

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)**

**ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО**

**ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ**

**ПИ (филиал) ДГТУ в г. Таганроге**

УТВЕРЖДАЮ

Директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.К. Исаев

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г

Рег. № \_\_\_\_\_\_\_\_\_\_\_\_

**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ**

к практическим занятиям

По дисциплине: ОП.04 Основы алгоритмизации и программирования

По специальности: 09.02.07 «Информационные системы и программирование»

Таганрог

2020**Лист согласования**

Учебно-методическое пособие по учебные дисциплины ***Основы алгоритмизации и программирования*** разработана на основе Федерального государственного образовательного стандарта (далее – ФГОС) для специальности среднего профессионального образования (далее – СПО) 09.02.07 «Информационные системы и программирование».

**Разработчик(и):**

Преподаватель Т.М. Марданова

Методические указания рассмотрены и одобрены на заседании цикловой методической комиссии специальности 09.02.05Прикладная информатика (по отраслям)

Протокол № 7 от «04» февраля 2020г

Председатель цикловой методической комиссии О.В. Андриян

**Рецензенты:**

ЧОУ ВО «ТИУиЭ» начальник информационно-аналитического управления, к.т.н., доцент О.И. Овчаренко

АО «Красный гидропресс»зам. начальника отдела ИТ С.С. Пирожков

**Согласовано:**

Заведующий УМО

Т. В. Воловская

**Введение**

В учебно-методическом пособии к практикуму по курсу «Основы алгоритмизации и программирования» изложены сведения, необходимые для успешного выполнения практических занятий по данному курсу. Описан процесс работы с инструментарием, применяемым на практических занятиях, представлен ряд типичных задач и подходы к их решению. Практические занятия посвящены формированию алгоритмов разработки программных модулей в соответствии с техническим заданием. Разрабатывать программные модули в соответствии с техническим заданием. Выполнять отладку программных модулей с использованием специализированных программных средств. Выполнять тестирование программных модулей. Осуществлять рефакторинг и оптимизацию программного кода.

Цель настоящего пособия – помочь обучающимся при выполнении практических работ, выполняемых для закрепления знаний по теоретическим основам и получения практических навыков работы на компьютерах.

Обучающийся должен знать: понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции. Эволюцию языков программирования, их классификацию, понятие системы программирования. Основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти. Подпрограммы, составление библиотек подпрограмм Объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения.

Обучающийся должен уметь: разрабатывать алгоритмы для конкретных задач. Использовать программы для графического отображения алгоритмов. Определять сложность работы алгоритмов. Работать в среде программирования. Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования. Оформлять код программы в соответствии со стандартом кодирования. Выполнять проверку, отладку кода программы.

Данное учебно-методическое пособие предназначено для обучающихся 2 курса.

**Правила выполнения практических занятий**

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

**Практическая работа №1**

**Знакомство со средой программирования. Составление программ линейной структуры. Составление программ разветвляющейся структуры.**

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с разветвляющейся структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

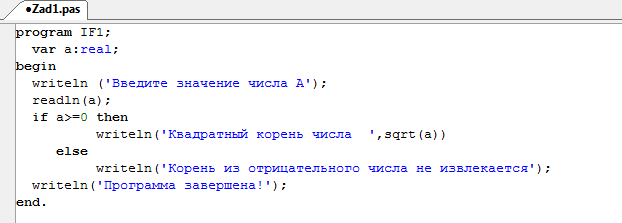
**Время выполнения:** 4 ч

##### Задание 1.

В системе программирования **PascalABC.Net** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

##### Методические указания по выполнению задания:

* 1. Запустите систему программирования **PascalABC.Net**.
  2. В папке своей группы на диске создайте папку **IF**, в дальнейшем все программы необходимо сохранять в эту папку.
  3. В окне редактора наберите текст программы, котораявычисляет квадратный корень из числа. Сохраните файл под именем **Zad1.pas**.



* 1. Проверьте правильность работы программы, задав в качестве исходных данных **А** следующие значения: **655536; -256; 125,44**.
  2. Поставьте «**;**» после оператора **writeln('Квадратный корень числа',sqrt(a))**в строке **7**. Запустите программу на исполнение. Появилось сообщение об ошибке. Почему это произошло? Исправьте ошибку.
  3. Внесите изменения в программу в строки **9-10**, добавив составной оператор **begin-end**.
  4. Используя программу **DiagramDesigner,** начертите блок-схему, которая соответствует полученной программе. Сохраните файл с блок-схемой в папке **IF** под именем **Zad1.ddd**.
  5. Исправьте программу так, чтобы в случае равенства числа нулю программа выводила на экран монитора сообщение **«Значение квадратного корня равняется нулю»**. Сохраните программу.

##### Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **If6**, **If8**.

##### Методические указания по выполнению задания:

1. Создайте новое окно, выполнив команду **Файл – Новый**.
2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотручебных заданий** в поле группа заданий выберите **If**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **If6 (If8)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу еще два раза.

9. Сохраните созданные программы в папке **IF**.

##### Задание 3.

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **DiagramDesigner.** Созданные программу и её блок-схему сохранить в папку **IF**, в названии файла укажите номер задачи.

##### Варианты заданий:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **мер**  **варианта** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **мера**  **заданий** | ,25 | ,24 | ,23 | ,22 | ,21 | ,20 | ,19 | ,18 | ,17 | 0,16 | 1,15 | 2,14 | 3,1 | 4,2 | 5,3 |

**Задания:**

1. Даны три действительные числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень — отрицательные.
2. Даны две точки *А*(𝑥1, 𝑦1) и *В*(𝑥2, 𝑦2). Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.
3. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник. Если да, то будет ли он прямоугольным.
4. Даны действительные числа x и y, не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее — их удвоенным произведением.
5. На плоскости 𝑋𝑂𝑌 задана своими координатами точка А. Указать, где она расположена: на какой оси или в каком координатном угле.
6. Даны целые числа 𝑚, 𝑛. Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.
7. Дано двухзначное число 𝑁. Проверить, будет ли сумма его цифр четным числом.
8. Дано двухзначное число 𝑁. Проверить, будет ли сумма больше 10.
9. Определить, является ли треугольник со сторонами 𝑎, 𝑏, 𝑐 равносторонним.
10. Определить, является ли треугольник со сторонами 𝑎, 𝑏, 𝑐 равнобедренным.
11. Определить, имеется ли среди чисел a, b, c хотя бы одна пара взаимно противоположных чисел.
12. Подсчитать количество отрицательных чисел среди чисел 𝑎, 𝑏, 𝑐.
13. Подсчитать количество положительных чисел среди чисел 𝑎, 𝑏, 𝑐.
14. Подсчитать количество целых чисел среди чисел 𝑎, 𝑏, 𝑐.
15. Дано целое число. Если оно является положительным, то прибавить к нему1; в противном случае не изменять его. Вывести полученное число.
16. Дано целое число. Если оно является положительным, то прибавить к нему1; в противном случае вычесть из него2. Вывести полученное число.
17. Дано целое число. Если оно является положительным, то прибавить к нему1; если отрицательным, то вычесть из него2; если нулевым, то заменить его на10. Вывести полученное число.
18. Даны два действительных числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.
19. Даны два действительных числа.Заменить первое число нулем, если оно меньше или равно второму, и оставить числа без изменения в противном случае.
20. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу(1, 3).
21. Есть окружность с центром в начале координат и радиусом 5 единиц. Ввести с клавиатуры два числа, как координаты точки на той же плоскости, что и окружность. Вывести одно из сообщений: точка внутри окружности; точка на окружности; точка вне окружности.
22. Найти действительные корни квадратного уравнения 𝑎𝑥2 + 𝑏𝑥 + 𝑐 = 0для любых вводимых значений коэффициентов. Рассмотреть ситуации:𝑎 = 0, 𝐷 < 0 и обычное решение с двумя корнями.
23. Решить уравнение 𝐴𝑥 = 𝐵. Предусмотреть случаи отсутствия решения и бесконечного множества решений.
24. Вычислить площадь треугольника по формуле Герона для любых вводимых длин сторон А, В и С: 𝑝 =

𝑎+𝑏+𝑐; 𝑆 = √𝑝 ∙ (𝑝 − 𝑎) ∙ (𝑝 − 𝑏) ∙ (𝑝 − 𝑐). Предусмотреть проверку существования треугольника.

2

1. Даны три числа. Найти сумму двух наибольших из них.

##### Задание 4.

Проверьте файлы, которые были сохранены в папку **IF** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

##### Задание 5.

В системе программирования **PascalABC.Net**составьте программу с усложненной разветвляющейся структурой по заданию. Протестируйтеработу программы. Выполните задания на модификацию созданной программы.

##### Задание 6.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **If26**, **Case10**.

##### Методические указания по выполнению задания:

* + 1. Создайте новое окно, выполнив команду **Файл – Новый**.
    2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотручебных заданий** в поле группа заданий выберите **If**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
    3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **If26 (Case10)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
    4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
    5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
    6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу еще несколько раз.
    7. Сохраните созданные программы в папке **IF2**.

**Практическая работа №2**

**Составление программ циклической структуры. Обработка одномерных массивов. Обработка двумерных массивов.**

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с циклической структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

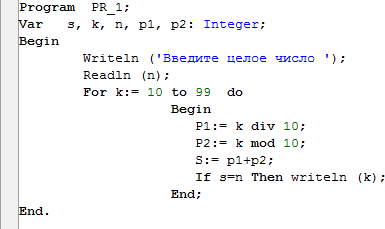
**Время выполнения:** 4 ч

##### Задание 1.

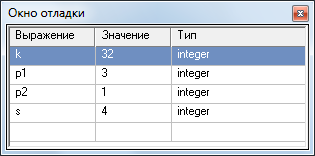
В системе программирования **PascalABC.Net** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

##### Методические указания по выполнению задания:

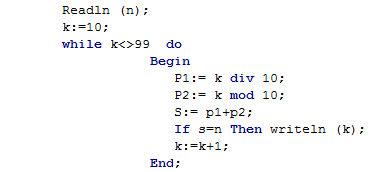
1. Запустите систему программирования **PascalABC.Net**.
2. В папке своей группы на диске создайте папку **Циклы**, в дальнейшем все программы необходимо сохранять в эту папку.
3. В окне редактора наберите текст программы, которая из чисел от 10 до 99 выводит на экран те, сумма цифр которых равна 𝑛 (0 < 𝑛 ≤ 18). Сохраните файл под именем **Zad1.pas**.



1. Проанализируйте текст программы и ответьте на вопросы:
   * Каким образом выделили последнюю (младшую) цифру числа?
   * Каким действием в программе выделили первую (старшую) цифру числа?
2. Объясните обозначение переменных.
3. Рассмотрим выполнение программы в пошаговом режиме для 𝑛 = 2. Режим пошагового выполнения предназначен для отладки программы. Для выполнения одного шага (одной строки) программы следует нажать клавишу **F8** (шаг без входа в подпрограмму), либо клавишу **F7** (шаг со входом в подпрограмму). Окно отладки позволяет просматривать во время пошагового исполнения программы значения переменных. По умолчанию оно располагается в правом верхнем углу окна редактора. Для добавления переменной или выражения в окно отладки следует нажать комбинацию клавиш **Ctrl-F5** или выполнить команду **Программа – Добавить переменную**. Добавьте в окно отладки переменные **k, p1, p2** и **s** и запустите программу в пошаговом режиме.



1. В результате работы программы на экране появится 2 числа: 11, 20.
2. В данной программе используется цикл со счетчиком, изменим программу так чтобы в программе использовался цикл с предусловием. Для этого необходимо внести изменения в строки программы **6- 11**.



1. Запустите программу на исполнение при 𝑛 = 2. Проанализируйте полученные результаты.
2. Самостоятельно выполните изменение программы так, чтобы в ней использовался цикл с постусловием.

##### Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач

##### While1, While4, While11, For5, For12.

**Методические указания по выполнению задания:**

1. Создайте новое окно, выполнив команду **Файл – Новый**.
2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле группа заданий выберите **While (For)**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **While1 (While4, While11, For5, For12)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу необходимое количество раз.
7. Сохраните созданные программы в папке **Циклы**.

**Задание 3.**

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **Diagram Designer.** Созданные программу и её блок-схему сохранить в папку **Циклы**, в названии файла указать номер задачи. Каждый студент должен выполнить все задания.

* С помощью оператора while напишите программу определения суммы всех нечетных чисел в диапазоне от 1 до 99 включительно.
* Найти нечетные и кратные трем числа в диапазоне от 30 до 60 включительно. Распечатать их в порядке убывания.
* В сберкассу на трехпроцентный вклад положили S рублей. Какой станет сумма вклада через N лет?

##### Задание 4.

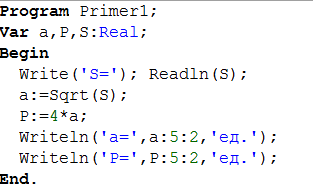
Проверьте файлы, которые были сохранены в папку **Циклы** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

##### Задание 5.

В системе программирования **PascalABC.NET** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

##### Методические указания по выполнению задания:

1. Запустите систему программирования **PascalABC.NET**.
2. В окне редактора наберите текст программы, которая вычисляет сторону и периметр квадрата, если известна его площадь.



В **PascalABC** используется несколько типов представления числовых значений, на начальном этапе будут рассмотрены лишь некоторые из них: **Integer** – целые числа в интервале от -2147483648 до 2147483647; **Real** – вещественные – целые и дробные положительные и отрицательные числа

Описания констант в декларативной части производится перед переменными, и предусматривают определенную форму записи чисел (дополнительно тип константы не оговаривается): если константа записана с точкой, тип константы считается Real. При записи значения константы используется знак равенства.

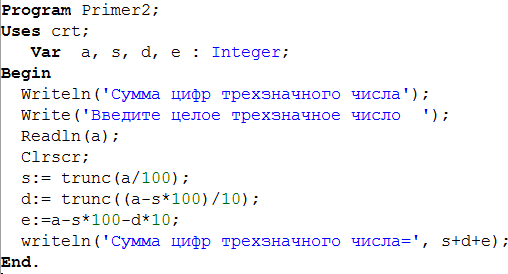
Переменная – это вид объектов в программе, предназначенный для хранения информации во время выполнения программы. По правилам **PascalABC** каждая переменная должна быть объявлена, т.е. описана в декларативной части программы. Переменная не имеет какого-либо конкретного значения до тех пор, пока компьютеру не будет дано точное предписание, поместить что-либо определенное в соответствующую ячейку памяти.

Описание переменных следует за описанием констант. В описании переменных после двоеточия указывается тип переменной/

В **PascalABC** возможны следующие действия (группы операций записаны в порядке приоритета): операция возведения в степень; умножение, деление, деление целочисленное, получение остатка от целочисленного деления; сложение, вычитание.

В пределах одной группы приоритета порядок выполнения операций, если нет скобок, определяется последовательностью записи.

1. Сохраните созданную программу в папке **Begin**, созданную ранее под именем **Zad1.pas**.
2. Выполните команду файл – Новый и наберите следующую программу. Разберитесь в её работе.

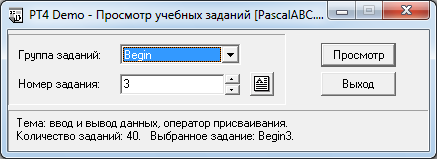


1. Определите с помощью каких стандартных операций вычисляется количество сотен и количество десятков. Запишите в тетрадь синтаксис, используемой для этого функции.
2. Измените программу так, чтобы для определения количества сотен и количества десятков использовались операции целочисленного деления и получения остатка от целочисленного деления.
3. Сохраните созданную программу в папке **Begin** под именем **Zad2.pas**.

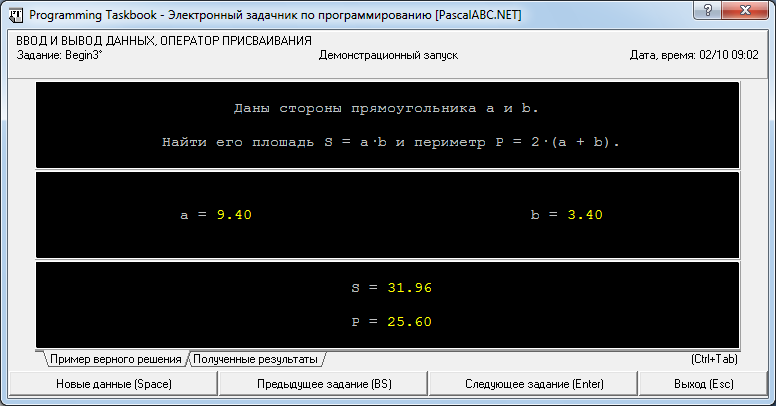
**Задание 6.** Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **Begin3**, **Begin4, Begin5.**

##### Методические указания по выполнению задания:

1. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле номер задания задайте значение **3** и нажмите кнопку **Просмотр**.



1. Окно просмотра текста задачи состоит из трех разделов. В первом записано условие задачи, во втором представлены значения исходных данных, а в третьем разделе окна указаны результаты решения задачи.



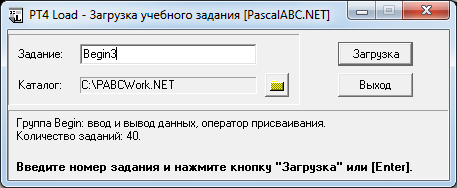
Условие

Исходные

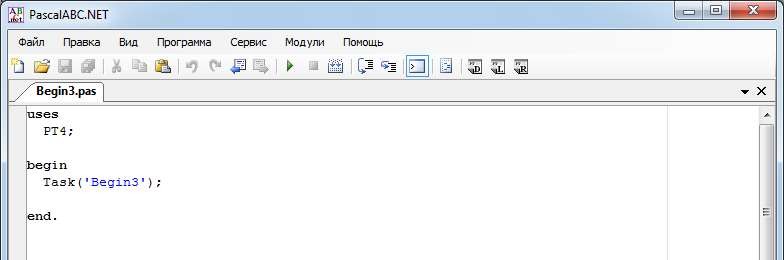
данные

Результаты

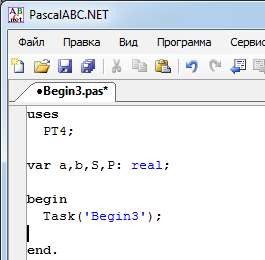
1. Решим данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **begin3** и нажмите кнопку **Загрузка**.



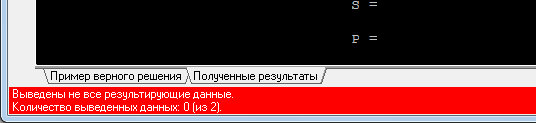
1. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.



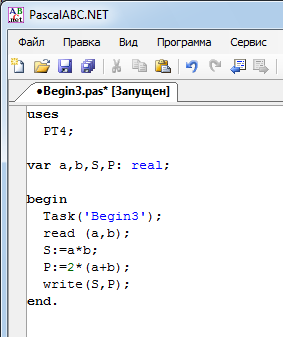
1. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – выполнить** или нажмите соответствующую кнопку на панели инструментов.
2. При написании программы для решения задачи нам необходимо описать четыре переменные, две из них этого стороны прямоугольника **a, b**, именно их значения указаны в окне исходных данных, и переменные S и P, для результатов вычислений, их значения указаны в окне результатов. Описание переменных осуществляется в разделе **var**. Проанализировав значения исходных данных и полученных результатов в задачнике, можно сделать вывод, что переменные имеют тип **real**. Перейдите к окну шаблона программы и выполните описание переменных.



1. Далее перейдем к написанию программы. Нам необходимо ввести значения переменных **a, b**, для этого необходимо считать их значения с помощью оператора ввода **read**. Запустите программу на исполнение, программа выдаст сообщение, что **выведены не все результирующие данные**, т.е. ввод прошел нормально, а вывод значений был не сделан.



1. Для того чтобы выполнить вывод результирующих значений на экран нам необходимо выполнить расчет площади и периметра, формулы для вычисления представлены в условии задачи **S=ab, P=2(a+b)**. Запишите данные формулы, используя оператор присваивания.
2. Выполните запуск программы, изменился ли текст сообщения? Почему не изменился? Какой оператор необходимо добавить в текст программы?
3. Добавьте в тело программы оператор вывода, обратите внимание на последовательность переменных при выводе.



1. Запустите программу, изменился ли текст сообщения?
2. Выполните самостоятельно решение задач **Begin4, Begin5**.

**Задание 3.**

Составьте программы для решения следующих задач:

* Рассчитайте и выведите на экран количество рабочих часов в месяце, если продолжительность рабочего дня равна 8 часам в день, а число рабочих дней в месяце запрашивается у пользователя вашей программы.
* Скорость передачи данных по локальной сети запрашивается у пользователя и измеряется в битах в секунду. Ученик качал игру Т минут. Рассчитайте и выведите на экран размер файла, который скачал ученик и сколько денег придѐтся заплатить ему за трафик, если первый Гбайт не оплачивается, а всё то, что сверху - по у рублей за Гбайт.
* Жёсткий диск имеет объём свободного пространства Х Гбайт – запрашиваемая величина. Сколько книг, каждая из которых состоит из 350 страниц, на каждой странице по 35 строк, в каждой строке по 55 символов, можно записать на жёсткий диск, если для хранения кода одного символа отводится 2 байта?

##### Задание 4.

Проверьте файлы, созданные в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

**Практическая работа №3**

**Работа со строками. Работа с данными типа множество. Файлы последовательного доступа.**

**Цель:**изучить возможности применения множеств в языке программирования, операции над множествами.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Задание.** Решить задачи и оформить согласно этапам решения задач на ЭВМ.

1. Вычислить сумму тех элементов матрицы *A*, номера строк и столбцов которых принадлежат к заданным множествам целых чисел *S1*и *S2*.

2. Дан текст из цифр и латинских букв. Определить, каких букв больше: гласных или согласных.

3. Написать программу, формирующую случайным образом множество целых чисел и определяющую мощность этого множества.

4. Написать программу, формирующую случайным образом два числовых множества и определяющую, в каком отношении находятся эти множества.

5. В городе *N*имеется 100 кондитерских магазинов. Известно, что в каждом из этих магазинов не более 20 видов сластей в ассортименте. Какие виды сластей есть во всех имеющихся магазинах? Существует ли магазин, торгующий уникальной продукцией? Перечислите пять видов сластей, которые есть в большинстве магазинов города *N*(ассортимент кондитерских магазинов рассматривать как данные перечисляемого типа).

**Итог работы:**отчет, защита работы.

**Практическая работа №4**

**Типизированные файлы. Нетипизированные файлы. Организация процедур.**

**Цель работы:**

- Знакомство с файловыми типами в Паскале.

- Получение навыков операций с файлами

- Применение навыков проектирования сложных программ

- Применение навыков создания пользовательского интерфейса

## Содержание отчета

Отчет должен содержать:

1. Постановка задачи
2. Таблица спецификаций, включая структуру записи файла.
3. Схема иерархии модулей.
4. Блок-схемы основных модулей или словесное описание алгоритма по пунктам.
5. Таблица тестов.
6. Результаты тестирования (копии экранов).
7. Листинг основных подпрограмм.

## Задания

### Общее условие

Программа должна включать следующие функции: открытие (создание в случае его отсутствия), сохранение файла, добавление, изменение полей записи, удаление записи, просмотр содержимого файла, обработка запросов. Выбор действия осуществляется посредством меню (MainMenu). **Обработка ошибок пользовательского ввода обязательна!** При выполнении данного задания следует показать умение пользоваться изученными компонентами.

Для открытия/сохранения файлов использовать стандартные диалоговые окна (компоненты OpenDialog и SaveDialog со страницы Dialogs). Данные читать в переменную типа record, структура которой соответствует варианту задания (использовать типизированный файл). Для отображения данных использовать компонент StringGrid.

Также на форме следует разместить кнопки «Изменить», «Удалить», «Добавить» или использовать контекстное меню (PopupMenu). При выборе редактирования/добавления новой записи следует вызывать форму, на которой размещаются компоненты (Edit, MaskEdit, CheckBox, RadioGroup и др. в соответствии с заданием). Для запоминания выбранной строки в StringGrid для редактирования использовать событие onSelectCell (параметр ARow – номер выбранной строки).

Следует учитывать, что в StringGrid для доступа к конкретной ячейке используется метод Cell[j,i], в котором сначала указывается столбец, а потом строка (счет с 0).

Все изменения, производимые во время выполнения приложения, хранятся в StringGrid, при выборе пункта «Сохранить» – заносятся в файл.

*Дополнительное задание*: переделать приложение таким образом, чтобы все изменения параллельно отражались на форме и в файле (для удаления записи из файла необходимо создать временный файл, в который перенести все записи, кроме удаляемой; затем старый файл удалить, а временный переименовать по имени удаленного).

### Варианты заданий

1 (17). Дан файл student.dat. Поля записи содержат следующие данные: фамилия, имя, отчество, шифр группы, пол, семейное положение (0/1), номер комнаты в общежитии (0-если не в общежитии).

Обработка следующих запросов:

* по номеру комнаты вывести список жильцов;
* вывести список всех семейных студентов;
* по ФИО и шифру группы указать N комнаты проживания;
* выдать список студентов с фамилией на заданную букву.

2 (18). База данных heap.dat. Структура записей: номер изделия, наименование изделия, количество изделий, цена 1 изделия, стоимость.

Обработка следующих запросов:

* по номеру изделия выдать справку о наличии его на складе;
* выдать общую стоимость всех изделий, хранящихся на складе
* выдать список всех изделий c номерами в заданном интервале;
* найти изделие с самой большой ценой.
* выдать список изделий на складе, наименование которых начинается с заданной буквы.

3 (19). Записи файла plan.dat имеют вид: шифр специальности, наименование дисциплины, шифр кафедры [БукваЦифраЦифра], количество лекций, количество практик, количество лабораторных, признак: 2 – экзамен, 1 – зачет, 0 – нет ничего.

Обработка следующих запросов:

* по шифру кафедры подсчитать общее количество проводимых ею занятий;
* вывести список дисциплин для заданной специальности;
* найти план занятий по заданным специальности и дисциплине;
* вывести список дисциплин, читаемых заданной кафедрой.

4 (20). База данных table.dat содержит данные о расписании занятий на факультете. Структура записей: шифр группы, день недели, время начала занятия, предмет, номер аудитории, фамилия преподавателя.

Обработка следующих запросов:

* выдать расписание на всю неделю для указанной группы;
* выдать расписание для указанной группы в указанный день;
* найти выходные дни для указанной группы;
* выдать номера аудиторий, где проходят занятия по заданному предмету.

5 (21). Справочная gai.dat содержит записи следующего наполнения: марка автомобиля, номер, цвет, год регистрации в ГАИ, фамилия владельца.

Обработка следующих запросов:

* выдать фамилии владельцев и номера автомобилей заданной марки;
* выдать количество автомобилей заданной марки;
* выдать список всех а/м заданной марки, зарегистрированных не раньше заданного года;
* поиск владельца а/м по цвету, по марке и по первой букве номера а/м.

6 (22). База данных study.dat содержит записи вида: номер зачетки, фамилия И.О., номер варианта, отметка о выдаче задания 1 (1/0), отметка о выполнении задания 1 в баллах, отметка о выдаче задания 2 (1/0), отметка о выполнении задания 2 в баллах, отметка о выдаче задания 3 (1/0), отметка о выполнении задания 3 в баллах.

Требуется обеспечить выдачу:

* общей ведомости;
* справки о студенте по номеру зачетки;
* ведомости о выполнении студентами задания с указанным номером;
* списка студентов с заданным номером варианта, имеющих заданную сумму баллов по всем 3 задачам;
* списка студентов, не получивших задание с указанным номером.

7 (23). База данных table.dat содержит данные о расписании занятий на факультете. Структура записей: шифр группы, день недели, время начала занятия, предмет, номер аудитории, фамилия преподавателя.

Обработка следующих запросов:

- выдать расписание на всю неделю для указанного преподавателя;

- выдать номер аудитории для указанной группы в заданные день и время;

- найти выходные дни для указанного преподавателя;

- выдать список групп, занимающихся в указанный день в заданной  
аудитории.

8 (24). В базе book.dat хранятся записи, содержащие: номер каталога, номер книги, наименование книги, год издания, цена 1 экземпляра.

Обеспечить выдачу следующих справок:

* по автору книги найти все его книги;
* по номеру книги найти наименование книги и автора;
* выдать список книг, изданных в указанном году;
* найти книги по фамилии автора, году издания.

9 (25). В базе bus.dat хранятся записи, содержащие: номер маршрута, начало маршрута, конец маршрута, тип автобуса, количество автобусов на линии, номер автобазы.

Обеспечить выдачу следующих справок:

* определить общее число городских автобусов;
* по номеру автобуса определить его маршрут;
* выдать список номеров автобусов, маршруты которых проходят через заданный пункт;
* выдать список маршрутов, которые обслуживает автобаза с указанным номером.

10 (26). В базе export.dat содержатся сведения об экспорте товаров: наименование товара, страна, импортирующая товар, объем поставляемой партии в штуках, страна-производитель.

Обработка следующих запросов:

* найти страны, в которые экспортируется данный товар;
* вывести список товаров, поставляемых указанной страной;
* вывести наименование товаров, начинающихся с заданной буквы;
* вывести общий объем товаров, поставляемых в указанную страну.

11 (27). Дан файл toys.dat, содержащий сведения об игрушках: указывается название игрушки (кукла, мяч, кубики и т.д.), ее стоимость в тыс. руб. и возрастные границы детей, для которых игрушка предназначена.

Получить следующую информацию:

* названия самых дорогих игрушек (цена которых отличается от цены самой дорогой не более чем на 1 тыс. руб.);
* цены всех игрушек с заданным названием;
* найти игрушки, предназначенные детям от N лет;
* найти игрушку на заданную букву.

12 (28). База содержит phone.dat номера телефонов сотрудников учреждения: указывается фамилия, инициалы сотрудника, номер телефона и его адрес.

Обработка следующих запросов:

* найти телефон сотрудника по его фамилии;
* найти список телефонов сотрудников, чьи фамилии начинаются с заданной буквы;
* найти список сотрудников, чьи телефоны начинаются с заданной цифры;
* найти адрес сотрудника по номеру его телефона.

13 (29). В базе apteka.dat хранятся записи, содержащие информацию о лекарствах: номер аптеки, название лекарства, цена, количество.

Обработка следующих запросов:

* получить сведения о наличии лекарства в определенной аптеке;
* получить общую стоимость всех лекарств во всех аптеках;
* по названию лекарства выдать список аптек, в которых оно имеется;
* найти самое дешевое лекарство.

14 (30). В базе данных school.dat хранятся сведения о годовых оценках учеников школы: ФИО ученика, класс (4-11), индекс класса (А-Ж), оценки за год по литературе, русскому языку, математике, физкультуре, истории.

Обработка следующих запросов:

* выдать список учеников-отличников определенного класса;
* выдать названия классов, в которых учеников больше 20;
* подсчитать неуспевающих учеников в школе;
* подсчитать число медалистов.

15 (31). База данных worker.dat содержит записи вида: номер отдела, название отдела, ФИО, пол, год рождения, стаж работы, количество детей.

Требуется выдать:

* по номеру отдела информацию обо всех сотрудниках;
* фамилии сотрудников, имеющих детей;
* список сотрудников со стажем работы больше заданного;
* список сотрудников с заданным годом рождения.

16 (32). Дан файл reader.dat с записями вида: индекс издания, наименование, период подписки (количество месяцев), цена 1-го месяца, ФИО подписчика, адрес подписчика, стоимость подписки, N почтового отделения связи.

Обеспечить выдачу следующих справок:

* для заданного N почтового отделения создать файл reestr.dat с записями вида: индекс издания, наименование, количество подписчиков по месяцам, общая стоимость подписки;
* по фамилии подписчика выдать справку обо всех выписанных им изданиях;
* по индексу издания выдать список всех подписавшихся на него;

найти ФИО подписчика, который выписал издания на самую большую сумму в заданном почтовом отделении.

**Практическая работа №6**

**Изучение интегрированной среды разработчика.**

**Цель работы**: Обобщить знания по интегрированной среде разработке (ИСР) Delphi; получить практические навыки работы в ИСР, получить практические навыки программирования консольных приложений

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Задание.** Изучить интегрированную среду разработки Delphi. Создать консольное приложение.

Теоретические сведения

ИСР (Integrated Development Environment) - это среда, в которой есть все необходимое для проектирования, запуска и тестирования приложений и где все нацелено на облегчение процесса создания программ. ИСР интегрирует в себе редактор кодов, отладчик, инструментальные панели, редактор изображений, инструментарий баз данных.

При запуске Delphi открывается основное окно ИСР.

В верхней части ИСР располагается полоса главного меню. Ее состав несколько различается от версии к версии , кроме того зависит от варианта Delphi.

Ниже полосы главного меню расположены две инструментальные панели. Левая панель содержит два ряда быстрых кнопок, дублирующих некоторые наиболее часто используемые команды меню. Права панель содержит палитру компонентов библиотеки визуальных компонентов. Палитра компонентов содержит ряд страниц, закладки которых видны в ее верхней части. Состав палитры зависит от версии Delphi.

Правее полосы главного меню размещена еще одна небольшая инструментальная панель, содержащая выпадающий список и две быстрые кнопки. Это панель сохранения и выбора различных конфигураций окна ИСР.

В основном пол окна слева располагаются Дерево Объектов (Object tree View), под ним – Инспектор Объектов (Object Inspector). Окно Дерево Объектов отображает иерархическую связь визуальных и невизуальных компонентов и объектов разрабатываемого приложения. Инспектор Объектов – это основной инструмент, с помощью которого можно задавать свойства компонентов и обработчики событий.

Правее этих окон располагается окно пустой формы, готовой для переноса на нее компонентов.

Полоса главного меню

Разделы меню File(файл) позволяет создать новый проект, новую форму, фрейм, модуль данных, открыть ранее созданный проект или форму, сохранить проекты или формы в файлах с заданными именами.

Разделы меню Edit(правка, редактирование) позволяют выполнять обычные для приложений Windows операции обмена с буфером, а также дают возможность выравнивать группы размещенных на форме компонентов по размерам и месторасположению.

Разделы меню Search(поиск) позволяют осуществлять в коде разрабатываемого приложения поиск и контекстные замены, которые свойственны большинству известных тестовых редакторов.

Разделы меню View(просмотр) позволяют вызывать на экран различные окна необходимые для проектирования.

Разделы меню Project позволяют добавлять и убирать из проекта формы, задавать опции проекта, компилировать проект без его выполнения и делать много других полезных операций.

Разделы меню Run(выполнение) дает возможность выполнять проект в нормальном или отладочном режимах, продвигаясь по шагам, останавливаясь в указанных точках кода, просматривая значения переменных и т.д.

Меню Component (компонент) позволяет создавать и устанавливать новые компоненты, конфигурировать палитру компонентов, работать с пакетами packages.

Разделы меню DataBase (база данных) позволяет использовать инструментарий для работы с базами данных.

Меню Tools (инструментарий) включает ряд разделов, позволяющих выполнять настройки ИСР и вызывать различные вспомогательные программы.

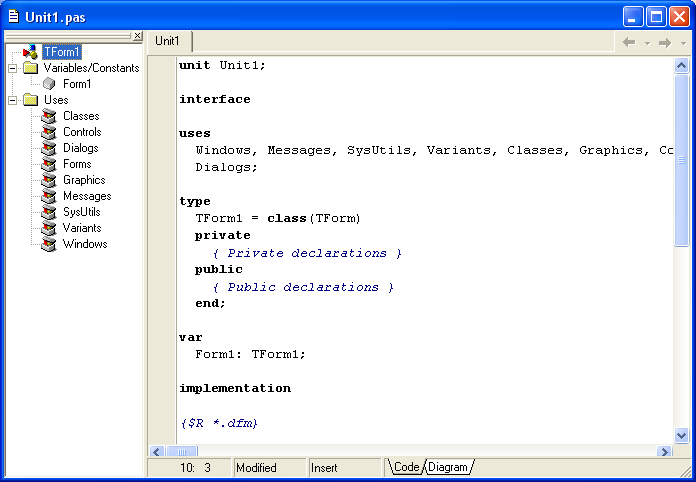
Меню Windows (окно) имеется только начиная с 6 версии. Разделы этого меню позволяют ориентироваться среди массы окон, обычно одновременно открытых в процессе проектирования и переключаться в нужное окно.

Меню Help (справка) содержит разделы, помогающие работать со встроенной в Delphi справочной системой.

Палитра компонентов

Перечислим некоторые из них:

|  |  |
| --- | --- |
| Standard | Стандартная, одержащая наиболее часто используемые компоненты |
| Additional | Дополнительная, являющаяся дополнением страндартной |
| Win32 | 32-битные компоненты в стиле Windows 95/98/200 и NT |
| System | Системная, содержащая такие компоненты, как таймеры, плееры и ряд других |
| Internet | Компоненты для создания серверов Web |
| QReport | Компоненты для подготовки отчетов |
| Dialogs | Диалоги, системные диалоги типа «Открыть файл» и др. |
| Win 3.1 | Windows 3.x, компоненты в стиле Windows 3.x (оставлены для обратной  совместимости с Delphi 1) |
| Samples | Образцы, различные интересные, но не до конца документированные компоненты |
| ActiveX | Примеры компонентов ActiveX |
| COM+ | Компонент, дающий доступ к катологу COM+, содержащему данные по  конфигурации COM+ |
| IndyClients | Компоненты клиентских приложений Internet Direct (Indy), дающих доступ к  различным протоколам Интернет из приложений Delphi, C++Builder, Kylix |
| IndyServers | Компоненты серверных приложений Internet Direct (Indy) |
| IndyMisk | Различные вспомогательные компоненты приложений Internet Direct (Indy) |
| Servers | Оболочки VCL для распространенных серверов COM. |



Редактор кода включает в себя две страницы: Code (код) и Diagram (диаграммы).

Страница Code содержит коды модулей приложения и тексты других файлов, которые открыты в процессе проектирования. Является полноценным программным редактором. В заголовке окна Редактора кода отображается имя текущего файла.

В нижней части окна Редактора Кода располагается строка состояния. В самой левой ее позиции находится индикатор строки и колонки, который помогает определить к каком месте кода вы находитесь. Второй элемент строки

состояния- индикатор модификации, который указывает были ли сделаны изменения в коде. Третий элемент строки состояния- индикатор режима вставки, который показывает, будут ли вводимые символы вставляться в текст или писаться поверх текста. Переключение режима вставки производится клавишей Insert.

Страница Diagram позволяет строить диаграммы, иллюстрирующие взаимоотношения компонентов в приложении.

Инспектор объектов

Обеспечивает простой и удобный интерфейс для изменения свойств объектов Delphi и управления событиями, на которые реагирует объект.

Окно Инспектора Объектов имеет две страницы. В Верхней части окна имеется выпадающий список всех компонентов, размещенных на форме. В нем можно выбрать тот компонент, свойства и события которые вас интересуют.

Страница свойств (Properties) Инспектора Объектов показывает свойства того объекта, который в данный момент выделен.

Страница событий (Events) показывает все события, на которые может реагировать выбранный объект.

Консольного приложения

Используя ИСР Delphi можно создать в ОС MS Windows приложение, которое будет работать как программа MS Dos. Такие приложения называются консольными.

Консольное приложение — программа, предназначенная для работы в операционной системе MS-DOS (или в окне DOS), для которой устройством ввода является клавиатура, а устройством вывода — монитор, работающий в режиме отображения символьной информации (буквы, цифры и специальные знаки).

Чтобы создать консольное приложение необходимо выполнить команду File| New | Other, затем на вкладке New выбрать Console Application. В результате на экране появится окно Projectl.dpr, в котором находится шаблон главной процедуры консольного приложения. В этом окне можно набирать инструкции программы.

Следует обратить внимание на то, что в DOS используется кодировка ASCII, а в Windows — ANSI, буквы русского алфавита в которых имеют разные коды. Это приводит к тому, что вместо сообщений на русском языке консольное приложение выводит непонятные символы. Поэтому консольные приложения должны выводить сообщения на английском или необходимо разработать подпрограмму преобразования сроки Ansi в строку ASCII.

Начинается текст программы строкой {$APPTYPE CONSOLE} - это директива предназначена для компилятора. Следуя ее указаниям, компилятор генерирует исполняемую программу как консольное приложение.

Компиляция консольного приложения выполняется обычным образом, т. е. выбором из меню Project команды Compile. После успешной компиляции программа может быть запущена выбором из меню Run команды Run. При запуске консольного приложения на экране появляется стандартное окно DOS- программы. Процесс сохранения проекта консольного приложения стандартный. В результате выбора из меню File команды Save на экране появляется диалоговое окно Save Project, в котором нужно ввести имя проекта.

Пример выполнения работы Задание. Написать программу пересчет веса из фунтов в килограммы

1. Листинг программы program funt2kg;

{$APPTYPE CONSOLE}

function Rus(mes: string):string; // Функция Rus преобразует ANSI-строку в ASCII-строку var i: integer; // номер обрабатываемого символа

begin

for i: mes[i] of

'A'..'п' : mes[i] := Chr(Ord(mes[i]) - 64);

'р'..'я' : mes[i] .:= Chr (Ord(mes [i] ) -16); end;

rus := mes; end;

// основная программа

Var f, w :real; // вес в фунтах и в граммах

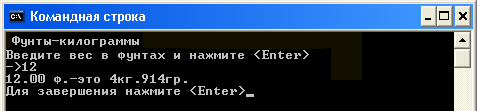
k, g:integer; // кол-во килограммов и кол-во граммов begin

writeln(Rus('Фунты-килограммы')); writeln(Rus('Введите вес в фунтах и нажмите <Enter>')); write('-> '); readln(f);

w := f \* 0.4095; // один фунт — это 409,5 гр. writeln(w, Rus(' кг')};

write(Rus('Для завершения нажмите <Enter>')); readln;end.

1. Результат работы программы



**Задание:**

|  |  |
| --- | --- |
| Вари  ант | Задание |
| 1 | Треугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3). Найти площадь треугольника АВС |
| 2 | Найти сумму n членов арифметической прогрессии, первый член которой равен а, а разность равна q.  Найти третий, тринадцатый и двадцать третий члены прогрессии |
| 3 | Трапеция задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3), D(x4,y4). Найти значение средней линии трапеции. |
| 4 | Квадрат задается координатами своей стороны АВ: A(x1, y1), B(x2, y2). Найти его периметр. |
| 5 | Треугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3). Найти длины медиан треугольника АВС |
| 6 | Дан радиус основания и высота цилиндра . Найти его площадь боковой поверхности и объем. |
| 7 | Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности. |
| 8 | Найти площадь кольца по заданным внешнему и внутреннему радиусам. |
| 9 | Треугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3). Найти длины высот треугольника АВС |
| 10 | Даны катеты прямоугольного треугольника. Найти его площадь и периметр |
| 11 | Квадрат задается координатами своей стороны АВ: A(x1, y1), B(x2, y2). Найти его площадь. |
| 12 | Дана длина стороны равностороннего треугольника. Найти радиусы вписанной и описанной  окружностей. |
| 13 | Треугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3). Найти длины биссектрис треугольника АВС |
| 14 | Дано шестизначное число. Найти произведение его цифр. |
| 15 | Прямоугольный параллелепипед задается длинами a,b,c. Вычислить площадь поверхности и объем параллелепипеда. |
| 16 | Треугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3). Найти периметр треугольника АВС. |
| 17 | Вычислить скалярное произведение двух векторов, если известны координаты векторов (x1, y1, z1) и (x2, y2, z2). |
| 18 | Ромб задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3), D(x4,y4). Найти площадь ромба АВСD |
| 19 | Дано семизначное число. Найти произведение его цифр. |
| 20 | Вычислите длину окружности, площадь круга, объем шара заданного радиуса |
| 21 | Дано натуральное число n (n>999). Определить число единиц, десятков, сонет, тысяч в нем. |
| 22 | Дано положительное число а. Вычислить площадь равностороннего треугольник со стороной а;  площадь квадрата со стороной а; площадь круга, радиус которого равен а. |
| 23 | Дано шестизначное число. Найти сумму его цифр. |
| 24 | Найти сумму n членов арифметической прогрессии, первый член которой равен а, а разность равна q.  Найти пятый, двадцатый и пятидесятый члены прогрессии |
| 25 | Трапеция задается длинами оснований и высотой. Найти площадь трапеции АВСD. |
| 26 | Квадрат задается координатами своей стороны АВ: A(x1, y1), B(x2, y2). Найти его диагональ. |
| 27 | Прямоугольник задается координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3), D(x4,y4). Найти длину диагонали прямоугольника АВСD |
| 28 | Найти сумму n членов геометрической прогрессии, первый член которой равен а, а знаменатель  равен q. Найти третий, пятнадцатый и шестнадцатый члены прогрессии |
| 29 | Треугольник, заданный координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3), описывает окружность. Найти радиус окружности |
| 30 | Дано пятизначное число. Найти произведение его цифр. |
| 31 | В треугольник, заданный координатами своих вершин на плоскости: A(x1, y1), B(x2, y2), C(x3,y3), вписана окружность. Найти радиус окружности |

**Контрольные вопросы:**

1. Интегрированная среда разработки Delphi. Определение ИСР и вид главного окна ИСР.
2. Полоса главного меню. Вид и назначение.
3. Палитра компонентов. Вид и назначение.
4. Редактор кода. Вид и назначение
5. Инспектор объектов. Вид и назначение.
6. Консольное приложение. Определение. Создание. Запуск

**Практическая работа №7**

**Создание проекта с использованием компонентов для работы с текстом.**

**Цель работы:** сформировать умения по использованию компонентов для работы с текстом в Delphi, сформировать умения по созданию приложений с компонентами для работы с текстом в Delphi.

**Оборудование, технические и программные средства:** персональный компьютер, интегрированная среда разработчика **Turbo Delphi**.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

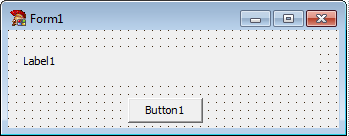
**Время выполнения:** 4 ч

##### Задание 1.

Разработать программу, которая при нажатии на кнопку **«Output»** выводит сообщение **«Моя первая программа на языке Delphi»**, а затем при повторном нажатии на эту же кнопку сообщение исчезает. При повторном выводе размер надписи должен увеличиваться.

##### Методические указания по выполнению задания:

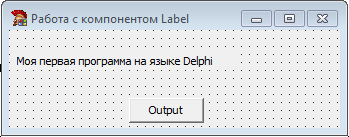
* + 1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
    2. Создайте в папке своей группы новую папку и назовите её **Label**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
    3. Разместите на форме компонент **Label** и кнопку **Button** вкладки палитры компонентов **Standard**.

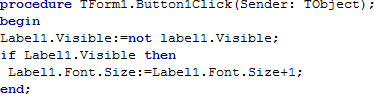


Компоненты ввода — вывода данных можно условно разделить на несколько различных блоков: компоненты вывода текстовой информации на экран; однострочные поля ввода текстовой и числовой информации; многострочные поля ввода.

Для вывода определенной информации на экран, кроме уже ранее используемого компонента **Label**, есть и другие компоненты. Текст, который будет отображен, можно задавать как на этапе разработки формы, так и в процессе выполнения программы, присвоив значение свойству **Caption**.

* + 1. Задайте для формы заголовок **«Работа с компонентом Label»**.
    2. Выделите надпись **Label1**, найдите в **Object Inspector** свойство **Caption** и вставьте новое название надписи **Моя первая программа на языке Delphi.**
    3. Выделите кнопку **Button1**, найдите в **Object Inspector** свойство **Caption** и вставьте новое название кнопки **Output**.



* + 1. Перейдите в **Object Inspector** на страницу **Events**, найдите событие **OnClick** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре кнопки **Button1**, напишите следующий программный код:

В этой программе при каждом очередном нажатии происходит изменение свойства **Visible**, вследствие чего надпись то появляется, то исчезает с экрана, а также происходит увеличение свойства **Size**.

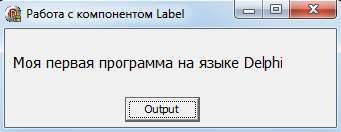
Для вывода определенной информации на экран, кроме уже описанного компонента **Label**, может быть также использован компонент **Panel** с той же самой вкладки или **StaticText** со страницы **Additional**. Они имеют незначительные отличия от компонента **Label**.

Остальные компоненты позволяют вводить и редактировать информацию, включая возможность выделения, копирования, удаления и вставки фрагментов текста. Отметим общие для всех редакторов методы: **Clear** — удалить весь текст, помещенный в редакторе; **ClearSelect** — удалить выделенный фрагмент текста; **CopyToClipboard** — копировать в буфер выделенный фрагмент, **CutToClipboard** — удалить из текста выделенный фрагмент и поместить его в буфер, **PasteFromClipboard** — копировать текст из буфера в то место редактора, где в данный момент находится курсор.

Для всех редакторов определено дополнительное событие **OnChange**, возникающее, когда изменяется текст, находящийся в редакторе.

Для ввода или вывода одной строки могут использоваться компоненты **Edit** со страницы **Standart** и **MaskEdit** со страницы **Additional**. Основное свойство данных компонентов — это строка, которая либо вводится, либо выводится. Данное свойство имеет имя **Text** и тип **String**, доступное как во время подготовки, так и время выполнения. Логическое свойство **ReadOnly** позволяет запретить изменения, а целочисленное свойство **GetTextLen** выдает текущую длину строки.

* + 1. Сохраните изменения и запустите проект. Протестируйте его работу.

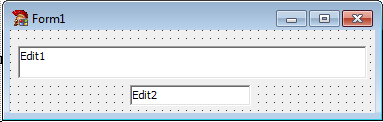


##### Задание 2.

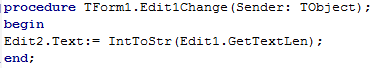
Разработать программу, которая при вводе текста в первый компонент **Edit1**, во втором компоненте **Edit2** отображает реальную длину вводимой строки. Кроме этого, при выходе из компонента **Edit1** его содержимое копируется в буфер обмена и удаляется, а при возвращении в программу появляется снова.

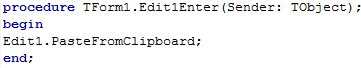
##### Методические указания по выполнению задания:

1. Создайте в папке своей группы новую папку и назовите её **Edit1**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
2. Разместите на форме два компонент **Edit** вкладки палитры компонентов **Standard**.

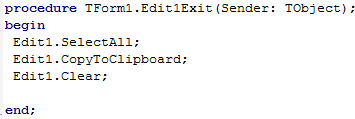


1. Задайте для формы заголовок **«Работа с компонентом Edit»**.
2. Выделите текстовое поле **Edit1**, найдите в **Object Inspector** свойство **Text** и оставьте его пустым. Аналогичные действия выполните со вторым текстовым полем.
3. Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnChange** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код:

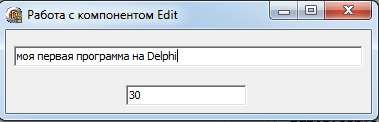


Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnEnter** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код: 

1. Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnExit** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код:



1. Сохраните изменения и запустите проект. Протестируйте его работу.

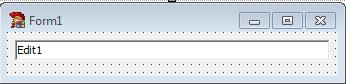


##### Задание 3.

Разработать программу, которая запрещает ввод в компонент **Edit1** подряд двух одинаковых символов.

##### Методические указания по выполнению задания:

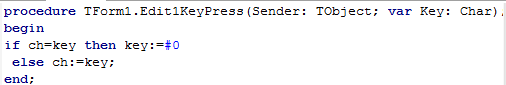
1. Создайте в папке своей группы новую папку и назовите её **Edit2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
2. Разместите на форме компонент **Edit** вкладки палитры компонентов **Standard**.



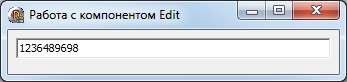
1. Перейдите в код программы (на клавиатуре нажмите клавишу **F12**). Введите в раздел **VAR** глобальную переменную **ch** типа **char**, в которой будет храниться последний нажатый символ.



1. Задайте для формы заголовок **«Работа с компонентом Edit»**.
2. Выделите текстовое поле **Edit1**, найдите в **Object Inspector** свойство **Text** и оставьте его пустым.
3. Создайте процедуру обработки события **KeyPress** текстового поля **Edit1**, параметр **Key** данной процедуры содержит символ нажатой клавиши. Если вновь введенный символ совпадает с только что нажатым символом, то он игнорируется. В противном случае, новый символ запоминается в переменной **ch**.



1. Сохраните изменения и запустите проект. Протестируйте его работу.



##### Задание 4.

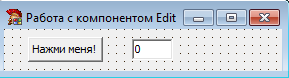
Разработать программу, которая считает количество нажатий на кнопку и выдает это значение в компоненте **Edit**.

##### Методические указания по выполнению задания:

1.Создайте в папке своей группы новую папку и назовите её **Edit3**. Сохраните проект в созданную ранее папку, выполнив команду **File – Save Project as…**

2.Разместите на форме компонент **Edit** и кнопку **Button** вкладки палитры компонентов **Standard**.

3.Используя **Object Inspector,** задайте значения свойств компонентов формы в соответствии с рисунком.



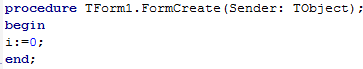
4.Если в целочисленной переменной **i** будем считать количество нажатий, то процедура обработки события **OnClick** кнопки может быть записана в виде:



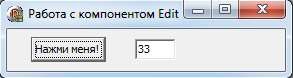
Однако остается вопрос, где описывать данную переменную **i**. Если сделать это внутри данной процедуры, то также необходимо осуществлять обнуление переменной, а это приведет к получению одного и того результата, равного единице. Следовательно, переменная **i** должна быть глобальной переменной в модуле, а ее начальная инициализация должна происходить в процедуре, которая выполняется всего один раз, и всего один раз происходит это событие. Таким событием является создание формы **OnCreat**, данное событие произойдет один раз и процедура **FormCreate(Sender:TObject)** будет вызвана всего один раз.

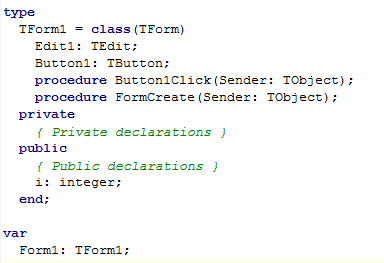
5.Перейдите в код программы (на клавиатуре нажмите клавишу **F12**). Введите в раздел **VAR** глобальную переменную **i** типа **integer**, в которой будет храниться последний нажатый символ.

6.Выделите форму, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnCreate** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре формы **Form1**, напишите следующий программный код:



7.Сохраните изменения и запустите проект. Протестируйте его работу.



При возникновении необходимости сделать данную переменную **i** общедоступной, можно поместить описание переменной в интерфейсной части модуля после служебного слова **public**. Именно так, как правило, и поступают. В модуле необходима всего одна переменная — форма, а все остальные описываются в виде полей. В этом случае описание формы будет иметь вид: 

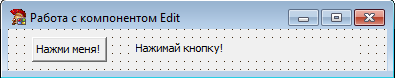
В программе для обращения к переменной **i** необходимо писать ее полное имя **Form1.i**. Однако код процедур обработки событий можно и не переписывать, поскольку процедуры обработки описаны непосредственно в формы, следовательно, данное числовое поле доступно непосредственно.

8. Внесите изменения в программный код в соответствии с рисунком. Сохраните проект и протестируйте работу приложения.

9.Данную программу можно легко модифицировать так, чтобы после определенного количества нажатий появлялось некоторое сообщение или кнопка блокировалась, или приложение автоматически закрывалось. Результат можно отображать не только посредством компонента **Edit**, но и через не редактируемый текст, т. е. компонент **Label**, что в данном случае является более естественным.

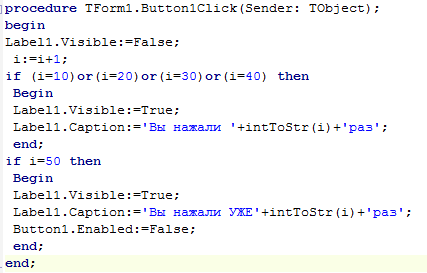
Свойству **Visible** компонента **Label** присваиваем **False**, т. е. при открытии формы надпись отражаться не будет. Затем, как и ранее, при нажатии на кнопку переменная **i** увеличивается на **1**. Когда значение переменной **i** будет равно **10, 20, 30** или **40** компонент **Label** становится видимым, а свойству **Caption** надписи присваиваем значение «**Вы нажали i раз**». При следующем нажатии надпись становится невидима. Когда **i** станет равной **50**, кнопку необходимо сделать неактивной, для чего необходимо изменить значение свойства **Enabled** с **True** — включено на **False** — выключено.

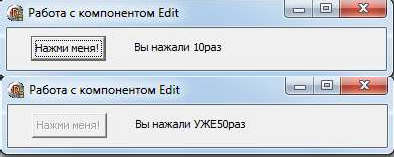
10.Разместите на форме компонент **Label**. Задайте свойство **Caption** равным «**Нажимай кнопку!**». Удалите с формы компонент **Edit1**.



11.Перейдите в окно редактирования кода и внесите изменения в код процедуры обработки события

**OnClick** кнопки.



12.Сохраните проект и протестируйте работу приложения. 

##### Задание 5.

Разработать программу, которая считывает строку под определенным номером и помещает её в текстовое поле.

##### Методические указания по выполнению задания:

1.Создайте в папке своей группы новую папку и назовите её **Memo1**. Сохраните проект в созданную ранее папку, выполнив команду **File – Save Project as…**

2.Для ввода или вывода нескольких строк могут использоваться компоненты **Memo** со страницы **Standard** и **RichEdit** со страницы **Win32** (полный текстовый редактор для **RTF**-файлов). Многие свойства у данных компонентов аналогичны свойствам компонента **Edit**, однако для возможности доступа к строкам вместо свойства **Text** имеется свойство **Lines**, при выборе которого во время проектирования задается начальное значение строк.

Для доступа к строкам во время выполнения программы также используется свойство **Lines** класса **TString**. Подробнее остановимся на классе **TString**, с которым в последствии мы будем еще встречаться. А именно, этот класс обладает свойствами: **Count** — целочисленное свойство, определяющее количество элементов в списке (в данном случае это будет количество строк в компоненте **Memo**), **Text** — свойство, содержащее все строки списка, **String[Index:Integer]** — свойство, определяющее строку с номером **Index**. Учитывая, что нумерация строк начинается с 0 и свойство **String** является свойством по умолчанию, можно утверждать, что свойства **Memo1.Lines.String[3] и Memo1.Lines[3]** эквивалентны и указывают на четвертую строку в компоненте **Memo1**.

Класс **TString** обладает также рядом методов, среди которых отметим следующие: **Add(St:String):integer** добавляет строку **St** и возвращает номер этой строки; **Delete(Index:Integer)** удаляет строку с номером **Index; Insert (Index:Integer, St:String)** вставляет строку с номером **Index, Clear** полностью уничтожает все содержимое компонента.

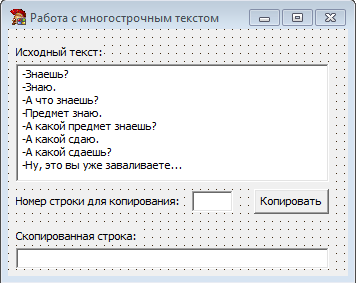
Все содержимое компонента можно записать в файл с помощью метода **SaveToFile** или прочитать из файла посредством методом **LoadFromFile**. Аналогичным образом можно поступить и с потоком, направив в него весь файл, или прочитать файл из потока.

Важным свойством компонентов **Memo** и **RichEdit** является **ScrollBar**, которое определяет, будет ли окно содержать горизонтальные или вертикальные линейки прокрутки.

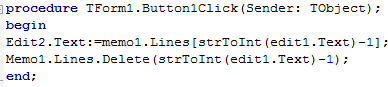
Компонент **RichEdit** обладает всеми характеристиками, присущими компоненту **Memo**, однако имеет богатые возможности для работы с текстовым форматом **RTF**. Данный формат предполагает возможность разбивать текст на параграфы. Для этого существуют специальные свойства: **SelAttributes** определяет атрибуты выделенного фрагмента и **Paragraph** — атрибуты абзаца.

3.Разместите на форме компоненты **Edit,** компонент **Memo,** компонент **Label**, кнопку **Button** вкладки палитры компонентов **Standard**.

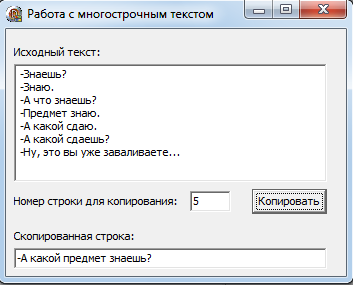
4.Используя **Object Inspector,** задайте значения свойств компонентов формы в соответствии с рисунком.



5.Для реализации решения задачи процедура обработки события **OnClick** кнопки может быть записана в виде:



6.Сохраните проект и протестируйте работу приложения.



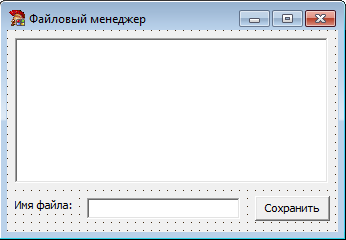
##### Задание 6.

Разработать программу, которая сохраняет текст, набранный в поле **Memo** в файл, имя которого задано в текстовом поле **Edit**.

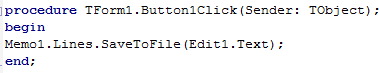
##### Методические указания по выполнению задания:

1.Создайте в папке своей группы новую папку и назовите её **Memo2**. Сохраните проект в созданную ранее папку, выполнив команду **File – Save Project as…**

2.Разместите на форме компонент **Edit,** компонент **Memo,** компонент **Label**, кнопку **Button** вкладки палитры компонентов **Standard**.

3.Используя **Object Inspector,** задайте значения свойств компонентов формы в соответствии с рисунком. 

4.Процедура обработки события **OnClick** кнопки **Button** будет состоять из одной строки, и соответственно, листинг будет иметь следующий вид:



5.Сохраните проект и протестируйте работу приложения.



**Практическая работа №8**

**Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.**

**Цель работы:** сформировать навыки разработки приложений с использованием компонент ввода и отображения чисел, дат и времени в среде визуального программирования Delphi, изучить особенности их использования

**Оборудование, технические и программные средства:** персональный компьютер, интегрированная среда разработчика **Turbo Delphi**.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

##### Задание 1.

Разработать программу, демонстрирующую действие процедур и функций, оперирующих с системными значениями даты и времени.

##### Методические указания по выполнению задания:

1.Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.

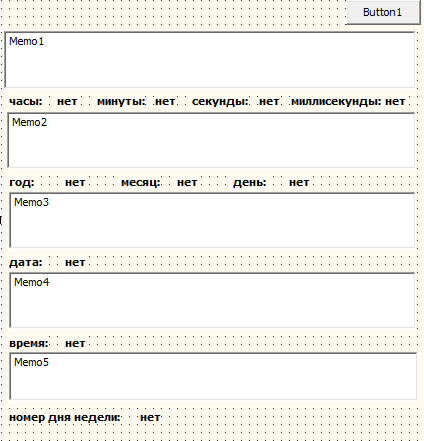
2.Создайте в папке своей группы новую папку и назовите её **Date1**. Сохраните проект в созданную ранее папку, выполнив команду **File – Save Project as…**

3.Задайте для формы следующие свойства:

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| BorderStyle | bsNone |
| ClientHeigth | 441 |
| ClientWidth | 423 |
| Color | clCream |
| Left | 102 |
| Position | poScreenCenter |
| Top | 98 |

4.Разместить на форме компоненты в соответствии с рисунком 2.



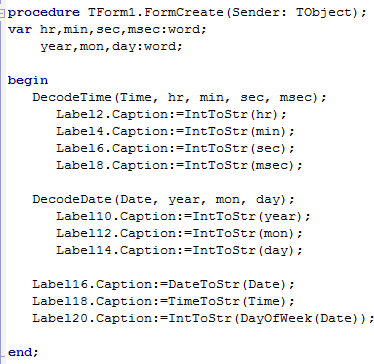
5.Задайте для элементов управления **Label** значение свойства **Caption** в соответствии с рисунком. Для этого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства). 

6.Задайте для элементов управления **Memo** значение свойства **Lines** в соответствии с таблицей. Для этого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства).

|  |  |
| --- | --- |
|  | **Значение** |
| Memo1 | **DecodeTime (Time, hr, min, sec, msec)** процедура, преобразующая текущее значение  времени в формат целых чисел по соответствующим переменным. |
| Memo2 | **DecodeDate (Date, year, mon, day)** процедура, преобразующая текущее значение даты в  формат целых чисел по соответствующим переменным. |
| Memo3 | **DateToStr (Date)** – функция, возвращающая строку с текущим значением даты в формате  day.mon.year |
| Memo4 | **TimeToStr (Time)** – функция, возвращающая строку с текущим значением времени в  формате hr:min:sec |
| Memo5 | **DayOfWeek (Date)** - функция, возвращающая номер дня недели по следующему соответствию: 1 – воскресенье, 2 – понедельник, 3 – вторник, 4 – среда, 5 – четверг, 6 –  пятница, 7 – суббота. |

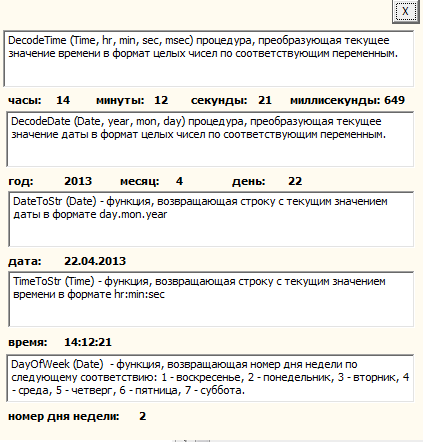
7.Задайте для элемента управления **Button1** значение свойства **Caption** равным **X**.

8.Введите в окне кода процедуру для отображения даты и времени календаря компьютера.



9.Запишите для командной кнопки **Button1** процедуру, которая по щелчку на эту кнопку закрывает форму.

10.Сохраните изменения и запустите проект, убедитесь в его работоспособности.



##### Задание 2.

Разработать приложение, с помощью которого пользователь в одностраничном блокноте выбирает одну из представленных закладок. На рабочем поле блокнота высвечивается соответствующая надпись: год, месяц или день календаря компьютера.

Компонент приложения, содержащий несколько страниц, каждая из которых имеет ярлычок в виде закладки, называется элементом с закладками. Страницы пользователь может выбирать, щелкая по закладкам (корешкам или ярлычкам).

##### Методические указания по выполнению задания:

1.В **Delphi** на странице **Win32** Палитры компонентов расположены две составляющие, работающие на программирование элементов с закладками: **TabControl** – одностраничный блокнот; и **PageControl** – многостраничный блокнот. Данные компоненты являются контейнерами и могут содержать в себе другие элементы или группы.

* + **TabControl** имеет несколько стилей отображения (свойство **Style**): **tsTabs** – стандартные закладки объемного вида;
  + **tsButtons** – закладки в виде кнопок;
  + **tsFlatButtons** – закладки в виде плоских кнопок. Свойство **Tabs** задает число и названия закладок блокнота.

Свойство **TabIndex** указывает текущую закладку в массиве **Tabs**. Программист может использовать данное свойство для переключения на нужную закладку блокнота. Нумерация начинается с 0. Если значение **TabIndex** равно – 1, то ни одна закладка не выбрана.

Свойство **HotTrack** задает подсвечивание названия закладки при наведении на неё курсора мыши.

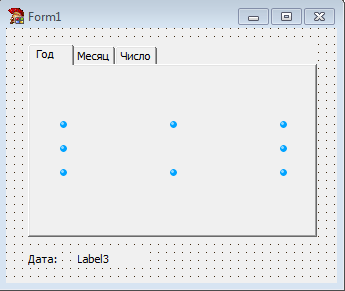
2.Создайте новый проект.

3.Создайте в папке своей группы новую папку и назовите её **Date2**. Сохраните проект в созданную ранее папку, выполнив команду **File – Save Project as…**

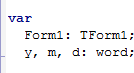
4.Разместите на форме компоненты **Label**, компонент **TabControl**. В **Label1** будет размещаться надпись блокнота, в **Label2**, **Label3** отображаются текущая дата календаря компьютера.

5.Задайте для элементов управления значение свойства **Caption** в соответствии с рисунком. Для этого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства).

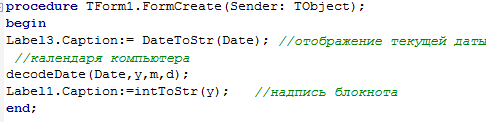
6.Выделите элемент управления **TabControl1**, в **Object Inspector** перейти на страницу **Properties** (свойства) и найти свойство **Tabs**, щелкнуть на кнопке . В открывшемся окне **String List Editor** введите 3 строки со значениями год, месяц и число соответственно.



7.Перейдите в окно редактирования кода и опишите в разделе **VAR** переменные **y, m, d** типа **word** как глобальные переменные модуля для хранения значений даты календаря компьютера.



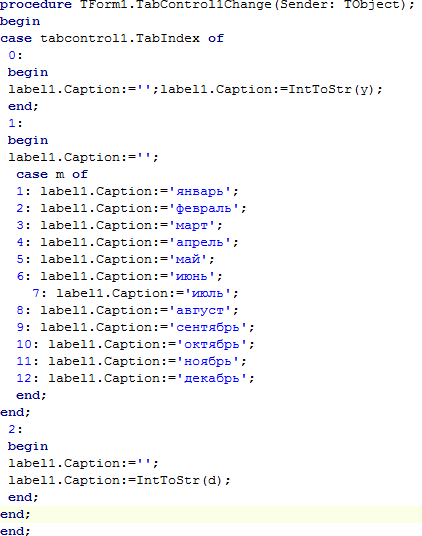
8.Введите в окне редактора кода процедуру для отображения значений даты календаря компьютера:



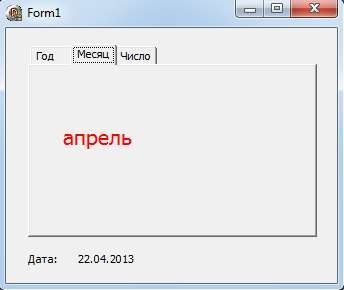
9.Выделите элемент управления **TabControl1,** в **Object Inspector** перейдите на страницу **Events**

(события) и найти событие **OnChange.**

10.Введите в окне редактора кода процедуру для работы с элементом управления **TabControl1**.

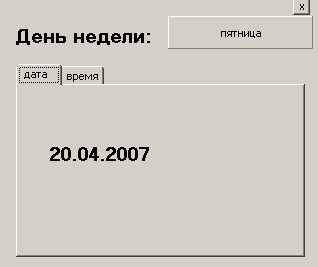


11.Сохраните изменения и запустите проект, убедитесь в его работоспособности.



##### Контрольное задание .

1. Разработайте приложение, с помощью которого пользователь в одностраничном блокноте выбирает одну из представленных вкладок. На рабочем поле блокнота высвечивается соответствующая надпись: дата, время календаря компьютера. На панели отображается день недели.



1. Измените, проект так, чтобы каждый день недели был раскрашен в разный цвет.

**Практическая работа №9**

**События компонентов (элементов управления), их сущность и назначение. Создание процедур на основе событий.**

* **ЦЕЛЬ РАБОТЫ**

Изучить визуальные компоненты оконных приложений и их события*.*

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

* **ОБЕСПЕЧЕННОСТЬ ЗАНЯТИЯ**

Оборудование лаборатории и рабочих мест лаборатории:

компьютеры, принтер, сканер, проектор, программное обеспечение общего и профессионального назначения, комплект учебно-методической документации.

Реализация профессионального модуля предполагает обязательную учебную практику.

Оборудование и технологическое оснащение рабочих мест:

1.Комплект ТС компьютера IBM-PC

2.Методические указания для выполнения практических работ

4. Microsoft Visio.

5. Microsoft Visual Studio.

6.Microsoft Office.

* **ЛИТЕРАТУРА**

Хейлсберг А. Язык программирования C#. / Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Классика Computers Science. 4-е изд. — СПб.: Питер, 2012. — 784 с.

* **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.**

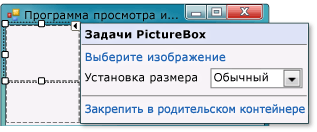
Откройте предыдущий проект.

Перейдите на вкладку "Панель элементов" (находящуюся в левой части интегрированной среды разработки Visual Studio) и разверните группу**Стандартные элементы управления**. В результате этого действия отображается большая часть стандартных элементов управления, которые можно увидеть в формах.

Выберите элемент управления TableLayoutPanel в форме. Чтобы подтвердить, что элемент TableLayoutPanel выбран, убедитесь, что его имя отображается в раскрывающемся списке в верхней части окна **Свойства**. Выбирать элементы управления в форме можно также с помощью раскрывающегося списка в верхней части окна **Свойства**. Выбирать элементы управления таким образом зачастую проще, чем выбрать крошечный элемент управления с помощью мыши.

Дважды щелкните элемент **PictureBox**, чтобы добавить в форму элемент управления PictureBox. Поскольку элемент управления TableLayoutPanel закреплен так, чтобы заполнять собой форму, интегрированная среда разработки добавляет элемент управления PictureBox в первую пустую ячейку (левый верхний угол).

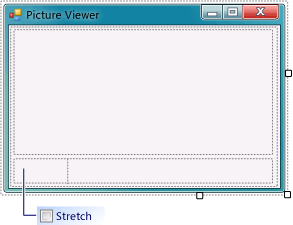
Щелкните новый элемент управления PictureBox, чтобы выбрать его, а затем щелкните черный треугольник на новом элементе управления PictureBox, чтобы отобразить его список задач, как показано на следующем рисунке.

  
Задачи элемента управления PictureBox

Выберите ссылку **Закрепить в родительском контейнере**. В результате этого действия у элемента управления PictureBox свойство **Dock** принимает значение**Fill**. Чтобы это увидеть, выберите элемент управления PictureBox, перейдите к окну **Свойства** и убедитесь, что свойство **Dock** имеет значение **Fill**.

Сделайте так, чтобы элемент управления PictureBox занимал два столбца с помощью его свойства **ColumnSpan**. Выберите элемент управления PictureBox и установите для его свойства **ColumnSpan** значение **2**. Также необходимо, чтобы когда элемент управления PictureBox был пустым, отображалась пустая рамка. Установите для его свойства **BorderStyle** значение **Fixed3D**.

Выберите элемент управления TableLayoutPanel в форме и добавьте в форму элемент управления **CheckBox**. Двойным щелчком выберите элемент **CheckBox**на панели элементов, чтобы добавить новый элемент управления CheckBox в следующую свободную ячейку таблицы. Поскольку элемент управления PictureBox занимает первые две ячейки в TableLayoutPanel, элемент управления CheckBox добавляется в нижнюю левую ячейку. Выберите свойство **Text** и введите слово Stretch, как показано на следующем рисунке.

Элемент управления TextBox со свойством Stretch

Выберите в форме элемент управления TableLayoutPanel, а затем на панели элементов перейдите к группе **Контейнеры** (из которой был взят элемент управления TableLayoutPanel) и дважды щелкните элемент управления**FlowLayoutPanel**, чтобы добавить новый элемент управления в последнюю ячейку в элементе управления PictureBox (справа внизу). Затем закрепите FlowLayoutPanel в TableLayoutPanel (выбрав **Закрепить в родительском контейнере** в списке задач FlowLayoutPanel, раскрываемом с помощью черного треугольника, или установив свойству **Dock** FlowLayoutPanel значение **Fill**).

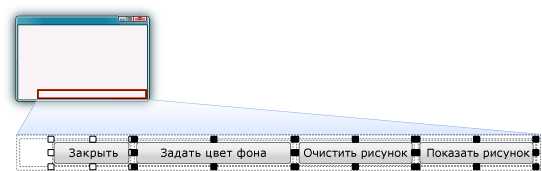
Выберите только что добавленный элемент управления FlowLayoutPanel. Перейдите к группе **Стандартные элементы управления** на панели элементов и двойным щелчком выберите элемент **Кнопка**, чтобы добавить в элемент управления FlowLayoutPanel кнопку с именем **button1**. Чтобы добавить другую кнопку, повторите это действие. Среда интегрированной разработки определяет, что уже существует кнопка с именем **button1** и называет следующую кнопку как**button2**.

Обычно другие кнопки добавляются при помощи панели элементов. На этот раз выберите **button2**, затем в строке меню выберите **Правка**, **Копировать** (или нажмите сочетание клавиш CTRL+C). В строке меню выберите **Правка**, **Вставить**(или нажмите сочетание клавиш CTRL+V), чтобы вставить копию кнопки. Повторите вставку еще раз. Интегрированная среда разработки добавила кнопки**button3** и **button4** в FlowLayoutPanel.

Выберите первую кнопку и установите для ее свойства **Text** значение "Показать рисунок". Затем установите для свойства **Text** следующих трех кнопок значения "Очистить рисунок", "Установить цвет фона" и "Закрыть".

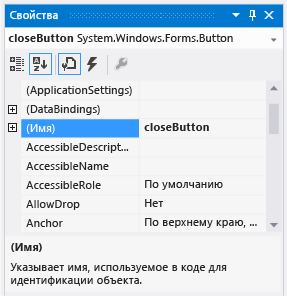
Следующий шаг — задать размер кнопок и разместить их таким образом, чтобы они были выровнены по правой стороне панели. Выберите элемент управления FlowLayoutPanel и обратите внимание на его свойство **FlowDirection**. Измените его значение на **RightToLeft**. После этого действия кнопки должны сами выровняться по правой стороне ячейки и изменить свой порядок таким образом, чтобы кнопка **Показать рисунок** располагалась с правой стороны.

Щелкните кнопку **Закрыть**, чтобы выбрать ее. Удерживая клавишу CTRL, щелкните три другие кнопки, чтобы все они были выбраны. При выделенных кнопках перейдите к окну **Свойства** и прокрутите его вверх до свойства **AutoSize**. Это свойство указывает кнопке автоматически изменять свой размер так, чтобы весь текст мог разместиться на ней. Задайте значение **true**. Кнопки теперь должны иметь соответствующий размер и быть расположены в правильном порядке. (пока выделены все четыре кнопки, можно одновременно изменить все четыре свойства **AutoSize**). На следующем рисунке показаны эти четыре кнопки.

  
Программа просмотра изображений с четырьмя кнопками

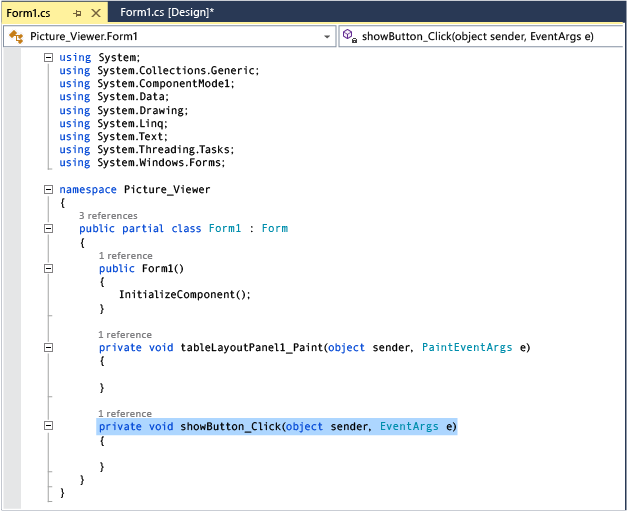
Теперь снова запустите программу, чтобы увидеть обновленную компоновку формы. При нажатии кнопок и установке флажка пока ничего не происходит, однако вскоре все заработает.

В форме нажмите кнопку **Закрыть**. (Если все еще выделены все кнопки, для отмены выделения нажмите клавишу ESC.) Прокрутите содержимое окна**Свойства**, пока не появиться свойство **(Name)**. Свойство **(Name)** расположено в верхней части, когда свойства расположены в алфавитном порядке. Измените имя на **closeButton**, как показано на рисунке ниже.

Окно "Свойства" с именем closeButton

Переименуйте другие три кнопки как**backgroundButton**, **clearButton**, **showButton**. Имена можно проверить в раскрывающемся списке селектора элементов управления в окне **Свойства**. Отобразятся новые имена кнопок.

Двойным щелчком нажмите кнопку **Показать рисунок** в форме. В качестве альтернативы можно нажать кнопку **Показать рисунок** в форме, а затем нажать клавишу ВВОД. При этом интегрированная среда разработки открывает в главном окне дополнительную вкладку, которая называется **Form1.cs** (**Form1.vb**, если используется Visual Basic). На этой вкладке отображается файл кода для формы, как показано на следующем рисунке.

  
Вкладка Form1.cs с кодом Visual C#

Обратите внимание на эту часть кода. (Откройте вкладку **VB** ниже, если используется Visual Basic для просмотра Visual Basic-версии кода.)

C#

[VB](https://infourok.ru/go.html?href=%23code-snippet-1)

private void showButton\_Click(object sender, EventArgs e)

{

}

Вы видите код с именем showButton\_Click(). Интегрированная среда разработки добавила его в код формы при открытии файла кода для кнопки **showButton**. Во время разработки при открытии файла кода для элемента управления в форме для элемента управления создается код, если он еще не существует. Этот код, известный как *метод*, выполняется при запуске программы и использовании этого элемента управления (в данном случае — кнопка **Показать рисунок**).

Снова выберите вкладку конструктора Windows Forms (**Form1.cs [Design]** в Visual C#, **Form1.vb [Design]** в Visual Basic), а затем откройте файл кода для кнопки**Очистить рисунок**, чтобы создать метод для нее в коде формы. Повторите это действие для двух оставшихся кнопок. Каждый раз при этом действии среда интегрированной разработки добавляет в файл кода формы новый метод.

Чтобы добавить еще один метод, откройте файл кода для элемента управления CheckBox в конструкторе Windows Forms, чтобы интегрированная среда разработки создала метод checkBox1\_CheckedChanged(). Этот метод вызывается каждый раз, когда пользователь устанавливает или снимает флажок.

Ниже показан новый код, который представлен в редакторе кода.

C#

[VB](https://infourok.ru/go.html?href=%23code-snippet-2)

private void clearButton\_Click(object sender, EventArgs e)

{

}

private void backgroundButton\_Click(object sender, EventArgs e)

{

}

private void closeButton\_Click(object sender, EventArgs e)

{

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

}

Пять методов, которые были добавлены, называются *обработчики событий*, так как программа вызывает их каждый раз, когда происходит событие (например, пользователь нажимает кнопку или устанавливает флажок).

При просмотре кода для элемента управления в интегрированной среде разработки во время разработки Visual Studio добавляет метод обработчика событий для элемента управления, если он не существует. Например, при двойном щелчке по кнопке интегрированная среда разработки добавляет обработчик события Click, который вызывается каждый раз, когда пользователь нажимает кнопку. Если дважды щелкнуть флажок, интегрированная среда разработки добавляет обработчик события CheckedChanged, который вызывается каждый раз, когда пользователь устанавливает или снимает флажок.

После добавления обработчика событий для элемента управления к нему можно вернуться в любой момент из конструктора Windows Forms с помощью двойного щелчка по элементу управления или путем выбора пунктов **Вид**, **Код** в строке меню.

Имена являются важными при выполнении построения программы, и методы (включая обработчики событий) могут иметь любые имена, которые нужны. При добавлении обработчика событий с помощью интегрированной среды разработки она создает имя на основе имени элемента управления и обрабатываемого события. Например, событие Click для кнопки с именем**showButton** вызывает метод обработчика событий showButton\_Click(). Также обычно после имени метода добавляются открывающая и закрывающая круглые скобки () для индикации того, какие методы рассматриваются. Если вы примете решение изменить имя переменной кода, щелкните правой кнопкой мыши по переменной в коде, а затем выберите команду **Рефакторинг**, **Переименовать**. Все экземпляры этой переменной в коде будут переименованы.

* **ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

**Практическая работа №10**

**Создание проекта с использованием кнопочных компонентов.**

**Цель работы:** Обобщить знания по управляющим элементам ИСР Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Теоретические сведения**

1. Компонент RadioButton (вкладка Standard)- позволяет выбрать единственное значение из определенного множества значений, представленного группой переключателей.

Таблица. Свойства компонента

|  |  |
| --- | --- |
| Свойство | Описание |
| Caption | Текст, который находится справа от кнопки |
| Checked | Состояние, внешний вид кнопки: если кнопка выбрана, Checked=True; если  кнопка не выбрана, то Checked=False |
| Enabled | Признак доступности флажка. Если значение свойства равно false, то флажок  не доступен. |
| Font | Шрифт, используемый для отображения поясняющего текста |
| Left | Расстояние от левой границы флажка до левой границы формы |
| Top | Расстояние от верхней границы флажка до верхней границы формы |
| Height | Высота поля вывода поясняющего текста |
| Width | Ширина поля вывода поясняющего текста |

1. Компонент CheckBox (вкладка Standard) - отличается от переключателя тем, что в группе флажков одновременно можно установить флажки в любой комбинации (в том числе могут быть установлены или сброшены все флажки)

Таблица. Свойства компонента

|  |  |
| --- | --- |
| Свойство | Описание |
| Caption | Текст , который находится справа от флажка |
| Checked | Состояние, внешний вид флажка: если флажок установлен (в квадратике есть  «галочка», Checked=True; если флажок сброшен (нет «галочки»), то Checked=False |
| State | Состояние флажка. В отличие от свойства Checked позволяет различать установленное, сброшенное и промежуточное состояния. Состояние флажка определяет одна из констант: cbChecked (установлен); cbGrayed (  неопределенное состояние); cbUnChecked (сброшен). |
| AllowGrayed | Свойство определяет, может ли флажок быть в промежуточном состоянии: если AllowGrayed=False,то флажок может быть только установленном или сброшенным; если AllowGrayed=True, то допустимо промежуточное  состояние. |
| Enabled | Признак доступности флажка. Если значение свойства равно false, то флажок  не доступен. |
| Font | Шрифт, используемый для отображения поясняющего текста |
| Left | Расстояние от левой границы флажка до левой границы формы |
| Top | Расстояние от верхней границы флажка до верхней границы формы |
| Height | Высота поля вывода поясняющего текста |
| Width | Ширина поля вывода поясняющего текста |

1. Компонент ListBox (вкладка Standard)- простой список представляет собой прямоугольную область, в которой располагаются его строковые элементы.

Таблица. Свойства компонента

|  |  |
| --- | --- |
| Свойство | Описание |
| Items | Элементы списка- массив строк |
| Count | Количество элементов списка |
| Sorted | Признак необходимости автоматической сортировки (ture) списка после  добавления очередного элемента |
| ItemIndex | Номер выбранного элемента (элементы списка нумеруются с нуля). Если в  списке ни один из элементов не выбран, то значение свойства равно «-1» |
| MultiSelect | Разрешает(true) или запрещает (false) выбирать несколько элементов |
| SelCount | Число выделенных строк списка |
| Columns | Определяет количество колонок элементов в списке |
| TopIndex | Индекс первого видимого в окне элемента |
| Left | Расстояние от левой границы поля вывода до левой границы формы |
| Top | Расстояние от верхней границы поля вывода до верхней границы формы |
| Height | Высота поля вывода |
| Width | Ширина поля вывода |
| Font | Шрифт, используемый для отображения выводимого текста. |
| Enabled | Используется для ограничения возможности изменить текст в поле редактирования. Если значение свойства равно false, то текст в поле  редактирования изменить нельзя. |
| Visible | Позволяет скрыть текст (false) или сделать его видимым (True) |

1. Компонент ComboBox (вкладка Standard)- комбинированный список представляет собой вариант простого писка с присоединенным полем, в котором отображается выбранный элемент списка. Это же поле может использоваться для ввода новых элементов или для быстрого поиска элемента по начальным символам.

|  |  |
| --- | --- |
| Свойство | Описание |
| Text | Текст, находящийся в поле- редактирования |
| Items | Элементы списка- массив строк |
| Count | Количество элементов списка |
| Sorted | Признак необходимости автоматической сортировки (ture) списка после  добавления очередного элемента |
| ItemIndex | Номер выбранного элемента (элементы списка нумеруются с нуля). Если в  списке ни один из элементов не выбран, то значение свойства равно «-1» |
| DropDown  Count | Количество отображаемых элементов в раскрытом списке. |
| DroppedDo  wn | Определяет, раскрыт ли в данный момент список |
| Left | Расстояние от левой границы поля вывода до левой границы формы |
| Top | Расстояние от верхней границы поля вывода до верхней границы формы |
| Height | Высота поля вывода |
| Width | Ширина поля вывода |
| Font | Шрифт, используемый для отображения выводимого текста. |
| Enabled | Используется для ограничения возможности изменить текст в поле  редактирования. Если значение свойства равно false, то текст в поле редактирования изменить нельзя. |
| Visible | Позволяет скрыть текст (false) или сделать его видимым (True) |

Задание. Создать приложение в соответствии с вариантом. При разработке проекта использовать компоненты ListBox, ComboBox, RadioButton, CheckBox.

|  |  |
| --- | --- |
|  | Задание |
| 1 | Дана строка. Выполнить следующие действия:   * заменить ‘а’ на ‘\*’ * определить количество гласных букв |
| 2 | Дано число а. Определить   * площадь круга * площадь равностороннего треугольника |
| 3 | Даны два числа a и b. Выполнить   * целый остаток деления a на b * возвести число а в степень b ab |
| 4 | Дана строка. Выполнить следующие действия:   * подсчитать количество вхождений в него символов ‘+’ и ‘-’ * заменить пробелы на тире |
| 5 | Даны целые числа от а до b. (a<=b). Найти:   * произведение чисел * среднее арифметическое нечетных чисел |
| 6 | Дано число а. Определить   * площадь равностороннего треугольника * высоту равностороннего треугольника |
| 7 | Дана строка. Выполнить следующие действия:   * заменить ‘м’ на ‘н’ * подсчитать количество символов ‘\*’ |
| 8 | Дано число а. Определить   * длину окружности * периметр квадрата |
| 9 | Даны два числа a и b. Найти   * площадь прямоугольника * площадь равностороннего треугольника |
| 10 | Даны три сопротивления R1, R2 и R3. Найти сопротивление соединения, если:   * сопротивления соединены параллельно * сопротивления соединены последовательно |
| 11 | Дана строка. Найти долю в процентах:   * цифр * знаков препинаний |
| 12 | Дано число а- длина стороны равностороннего треугольника . Определить:   * радиус вписанной окружности r * радиус описанной окружности R |
| 13 | Дана строка. Выполнить следующие действия:   * определить каких букв больше ‘м’ или ‘н’ * определить число вхождений в него буквосочетаний ‘ро’ |
| 14 | Даны целые числа от а до b. (a<=b). Найти:   * произведение четных чисел * среднее арифметическое чисел |
| 15 | Дано число а. Определить   * площадь квадрата * площадь равностороннего треугольника |
|  | Задание |
| 16 | Дана строка. Выполнить следующие действия:   * найти количество гласных букв * найти количество согласных букв |
| 17 | Дана строка. Выполнить следующие действия:   * найти количество глухих согласных букв * найти количество звонких согласных букв |
| 18 | Дано число а- длина катетов прямоугольного равнобедренного треугольника . Определить:   * гипотенуза b * площадь S |
| 19 | Дана строка. Выполнить следующие действия:   * заменить ‘о’ на ‘=’ * определить количество согласных букв |
| 20 | Дана строка. Выполнить следующие действия:   * долю в процентах знаков препинания * заменить все знаки препинания на пробел |
| 21 | Дано число а. Определить   * площадь равностороннего треугольника * высоту равностороннего треугольника |
| 22 | Дана строка. Найти процентное отношение:   * строчных и прописных букв * гласных и согласных букв |
| 23 | Даны два числа a и b. Выполнить   * умножение деление a\*b * извлечь корень степени b из числа а |
| 24 | Дана строка. Выполнить следующие действия:   * заменить ‘а’ на ‘\*’ * определить количество гласных букв |
| 25 | Дана строка. Выполнить следующие действия:   * определить долю в процентах букв ‘Т’ * заменить ‘т’ на ‘Т’ |
| 26 | Дано число а- длина стороны равностороннего треугольника . Определить:   * высоту h * площадь S |
| 27 | Дана строка. Найти долю в процентах:   * букв от ‘а’ до ‘й’ * букв от ‘к’ до ‘я’ |
| 28 | Даны два числа a и b. Выполнить   * деление нацело а на b * возвести число а в степень b ab |
| 29 | Даны три сопротивления R1, R2 и R3. Найти сопротивление соединения, если:   * сопротивления соединены параллельно * сопротивления соединены последовательно |
| 30 | Дана строка. Выполнить следующие действия:   * заменить ‘а’ на ‘о’ * заменить все вхождения буквосочетаний ‘да’ на ‘не’ |

**Контрольные вопросы:**

1. События и процедура обработки событий. Определения, применение

2. Компоненты Button и BitBtn. Назначение, свойства, события и методы

3. Компонент CheckBox. Назначение, свойства, события и методы

4. Компонент RadioButton. Назначение, свойства, события и методы

5. Компонент RadioGroup. Назначение, свойства, события и методы

6. Компонент ListBox. Назначение, свойства, события и методы

7.Компонент ComboBox. Назначение, свойства, события и методы

**Практическая работа №11**

**Создание проекта с использованием компонентов стандартных диалогов и системы меню.  
Цель работы:** сформировать умения использования компонентов стандартных диалогов и системы меню, изучить основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

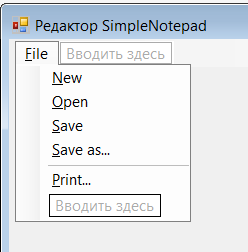
**Время выполнения:** 4 ч

##### Задание:

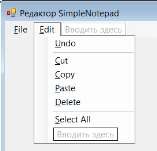
Разработать приложение **SimpleNotepad**, представляющее собой простейший текстовый редактор. Использовать в приложении компонент для работы с меню и стандартные окна диалога.

##### Методические указания по выполнению задания:

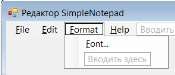
1. Запустите среду программирования **Visual Studio**. Создайте новое **Приложение Windows Forms**. Имя проекта и приложения **SimpleNotepad**.
2. Измените свойство **Name** формы на **SimpleNotepadForm.**
3. Перейдите в окно **Обозреватель решений** и переименуйте файл **Form1.cs** на **SimpleNotepadForm.cs.**
4. Измените размеры окно формы. Измените заголовок окна на **Редактор SimpleNotepad**.
5. Добавьте в окно приложения элемент управления **MenuStrip**.
6. В поле **Вводить здесь** меню введите строку **&File**. В результате в окне приложения появится меню **File**. Обратите внимание, что первая буква в названии подчеркнута. Это получилось потому, что перед ней стоит префикс &. Этим префиксом отмечается буква, предназначенная для ускоренного выбора меню при помощи клавиатуры.
7. Создайте меню **File** в соответствии с рисунком. Для вставки разделительной черты необходимо вызвать контекстное меню и выбрать команду **Separator**.



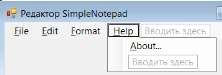
1. Создайте меню **Edit** в соответствии с рисунком. В меню **Edit** реализованы стандартные функции приложения **Блокнот**.

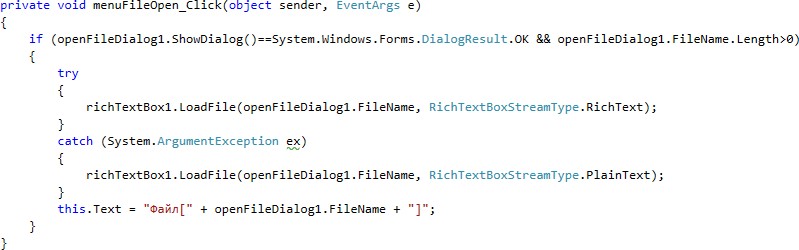


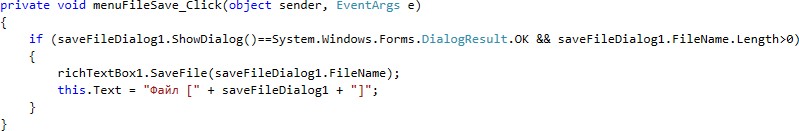
1. Меню **Format** состоит только из одной строки **Font**, с помощью которой пользователь может изменить шрифт текста.



1. Меню **Help** состоит только из одной строк **About**.



1. Переименуйте идентификаторы меню и строк меню таким образом, чтобы с ними было удобнее работать в программе. Для этого в окне дизайнера формы щелкните правой кнопкой мыши главное меню приложения и затем выберите из контекстного меню строку **Правка DropDownItems**. В окне **Редактор коллекций элементов** отредактируйте имена меню и строк меню, изменив значение свойства **Name** соответствующего элемента. При этом меню верхнего уровня должны называться **menuFile**, **menuEdit**, **menuFormat**, **menuHelp**. Имена строк формируются путем добавления к имени меню текста, отображаемого в строке меню. Например, строка **New** называется **menuFileNew**.
2. Запустите приложение. Убедитесь, что меню корректно отображается, и вы можете выбирать его строки.
3. Редактирование текста в приложении будет выполнять компонент **RichTextBox**. Перетащите компонент **RichTextBox** с панели элементов в окно приложения. Задайте значение свойства **Dock** данного компонента равным **Fill**, для того чтобы он занимал все окно приложения.
4. Создайте обработчики событий, необходимые для выполнения таких функций как создание нового документа, открытие и сохранение документа.
5. Добавьте на форму компонент **OpenFileDialog** с вкладки **Диалоговые окна** панели элементов. Выделите пункт меню **Open...** и дважды щелкните по нему левой кнопкой мыши, для того чтобы добавить обработчик события. В обработчик события добавьте следующий код для отображения диалогового окна открытия файла.
6. Выделите компонент **openFileDialog1**, на панели **Свойства** найдите свойство **Filter** и введите следующую строку **RTF files|\*.rtf|Text files|\*.txt|All files|\*.\***
7. Добавьте на форму компонент **SaveFileDialog** с вкладки **Диалоговые окна** панели элементов. Выделите пункт меню **Save As…** и дважды щелкните по нему левой кнопкой мыши, для того чтобы добавить обработчик события. В обработчик события добавьте следующий код для отображения диалогового окна **Сохранение файла**. Этот код сохраняет текст введенный в элемент управления **richTextBox**, в текстовый файл в указанной папке. Метод **ShowDialog** отображает диалоговое окно, а затем с помощью поля **DialogResult**.**OK** осуществляется проверка нажата ли пользователем кнопка **ОК**.



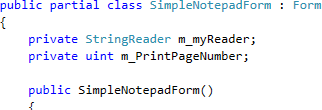
1. Выделите компонент **saveFileDialog1**, на панели **Свойства** найдите свойство **Filter** и введите следующую строку **RTF files|\*.rtf**, найдите свойство **FileName** и задайте значение **doc1.rtf**
2. Добавьте в окно дизайнера форм компоненты **PrintDocumet**, **PrintDialog** вкладки **Печать** панели элементов.



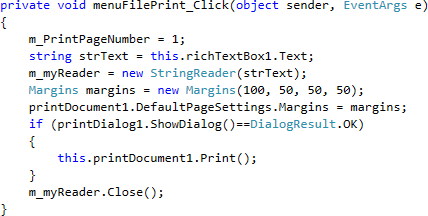
1. Компонент **PrintDocumet** предназначен для вывода данных документа на принтер. Свойства компонента **PrintDocumet** описывают, как именно нужно распечатывать документ. В свойство **DocumentName** данного компонента запишите строку **SimpleNotepad Document**. Эта строка будет идентифицировать документ при отображении состояния очереди печати. Значения остальных свойств оставьте без изменения.
2. С помощью компонента **PrintDialog** приложение выведет на экран стандартное диалоговое окно печати документа. Задайте для данного компонента значение свойства **Document** равным **printDocument1**. Этим обеспечивается связь компонента **printDialog** с компонентом **PrintDocumet**.
3. Для работы с классами, предназначенными для выполнения операций с потоками и печати, добавьте в начало программы следующие строки:



1. Добавьте в класса **SimpleNotepadForm** поля **m\_myReader** (для печати содержимого редактора текста) и **m\_PrintPageNumber** (номер текущей распечатываемой страницы документа).



1. Создайте обработчик события печати документа.



Печать документа будет начинаться с первой страницы, поэтому в поле **m\_PrintPageNumber** записывается значение 1. Далее выполняется чтение текущего содержимого окна редактирования текста в поток **m\_myReader** класса **StringReader**. Далее задаются границы отступов на распечатываемой странице и отображается диалоговое окно печати документа. Если пользователь щелкает в этом окне кнопку **OK**, документ **printDocument1** отправляется на печать методом **Print**. Далее ненужный более поток **m\_myReader** закрывается методом **Close**.

1. На данном этапе приложение еще не в состоянии распечатать документ. Причина этого в том, что приложение пока еще не знает, каким именно образом нужно печатать документ.
2. Чтобы в нашем приложении заработала функция печати, необходимо создать обработчик события **PrintPage**. Для этого нужно дважды щелкнуть левой клавишей мыши значок компонента **printDocument1**. В тело обработчика событий вставьте программный код из текстового файла **Print.txt**. Комментарии в тексте обработчика событий **PrintPage** поясняют назначение отдельных программных строк. Заметим, что для полного понимания действий, выполняемых нашим обработчиком событий, требуется предварительное знакомство с графической подсистемой **Graphics Device Interface Plus (GDI+)**, реализованной компанией **Microsoft** в рамках библиотеки классов **.NET Framework**. Пока же нужно отметить, что приложение распечатывает текст построчно в цикле. После завершения печати всех строк текущей страницы обработчик событий **PrintPage** печатает верхний и нижний колонтитулы, а также рисует горизонтальные линии, отделяющие текст колонтитулов от текста документа.
3. Измените меню **File**, добавив пункт **Exit,** при выборе которого окно редактора текста должно быть закрыто. Добавьте обработчик события для данного пункта меню:



1. Однако при закрытии окно приложения возникает проблема: окно редактора текста будет закрыто и в том случае, если пользователь не сохранил сделанные им изменения. Чтобы решить эту проблему, нам нужно каким-то образом отслеживать наличие изменений в окне редактирования текста. Определите в классе **SimpleNotepadForm** поле **m\_DocumentChanged**, в котором будет храниться флаг, отмечающий изменения, сделанные пользователем в документе. В новом или только что загруженном документе изменений нет, поэтому начальное значение этого флага равно **false**.



1. Создайте обработчик события **richTextBox1\_TextChanged**, выполнив двойной щелчок по компоненту **richTextBox**. Этот обработчик получит управление, как только пользователь внесет любые изменения в содержимое редактируемого документа.

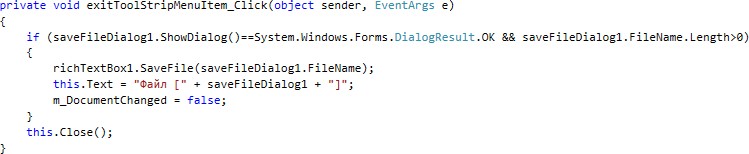


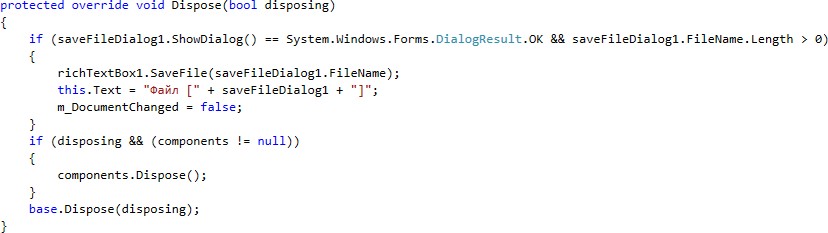
1. Если пользователь редактировал документ, а потом решил создать новый, выбрав из меню **File** строку

**New**, изменения, внесенные в старый документ, могут быть потеряны.

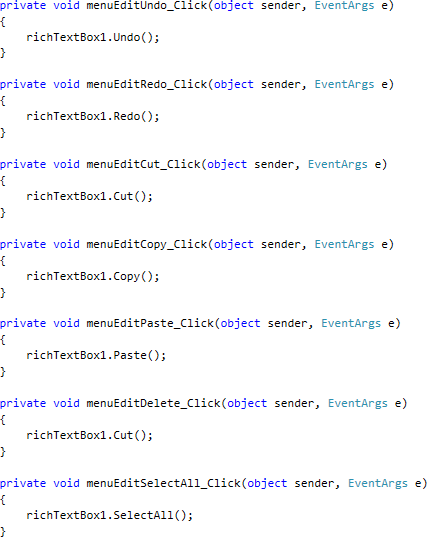
1. Чтобы избежать этого, проверьте флаг **m\_DocumentChanged** перед тем как очищать содержимое редактора текста. Если в редакторе есть не сохраненные изменения, необходимо выполнить сохранение документа. Отредактируйте код обработчика события пункта меню **New** следующим образом:
2. Отредактируйте код обработчика события пунктов меню **Save** и **Save As**, добавив строку кода

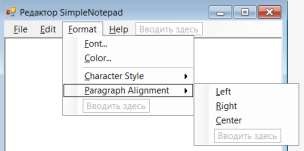
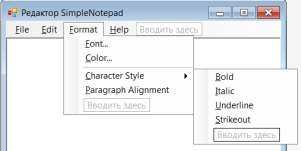
##### m\_DocumentChanged=false

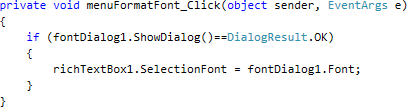
1. Измените обработчик события пункта меню **Exit** следующим образом:
2. Надо выполнить еще одну проверку флага **m\_DocumentChanged** в методе **Dispose**, который вызывается при закрытии окна приложения. Для этого на панели Обозреватель решений выделите **SimpleNotepadForm.Designer.cs**, найдите метод **Dispose** и внесите необходимые изменения.



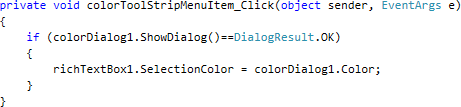
Теперь, когда пользователь попытается закрыть программу с помощью соответствующей кнопки заголовка окна, не сохранив сделанные изменения, на экране появится стандартное диалоговое окно, предлагающее ему сохранить документ.

1. Создайте обработчики событий пунктов меню **Edit**. Реализация данных функций является простой, поскольку элемент управления **RichTextBox** имеет все необходимые методы. Добавьте в пункт меню **Edit** строку **Redo**. Подготовьте обработчики событий для всех строк меню **Edit**.
2. Измените меню **Format**, добавив в него строку **Color**, а также два меню второго уровня – **Character Style** и **Paragraph Alignment**. Измените имена строк меню таким образом, чтобы с ними было легче работать в программе.



37.Добавьте на форму компонент **FontDialog** с вкладки **Диалоговые окна** панели элементов. Этот компонент отображает на экране стандартное диалоговое окно выбора шрифта. Создайте обработчик события пункта меню **Font** и вставьте следующий код: 

После того как пользователь выбрал нужный ему шрифт, обработчик события переписывает этот шрифт из свойства **fontDialog1.Font** в свойство **richTextBox1.SelectionFont**. Свойство **SelectionFont** позволяет изменить шрифт фрагмента текста, выделенного пользователем (или программой) в окне редактирования.

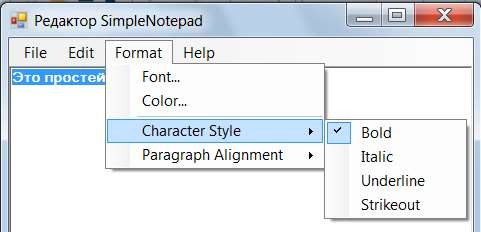
38.Добавьте на форму компонент **ColorDialog** с вкладки **Диалоговые окна** панели элементов. Этот компонент отображает на экране стандартное диалоговое окно выбора шрифта. Создайте обработчик события пункта меню **Color** и вставьте следующий код:

39.Создайте обработчик события для пункта меню **Bold** и вставьте следующий код:

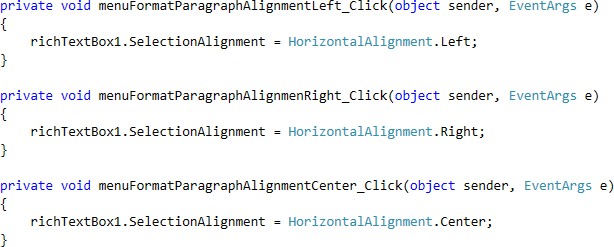


Получив управление данный метод прежде всего, определяет шрифт фрагмента текста, выделенного пользователем, анализируя свойство **richTextBox1.SelectionFont**. Если шрифт определить не удалось, и это свойство содержит значение **null**, программа не делает никаких изменений. В противном случае программа сохраняет текущий шрифт в переменной **currentFont** класса **System.Drawing.Font**. Далее метод проверяет, был ли выделен фрагмент текста жирным шрифтом, анализируя свойство **richTextBox1.SelectionFont.Bold**. Если это свойство содержит значение **true**, то метод снимает выделение, если нет, то устанавливает его. Для снятия выделения программа записывает в переменную **newFontStyle** значение **FontStyle.Regular**, а для установки — значение **FontStyle.Bold**.

40.При выделении фрагмента текста жирным шрифтом в меню одновременно отмечается «галочкой» строка **Bold**. Для этого необходимо создать **CheckMenuFontCharacterStyle**. Для этого перейдите в окно кода и вставьте программный код процедуры из текстового документа **CheckMenu.txt.** Метод **CheckMenuFontCharacterStyle** по очереди проверяет стилевое оформление выделенных фрагментов, отмечая «галочками» соответствующие строки меню **CharacterStyle** или снимая со строк этого меню отметки.

41.Запустите программу и протестируйте её работу. 

42.Аналогично создайте обработчики для пунктов меню **Italic, Underline, Strikeout**.

43.Подготовьте обработчики событий для строк меню **Paragraph Alignment** следующим образом:

Все эти обработчики событий строк меню **Paragraph Alignment** изменяют значение свойства **richTextBox1.SelectionAlignment**, задающего выравнивание параграфа текста, выделенного пользователем (для выделения параграфа с целью изменения выравнивания достаточно установить в него текстовый курсор).

44.Сохраните изменения и протестируйте работу приложения.

**Практическая работа №12**

**Разработка функциональной схемы работы приложения.**

**Цель работы:** ознакомиться с правилами написания технического задания. ГОСТ 19.201-78. Настоящий стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Индивидуальные задания:**

1. Разработка программного комплекса «Автотранспорт».

2. Разработка программного комплекса «Деканат института».

3. Разработка программного комплекса «Обслуживание банкомата».

4. Разработка программного комплекса «Управление гостиницей».

5. Разработка программного комплекса «Выдача кредитов в банке».

6. Разработка программного комплекса «Строительная фирма».

7. Разработка программного комплекса «Управление библиотечным фондом».

8. Разработка программного комплекса «АРМ работника склада»

9. Разработка программного комплекса «АРМ администратора ателье по ремонту оргтехники»

10. Разработка программного комплекса «АРМ администратора автосалона».

11. Разработка программного комплекса «АРМ администратора ресторана».

12. Разработка программного комплекса «АРМ сотрудника ЖЭСа».

13. Разработка программного комплекса «АРМ администратора аэропорта».

14. Разработка программного комплекса «АРМ работника отдела кадров».

15. Разработка программного комплекса «АРМ администратора спорткомплекса».

**Практическая работа №13**

**Разработка оконного приложения с несколькими формами.**

**Цель работы:** Обобщить знания по управляющим элементам ИСР Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Теоретические сведения:**

Любая программа в Delphi имеет как минимум одну форму, которая называется главной. При формировании проекта автоматически создается одна форма. Если нет необходимости в создании нескольких форм, то в этом случае следует только задать главной форме необходимые характеристики и заполнить ее компонентами. Для добавления в проект новой формы следует воспользоваться командой меню Flle\New Form.

Все формы создаются на основе класса TForm. Класс TForm предоставляет возможность изменять поведение и внешний вид формы с помощью ряда свойств, методов и событий.

Таблица. Свойства класса TForm

|  |  |
| --- | --- |
| Название | Описание |
| Active | Определяет состояние формы (True -активная) |
| Borderlcons | Отвечает за наличие кнопок в строке заголовка (biSystemMenu - значок вызова системного меню, biMinimize - кнопка минимизации, biMaximize - кнопка максимизации, biHelp - кнопка вызова справки, которая появляется на экране, если  отключить biMinimize и biMaximize) |
| BorderStyle | Выбирает изображение рамки формы   * bsNone - форма не имеет рамки, ее размер постоянен и отсутствуют кнопки системного меню, а также кнопки минимизации и максимизации окна; * bsSingle - размеры формы неизменны, имеется рамка толщиной 1 пиксель; * bsDialog - стандартная рамка, размеры нельзя изменять; * bsSizeable - размеры формы можно изменять, стандартная рамка; * bsToolwindow - сходный с bsSingle, но строка заголовка меньше; * bsSizeToolwin - сходный с bsSizeable, но заголовок по высоте уменьшен |
| Canvas | Обеспечивает доступ к форме для ее прорисовки |
| ClientHeight | Задает высоту формы |
| ClientWidth | Устанавливает ширину формы |
| HelpFile | Содержит имя файла помощи. По умолчанию - файл помощи приложения |
| Icon | Содержит пиктограмму, которая будет отображаться при минимизации формы |
| Menu | Содержит главное меню формы |
| ModalResult | Используется для закрытия модальных форм |
| Visible | Отвечает за видимость формы (True -видимая, False - невидимая) |
| WindowState | Отвечает за внешний вид формы на экране   * wsNormal - обычные размеры формы; * wsMaximized - форма развернута на весь экран; * wsMinimized - форма свернута в значок на панели задач Windows |

Таблица . Методы класса TForm

|  |  |
| --- | --- |
| Название | Описание |
| Close | Закрывает форму (если она главная, то закрывает приложение) |
| CloseQuery | Определяет, можно закрыть сейчас форму или нет (True - можно) |
| FocusControl | Активизирует указанный компонент, помещенный в форму |
| Next | Делает активной следующую MDI-форму |
| Previous | Активизирует предыдущую MDI-форму |
| Print | Распечатывает форму на принтере |
| Release | Удаляет форму с экрана и из динамической области памяти |
| Show | Вызывает форму в немодальном режиме |
| ShowModal | Делает форму видимой и модальной |

Таблица. События класса TForm

|  |  |
| --- | --- |
| Название | Описание |
| OnActive | Возникает при активизации формы |
| OnClose | Вызывается перед закрытием формы. Параметр Action определяет последствия обработки события: caNone - форма не закрывается, caHide - спрятать форму, caFree  - удаление из приложения и динамической памяти, caMinimize -минимизация формы |
| OnCloseQuery | Вызывается перед закрытием окна. Параметр CanClose определяет возможность  закрытия окна |
| OnCreate | Появляется при создании формы, но перед ее появлением на экране |
| OnDeасtivate | Генерируется, когда форма становится неактивной |
| OnDestroy | Обрабатывает событие, возникающее перед удалением формы из динамической  Памяти |
| OnHelp | Возникает при обращении к справке |
| OnHide | Срабатывает перед удалением формы с экрана |
| OnPaint | Является обработчиком события, возникающего при необходимости перерисовки  Формы |
| OnResize | Реагирует на изменение размеров формы |
| OnShow | Возникает, когда форма становится видимой на экране |

Многостраничный блокнот PageControl.

Компонент PageControl является управляющим элементом, включающим набор из нескольких страниц, размещаемых одна под другой. Каждая страница имеет закладку, которая является неотъемлемой частью данной страницы, в отличие от одностраничного блокнота. Страницы используются для объединения различных управляющих элементов в группы, обеспечивая их компактное размещение и простое переключение между ними.

Многостраничный блокнот является более сложным управляющим элементом, чем компонент TabControl, однако многие свойства этих двух элементов совпадают. Поэтому будут рассмотрены только свойства, специфичные для компонента PageControl.

|  |  |
| --- | --- |
| Свойство | Описание |
| ActivePage  (тип TTabSheet) | определяет название текущей (выбранной) страницы компонента  PageControl. |
| ActivePagelndex (тип Integer) | служит для определения индекса текущей страницы. С помощью свойств  ActivePage и ActivePagelndex можно программно устанавливать новую активную страницу |
| PageCount  (тип Integer) | позволяет определить количество страниц многостраничного блокнота. |
| Pages [Index: Integer] (тип TTabSheet) | представляет собой список всех страниц управляющего элемента PageControl. Используя данное свойство во время выполнения приложения, можно получить доступ к любой странице блокнота по ее  номеру, задаваемому параметром index. |

Добавление и удаление страниц, а также перемещение между страницами компонента PageControl в процессе проектирования приложения осуществляется с помощью вызова контекстного меню (нажатием правой кнопки мыши в поле компонента)и дальнейшего выбора соответствующего пункта меню. Перемещаться между страницами можно также простым нажатием на закладке необходимой страницы. Это возможно, потому что каждая страница (включая закладку) является отдельным независимым объектом.

Индивидуальное задание

1. На главную форму приложения поместить компонент TAnimate. Компонент должен реализовать просмотр пользовательской анимации, путь которому прописывается в свойстве FileName.
2. На главной форме реализовать прослушивание музыкальной композиции при помощи компонента TMediaPlayer.
3. Поместить на форму главное меню, содержащее следующие пункты:
   * задание 1
   * задание 2
   * выход

При выборе пункта меню «Задание 1», необходимо реализовать открытие формы «График» (в модальном режиме). На форме «График» разместить 2 вкладки:

* «Таблица»- содержит таблицу, содержащее протабулированное значение функции y=f(x) на отрезке [a,b]. Количество значений функций равно n. Шаг вычисляется по формуле h=(b-a)/n..
* «График» - содержит график функции y=f(x), начерченный с помощью компонента TChart.

При выборе пункта меню «Задание 2», необходимо открытие формы «Теория» (в немодальном режиме). На форме «Теория» расположить 2 компонента TreeView и WebBrowser. Узлами дерева TreeView являются темы теоретического материала. По щелчку названию темы необходимо осуществить загрузку соответствующей страницы html в компонент WebBrowser.

При выборе пункта меню «Выход», необходимо закрыть приложение.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Интервал | n |
| 1. | Y=2sin(x/3) | [** / 2;** / 2] | 30 |
| 2. | Y=2cos(x/4) | [0;3** / 2] | 40 |
| 3. | Y=|sin(x)+cos(x)| | [0; ** ] | 40 |
| 4. | Y=|sin(x)-cos(x)| | [0; ** ] | 40 |
| 5. | Y=2sin(x)+3cos(x) | [-;] | 50 |
| 6. | Y=sin(x)+cos2(x) | [-;] | 50 |
| 7. | Y=2-cos(x) | [0;3** / 2] | 40 |
| 8. | Y=sin( 2*х* )+cos(x) | [0;2] | 50 |
| 9. | Y=2sin(2x)+1 | [** / 2;** / 2] | 50 |
| 10 | Y=sin(x)+cos(x)-1 | [-;] | 40 |
| 11 | Y= *х*2  2*х* | [0;5] | 40 |
| 12 | Y=10/(1+ *х*2 ) | [-3;3] | 50 |
| 13 | Y=(x-3)/( *х*2  2 ) | [-1;4] | 50 |
| 14 | Y=xcos(2x) | [-;2] | 50 |
| 15 | Y= *x*2*e* *x* | [-1;3] | 40 |
| 16 | Y= ( *х*2  4*х*  4)3 | [-2;4] | 40 |
| 17 | Y=2sin(x)+1 | [-;** / 2] | 40 |
| 18 | Y=|2cos(x)-1| | [0; ] | 50 |
| 19 | Y= *x* 10 | [1;3] | 40 |

|  |  |  |  |
| --- | --- | --- | --- |
| 20 | Y= ln( *x*2 ) | [2;4] | 40 |
| 21 | Y=tg(x)/2 | [** / 2;** / 2] | 45 |
| 22 | Y=sin ( 2*x* ) +cos  ( *x*) | [0; 2] | 50 |
| 23 | Y= *ex*  *e* *x*  2 | [0;2] | 45 |
| 24 | Y=(x-3)/( *x*2 +2) | [-1;4] | 50 |
| 25 | Y=arccos(x)\*(1-  *x*2 ) | [-1;1] | 30 |
| 26 | Y=arcsin(x)\*(1- *x*2 ) | [-1;1] | 30 |
| 27 | Y=2cos(x)+1 | [** / 2;** / 2] | 50 |
| 28 | Y= *x*  *e x* | [-1;3] | 45 |
| 29 | Y=lg(x)+1 | [1;5] | 40 |
| 30 | Y= *ex*  10 | [-1;3] | 40 |

**Контрольные вопросы:**

1.Назначение, свойства, события и методы компонентов, используемых при разработке приложения

2. Компонент Form. Назначение, свойства, события и методы

3. Модальный и немодальный режим открытия окна. Различие, назначение.

**Практическая работа №14**

**Разработка игрового приложения.**

**Цель работы**: получение практических навыков по разработке игровых  программ на языке программирования Visual Basic 6.0.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

## Краткие сведения из теории

Visual Basic 6.0 позволяет разрабатывать игровые программы, включающие графику и мультимедиа. Для представления графических объектов, таких как линии, окружности и др. необходимо также присваивать значения их свойствам (длина, радиус, координаты и т. д.).

Рассмотрим элемент управления SHAPE, который предназначен для рисования различных фигур: эллипс, окружность, прямоугольник и др.

Вид фигуры задается свойством Shape, которое принимает следующие значения.

Shape = 0              вычерчивает прямоугольник

Shape = 1              вычерчивает квадрат

Shape = 2              вычерчивает овал

Shape = 3              вычерчивает окружность

Shape = 4              вычерчивает скругленный прямоугольник

Shape = 5              вычерчивает скругленный квадрат

Важной характеристикой графических объектов является цвет, который можно задавать с помощью функции RGB ( ) или QBColor ( )

RGB ( ) функция строит цвет из red, green и blue цветов с различным насыщением смешивания..

Синтаксис RGB (red, green, blue)

Где:

Red - число из диапазона 0 - 255, представляет величину красной компоненты цвета.

Green - число от 0 - 255, представляет величину зеленой компоненты цвета.

Blue - число от 0 - 255, представляет величину синей компоненты цвета.

Функция QBColor -возвращает  код, соответствующий номеру цвета.

Синтаксис QBColor (color), где: color - целое число от 0 до 15

**Использование файла с картинкой**

Файл с картинкой можно поместить:

-на форму

-на элемент управления Image

 -на элемент управления Picture

Картинку можно поместить  или во время проектирования или в процессе работы программы.

Для вставки картинки в процессе создания формы нужно щелкнуть на свойство Picture и выбрать тип файла. Например .bmp или .gif.

На картинку можно поместить элементы управления.

Для вставки картинки во время работы программы можно написать код

FrmMyForm.Picture = Load Picture ("C:\VB\WORKS\ball.bmp")

Чтобы убрать картинку нужно выполнить команду

FrmMyForm.Picture = Load Picture("")

Картинку можно вставить в элемент управления Picture или Image.

Пример.

ImgMyImage.Picture = Load Picture("C:\…ball.bmp")

Для очистки картины

ImgMyImage.Picture = Load Picture("")

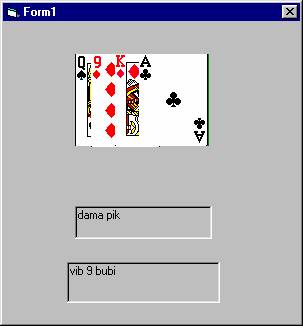
Для изменения размеров картинки установить свойство Stretch = True.

ImgMyImage.Stretch = True

Картинку на элемент управления Image можно вставлять из буфера обмена.Для этого достаточно выделить вставляемый фрагмент картинки в каком либо графическом редакторе, например Paint, скопировать фрагмет в буфер, затем выделить элемент Image на форме и вставить из буфера обмена фрагмент картинки.

Ниже приведен пример программы, в которой использованы картинки с картами, вырезанными через буфер обмена в элемент управления Image.

Внешний вид формы.



Код программы имеет вид.

Private Sub Form\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Dim prov As Boolean

With Image1

prov = (Image1.Left <= X) And (X <= Image4.Left + Image4.Width) And (.Top <= Y) And (Y <= .Top + .Height)

End With

If prov = False Then Label1.Caption = Empty

End Sub

Private Sub Image1\_Click()

Label2.Caption = "vibrana dama pik"

End Sub

Private Sub Image1\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Label1.Caption = "dama pik"

End Sub

Private Sub Image2\_Click()

Label2.Caption = "vib 9 bubi"

End Sub

Private Sub Image2\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Label1.Caption = "9 bubi"

End Sub

Private Sub Image3\_Click()

Label2.Caption = "vib korol bub"

End Sub

Private Sub Image3\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Label1.Caption = "korol bub"

End Sub

Private Sub Image4\_Click()

Label2.Caption = "vib tuz krest"

End Sub

Private Sub Image4\_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Label1.Caption = "tuz krest"

End Sub

# Содержание работы

1.     Создать внешний вид формы программы

2.     Написать программные коды для созданной формы

3.     Запустить и отладить программу

# Содержание отчета

1.     Постановка задачи.

2.     Описание внешнего вида и работы программы.

3.     Инструкция программиста.

4.     Инструкция оператора (пользователя).

5.     Листинг программы.

# Варианты  работ

1.     Игра «Угадай карту». Дано пять карт, четыре короля и дама пик, перевернутые «рубашкой вверх». Цель игры: угадать, где находится дама пик за наименьшее число ходов. Щелчок по карте (ход)  переворачивает ее. Добавить звуковое сопровождение. (3 балла).

2.     Игра «Где яйцо». Даны три корзины. В одной из корзин спрятано яйцо. Нужно угадать в какой корзине яйцо. Щелчком мыши корзина переворачивается. Добавить звуковое сопровождение. (3 балла).

3.     Игра «Крестики-нолики». Предусмотреть два варианта. Игра с компьютером и игра двух игроков. (4 балла).

4.     Игра «Шашки». Игра для двух игроков. Шашки передвигаются с помощью мыши. Шашки удаляются с поля двойным щелчком мыши. (5 баллов).

5.     Игра «Сапер». На поле размером 8x8 случайно размещается N мин. Задача игрока найти все мины не подорвавшись. (6 баллов).

6.     Игра «Пятнадцать». На поле 4Х4 размешаны числа от 1 до 15. Требуется с помощью пустого квадрата расставить их в порядке возрастания. (7 баллов).

7.     Игра «Shifter». На поле 5х5 случайно расставлены красные и зеленые квадраты. Задача собрать заданный узор. Квадраты можно сдвигать влево или вправо, а также вверх или вниз. Образцы узоров можно выбирать. (7 баллов).

8.     Игра «Скачки». На ипподроме расположены четыре лошади с всадниками. после старта они двигаются к финишу. Заранее неизвестно, какая лошадь придет первой. Добавить звуковое сопровождение. (8 баллов).

9.     Игра «Шары». Дано 5 красных и 5 зеленых шаров. Цель игры – оставить противнику один красный шар. За один ход можно забрать или один шар любого цвета, или два шара одного цвета. Предусмотреть два варианта игры: игра с компьютером и игра двух игроков. (8 баллов).

10. Игра «Лабиринт». В лабиринте находятся шарик (колобок) и мешочки с деньгами. Задача колобка собрать все мешочки. Движение колобка задается клавишами со стрелками. (9 баллов).

11. Игра "Пирамида". Имеются три стержня A, B, C и  5 дисков, пронумерованных от 1 до 5 в порядке возрастания их размеров. Сначала все диски надеты на стержень  A от  5 (нижний диск) до 1. Игрок должен перенести все диски с  A  на C, с использованием стержня В, при этом диски можно переносить только по одному, больший диск нельзя ставить на меньший. (9 баллов).

12. Игра «Мозаика». Выбранный рисунок разбивается на N квадратов, перемешанных случайным образом. Перетаскиванием квадратов нужно собрать исходный рисунок. (10 баллов).

# Контрольные вопросы

1.     Как нарисовать окружность на экране?

2.     Как закрасить элемент Shape в нужный цвет?

3.     Назначение функции RGB ( ).

4.     Какие способы задания цвета существуют в Visual Basic 6.0?

5.     На какие объекты можно поместить файл с картинкой?

6.     Как вставить картинки во время работы программы?

7.     Как удалить картинки во время работы программы?

8.     Какой элемент управления предназначен для работы с таблицами?

9.     Как создать массив элементов управления?

10. Назначение свойства Stretch элемента управления Image.

11. Какие расширения имеют графические файлы?

12. Как добавить новую форму в проект?

**Практическая работа №15**

**Создание процедур обработки событий. Компиляция и запуск приложения. Разработка интерфейса приложения.**

**Цель урока:**освоить составление программы на обработку событий.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Задание.**

Создать программу, выполняющую следующие действия:

1. По щелчку мышью на кнопке кнопка либо останавливается, либо двигается.



Рис. 1.

2. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

Описание плана разработки программы

1. Открыть новый проект.

2. Разместить на форме экземпляры компонентов: кнопку Button, таймер Timer. Кнопка включает и выключает таймер, а таймер двигает кнопку.

3. Выполнить следующие действия:

Таблица 1.

Выделенный объект

Вкладка окна Object Inspector

Имя свойства/ имя события

Действие

Forml

Properties

Caption

Установка имени формы "Двигающаяся кнопка"

Timerl

Properties

Enabled

Установить значение свойства Enabled = false

Свойство Enabled определяет, включен или выключен таймер (по умолчанию, он включен).

Interval

Interval = 100

Свойство Interval определяет интервал в миллисекундах между возникновением событий OnTimer (по умолчанию интервал равен 1 секунде).

Events

OnTimer

Button1.Left := Button1.Left - 5; if Button1.Left < 10

then Button1.Left := 100;

Buttonl

Properties

Caption

Установка имени кнопки "Сменить заголовок окна"

Default

Выбрать в раскрывающемся списке значение True

Events

OnClick

Timer1.Enabled := not Timer1.Enabled;

4. Сохраните проект, запустите и протестируйте его.

**Практическая работа №16**

**Тестирование, отладка приложения.**

**Цель работы:** Осуществить тестирование и отладку разработанной ранее конкретной программы на алгоритмическом языке высокого уровня.

**Форма отчета:**

Во время выполнения лабораторной работы необходимо соста­вить набор тестов к разработанной ранее программе и провести ее отладку.

Составленный набор тестов необходимо представить в отчете.

**Время выполнения:** 4 ч

**Теоретические сведения.**

Тестирование - это процесс выполнения программы с целью определения места некорректного ее функционирования. Оно включает преднамеренное конструирование трудных наборов входных данных, создающих наибольшие возможности для отказа программного изделия. Тестирование является основным методом обнаружения ошибок в программе. Результаты тестирования являются исходными данными для отладки.

Отладка программы - это этап ее разработки, на котором устраняются недостатки только что созданной программы.

Если программа правильно ведет себя для солидного набора тестов, нет оснований, утверждать, что в ней нет ошибок. Просто неизвестно, когда она не работает и можно говорить лишь о некотором уровне ее правильности.

Тесты, не способствующие обнаружению ошибок и только подтвер­ждающие корректность функционирования программы, являются не эф­фе­ктивными, т.к. приводят к бесполезным затратам ресурсов и времени.

Тест - это просчитанный вручную или другим способом пример, промежуточные и конечные результаты которого используются для контроля правильности (живучести) программного изделия. Тест состоит из исходных данных и тех значений, которые дол­жны выдать отладочные печати при работе по этому тесту. Эти знач­ения должны быть записаны в точности в том виде, в котором их должна выдать ЭВМ.  Эти значения желательно получить любым путем, но не тем, который реализован в программе, т.к. в последнем случае можно не заметить ошибки в алгоритмизации.

Комплект тестов должен быть таким:

-  чтобы проверить все варианты внешнего эффекта программы и ва­рианты ее внутренней работы алгоритма;

-   чтобы все ветви алгоритма были пройдены, по крайней мере, по од­ному разу.

-   чтобы проконтролировать предельные и вырожденные случаи.

Тестовые данные должны подбираться таким образом, чтобы прог­раммист был в состоянии вычислить правильный результат ещё до на­чала тестирования.

Процесс тестирования программы можно разделить на три этапа:

1. Проверка в нормальных условиях.

2. Проверка в экстремальных условиях.

3. Проверка в исключительных  ситуациях.

Каждый из этих трех этапов проверки должен гарантировать по­лучение верных результатов при правильных входных данных и выдачу сообщений об ошибках при неправильных входных данных.

Проверка в нормальных условиях

Случаи, когда программа должна работать со всеми возможными исходными данными, чрезвычайно редки. Обычно имеют место конкрет­ные ограничения на область изменения данных, в которой программа должна сохранять свою работоспособность. Программа  должна  выда­вать правильные результаты для характерных совокупностей исход­ных данных.

Проверка в экстремальных условиях

Тестовые данные этого этапа включают граничные значения об­ласти изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичные примеры очень боль­шие числа, очень малые числа или отсутствие информации.

Проверка в исключительных ситуациях.

Используются исходные данные, значения которых лежат за пре­делами допустимой области их изменения. Наихудшая ситуация когда программа воспринимает неверные данные как правильные и выдаёт неверный, но правдоподобный результат. Программа должна отвергать любые данные, которые она не в состоянии обрабатывать верно.

### Контрольные вопросы

Тестирование и отладка программы.

1.     Ошибки - их причины появления и последствия.  Классы ошибок

2.     Первичные и вторичные ошибки. Синтаксические и семантические ошибки. Защитное программирование. Программирование без ошибок.

3.     Тестирование ПО. Понятие теста. Комплект тестов.Пример теста.

4.     Методы тестирования. Этапы тестирования. Аксиомы тестирования.

**5.**Отладка ПО. Этапы, методы и инструменты отладки программ.

**Практическая работа №17**

**Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.**

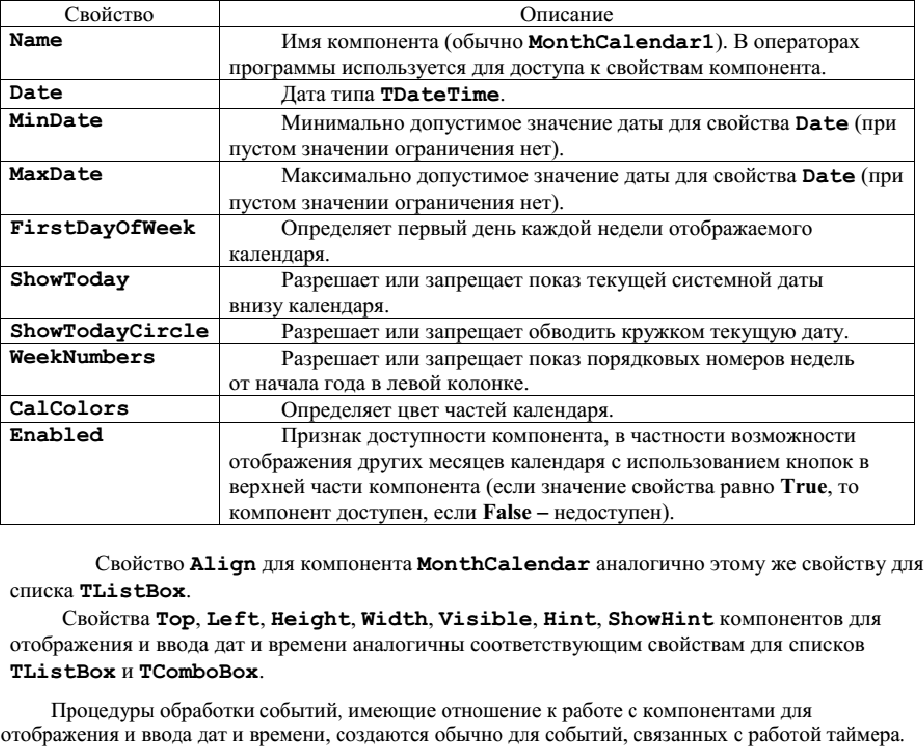
**Цель работы:** Обобщить знания по управляющим элементам ИСР Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений.

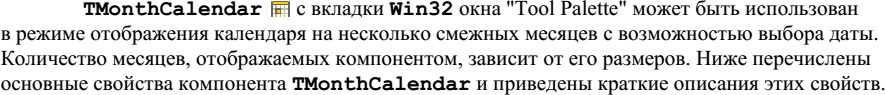
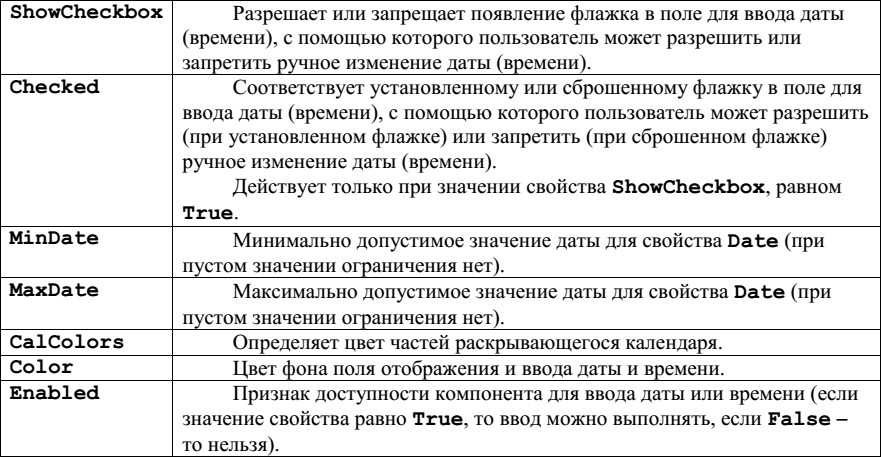
**Форма отчета:**

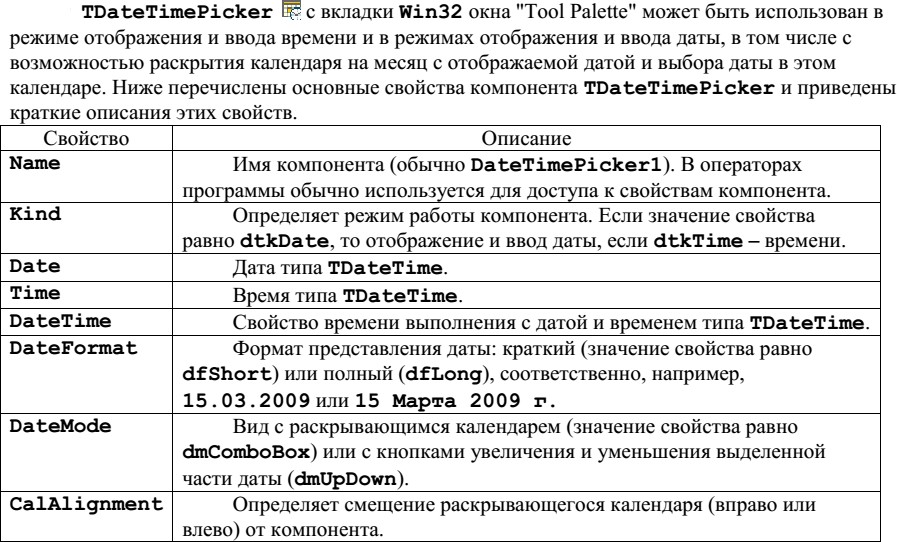
* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

**Теоретические сведения:**







Индивидуальное задание Задание. Создайте приложение, при выполнении

которого на экране появляется окно с тем же заголовком и с теми же компонентами, отображающими дату и время. Цвета частей календарейдолжны отличаться от стандартного цвета. Компоненты должны отображать текущую дату или текущее время. Компонент TMonthCalendar должен быть установлен с текущим месяцем в середине этого компонента с использованием кнопок в его верхней части. При демонстрации выполнения приложения один из компонентов TDateTimePicker должен быть раскрыт так же, как на рисунке.

**Контрольные вопросы:**

1. Компонент DateTimePacker. Назначение и свойства компонента
2. Компонент MonthCalendar. Назначение и свойства компонента

**Практическая работа №18**

**Классы ООП: виды, назначение, свойства, методы, события. Объявления класса. Создание наследованного класса. Программирование приложений. Перегрузка методов.**

**Цель работы** - создать программу, которая переводит значение температуры по Цельсию в значения температуры по Фаренгейту.

**Форма отчета:**

* Выполнить задания;
* Показать решенные задачи преподавателю;
* Ответить на вопрося преподавателя;

**Время выполнения:** 4 ч

Введите графический объект изображения значения температуры.

1. Поместите на форму два поля ввода Edit и четыре кнопки.

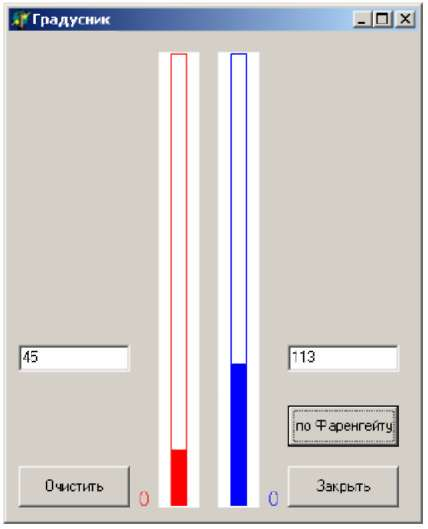


2. Ввести число в левое поле. По нажатию на кнопку "по Фаренгейту" в правом поле выводится преобразованное число. По нажатию кнопки "Очистить" очищаются поля ввода.

3. Ограничьте вводимые температуры диапазоном от 0°C до 100°C (если введено значение, превышающее 100°C или 212°F, то при нажатии кнопки в полях ввода должно отобразиться 100 и 212 соответственно).

4. Отобразить столбик термометра графически. Добавьте компоненты Image.

5. Сделать видимой только ту кнопку, которая необходима для преобразования.



**Листинг программы**

unit Unitl; interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls;

type

TForml = class(TForm) Editl: TEdit; Edit2: TEdit; Buttonl: TButton; Button2: TButton; Button3: TButton; Button4: TButton; Imagel: TImage; Image2: TImage; Labell: TLabel; Label2: TLabel;

procedure FormCreate(Sender: TObject); procedure Button2Click(Sender: TObject); procedure ButtonlClick(Sender: TObject); procedure Button3Click(Sender: TObject); procedure Button4Click(Sender: TObject);

private

{ Private declarations } public

{ Public declarations } end; var

Forml: TForml;

implementation

{$R \*.dfm}

procedure TForml.FormCreate(Sender: TObject); begin

Editl.Text := ''; Edit2.Text := '';

Imagel.Canvas.Pen.Color := clRed; Imagel.Canvas.Brush.Color := clWhite; Imagel.Canvas.Rectangle(l0,l,23,360); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clWhite; Image2.Canvas.Rectangle(l0,l,23,360); end;

procedure TForml.Button2Click(Sender: TObject);

var a, b, c : real;

begin

a := StrToFloat(Editl.Text);

if (a > 0) or (a < l00) then b := l.8 \* a + 32;

if a > l00 then b := 2l2;

if a < 0 then b := 32;

Edit2.Text := FloatToStr(b);

Image1.Canvas.Pen.Color := clRed; Image1.Canvas.Brush.Color := clWhite; Image1.Canvas.Rectangle(10,1,23,360); Image1.Canvas.Pen.Color := clRed; Image1.Canvas.Brush.Color := clRed; Image1.Canvas.Rectangle(10,round(360-a),23,360); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clWhite; Image2.Canvas.Rectangle(10,1,23,360); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clBlue; Image2.Canvas.Rectangle(10,round(360-b),23,360);

end;

procedure TForm1.Button1Click(Sender: TObject);

var a, c, d : real;

begin

c := StrToFloat(Edit2.Text); if (c > 32) or (c < 212) then d := ( c - 32)/1.8; if c < 32 then d := 0; if c > 212 then d := 100; Edit1.Text := FloatToStr(d); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clWhite; Image2.Canvas.Rectangle(10,1,23,360); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clBlue; Image2.Canvas.Rectangle(10,round(360-c),23,360); Image1.Canvas.Pen.Color := clRed; Image1.Canvas.Brush.Color := clWhite; Image1.Canvas.Rectangle(10,1,23,360); Image1.Canvas.Pen.Color := clRed; Image1.Canvas.Brush.Color := clRed; Image1.Canvas.Rectangle(10,round(360-d),23,360); end;

procedure TForm1.Button3Click(Sender: TObject); begin

Edit1.Text := ''; Edit2.Text := '';

Image1.Canvas.Pen.Color := clRed; Image1.Canvas.Brush.Color := clWhite; Image1.Canvas.Rectangle(10,1,23,360); Image2.Canvas.Pen.Color := clBlue; Image2.Canvas.Brush.Color := clWhite; Image2.Canvas.Rectangle(10,1,23,360);

end;

procedure TForm1.Button4Click(Sender: TObject); begin

Close; end;

procedure TForm1.Edit1Enter(Sender: TObject); begin

Button2.Visible := true; Button1.Visible := False; end;

procedure TForm1.Edit2Enter(Sender: TObject); begin

Button1.Visible := True;

Button2.Visible := False; end; end.

**Контрольные вопросы.**

1. Из каких составных частей состоит интегрированная среда разработки программ Delphi?
2. Что представляет собой программа Delphi?
3. Какие действия необходимо выполнить для правильной ор- ганизации проекта?
4. Как создается обработчик событий для кнопки?
5. Какие приняты соглашения по именам составных частей проекта ?

**Список литературы**

1. Колдаев В.Д., Основы алгоритмизации и программирования: учебноепособие / под ред.проф. Л.Г. Гагариной. – М: ИД «ФОРУМ»: ИНФРА-М, 2017 – 416с., URL: https://znanium.com/bookread2.php?book=902236.
2. Колдаев В.Д., Основы алгоритмизации и программирования: учебное пособие / под ред.проф. Л.Г. Гагариной. – М: ИД «ФОРУМ»: ИНФРА-М, 2019 – 414с., URL: https://znanium.com/bookread2.php?book=980416;
3. Фризен И.Г., Основы алгоритмизации и программирования: учебное пособие / Фризен И.Г. — М: ИД «ФОРУМ»: ИНФРА-М, 2017 — 392 с. — Текст : электронный // Электронно-библиотечная система «Знания» : [Электронный ресурс]. — URL: https://znanium.com/bookread2.php?book=559358.
4. Ночка Е.И. Основы алгоритмизации и программирования на языке Питон: / Ночка Е.И. – М.: КУРС, 2017 – 59с. : ил. – [Электронный ресурс]. – URL: https://znanium.com/bookread2.php?book=772548.
5. Лубашева Т.В., Основы алгоритмизации и программирования: учебное пособие / Лубашева Т.В – Минкс: РИПО, 2016. – 378с., URL: https://znanium.com/bookread2.php?book=947743
6. Коврижных, А.Ю. Основы алгоритмизации и программирования : практикум : учебно-методическое пособие : в 2 частях / А.Ю. Коврижных, о.Е. К, Г.Е. Лузина. — Екатеринбург : УрФУ, [б. г.]. — Часть 1 : Задачи и упражнения — 2016. — 52 с. — ISBN 978-5-7996-1886-5. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/98290