с клавиатуры

6. Заменить две буквы X на У, причем X и У вводятся с клавиатуры.

**Вариант-2**

1. Из строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы.

2. Заменить А на АА во всем выражении.

3. На какую букву начинается больше слов в тексте?

4. Дана строка символов. Удалить из нее все знаки препинания.

5. Дана строка символов. Дано слово. Удалить из строки это слово.

6. Дана строка символов. Выделите подстроку между первой и последней точкой.

**Вариант-3**

1. Дала строка символов. Удалить из нее первый знак препинания.

2. Дана строка символов до точки. Определить, является ли она правильным скобочным выражением. Рассматривать только круглые скобки.

3. Составит программу, которая заменяет все буквы “О” на “ОО”.

4. Из строки удалить первую букву, если длина строки нечетная, иначе — удалить среднюю букву.

5. Сколько раз в тексте встречается заданное слово?

6. В данном предложении определите слова, которые начинаются с заданной буквы.

**Вариант-4**

1. Дана строка символов да точки. Группы символов в ней между пробелами считаются словами. Определить количество слов.

2. Дана строка символов до точки. Определить, является ли она записью десятичного числа кратного двум.

3. Дана строка символов до точки. Вывести ее на экран задом наперед.

4. Составить программу, которая заменяет все окончания “ый” на”ая”.

5.Подсчитать, сколько раз в данной строке встречается буква, вводимая с клавиатуры

6. Заменить две буквы Х на У, причем Х и У вводятся с клавиатуры.

**Вариант-5**

1. Из строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы.

2. Заменить А на АА во всем выражении.

3. На какую букву начинается больше слов в тексте?

4. Дана строка символов. Удалить из нее все знаки препинания.

5. Дана строка символов. Дано слово. Удалить из строки это слово.

6. Дана строка символов. Выделите подстроку между первой и последней точкой.

**Вариант-6**

1. Дала строка символов. Удалить из нее первый знак препинания.

2. Дана строка символов до точки. Определить, является ли она правильным скобочным выражением. Рассматривать только круглые скобки.

3. Составит программу, которая заменяет все буквы “О” на “ОО”.

4. Из строки удалить первую букву, если длина строки нечетная, иначе — удалить среднюю букву.

5. Сколько раз в тексте встречается заданное слово?

6. В данном предложении определите слова, которые начинаются с заданной буквы.

**Методика анализа результатов, полученных в ходе практической работы**

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

**Порядок выполнения отчета по практической работе**

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

***Практическая работа №9. События компонентов (элементов управления), их сущность и назначение. Создание процедур на основе событий. Создание проекта с использованием кнопочных компонентов. Создание проекта с использованием компонентов стандартных диалогов и системы меню.***

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

**Задание 1-го уровня**

1. Создать новый проект.
2. Составить эскиз интерактивной формы - калькулятора, способной выполнять простые арифметические действия (Рис. 1).
3. Задать значения свойств элементов управления, размещенных на интерактивной форме.
4. Для каждого элемента управления написать программный код, соответствующий событию активизации (нажатия) элемента управления.
5. Осуществить сборку и компиляцию модулей проекта.
6. Выполнить вычисления с помощью созданного калькулятора.

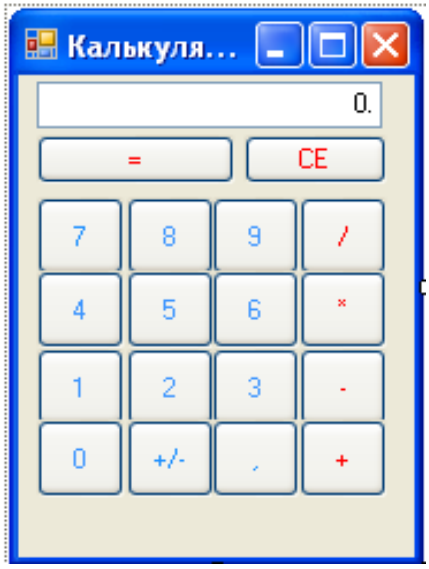



Рис. 1. Эскиз интерактивной формы

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

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

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

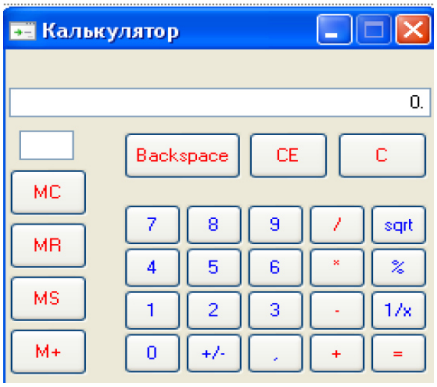



Рис. 3. Эскиз интерактивной формы (3-й уровень)

**Порядок выполнения работы (1-й уровень)**

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

2. Создать эскиз интерактивной формы.

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

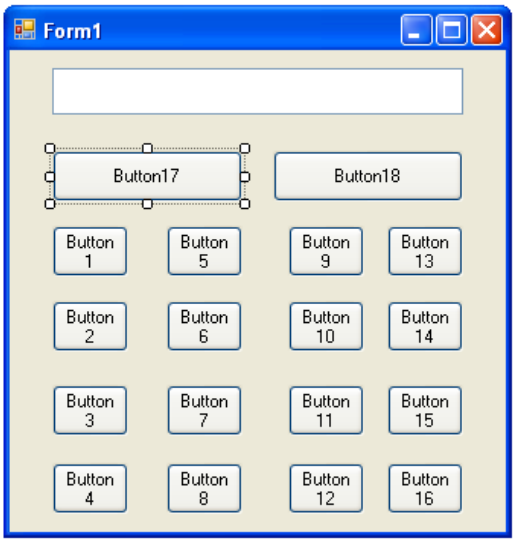



Рис. 4. Размещение элементов управления на форме

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

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

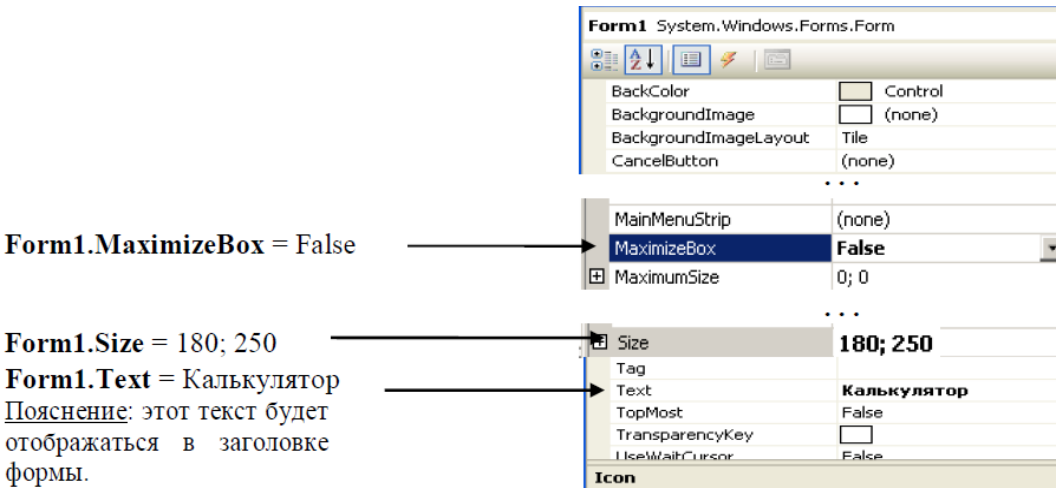



Рис. 5. Свойства Form1

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

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

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

Таблица 2

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

| | |
|---|---|
| Button8.ForeColor | Голубой |
| Button8.Text | +/- |
| Button9.Name | bt9 |
| Button9.ForeColor | Голубой |
| Button9.Text | 9 |
| Button10.Name | bt6 |
| Button10.ForeColor | Голубой |
| Button10.Text | 6 |
| Button11.Name | bt3 |
| Button11.ForeColor | Голубой |
| Button11.Text | 3 |
| Button12.Name | btpoint |
| Button12.ForeColor | Голубой |
| Button12.Text | , |
| Button13.Name | btdel |
| Button13.ForeColor | Красный |
| Button13.Text | / |
| Button14.Name | btpr |
| Button14.ForeColor | Красный |
| Button14.Text | * |
| Button15.Name | btmin |
| Button15.ForeColor | Красный |
| Button15.Text | - |
| Button16.Name | btplus |
| Button16.ForeColor | Красный |
| Button16.Text | + |
| Button17.Name | btis |
| Button17.ForeColor | Красный |
| Button17.Text | = |
| Button18.Name | btce |
| Button18.ForeColor | Красный |
| Button18.Text | CE |

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

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

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

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

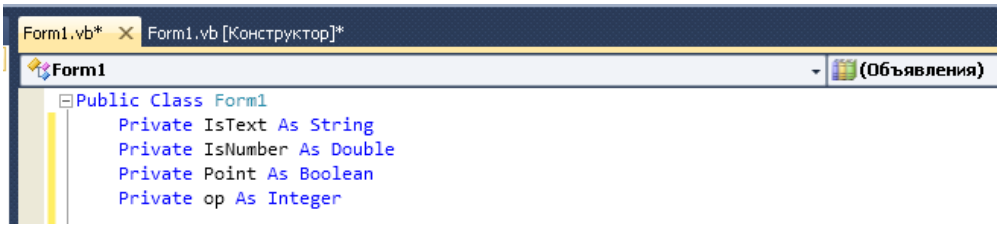



Рис. 6. Объявление переменных в блоке Form1 - Объявления

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1. Выполните сборку и компиляцию модулей проекта.
2. Запустите проект и протестируйте.

**Пояснения для выполнения задания 2-го уровня**

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

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

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

**Пояснения для выполнения задания 3-го уровня**

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

**Контрольные вопросы:**

1. Назначение функций Val() и CStr().
2. Поясните данный фрагмент кода:

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

3. Перечислите свойства кнопки, используемые для задания отображаемого текста и его цвета.
4. Поясните данный фрагмент кода:

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

5. Назначение конструкции **SelectCase**
6. Поясните данный фрагмент кода:

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

## Создание проекта с использованием компонентов для работы с текстом

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

**Задание 1-го уровня**

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

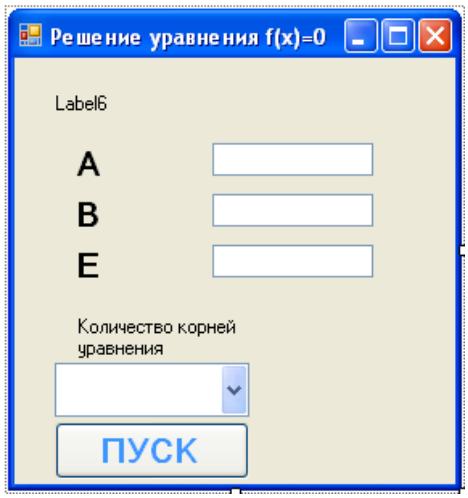



Рис. 1. Эскиз интерактивной формы

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

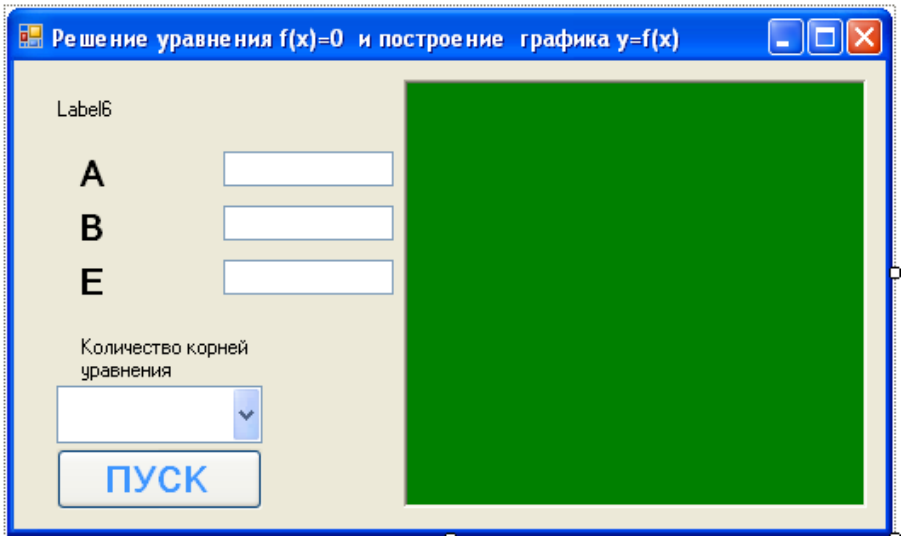



Рис. 2. Эскиз интерактивной формы с построением графика

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

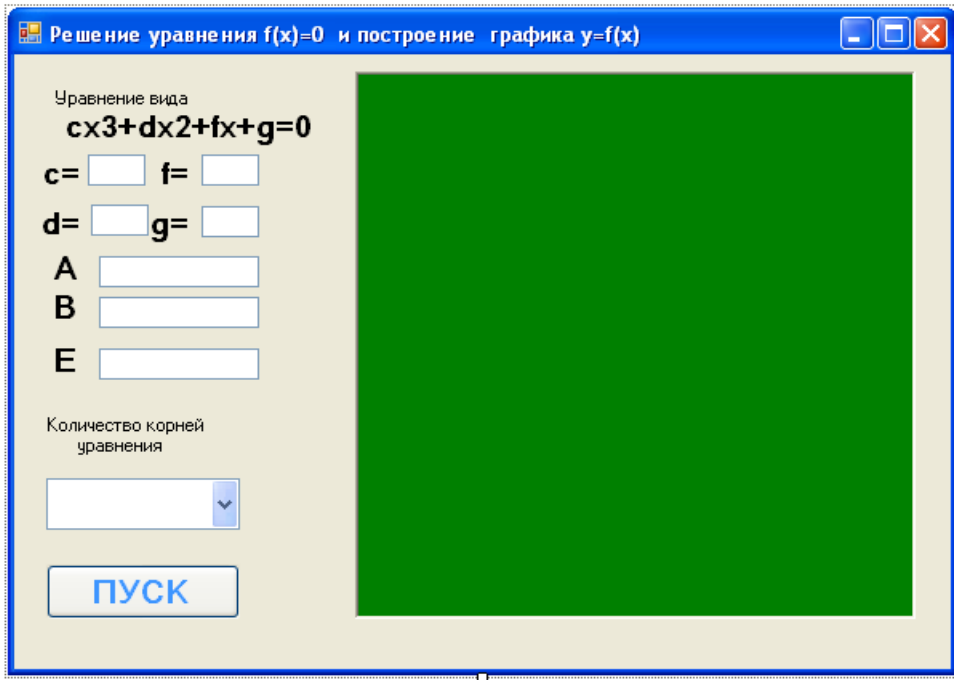



Рис. 3. Эскиз интерактивной формы с построением графика и пользовательским вводом функции

**Порядок выполнения работы (1-й уровень)**

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

2. Создать эскиз интерактивной формы.

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

Рис. 4. Размещение элементов управления на форме

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

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

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

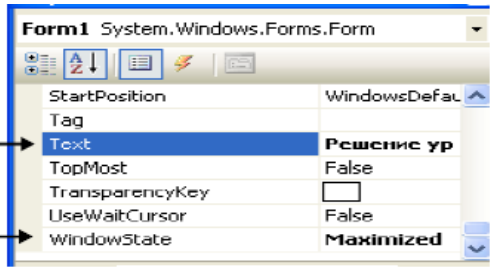
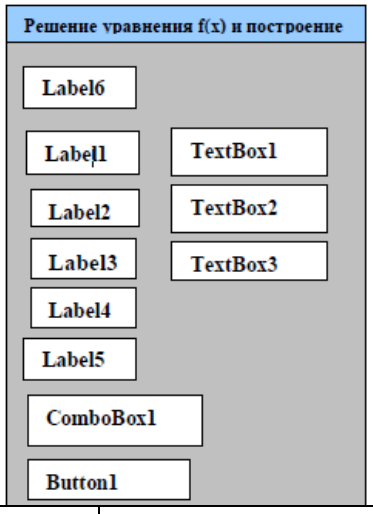



Рис. 5. Свойства Form1

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

Таблице 1




| Свойство | Значение |
|---|---|
| Label1.Text | A |
| Label2.Text | B |
| Label3.Text | E |
| Label1.Font | жирный, 16 пт. |
| Label2.Font | жирный, 16 пт. |

| | |
|---|---|
| Label3.Font | жирный, 16 пт. |
| Label4.Text | Количество корней |
| Label5.Text | Уравнения |
| Label6.Text | |

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

Таблица 2

| **Свойство** | **Значение** |
|---|---|
| TextBox1.Font | жирный, 16 пт. |
| TextBox2.Font | жирный, 16 пт. |
| TextBox3.Font | жирный, 16 пт. |

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

Таблице 3

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

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

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

Таблице 4

| **Свойство** | **Значение** |
|---|---|
| ComboBox1.DropDownStyle | DropDownList |
| ComboBox1.Font | жирный, 16 пт. |

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

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

4.1. Для объявления глобальных переменных выполните двойной щелчок левой кнопкой мыши на форме. В появившемся окне головного модуля *Form1.vb* выберете блок Объявление*(Declarations*), как показано на Рис. 6, и введите программный код, объявляющий переменные:

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

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

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

ведите программный код:

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

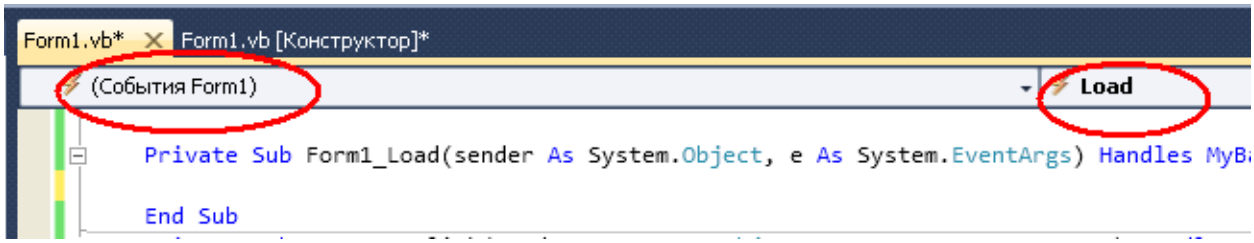



Рис. 7. Обработка события в блоке Form1 - Load

4.3. Написать программный код, обрабатывающий событие «нажатие кнопки пуск»(Button1_Click).

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

1. Сборка и компиляция модулей проекта выполняется командой *Построить решение (Build Windows Application)*из меню *Построение* (*Build)*. Запустить приложение на выполнение можно командой Начать отладку (*Start Debugging)*из меню *Отладка* (*Debug)*. В появившейся форме (Рис. 1) ввести с клавиатуры значения исходных данных: - **A** - начало интервала табулирования функции; - **B** - конец интервала табулирования функции; - **E** - шаг вычисления корней

уравнения. Для выполнения вычислений нажать кнопку ПУСК. Покажите преподавателю результаты работы.

**Пояснения для выполнения задания 2-го уровня**

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

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

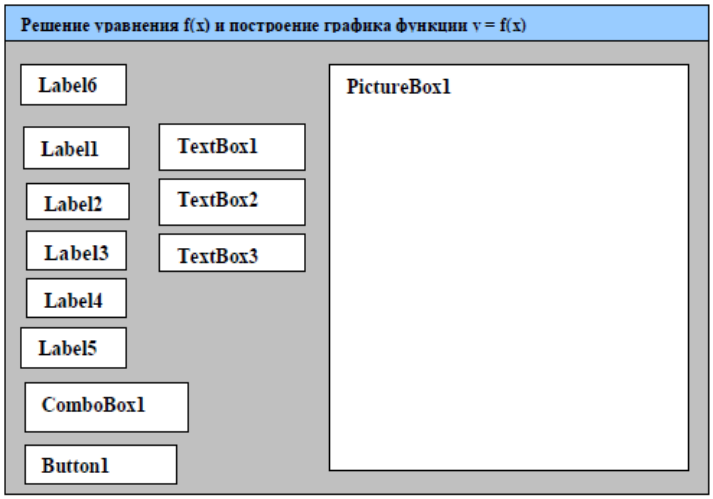



Рис. 8. Размещение элементов управления на форме

Таблица 5

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

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

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

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

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

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

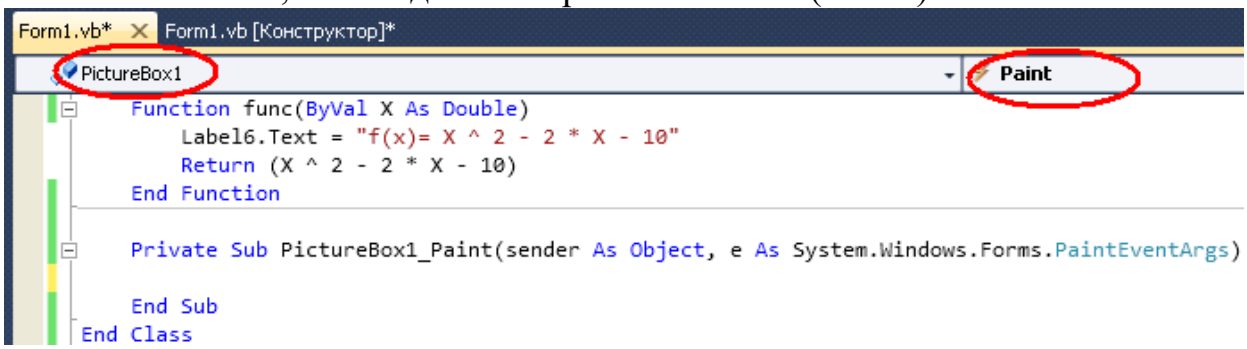



Рис. 9. Обработка события в блоке PictureBox1- Paint 35

И ввести код для подпрограммы **PictureBox1_Paint**:

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

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

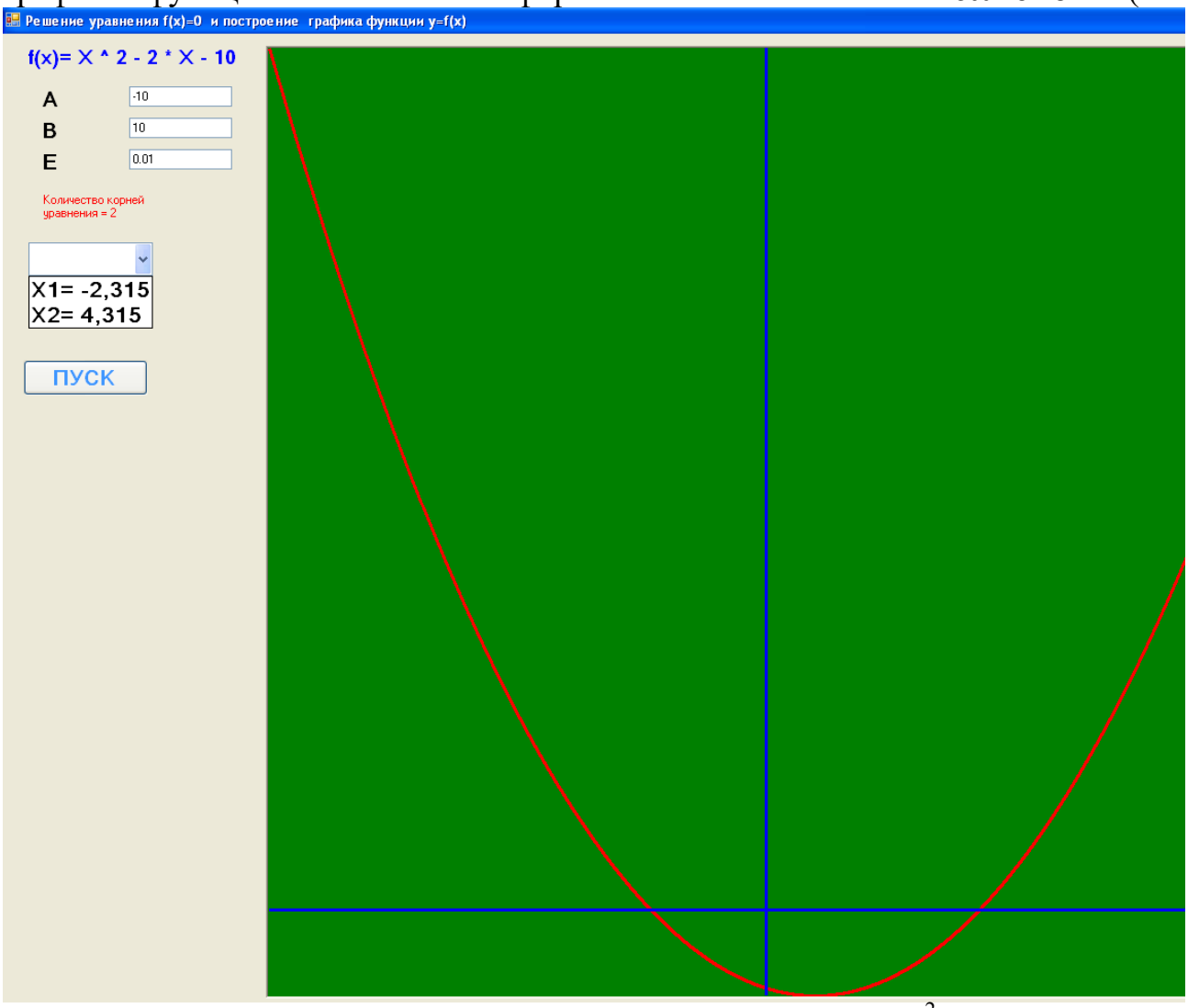



Рис. 10. Результат расчета функции $f(x) = x^2 - 2x - 10$

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

Таблица 6. Варианты заданий к лабораторной работе №6

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

**Контрольные вопросы:**

7. Перечислите элементы управления для работы с текстом.
8. Поясните данный фрагмент кода:

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

9. Перечислите свойства элементов управления, используемые для задания отображаемого текста и его цвета.
10. Укажите объявление глобальных переменных в коде программы.
11. Элемент управления для задания графических примитивов
12. Поясните данный фрагмент кода:

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

**Создание проекта с использованием кнопочных компонентов.**

**🕮Теоретическая часть**

## Отображение дат. Использование элементов управления MonthCalendar и DateTimePicker

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

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

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

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

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

**Константы для указания формата даты**

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

| | | |
|---|---|---|
| DateFormat.ShortTime | Отображает время в 24-часовом формате (чч:мм). | 12:00 |

**Свойства и функции системных часов**

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

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

**Порядок выполнения работы**

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

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

8. Вернитесь в режим конструктора и добавьте в форму элемент управления *DateTimePicker*, оставив имя по умолчанию *DateTimePicker1*.
9. Дважды щелкните элемент управления *DateTimePicker*, чтобы открыть обработчик событий по умолчанию в редакторе кода.

10. В обработчике событий DateTimePicker_ValueChanged добавьте следующий код для добавления элементов в список.

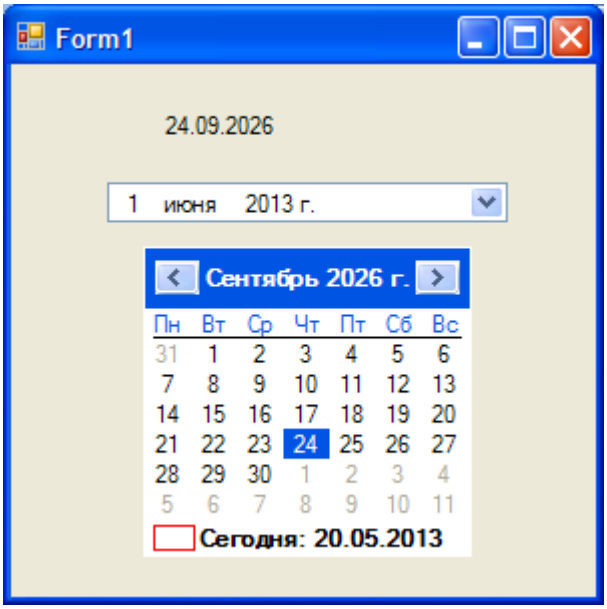
```
Me.Label1.Text = CStr(Me.DateTimePicker1.Value)
```

11. Нажмите клавишу F5 для запуска программы.

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

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

Дата и время отображаются в метке.




**Извлечение нескольких дат**

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

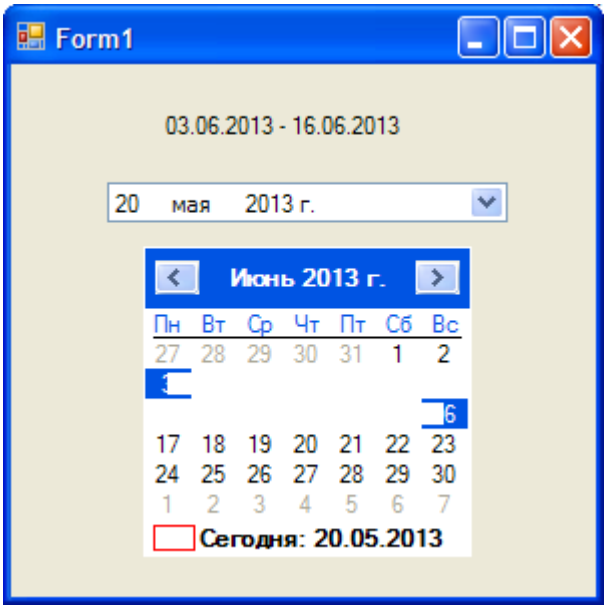
**Извлечение диапазона дат из элемента управления календарем месяца**

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

```
Me.MonthCalendar1.MaxSelectionCount = 14

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

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

Else

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

End If
```

2. Нажмите клавишу F5 для запуска программы.

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




**Форматирование дат**

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

**Форматирование даты в метке**

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

```
Me.MonthCalendar1.MaxSelectionCount = 14

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

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

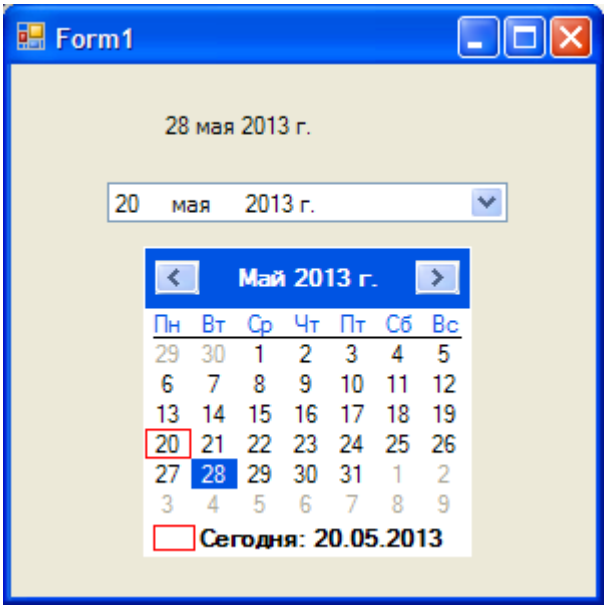
2. Замените код в обработчике событий DatePicker1_ValueChanged следующим. Этот код форматирует дату, возвращаемую в полном формате.

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

3. Нажмите клавишу F5 для запуска программы.

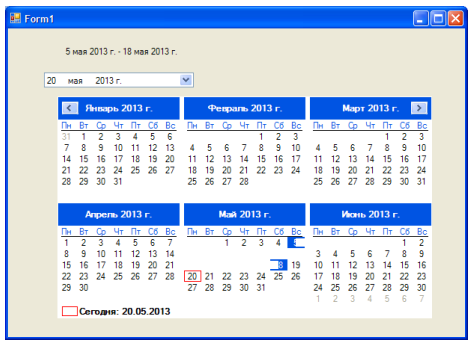
4. Когда появится форма, выберите дату или диапазон дат в элементе управления *MonthCalendar*. Убедитесь, что дата или диапазон дат отображается в метке в полном формате.

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




**Чтобы отобразить несколько месяцев**

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




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

**Программа Birthday**

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

1. В области элементов выберите элемент управления *Button*, и ниже объекта выбора даты и времени добавьте объект кнопки. Эта кнопка будет использована для показа дня рождения и для проверки правильности работы объекта выбора даты и времени.

**2.** В окне *Свойства(Properties)* измените свойство *Text* объекта кнопки на **Показать день моего рождения.**

3. Дважды щелкните мышью на объекте кнопки, а потом наберите следующий фрагмент программы между операторами Private Sub и End Sub в процедуре события Button1_Click:

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

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

Во второй строке `DateTimePicker1.Value.DayOfYear.ToString()` объект календаря используется для вычисления дня года, отсчитывая с 1 января. Это делается с

помощью свойства DayOfYear и метода ToString, который переводит числовой результат вычисления даты в текстовое значение, которое гораздо проще показать с помощью функции MsgBox.

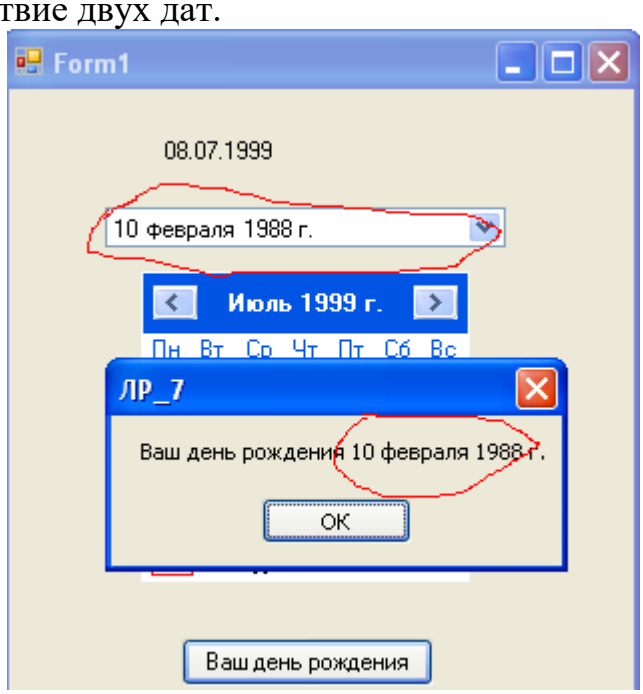
В третьей строке фрагмента, после перевода значения в строковое (или текстовое) представление, в окне сообщения показывается информация о точном времени.

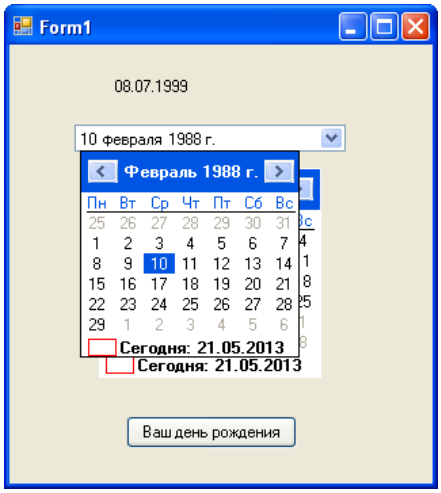
**Запуск программы Birthday**

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

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

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




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

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

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

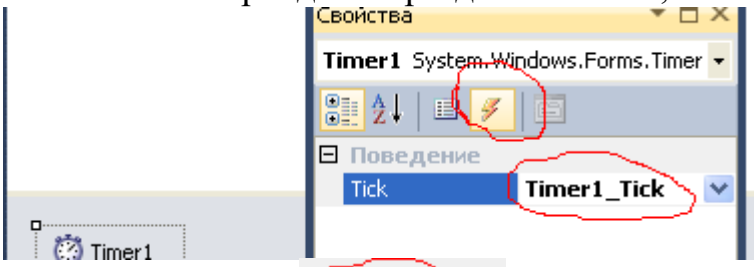
**Работа с датами в VB.Net. (Дополнительно)**

Реализовать программу которая будет узнавать текущую дату и время.

Чтобы узнать текущее время в VB.Net есть функция - **TimeString**, чтобы узнать дату - **DateString**. Кроме того есть функции работы со временем: **Hour()**(часы), **Minute()**(минуты), **Second()**(секунды). Эти функции вырезают часы,

минуты, секунды из указанно времени. Например, *Minute(TimeString)* - вырезает минуты из текущего времени. И функции для работы с датами: **Day()**(год), **Month()**(месяц), **Year**(год). Эти функции вырезают из текущей даты: день, месяц, год. Например, *Month(DateString)*, вырезает из текущей даты месяц.

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




7. Дважды щелкните в строке 

и введите в открывшемся окне код:

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

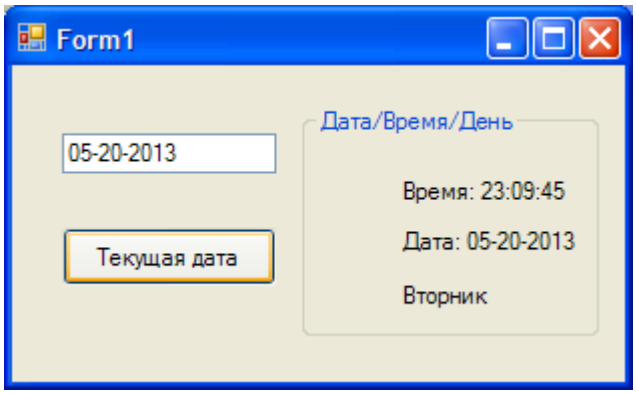
8. Для определения дня недели нужна функция ***WeekDay***. Напишите обработчик события загрузки формы:

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

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

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

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

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

11. Запустите проект.




**Контрольные вопросы:**

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

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

## Создание проекта с использованием компонентов стандартных диалогов и системы меню

**🕮Теоретическая часть**

### Добавление меню с помощью элемента управления MainMenu

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

**Соглашения о пунктах меню**

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

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

**Добавление клавиш доступа к командам меню**

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

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

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

**Использование элементов управления для диалоговых окон**

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

**Процедуры обработки событий, которые управляют общими диалоговыми окнами**

Чтобы показать в программе диалоговое окно, в процедуре обработки события для соответствующей команды меню нужно ввести оператор, состоящий из имени

диалогового окна и метода ShowDialog. Если это необходимо, то перед открытием диалогового окна нужно запрограммировать свойства этого окна. Наконец, в тексте программы должен присутствовать код, который после закрытия окна выполнит те или иные операции в зависимости от выбора или действий, которые пользователь выполнит в этом диалоговом окне.

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

**Управление выбором цвета с помощью установки свойств диалогового окна выбора цвета**

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

| **Свойство** | **Значение** |
|---|---|
| AllowFullOpen | Если присвоено значение True, в диалоговом окне будет работать кнопка DefineCustomColors (Определить цвет). |
| AnyColor | Если присвоено значение True, пользователь сможет выбрать любой цвет из показанных в диалоговом окне. |
| FullOpen | Если присвоено значение True, при открытии диалогового окна будет видна область CustomColors (Дополнительные цвета). |
| ShowHelp | Присвойте True, если нужно включить в диалоговом окне кнопку Help (Справка). |
| SolidColorOnly | Если присвоено значение True, пользователь сможет выбрать только "чистые" цвета (цвета со смешением будут отключены). |

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

**Порядок выполнения работы**

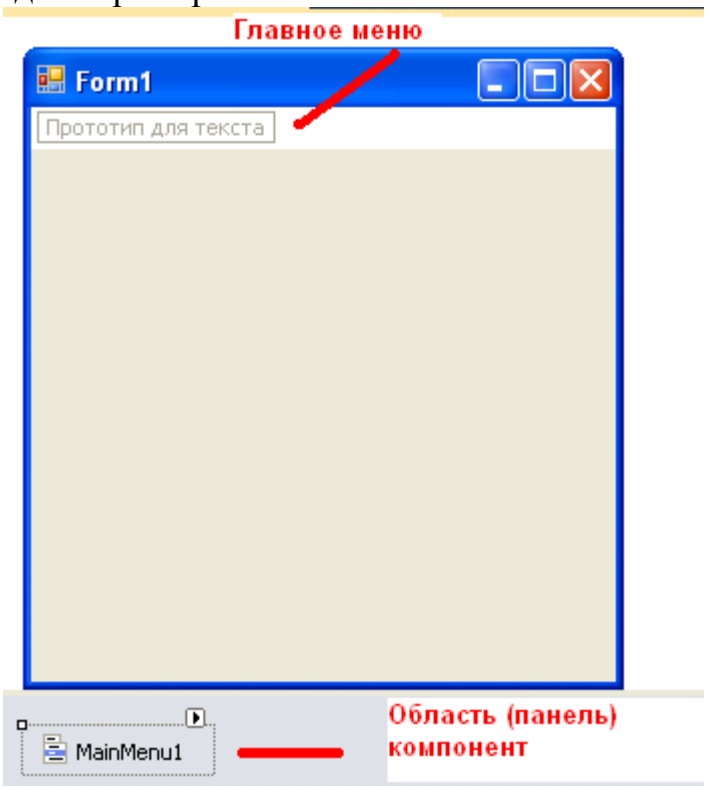
**Создание меню**

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

2. Выберите элемент *Приложение Windows Forms* и нажмите кнопку ОК.

3. Выберите элемент управления *MainMenu* на закладке WindowsForms окна области элементов и нарисуйте на поле формы элемент управления.

Форма будет выглядеть примерно так.




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

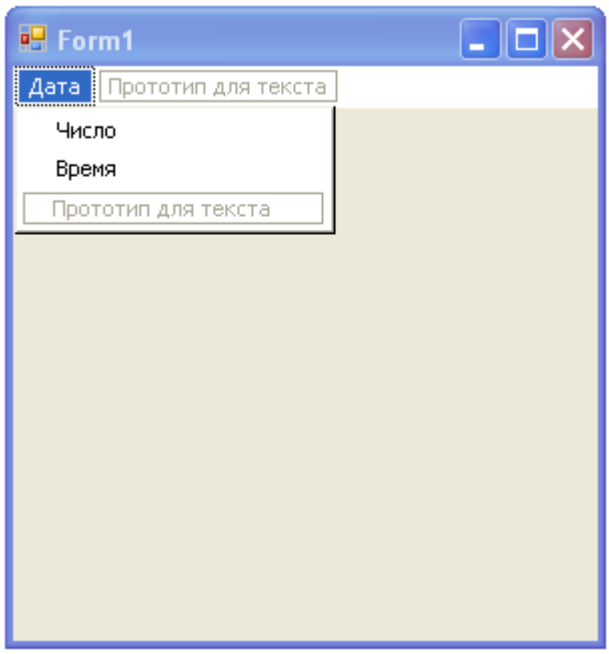
В дополнение к объекту меню, расположенному в области компонентов, создаваемое в Visual Studio.NET меню графически изображается в верхней части формы. Чтобы ввести название меню, нужно щелкнуть по полю Прототип для текста. После этого можно добавлять заголовки подменю и других меню, выбирая нужные поля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

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

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

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

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




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

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

**Добавление клавиш доступа**

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

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

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

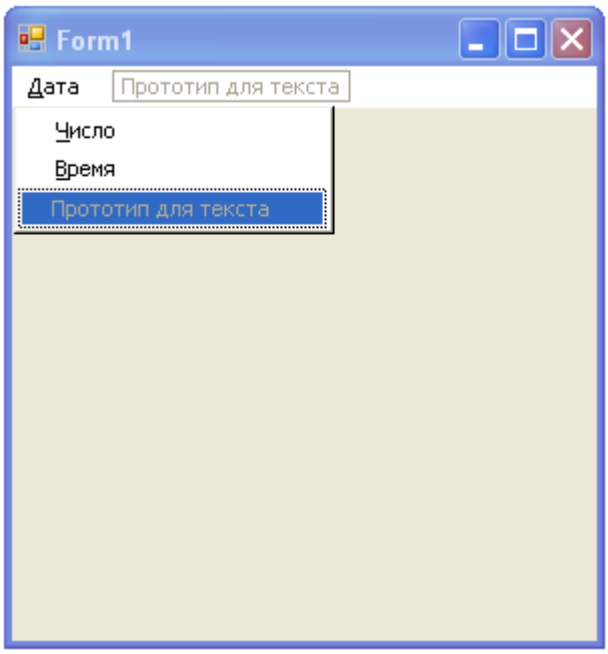
4. В списке меню выберите команду **Число**, а затем щелкните на ней еще раз, чтобы появился курсор редактирования текста.

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

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

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

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




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

**Изменение порядка элементов меню**

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

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

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

**Обработка выбора меню**

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

**Добавление на форму объекта надпись**

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

| Объект | Свойство | Значение |
|---|---|---|
| Label1 | BorderStyle | FixedSingle |

| | | |
|---|---|---|
| | Font | Microsoft Sans Serif, жирный , 14 пунктов |
| | Text | (empty) |
| | TextAlign | MiddleCenter |

**Редактирование процедур событий меню**

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

```
Label1.Text = TimeString
```

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

4. Нажмите на клавишу со стрелкой вниз. VisualBasic интерпретирует строку и, если потребуется, изменит заглавные буквы и добавит или удалит пробелы. VisualBasic проверяет каждую строку в процессе ее ввода и ищет в них синтаксические ошибки. Набор строки можно закончить, нажав клавишу (Enter), стрелку вверх или стрелку вниз.
5. В Обозревателе решений нажмите кнопку*Просмотреть конструктор*, а затем дважды щелкните мышью на команде **Число** в меню **Дата**. В редакторе кода откроется процедура обработки событий MenuItem2_Click. Эта процедура исполняется тогда, когда пользователь щелкает в меню **Дата** на команде **Число**.
6. Добавьте в программу следующий оператор

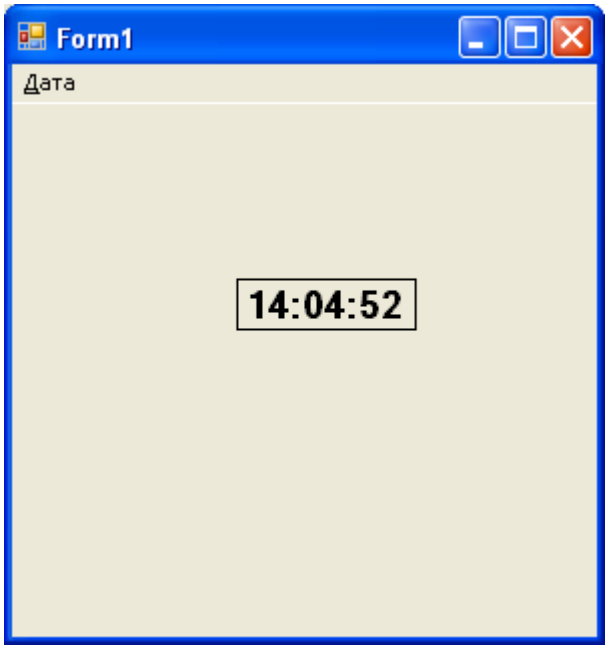
```
Label1.Text = DateString
```

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

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

**Запуск программы Menu**

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа Menu запустится в среде разработки.
2. В строке меню выберите пункт **Дата**. Появится меню **Дата**.
3. Выберите команду **Время**. В поле надписи появится текущее системное время, как показано ниже.




Теперь посмотрим, какое сегодня число.

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

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

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

**Добавление элементов управления OpenFileDialog и ColorDialog**

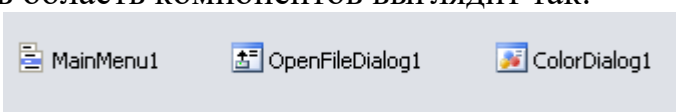
1. Добавьте в область компонентов, в которой уже находится объект главного меню *MainMenu1*, два элемента управления диалоговых окон. Элемент управления *OpenFileDialog* потребуется, чтобы открывать файлы с точечными изображениями, а элемент управления *ColorDialog* позволит изменять цвет для показа даты. В процессе разработки элементы управления диалоговых окон помещаются в области компонентов, а не на поле формы, так как во время выполнения они на форме не появляются.

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

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

В области компонентов появится объект диалогового окна для открытия файла.

3. В области элементов на закладке *WindowsForms* выберите элемент управления *ColorDialog*, а затем щелкните на области компонентов, расположенной ниже поля формы. Теперь область компонентов выглядит так.




Как и объект главного меню, объекты с диалогами открытия файла и выбора цвета можно настроить, задав их свойства.

Теперь с помощью элемента управления *PictureBox* создайте область для показа изображений. Этот объект показывает в поле формы содержимое из файлов изображений. На этот раз мы выведем на поле формы картинку, используя диалоговое окно для открытия файла.

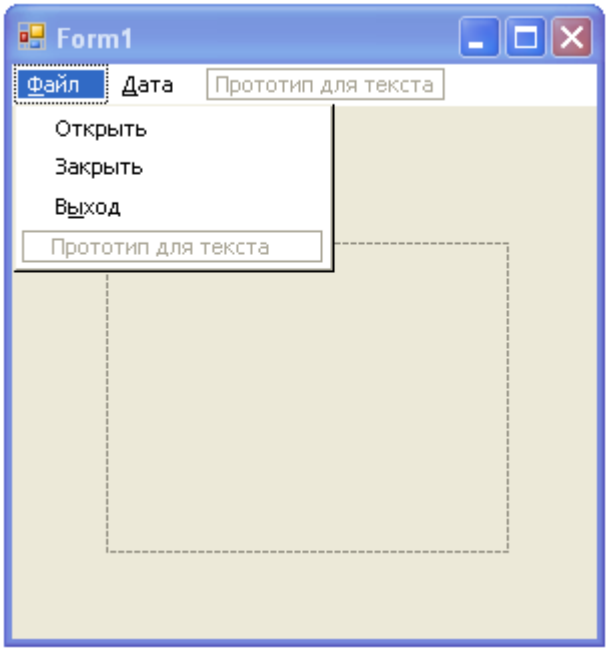
## Добавление объекта области показа изображения

1. В области элементов выберите элемент управления *PictureBox*.
2. В поле формы ниже надписи нарисуйте объект области показа изображений, и в окне Свойства для свойства *SizeMode* этого объекта выберите значение *StretchImage*.

Теперь с помощью *Конструктора меню* добавьте в вашу программу меню Файл.

**Добавление меню Файл**

1. В поле формы щелкните на меню **Дата**, затем на ячейке Прототип для текста, расположенной справа от этого меню. Теперь нужно добавить в программу меню **Файл**, в котором будут команды **Открыть**, **Закрыть** и **Выход**.
2. Чтобы создать меню **Файл** с буквой **"Ф"** в качестве клавиши доступа, введите **&Файл**.
3. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду Открыть ...с буквой **"О"** в качестве клавиши доступа, введите **&Открыть** .... Команда Открыть будет использоваться для загрузки точечных изображений. Так как эта команда должна будет открывать диалоговое окно, добавьте к ее имени многоточие.
4. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Закрыть** с буквой **"З"** в качестве клавиши доступа, введите **&Закрыть**. Команда **Закрыть** будет использоваться в программе для закрытия файла с изображением.
5. Нажмите клавишу со стрелкой вниз, а затем, чтобы создать команду **Выход** с буквой **"ы"** в качестве клавиши доступа, введите **В&ыход**. Команда **Выход** будет использоваться для закрытия программы. Обратите внимание, что в этом случае в качестве клавиши доступа для команды Выход была использована третья буква, как это сделано во многих приложениях для Windows.
6. Чтобы передвинуть меню **Файл** на первое место, просто перетащите его на меню **Дата**. В Конструкторе меню целые меню можно перемещать точно так же, как и отдельные команды внутри меню. Имеет смысл сделать меню **Файл** первым меню программы. Ваша форма должна выглядеть примерно так.




В следующем упражнении мы отключим команду **Закрыть** в меню **Файл**. (Команда **Закрыть** может использоваться только после того, как файл уже был открыт в программе). Далее в этой лекции мы добавим в процедуру обработки

события команды **Открыть** оператор, который в нужный момент включает команду **Закрыть**.

**Отключение команды Закрыть**

1. В меню **Файл** программы *Menu* выберите команду **Закрыть**.
2. В окне Свойства для свойства *Enabled* объекта *MenuItem7* выберите значение *False*.

Теперь, чтобы продемонстрировать, как работает диалоговое окно выбора цвета, необходимо добавить в меню **Дата** команду **Цвет текста**. Диалоговое окно выбора цвета с помощью свойства *Color* возвращает в программу вновь выбранный цвет. Это свойство будет использоваться для изменения цвета текста в объекте *Label1*.

**Добавление команды Цвет текста в меню Дата**

1. Выберите меню **Дата**, а затем щелкните на нижней ячейке Прототип для текста.
2. Чтобы добавить в это меню команду **Цвет текста** с клавишей "Ц" в качестве клавиши доступа, введите **&Цвет текста ...**. Команда будет добавлена в меню **Дата**. Эта команда заканчивается многоточием, которое указывает, что при ее выборе откроется диалоговое окно.

**Редактирование процедуры обработки событий команды Открыть**

1. В меню **Файл** нашей формы дважды щелкните мышью на команде **Открыть**. В редакторе кода появится процедура обработки событий mnuOpenItem_Click.
2. Между операторами Private Sub и End Sub добавьте следующие ниже операторы. Убедитесь, что вы набрали каждую строку в точности так, как они здесь напечатаны, а после набора последней строки нажмите на клавиатуре стрелку вниз.

```
OpenFileDialog1.Filter = "Точечный рисунок (*.bmp)|*.bmp"
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
    PictureBox1.Image = System.Drawing.Image.FromFile(OpenFileDialog1.FileName)
    MenuItem6.Enabled = True
End If
```

Первые три оператора в этой процедуре обработки события ссылаются на три свойства объекта диалога открытия файла. В первом операторе свойство Filter используется для определения списка допустимых файлов. В нашем случае этот список содержит только один элемент: *.bmp. Это достаточно важно для диалогового окна Открыть, так как объект показа изображения поддерживает шесть типов файлов: точечные изображения (файлы .bmp), метафайлы Windows (файлы .emf и .wmf), значки (файлы .ico), формат JointPhotographicExpertsGroup (файлы .jpg и .jpeg), формат PortableNetworkGraphics (файлы .png) и формат GraphicsInterchangeFormat (файлы .gif). Попытка показать в объекте изображения файл .txt приведет к возникновению ошибки в момент выполнения. Чтобы добавить в список Filter дополнительные элементы, между ними нужно ввести символ (|). Например, при следующем значении фильтра

*OpenFileDialog1.Filter = "Точечный рисунок (*.bmp)|*.bmp|Метафайл Windows (*.wmf)|*.wmf"*

в диалоговом окне **Открыть** будут показаны как файлы с точечными изображениями, так и метафайлы Windows.

Второй оператор в процедуре обработки события показывает в программе диалоговое окно **Открыть**. Метод *ShowDialog* - это новый метод Visual Basic .NET, он похож на метод Show из VisualBasic 6, но может использоваться для любой формы *WindowsForms*. Метод *ShowDialog* возвращает результат с именем *DialogResult*, который указывает, какую кнопку диалогового окна нажал пользователь. Чтобы

определить, щелкнул ли пользователь на кнопке **Открыть**, используется оператор If...Then, который проверяет, равно ли возвращенное значение *DialogResult.OK*. Если оно равно этому значению, то в свойстве *FileName* диалога открытия должен храниться путь к существующему файлу на диске.

В третьем операторе используется имя файла, которое было выбрано в диалоговом окне. Когда пользователь выбирает диск, папку и имя файла, а затем нажимает кнопку **Открыть**, полный путь передается в программу через свойство *OpenFileDialog1.FileName*. Затем используется метод *System.Drawing.Image.FromFile*, который копирует указанное точечное изображение в объект показа изображений. (Этот оператор разбит на две строки при помощи символа продолжения строки, так как он получился очень длинным.)

В четвертом операторе активируется команда **Закрыть** из меню **Файл**. Теперь, когда файл в программе был открыт, команда **Закрыть** должна быть доступна, чтобы пользователи могли закрыть этот файл.

Теперь введите код программы для процедуры обработки события *mnuCloseItem_Click*, которая выполняется при выборе команды **Закрыть** из меню **Файл**.

**Редактирование процедуры события команды Закрыть**

1. Снова откройте форму, а затем дважды щелкните мышью на команде **Закрыть** из меню **Файл**. В редакторе кода появится процедура обработки события для команды **Закрыть**.
2. Между операторами PrivateSub и EndSub добавьте следующие операторы программы:

```
PictureBox1.Image = Nothing
MenuItem6.Enabled = False
```

Первый оператор закрывает открытое точечное изображение, удаляя информацию, хранящуюся в свойстве *Image*. Ключевое слово *Nothing* используется здесь для того, чтобы отменить связь между текущим объектом точечного изображения и свойством *Image*, другими словами, *Nothing* задает для этого свойства нулевое значение, и изображение исчезает. (Далее в этой книге *Nothing* будет использоваться для сброса значений и других объектов и свойств.) Во втором операторе команду **Закрыть** из меню **Файл** отключается, так как открытых файлов больше нет. Использование этого оператора в программе равнозначно настройке свойства *Enabled* в окне *Properties (Свойства)*.

**Редактирование процедуры событий команды Выход**

1. Снова перейдите в конструктор формы, а затем дважды щелкните мышью на команде **Выход** из меню **Файл**. В редакторе кода появится процедура обработки событий для команды **Выход**.
2. Между операторами Private Sub и End Sub добавьте следующий оператор программы

```
End
```

Оператор End останавливает программу, когда пользователь заканчивает с ней работать.

**Редактирование процедуры обработки событий команды Цвет текста**

1. Перейдите в конструктор формы, а затем дважды щелкните мышью на новой команде **Цвет текста** из меню **Дата**. В редакторе кода появится процедура обработки событий для команды **Цвет текста**.
2. Добавьте в нее следующие операторы

```
ColorDialog1.ShowDialog()
Label1.ForeColor = ColorDialog1.Color
```

**Совет**. Диалоговое окно Color (Цвет) может использоваться для установки цвета любого элемента пользовательского интерфейса, который работает с цветом. Например, это может быть фоновый цвет формы, цвета геометрических фигур на форме, основной или фоновые цвет для различных объектов.

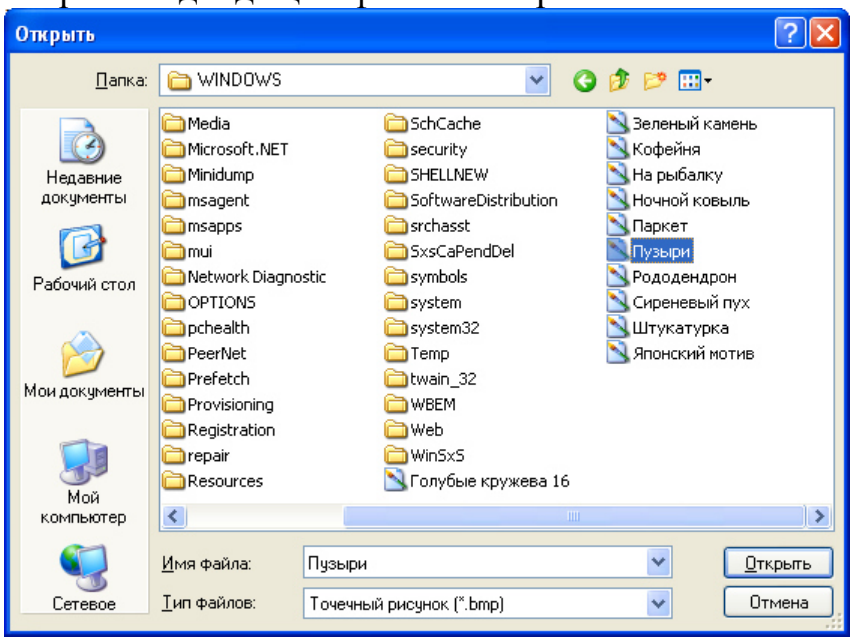
В первом операторе используется метод *ShowDialog*, который открывает диалоговое окно выбора цвета. Метод *ShowDialog* используется для открытия любой формы как диалогового окна, в том числе и формы, созданные стандартными диалоговыми окнами, которые предоставляет Visual Studio. Во втором операторе цвет, выбранный пользователем в диалоговом окне, присваивается свойству *ForeColor* объекта *Label1*. *Label1* - это поле надписи, которое используется в форме для показа текущих даты и времени. При работе программы вы будете выбирать в диалоговом окне цвет, и он будет использован для текста в этой надписи.

Теперь давайте запустим программу Menu и поэкспериментируем с созданными нами меню и диалоговыми окнами.

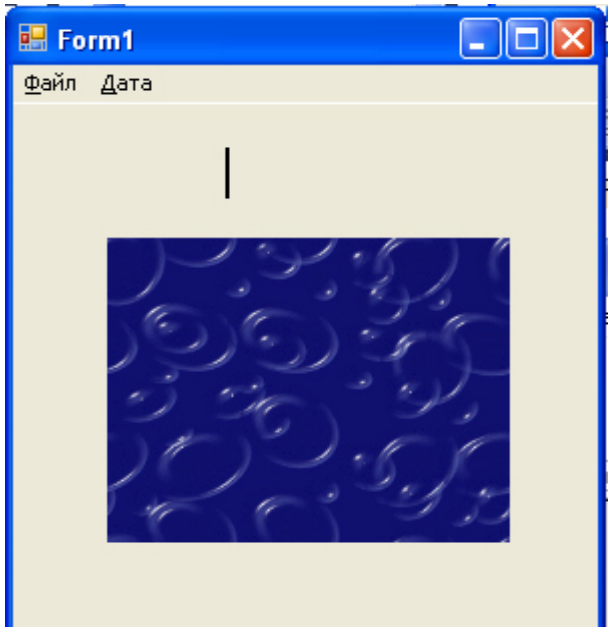
**Запуск программы Menu**

1. На стандартной панели инструментов нажмите кнопку Start (Начать). Программа запустится, и в строке меню появятся меню **Файл** и **Дата**.

2. В меню **Файл** запущенной программы выберите **Открыть**. Появится диалоговое окно **Открыть**. Обратите внимание на текст Точечный рисунок (*.bmp) в поле Тип файлов. Выберите подходящий файл с изображением.

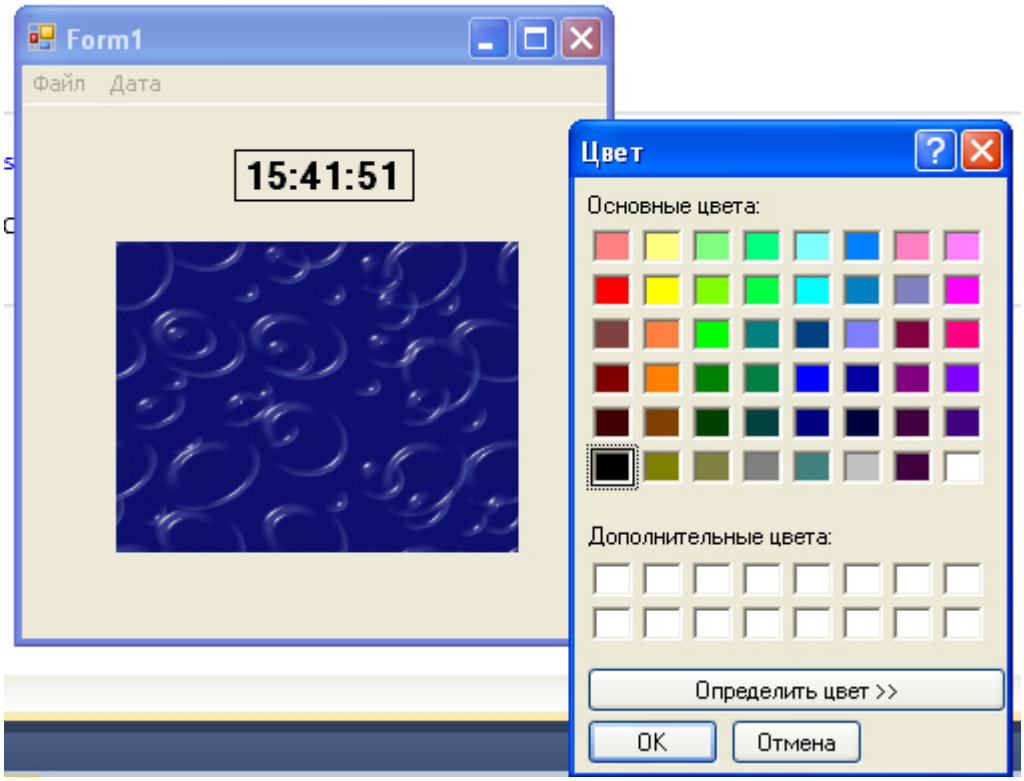



3. Выберите один из файлов *.bmp, а затем нажмите кнопку **Открыть**. Изображение из этого файла появится в поле показа изображений. Форма будет выглядеть примерно так.




4. В меню **Дата** выберите строку **Время**. В поле надписи появится текущее время.

5. В меню **Дата** выберите команду **Цвет текста**. Появится диалоговое окно Color (Цвет), показанное ниже.




Диалоговое окно Color (Цвет) содержит элементы, которые позволяют изменить цвет надписи в программе. По умолчанию в этом окне выбран текущий цвет - черный.

6. Щелкните на синем поле, а затем на кнопке **OK**.

Диалоговое окно Color (Цвет) закроется, а цвет текста в надписи изменится на синий.


7. В меню **Дата** щелкните на команде **День**.
8. Откройте меню **Файл**.

Обратите внимание, что команда **Закрыть** включена. (Мы включили ее с помощью свойства *Enabled* = True.)

9. Нажмите **З** (клавишу доступа для **Закрыть**, при необходимости переключите клавиатуру в русский регистр), чтобы закрыть изображение.

Файл закроется, и точечное изображение Windows исчезнет (это сработало ключевое слово *Nothing*.)

10. Откройте меню **Файл**. Теперь команда **Закрыть**отключена, так как в области вывода изображений картинки нет.
11. Выберите команду **Выход**. Программа закроется, и появится среда разработки Visual Studio.

**Следующий шаг: привязка сочетаний клавиш к пунктам меню**

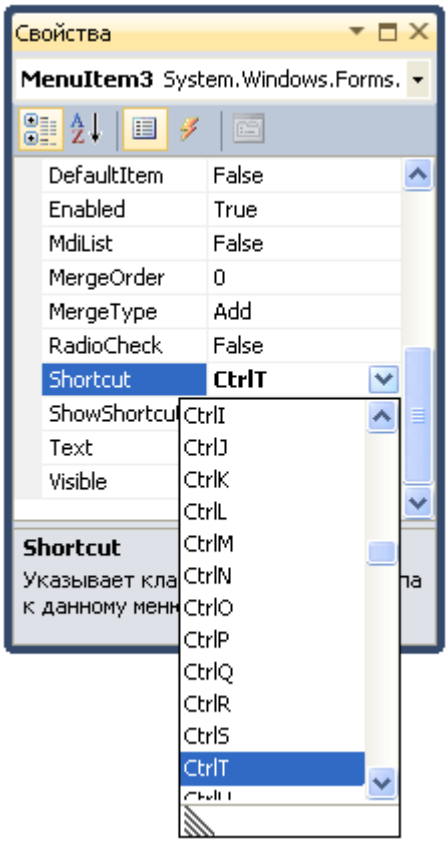
**Определение сочетания клавиш для меню Дата**

1. В меню **Дата** выберите команду **Время**.

Сочетание клавиш определяется в свойстве *Shortcut* выбранной команды, в окне Свойства.

2. Откройте окно Свойства, выберите свойство *Shortcut*, затем нажмите стрелку раскрывающегося списка в столбце справа, прокрутите этот список и выберите CtrlT. Окно Свойства будет выглядеть так, как показано на рисунке справа.

**Совет**. Обычно Visual Basic при выполнении программы показывает в меню сочетания клавиш, чтобы подсказать пользователям, какие клавиши следует нажимать. Чтобы скрыть от пользователя эти комбинации клавиш (например, если для них не хватает места), задайте для свойства ShowShortcut значение False. Сочетания клавиш по-прежнему будут работать, но подсказок в меню пользователи не увидят.

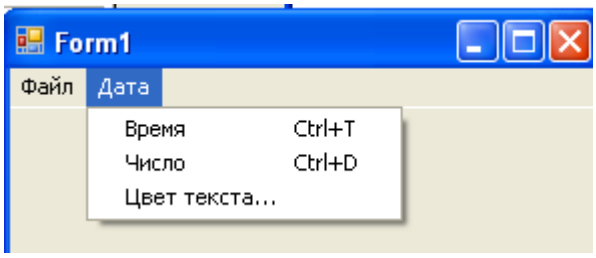



3. Выберите команду **Число**, а затем измените ее свойство *Shortcut* на значение CtrlD. Теперь давайте запустим программу и попробуем использовать сочетания клавиш.

4. На стандартной панели инструментов нажмите кнопку Начать.

5. Нажмите (Ctrl)+(T), чтобы выполнить команду **Время**. В программе появится текущее время.

6. Нажмите (Ctrl)+(D), чтобы выполнить команду **Число**. В программе появится текущая дата.




7. Щелкните на меню **Дата**. Рядом с командами **Время** и **Число** будут показаны сочетания клавиш. Visual Basic дописывает эти комбинации клавиш, когда они определены с помощью свойства *Shortcut*.

**Контрольные вопросы:**

1. Назовите элемент управления для добавления меню.
2. Рекомендации по созданию меню.
3. Добавление клавиш доступа. Как визуально отличаются пункты меню, для которых назначена клавиша доступа.
4. Назначение сочетаний клавиш для пунктов меню, свойство.
5. Перечислить элементы управления для создания диалоговых окон.
6. Метод, используемый для отображения диалогового окна.

# *Практическая работа №10. " Разработка функциональной схемы работы приложения. Разработка оконного приложения с несколькими формами. Разработка игрового приложения."*

**Цели:** получение навыков работы со средой разработки Visual Studio, создание приложений Windows Forms на языке C++ в Visual Studio.

***Задание № 1.*** Написать программу, реализующую игру в кости.

*Правила игры*

1. Играющий называет любое число в диапазоне от 2 до 12 и ставку, которую он делает в этот ход.

2. Программа с помощью генератора случайных чисел дважды выбирает числа от 1 до 6 (бросает кубик, на гранях которого цифры от 1 до 6).

3. Если сумма выпавших очков:

а) меньше либо равна 7 и играющий назвал число меньше либо равное 7, он выигрывает ставку;

б) больше 7 и играющий сделал ставку на число больше 7, он также выигрывает ставку;

в) равна названному игроком числу (игрок угадал сумму цифр), он получает в 4 раза больше очков, чем сделанная ставка;

г) в противном случае ставка проиграна (если ни одна из ситуаций а-в не имеет места).

В начальный момент у игрока и компьютера по 100 очков. Игра идет до тех пор, пока у кого-либо из играющих останется 0 очков.

## *Практическая работа №11. Создание процедур обработки событий. Компиляция и запуск приложения. Разработка интерфейса приложения. Тестирование, отладка приложения.*

**Цели:** получение навыков работы со средой разработки Visual Studio, создание приложений Windows Forms на языке C++ в Visual Studio.

**Теоретические вопросы**

Процедуры обработки событий.

Этапы разработки Windows-приложений.

***Задание № 1.*** Написать программу, которая вычисляет силу тока в электрической цепи. Программа должна быть спроектирована таким образом, чтобы кнопка Вычислить была доступна только в том случае, если пользователь ввел величину сопротивления.


***Задание № 2.*** Написать программу, которая вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно.


**Задание № 3.** Дана запись с именем STUDENT, содержащая следующие поля:

1) фамилия и инициалы;

2) номер группы;

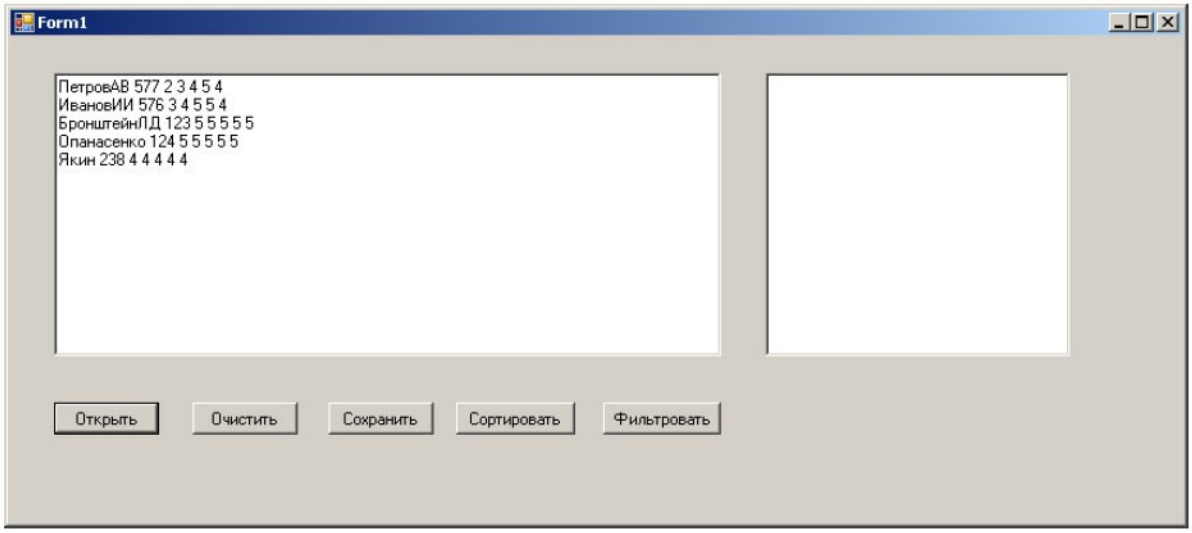
3) успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- чтение данных из файла и вывод их на экран;

вывод на экран фамилий, номеров групп и оценок для всех студентов, которые являютсякруглы• ми отличниками (если таких нет – вывести об этом сообщение);

- список должен быть упорядочен по возрастанию номера группы.




## *Практическая работа № 12-13. " Классы ООП: виды, назначение, свойства, методы, события. Объявления класса. Создание наследованного класса. Перегрузка методов "*

**Цели:** получение навыков работы со средой разработки Visual Studio, создание приложений сиспользованием классов на языке C++ в Visual Studio.

**Теоретические вопросы**

Наследование классов.

Ограничение доступа для родственных
классов.Конструкторы и деструкторы.

***Задание №1.*** Имеется базовый класс «Выпуклый четырехугольник» и производные классы,им порожденные: параллелограмм, ромб, квадрат. Описать для указанных фигур методы

«Вычисление углов», «Вычисление диагоналей», «Вычисление длин сторон», «Вычислениепериметра», «Вычисление площади».

```
#include
<iostream.h>
#include <math.h>

//Объявление базового класса четырехугольников
class FourAngle
{
protected:
```

```
        double xl,yl,x2,y2,x3,y3,x4,y4,
                        A,B,C,D,D1,D2,
                        Alpha,Beta,Gamma,
                        Delta,P,S;
public:
        void
        Init(void);
        void
        Storony(voi
        d);
        void
        Diagonal(void
        );void
        Angles(void);
        void
        Perimetr(void
        );void
        Ploshad(void)
        ;
        void PrintElements(void);
};
//Объявление класса параллелограммов – наследника четырехугольников
class Parall:public FourAngle
{
public:
        void
        Storony(void
        );      void
        Perimetr(voi
        d);     void
        Ploshad
        (void);


};
//Объявление класса ромбов – наследника параллелограммов
      class  Romb:public Parall
      {
```

```
        public:
            void
            Storony(void);
            void
            Perimetr(void);
        };
        //Объявление класса квадратов – наследника ромбов
class Kvadrat:public Romb
{
public:
            void
            Angles(void
            ); void
            Ploshad(voi
            d);
};
//Описания функций – членов класса FourAngle четырехугольников
void FourAngle::Init(void)
{
            cout<<"\nВведите координаты вершин:\n";
            cin>>xl>>yl>>x2>>y2>>x3>>y3>>x4>>y4;
        }
        void FourAngle::Storony(void)
        {
            A=sqrt((x2-xl)*(x2-xl)+(y2-yl)*(y2-yl));
            B=sqrt((x3-x2)*(x3-x2)+ (y3-y2)*(y3-y2));
            C=sqrt((x4-x3)*(x4-x3)+(y4-y3)*(y4-y3));
            D=sqrt((x4-xl)*(x4-xl)+(y4-yl)*(y4-yl));
        }
        void FourAngle::Diagonal(void)
        {
            D1=sqrt((xl-x3)*(xl-x3) + (yl-y3)*(yl-y3));
            D2=sqrt((x2-x4)*(x2-x4)+(y2-y4)*(y2-y4));
        }
        //Функция Ugol не является членом какого-либо класса.
        //Она выполняет вспомогательную роль для функции Angles.
        //Эта функция может независимым образом использоваться и в основной программе
```

```
        //для определения углов треугольника, заданного длинами сторон
        double Ugol(double Aa, double Bb, double Cc)
        {
            double VspCos,
            VspSin, Pi;
            Pi=4*atan(1.0);
            VspCos=(Aa*Aa+Bb*Bb-
            Cc*Cc)/2/Aa/Bb;VspSin=sqrt(1-
            VspCos*VspCos);
            if(abs(VspCos)>1.E-7)
            return
            (atan(VspSin/VspCos)+Pi*(VspCos<0))/Pi/180;
            else return 90.0;
}
void FourAngle::Angles(void)
{
            Alpha=Ugol(D,A,D2);
            Beta=Ugol(A,B,D1);
            Gamma=Ugol(B,C,D2)
            ;Delta=Ugol(C,D,D1);
}
void FourAngle::Perimetr(void)
{
           P=A+B+C+D;
}
void FourAngle::Ploshad (void)
{
            double
            Perl,Per2;
            Perl=(A+D+
            D2)/2;
            Per2=(B+C
            +D1)/2;
            S=sqrt(Perl*(Perl-A)*(Perl-D)*(Perl-
                    D2))+ sqrt(Per2*(Per2-B)*(Per2-
                    C)*(Per2-D1));
}
void FourAngle::PrintElements(void)
```

```
{
    cout<<"Стороны: \n"<<A<<"  "<<B<<"  "<<C<<"  "<<D<<"\n";
    cout<<"Углы:\n"<<Alpha<<"  "<<Beta<<"  "<<Gamma<<"  "<<Delta<<"\n";
    cout<<"Периметр:\n"<<P<<"\n";
    cout<<"Площадь:\n"<<S<<"\n";
    cout<<"Диагонали:\n"<<D1<<"  "<<D2<<"\n";
}
//Описания функций – членов класса Storony параллелограммов
void Parall::Storony (void)
{
    A=sqrt((x2-xl)*(x2-xl)+(y2-yl)*(y2-yl));
    B=sqrt((x3-x2)*(x3-x2)+(y3-
    y2)*(y3-y2));C=A; D=B;
}
void Parall::Perimetr(void)
{
    P=2*(A+B);
}
void Parall::Ploshad(void)
{
    double Per;

    Per=(A+D+D2)/2;
    S=2*sqrt(Per*(Per-A)*(Per-D)*(Per-D2));
}
//Описания функций – членов класса Romb ромбов
void Romb::Storony(void)
{
    A=B=C=D=sqrt((x2-xl)*(x2-xl)+(y2-yl)*(y2-yl));
}
void Romb::Perimetr(void)
{
    P=4*A;
}
//Описания функций – членов класса Kvadrat квадратов
void Kvadrat::Angles (void)
{
```

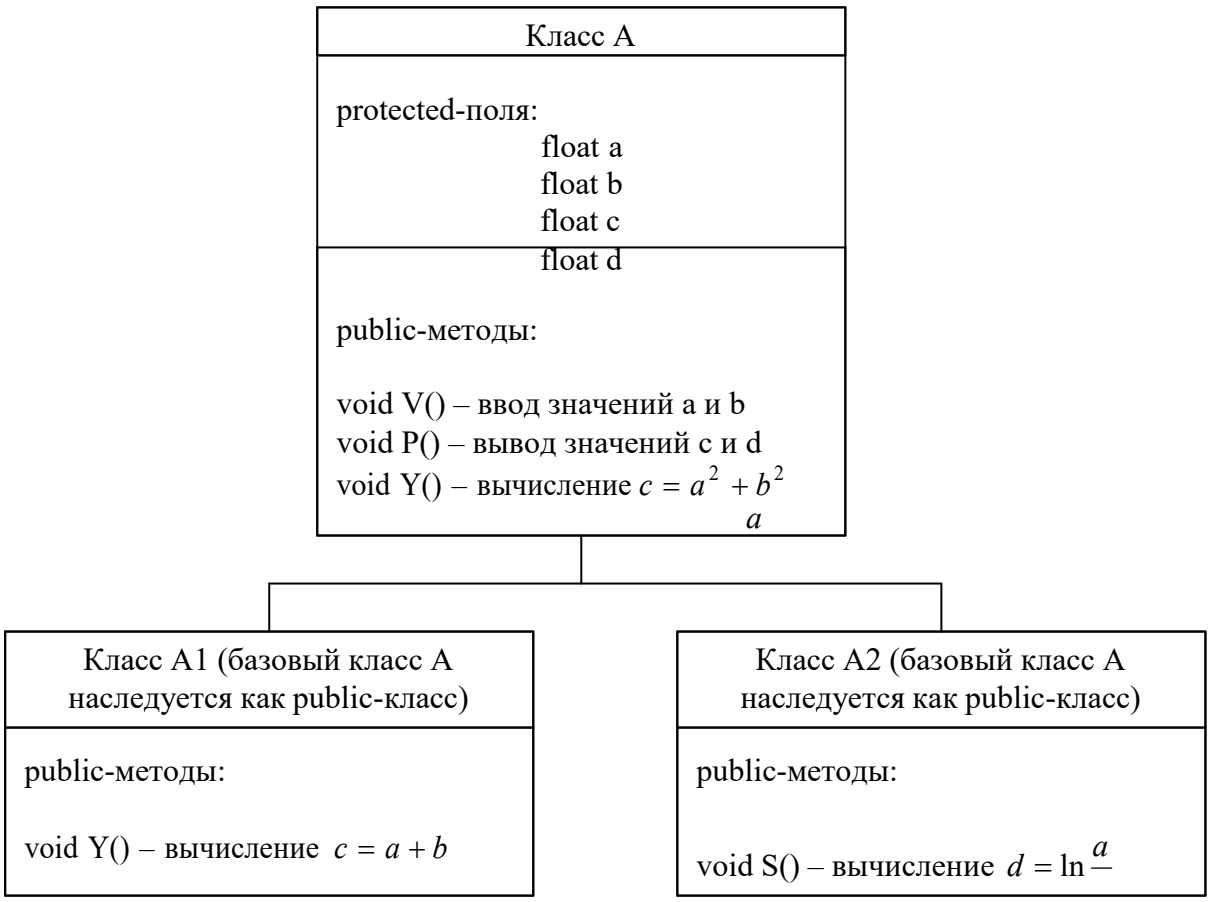
```
    Alpha=Beta=Gamma=Delta=90.0;
  }
  void Kvadrat::Ploshad(void)
  {
    S=A*A;
  }
  //Основная функция. По координатам вершин квадрата
  //вычисляет и выводит все его параметры
void main (void)
{
    Kvadrat  obj; //Объявление объекта класса "квадрат"
    obj.Init();
    obj.Storony()
    ;
    obj.Diagonal
    ();
    obj.Angles();
    obj.Perimetr(
    );
    obj.Ploshad()
    ;
    obj.PrintEle
    ments();
}
```

***Задание № 2.*** Объявить класс для строковых объектов. Конструктор с помощью операторanew резервирует блок памяти для указателя stringl. Освобождение занятой памяти выполняетдеструктор с помощью оператора delete.

```
#include
<iostream.h>
#include <string.h>
class string_operation
{
    char
    *strin
    gl;
    public
    :
    string_operation (int string_len)
```

```
    /конструктор

    {
        stringl=new char[string_len];}
    ~string_operation()                                    //деструктор
    {delete stringl;}
    void
    input_data(char*)
    ; void
    output_data(char
    *);
};
void string_operation::input_data(char *s)
{
    strcpy(stringl, s);
}

void string_operation::output_data(char *s)
{
    strcpy(s, stringl);
}



void main()
{
    char s1[10];

    string_operation a(10);
    a.input_data("Строка");
    a.output_data(s1);
    cout<<"Введена строка:
    "<<s1;
}
```

***Задание № 3.*** Составить программу на языке С++, определяющую следующую иерархиюклассов.




## *2. .УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ*

### 2.1 Основная литература


1. *Трофимов, В. В.* Основы алгоритмизации и программирования : учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2023. — 137 с. — (Профессиональное образование). — ISBN 978-5-534-07321-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/515434
2. Голицына, О. Л. Основы алгоритмизации и программирования : учебное пособие / О.Л. Голицына, И.И. Попов. — 4-е изд., испр. и доп. — Москва : ФОРУМ : ИНФРА-М, 2021. — 431 с. — (Среднее профессиональное образование). - ISBN 978-5-00091-570-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1150328 – Режим доступа: по подписке.
3. Колдаев, В. Д. Основы алгоритмизации и программирования : учебное пособие / В.Д. Колдаев ; под ред. проф. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 414 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0733-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1735805 – Режим доступа: по подписке


### 2.2. Дополнительная литература


1. *Паронджанов, В. Д.* Алгоритмические языки и программирование: ДРАКОН : учебное пособие для среднего профессионального образования / В. Д. Паронджанов. — Москва : Издательство Юрайт, 2023. — 436 с. — (Профессиональное образование). —

ISBN 978-5-534-14733-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/519246

2. *Кудрина, Е. В.* Основы алгоритмизации и программирования на языке C# : учебное пособие для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва : Издательство Юрайт, 2023. — 322 с. — (Профессиональное образование). — ISBN 978-5-534-10772-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/517324

**Периодические издания:**

1. Прикладная информатика [Электронный ресурс]. – Режим доступа: http://www.iprbookshop.ru/11770.html - ЭБС «IPRbooks»

***Программное обеспечение***

- Microsoft Windows или Яндекс 360
- Microsoft Office Professional Plus 2019
- Google Chrome или Яндекс.Браузер
- Консультант Плюс
- Visual Studio/Visual Studio Code;

## *Базы данных, информационно-справочные и поисковые системы, Интернет-ресурсы*

*Базы данных (профессиональные базы данных)*

–База данных IT специалиста– Режим доступа: http://info-comp.ru/

*Информационно-справочные системы*

–Информационно-справочная система для программистов – http://life-prog.ru

–Информационно-справочная система Федеральной службы по техническому и экспортному контролю (ФСТЭК) – https://fstec.ru/normotvorcheskaya/poisk-po-dokumentam

*Поисковые системы*

–https://www.yandex.ru/

–https://www.rambler.ru/

–https://www.google.ru

–https://www.yahoo.com/

*Электронные образовательные ресурсы*

– Корпорация Майкрософт в сфере образования [Электронный ресурс]– Режим доступа: https://www.microsoft.com/ru-ru/education/default.aspx

– Научная электронная библиотека «Киберленинка» – Режим доступа: http://cyberleninka.ru/

– Национальный открытый университет Интуит– Режим доступа: http://www.intuit.ru/

– Цифровой образовательный ресурс IPR SMART [Электронный ресурс]– Режим доступа: –https://www.iprbookshop.ru/

– Образовательная платформа Юрайт[Электронный ресурс]– Режим доступа: https://urait.ru/

Электронно-библиотечная система Znanium [Электронный ресурс]– Режим доступа: https://znanium.com/