**Министерство образования и науки Российской Федерации**

Старооскольский технологический институт им. А.А. УГАРОВА

(филиал) федерального государственного автономного образовательного учреждения

высшего образования

«Национальный исследовательский технологический университет «МИСиС»

**ОСКОЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ**

**Т.Н. Коренькова**

**Е.Н. Запасник**

**ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ**

**ЧАСТЬ 2**

**Учебное пособие для студентов всех форм обучения специальности**

**09.02.04 Информационные системы (по отраслям)**

# **ОДОБРЕНО НАУЧНО-МЕТОДИЧЕСКИМ СОВЕТОМ ОПК**

# **В КАЧЕСТВЕ УЧЕБНОГО ПОСОБИЯ**

**Старый Оскол 2016**

**ББК**

**УДК**

**Рецензент:**

преподаватель ОПК СТИ НИТУ «МИСиС» *Горюнова М.В.*

**Коренькова Т.Н., Запасник Е.Н.**

**Основы алгоритмизации и программирования. Часть 2:** учебное пособие. – Старый Оскол: СТИ НИТУ «МИСиС», 2016. – 92 с.

Учебное пособие составлено в соответствии с ФГОС по специальности 09.02.04 Информационные системы (по отраслям). Способствует развитию навыка программирования в среде TPascal, Delphi. Рассмотрены основные понятия алгоритмизации и программирования, представлены все виды задач обработки данных, приводятся таблицы сравнительного анализа форматов, операторов, процедур, описания данных для языков программирования. Дается развернутое и практически полное описание языков и систем программирования Pascal и Delphi. Все разделы насыщены примерами и задачами.

Учебное издание предназначено для студентов системы СПО.

©Коренькова Т.Н., Запасник Е.Н., 2016

©СТИ НИТУ «МИСиС»

# **Содержание**

Введение

# **Практическое занятие №18**

# Работа с файлом последовательного доступа. 4

# **Практическое занятие №19**

# Работа с файлом произвольного доступа. 4

# **Практическое занятие №20**

# Использование стандартных процедур и функций для работы с файлами. 7

# **Практическое занятие №21**

# Программирование модуля. 11

# **Практическое занятие №22**

# Создание библиотеки подпрограмм. 16

# **Практическое занятие №23.**

# Использование библиотеки подпрограмм. 16

# **Практическое занятие №24.**

# Изучение интегрированной среды разработчика. 19

# **Практическое занятие №25**

# Создание простого проекта. 26

# **Практическое занятие №26**

# Объявление класса, создание экземпляров класса. 32

# **Практическое занятие №27**

# Создание наследованного класса. 39

# **Практическое занятие №28**

# Перегрузка методов. 45

# **Практическое занятие №29**

# Создание проекта с использованием кнопочных компонентов. 52

# **Практическое занятие №30**

# Создание проекта с использованием компонентов для работы с текстом. 60

# **Практическое занятие №31**

# Создание проекта с использованием компонентов ввода

# и отображения чисел, дат и времени. 63

# **Практическое занятие №32**

# Создание проекта с использованием компонентов

# стандартных диалогов и системы меню. 68

**Практическое занятие №33**

Разработка оконного приложения. 69

# **Практическое занятие №34**

# Разработка оконного приложения с несколькими формами. 71

# **Практическое занятие №35**

# Разработка многооконного приложения. 74

**Сообщения и коды ошибок**. 77

**Литература**. 90

Введение

Трудно представить себе мир без компьютеров. Человек, не умеющий обращаться с ним, находиться в положении неграмотного, попавшего в библиотеку: на полках много интересных книг, а как их прочтешь? Культура общения с компьютером стала частью общей культуры человека. Но чтобы пользоваться услугами компьютера, надо уметь правильно сформулировать ему задание – четко, однозначно и на языке который он понимает. В настоящее время благодаря своей четкости и логичности Паскаль занял достойное место среди других зыков программирования. Он прекрасно подходит как для обучения программированию, так и для создания серьезных программ-приложений.

Пособие включает в себя: задания, разобранные примеры, комментарии к выполнению, контрольные задания и вопросы, справочные материалы и литературу.

Задачи сгруппированы по темам и охватывают все разделы базового курса программирования: от объявления переменных и программ с линейной структурой до работы с объектно-ориентированным языком программирования.

В каждой практической работе имеется справочник по языку программирования Паскаль, который содержит описание наиболее часто используемых процедур и функций.

*Обращение к студентам.*

Напоминаем, что научиться программировать можно только, программируя, решая конкретные задачи. Поэтому, чтобы получить максимальную пользу, вы должны работать активно. Решайте задачи. Изучайте приведенные решения. Вводите их в компьютер. Не бойтесь экспериментировать — вносите изменения в программы. Чем больше вы сделаете самостоятельно, тем большему вы научитесь! Желаем удачи!

*Как работать с пособием?*

Группы задач следуют в том порядке, в котором изучаются соответствующие разделы в курсе программирования. Перед тем как приступить к решению задач, нужно изучить соответствующую тему — прочитать теоретический материал. Писать программу лучше сначала на бумаге (в виде блок – схем), а уже затем вводить в компьютер. Задача считается решенной, если написанная программа работает так, как сказано в условии задачи.

*Оформление решений:*

Важно, чтобы решенная задача была правильно оформлена. Это предполагает:

1. использование несущих смысловую нагрузку имен переменных, констант и функций;
2. применение отступов при записи инструкций программы;
3. использование комментариев.

Правильно оформленную программу легче отлаживать, кроме того, она производит хорошее впечатление.

**Правила выполнения практических работ**

В соответствии с общими задачами обучения и развития к уровню подготовки предъявляются следующие требования:

1. Строго выполнять весь объем домашней подготовки, указанный в описании соответствующей практической работы;
2. Знать, что при выполнении каждой работы предшествует проверка готовности студента, которая проводится преподавателем;
3. После выполнения работы студент, должен представить отчет о проделанной работе с обсуждением полученных результатов и выводов.

В конце каждого занятия преподаватель ставит зачет, который складывается из результатов наблюдения за выполнением практической части работы, проверки отчета, беседы в ходе работы или после неё.

**Практическое занятие №18-19 Работа с файлом последовательного доступа. Работа с файлом произвольного доступа.**

# **Цель работы: научиться решать задачи и использовать процедуры и функции обработки файлов**

# **Теоретическая часть**

**Файл** – форма хранения информации вне среды.

*Файл удобен*:

- данные из файла можно использовать в нескольких программах;

- файл сохраняется после окончания работы программы;

- данные часто не убираются в ОЗУ (игры);

- с файлами можно выполнять множество полезных манипуляций.

**Типы файлов:**

1. Последовательные – чтение/запись от начала к концу.
2. Файлы прямого доступа – доступ к записи по адресу.
3. Индексированные файлы – доступ к записи по адресу и по ключу.

**Порядок работы с файлами.**

1. Объявить файл (varf: fileof тип или varf: text для текстового файла).
2. Связать файловую переменную с конкретным файлом на устройстве.
3. Открыть файловую переменную для записи или чтения.
4. Закрыть файловую переменную для записи или чтения.

По организации работы с данными различают три вида файлов: *типизированные*, *текстовые* и *нетипизированные*.

**Основные процедуры и функции обработки файлов (общие).**

**Assign(f,name);** - связь файловой переменной f с внешним файлом.

**Reset(f);** - открыть существующий файл для чтения.

**Rewrite(f);** - открыть файл для записи (если существует – стирается!).

**Close(f);** - закрыть файл для чтения/записи.

**Eof(f)** - функция «конец файла» (true, если из файла всё считано).

**Для типизированных файлов:**

**Read(f,a);** - чтение информации из файла в переменную а.

**Write(f,a); -** запись информации (значения переменной a) в файл.

**Seek(f,num);** - пропуск заданного количества записей в файле.

**Truncate(f); -** удаление части файла с текущего компонента до конца.

**FilePos(f)** - функция номера текущей записи в файле.

**FileSize(f)** - функция - количество записей в файле (размер файла).

**Для текстовых файлов:**

**Readln(f,a);** - чтение строки из файла.

**Writeln(f,a);** - запись строки в файл.

**Append(f);** - открыть файл для добавления в конец файла.

**Eoln(f)** - функция конец строки файла.

**SeekEoln(f)** - функция конец строки файла(не замечает пробелов в конце).

**SeekEof(f) -** функция конец файла(не замечает пробелов в конце файла).

*Задача 1.* Присвоить переменным числовые значения и сохранить их в файле. Прочитать файл, передав числовые значения другим переменным.

*Задача 2.* Создать текстовый файл данных из нескольких строк. Прочитать файл.

Прочитать числовое значение из текстового файла в числовую переменную и изменить полученное значение.

**Program fil; Uses crt;**

**var f: file of integer;**

**b,c,d,x,y,z,i:integer;**

**t:text; a:string;**

**Begin**

**clrscr; b:=3; c:=4; d:=5;**

**writeln('b=',b,' c=',c,' d=',d);**

**assign(f,'c:r.dat');**

**rewrite(f);**

**write(f,b,c,d);**

**close(f);**

**reset(f);**

**read(f,x,y,z);**

**write('x=',x,' y=',y,' z=',z);**

**close(f);**

**readkey; clrscr; assign(t,'c:\pascal\turbo7\work\prob.dat');**

**reset(t);**

**for i:=1 to 4 do**

**begin**

**readln(t,a);**

**writeln(a);**

**end;**

**readln(t,x); x:=x\*2; writeln('x=',x);**

**close(t);**

**readkey;**

**End.**

# **Практическая часть**

*Задачи для самостоятельного решения:*

1. Создать на диске С:\ текстовый файл (file.txt). Записать в этот файл небольшую информацию.
2. Составить полную цепочку команд текстового файла. Имя вводится с клавиатуры.
3. Создать файл в папке Bin. Переименовать этот файл. Имя создаваемого файла и имя нового файла вводятся с клавиатуры.
4. Проверить существует ли файл с именем fil.doc на диске.
5. Создать файл и записать в него фразу «Здравствуй мир!»
6. Создать текстовый файл (расширение .txt) и записать в него пять одинаковых строк.
7. Есть текстовый файл (из предыдущего задания), дописать в него 6 одинаковых строк.
8. В конец существующего текстового файла записать новую строку с текстом «До свидания, люди!».
9. Дан массив строк. Записать их в файл, расположив каждый элемент массива на отдельной строке с сохранением порядка.
10. Дан текстовый файл, состоящий из первых десяти натуральных чисел. Вывести на экран первых пять.
11. Удалите некоторый файл.
12. Имеется текстовый файл. Напечатать его первую строку.
13. Имеется текстовый файл. Напечатать его некоторую строку.
14. Дан текстовый файл (расширение .txt) посчитать количество символов в нем.
15. Дан текстовый файл (расширение .txt) посчитать количество строк в нем.
16. Дан текстовый файл (расширение .txt) посчитать количество символов в каждой строке.
17. Создать типизированный файл и записать в него число 500
18. Создать типизированный файл и записать в него число 4,5
19. Создать типизированный файл и записать в него слово «Привет»
20. Создать типизированный файл и записать в него 5 одинаковых целых чисел
21. Создать типизированный файл и записать в него 5 одинаковых вещественных чисел
22. Создать типизированный файл и записать в него 5 одинаковых слов
23. Создать типизированный файл и записать в него числа 10, 12, …,16
24. Создать типизированный файл и записать в него 7 целых чисел
25. Создать типизированный файл и записать в него 4 вещественных числа
26. Создать типизированный файл и записать в него 5 слов

**Контрольные вопросы.**

1. Верно ли, что элементы файла могут быть только одного типа?
2. Можно ли сравнивать файлы или присваивать один другому?

1. Установите, возможны ли приведенные ниже объявления файлов:

1) type WordFile=file of word;

   var MyFile : WordFile;

2) var MyFile : file of Word.

12.10. Установите, какая задача решается программой

var FileName : string; {имя файла}

Fvar :fileofchar; {переменная файлового типа}

Letter :char; {читаемый из файла символ}

begin write('Введите имя файла ');

readln(FileName);

assign(Fvar,FileName);

{$I-} {отключен автоматический контроль ввода/вывода}

reset(FVar);

{$I+} {включен автоматический контроль ввода/вывода}

if ioresult<>0

then begin writeln('Неоткрытфайл ',FileName); Halt end;

while not eof(Fvar) do

begin read(Fvar,Letter);

Letter:=upcase(Letter);

seek(Fvar,filepos(Fvar)-1);

write(Fvar,Letter)

end;

close(Fvar)

end.

# **Практическое занятие №20 Использование стандартных процедур и функций для работы с файлами.**

# **Цель работы: научиться использовать стандартных процедур и функций для работы с файлами.**

# **Теоретическая часть**

## Текстовые файлы.

Текстовый файл можно рассматривать как последовательность символов разбитую на строки длиной от 0 до 256 символов.

Описание:

var

f: text;

Для текстовых файлов существует специальный вид чтения и записи (read, write).

**Открытие текстового файла.**

* **rewrite(f);** - открыт для записи
* **reset(f);** - открыт для чтения
* **append(f);** - открывает уже существующий файл и позиционирует указатель обработки на конец файла. После такого открытия текстовый файл можно только дополнить информацией, начиная с конца строки.

**Процедуры и функции обработки текстовых файлов.**

1. **read(f,v1,[v2..vn])** – считывает одно или более значений из текстового файла в одну или более переменных, если переменные v1,[v2..vn] – строкового типа, то считывается строка или символы, если v1 – целого типа приведет к считыванию последовательности цифр, которая затем интерпретируется в число, значение которого будет присвоено переменной v1. В случае если вместо последовательности цифр идет любая другая последовательность символов выдается сообщение об ошибке;
2. **readln(f,v1,[v2..vn])** – выполняет те же действия, что и предыдущая процедура, а затем делает пропуск до начала следующей строки файла;
3. **write(f,v1,[v2..vn])** – записывает в текстовый файл одно или более значений;
4. **writeln(f,v1,[v2..vn])** - выполняет те же действия, что и предыдущая процедура, а затем добавляет к файлу метку конца строки.
5. **eoln(f);** - логическая процедура, возвращает для файла состояния конца строки;
6. **eof(f);** - логическая процедура, возвращает для файла состояния конца файла;
7. **seekeof(f);** - логическая процедура, возвращает для файла состояния конца файла;
8. **seekeoln(f);** - логическая процедура, возвращает для файла состояния конца строки, причем указатель файла пропускает все пробелы и знаки табуляции, предшествующие маркеру.

## Типизированные файлы.

К типизированным файлам относятся файлы строго определенного типа. Чаще всего это файлы, состоящие из записей. Они применяются для создания различных баз данных.

**Описание:**

type

fil = record

… { поля записи}

end;

var

f: file of fil;

В типизированных файлах содержимое рассматривается как последовательность записей определенной структуры. Единицей измерения такого набора данных является сама запись. Длина записи определяется как **sizeof(fil);**

**Seek(f, num)** процедура устанавливает текущую файловую переменную f на запись с номером num.

Типизированные файлы позволяют организовать работу в режиме чтения записи. Информация в типизированных наборах данных представлена в том же виде, как в памяти машины во время выполнения программы, поэтому не надо отслеживать управляющие последовательности типа конец строки или возврат каретки.

**Средства для работы с типизированными файлами.**

1. **truncate(f);** - уничтожает все компоненты файла f, начиная с места текущего положения файлового указателя;
2. **filepos(f): longint;** - функция возвращает для файла f текущую файловую позицию (номер записи, на которую она установлена) в виде значения типа longint;
3. **filesize(f): longint;**- функция нахождения размера файла (количество записей) в виде значения типа longint.

Для того, чтобы очередная запись могла быть записана в конец типизированного файла, необходимо перевести текущую файловую позицию в конец файла. При создании файла это происходит автоматически после формирования каждой новой записи. Если файл уже создан и файловая позиция, установленная по **Seek** , находится в начале файла **(filepos(f) < filesize(f))** , то в конец файла ее позволяет перевести вызов **seek(f,filesize(f))** .

## Нетипизированные файлы.

Нетипизированные файлы объявляются как файловые переменные типа file и отличаются тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы совместимыми с любыми другими файлами и позволяют организовать высокоскоростной обмен данными между диском и памятью.

var

f: file;

С помощью процедур reset, rewrite можно задать длину нетипизированного файла: **reset(f,512); rewrite(f, 65535);**по умолчанию длина равна 128 байтам.

**Средства для работы с нетипизированными файлами.**

1. Используют все процедуры и функции, рассмотренные ранее, кроме **read, write** , которые заменяются высокоскоростными.
2. **Blockread(f,buf,c);** - где buf – имя переменной, которая участвует в обмене данными с дисками, задает число, считываемых блоков;
3. **Blockwrite(f,buf,c);** - процедура предназначена для быстрой передачи в файл определенного числа записей из переменной buf.

Эти процедуры выполняют операции ввода – вывода блоками. Объем блока в байтах определяется по формуле: **объем=c\*recsize(f);,** где recsize – размер записи файла, заданный при его открытии.

**Практическая часть**

*Задачи для самостоятельного решения:*

1. Составить программу, которая создает файл таблицы значений sin(x) и tg(x) на отрезке [0,3] с шагом 0.01. Значения х записывать с одной цифрой в дробной части, значения функций sin(x) и tg(x) - с пятью.
2. Составьте программу, которая создает файл, состоящий из 100 случайных чисел целого типа в диапазоне от 0 до 300. Исследуйте получившийся файл с целью обнаружения в нем простых чисел. Простые числа вывести на экран.
3. Составьте программу, которая создает файл, состоящий из 30 случайных чисел целого типа в диапазоне от -50 до 100. Исследуйте получившийся файл с целью обнаружения в нем чисел, кратных 5. Если такие числа есть, то вывести их на экран.
4. Имеется текстовый файл. Напечатать первые 10 символов первой строки.
5. Напишите программу, которая создает файл данных, хранящий записи о владельцах автомототранспорта: марка автомобиля, номер регистрации в ГАИ, дата постановки на учет, ФИО владельца, домашний адрес, и обеспечивает обслуживание данного файла, запись, изменение данных, удаление, а также поиск данных по регистрационному номеру.
6. Имеется текстовый файл. Найти номер самой длинной строки. Если таких строк несколько, то найти номер первой из них.
7. Составьте программу, которая создает файл из 20 целых чисел. Замените число, стоящее на 15 месте числом 99. Элементы нового файла выведите на экран.
8. Составить программу, которая создает файл, состоящий из 35 случайных вещественных чисел в диапазоне от –1 до 1. Найти их среднее арифметическое.
9. Имеется текстовый файл. Напечатать самую длинную строку. Если таких строк несколько, то напечатать первую из них.
10. Имеется текстовый файл. Напечатать все его строки, в которых имеется более трех пробелов.
11. Составить таблицу значений для функции sin(x).
12. Имеется текстовый файл. Посчитать количество строк, начинающихся с буквы «А» или «а». Вывести на экран эти строки.
13. В существующем файле, элементами которого являются отдельные слова, изменить на заданное значение первое слово.
14. Имеется текстовый файл. Напечатать все его строки, содержащие более 30 символов.
15. В существующем файле, элементами которого являются числа, изменить на заданное число первый элемент.
16. Имеется текстовый файл. Напечатать все его строки, начинающиеся с буквы «Т». Посчитать их количество.
17. Имеется файл, элементами которого являются отдельные символы. Выяснить, образуют ли соседние символы файла буквосочетанием «ура».
18. Имеется текстовый файл. Напечатать все его строки, содержащие в качестве фрагмента заданный текст.
19. Имеется файл, элементами которого являются отдельные символы. Выяснить являются ли первые два символа цифрами.
20. Имеется текстовый файл. Посчитать количество строк, в которых имеется ровно пять букв «и». Вывести эти строки на экран.
21. Имеется типизированный файл из целых чисел. Составить программу упорядочения файла по возрастанию.
22. Имеется текстовый файл, в каждой строке которого первые два символа являются буквами. Получить слово, образованное р-ми символами некоторой строки.
23. Имеется файл, элементами которого являются числа. Напечатать все его элементы с нечетным порядковым номером.
24. Имеется текстовый файл, в каждой строке которого первые два символа являются буквами. Получить слово, образованное вторыми буквами каждой строки.
25. Составить программу, которая создает файл данных о жильцах дома, отображая в нем следующую информацию о каждом: номер квартиры, фамилию, имя, возраст, для лиц старше 18 лет в зависимости от рода занятий (учеба, работа, пенсия) – запись места учебы, места работы и трудового стажа, для пенсионеров – год выхода на пенсию. Программа должна обеспечивать ввод данных, поиск квартиры с максимальным числом жильцов, поиск самого юного и самого пожилого жильца, поиск студентов, пенсионеров
26. Имеется текстовый файл. Напечатать все его строки, содержащие более 10 символов.
27. Имеется файл с числами. Поменять местами второе и последнее числа.

**Контрольные вопросы:**

1.Какую работу выполняет функция? Напишите программу для создания файла, который может быть обработан представленной функцией, заполнив его подходящими случайными числами.

function max(var t:text):real;

var m,x:real;

begin reset(t);

read(t,m);

while not eof(t) do

begin read(t,x);

if x>m then m:=x

end;

max:=m

end.

2.Установите, есть ли ошибки в программе

var f:file of integer;

i:integer;

procedure pp(var fail:file of integer; var f:boolean);

var i:integer;

begin reset(fail);

i:=0;

while not eof(fail) do

begin read(fail,i);

inc(i)

end;

k:=filesize(fail);

if i=k then pp:=true else pp:=false

end;

begin write('Введитеимяфайла ');

readln(i);

assign(f,i);

pp(fail,f);

close(fail);close(f);

end.

1. Исправьте ошибку в программе

var f1,f2 : file of real;

r : real;

s1,s2 : string;

begin write('Введите имя исходного файла ');

readln(s1);

write('Введите имя создаваемого файла ');

readln(s2);

assign(f1,s1);reset(f1);

assign(f2,s2);rewrite(f2); while not eof(f1) do

begin read(f1,r);

write(f2,r);

end;

# end.

# **Практическое занятие №21 Программирование модуля.**

# **Цель работы: научиться программировать модуль, познакомиться с возможностями графических операторов**, **построение графических изображений в Паскале**

# **Теоретическая часть**

Стандартные модули Паскаля

В Турбо Паскале имеется 8 стандартных модулей, в которых содержится множество различных типов, констант, процедур и функций. Этимимодулямиявляются SYSTEM, DOS, CRT, GRAPH, OVERLAY, TURBO3, GRAPH3. Модули Паскаля GRAPH , TURBO 3, GRAPH 3 выделены в отдельные TPU -файлы, а остальные входят в состав библиотечного файла TURBO . TPL . Лишь один модуль Паскаля SYSTEM подключается к любой программе автоматически, все остальные становятся доступны только после указания их имен в списке подключаемых модулей.

Модуль Паскаля SYSTEM. В него входят все процедуры и функции стандартного Паскаля, а также встроенные процедуры и функции, которые не вошли в другие стандартные модули (например, INC , DEC , GETDIR и т.п.). Модуль Паскаля SYSTEM подключается к любой программе независимо от того, объявлен ли он в предложении USES или нет, поэтому его глобальные константы, переменные, процедуры и функции считаются встроенными в Турбо Паскаль.

Модуль Паскаля PRINTER делает доступным вывод текстов на матричный принтер. В нем определяется файловая переменная LST типа TEXT , которая связывается с логическим устройством PRN. После подключения данного модуля Паскаля можно выполнить, например, такое действие:

**Пример стандартного модуля Паскаля**

Uses printer;   
Begin   
   Writeln(lst, ‘ Турбо Паскаль ’);   
End.

Модуль Паскаля CRT. В нем сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом работы экрана. С его помощью можно перемещать курсор в любую точку экрана, менять цвет выводимых символов и фона, создавать окна. Кроме того, в данный модуль включены также процедуры «слепого» чтения клавиатуры и управления звуком.

Модуль Паскаля GRAPH . Содержит набор типов, констант, процедур и функций для управления графическим режимом работы экрана. Этот модуль позволяет создавать различные графические изображения и выводить на экран надписи стандартными или созданными программистом шрифтами.

Модуль Паскаля DOS . В модуле собраны процедуры и функции, открывающие доступ к средствам дисковой операционной системы MS - DOS .

Модуль Паскаля OVERLAY . Данный модуль необходим при разработке громоздких программ с перекрытиями. Турбо Паскаль обеспечивает создание программ, длина которых ограничивается лишь основной оперативной памятью. Операционная система MS - DOS оставляет программе около 580 Кбайт основной памяти. Память такого размера достаточна для большинства исполняемых программ, тем не менее, использование программ с перекрытиями снимает это ограничение.

Модули Паскаля TURBO 3 и GRAPH 3 введены для обеспечения совместимости с ранней версией системы Турбо Паскаль.

Язык паскаль предоставляет возможность рисования изображений. Чтобы паскаль смог это сделать необходимо подключить модуль Graf.

Шаблон графичекой программы выглядит следующим образом:

Uses   
    Graph;   
Var   
    grDriver,grMode,res:integer;   
Begin   
    grDriver:=detect;   
    init Graph (grDriver,'*путькфайлу BGI*');  
    *Операторыграфическихизображений*  
    Close graph;   
End.

**Библеотека GRAPH**

Библеотека (модуль) Graph содержит константы, процедуры и функции для управления графическим режимом работы монитора.

Константы цвета

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Black = 0 Blue = 1 Green = 2 Cyan = 3 Red = 4 Magneta = 5 Brown = 6 LightGray = 7 | {Черный} {Синий} {Зеленый} {Голубой} {Красный} {Фиолетовый} {Коричневый} {Светлосерый} |  | DarkGray = 8 LightBlue = 9 LightGreen = 10 LightCyan = 11 LightRed = 12 LightMagneta = 13 Yellow = 14 White = 15 | {Темносерый} {Яркосиний} {Яркозеленый} {Яркоголубой} {Розовый} {Малиновый} {Желтый} {Белый} |

Константы типов и толщины линий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SolidLn = 0 DottedLn = 1 CenterLn = 2 | {Сплошная} {Точечная} {Штрихпунктирная} |  | DashedLn = 3 NormWidth = 1 ThickWidrth = 3 | {Пунктирная} {Нормальная толщина} {Тройная толщина} |

Константы шаблона штриховки

|  |  |
| --- | --- |
| EmtyFill = 0 solidFill = 1 LineFill = 2 LtSlashFill = 3 SlashFill = 4 BkSlashFill = 5 LtBkSlashFill = 6 HatchFill = 7 XHatchFill = 8 InterLeaveFill = 9 WideDotFill = 10 CloseDotFill = 11 UserFill = 12 | {Заполнениецветомфона} {Сплошнаяштриховка} {Горизонтальнаяштриховка} {/// штриховка} {/// штриховкатолстымилиниями} {\\\ штриховкатолстымилиниями} {\\\ штриховка} {Заполнениепрямойклеткой} {ЗАпонениекосойклеткой} {Заполнениечастойсеткой} {Заполнениередкимиточками} {Заполнениечастымиточками} {Типзадаетсяпользователем} |

Процедуры

**Arc(X,Y:integer; U1,U2,R:Word**) Строит дугу окружности текущим ыветом стекущими праметрами линии. X, Y - координаты ыентра дуги, U1 - угол до начальной точки дуги, отсчитываемый против часовой стрелки от горизонтальной оси, направленной слева направо, U2 - угол до конечно йточки дуги, отсчитываемый так же, как U1, R - радиус дуги.

**Bar(X1,Y1,X2,Y2:integer)** Строит прямоугольник, закрашенный текущим ыветом с использованием текущего стиля (орнамента, штриховки). X1, Y1, X2, X2 - координаты левого верхнего и правого нижнего углов прямоугольника.

**Bar3D(X1,Y1,X2,Y2:integer;Glubina:word;Top:boolean)** Строит параллелепипед, используя текущий стиль и цвет. X1, Y1, X2, X2 - координаты левого верхнего и правого нижнего углов передней грани; Glubina - ширина боковй гранни (отсчитывая по горизонтали), Top - признак включения верхней грани (Если True - верхняя грань вычерчивается, False - не вычерчивается).

**Circle(X,Y:integer;R:word)** Рисует текущем цветом окружность радиуса R с центром в точке (X,Y).

**ClearDevice** Очищает грфический экран, закрашивает его в цвет фона.

**ClearViewPort** очищает выделенное графическое окно, заркрашивает его в цвет фона.

**CloseGraph** Закрывает графический режим, т.е. освобождает память,распределенную под драйверы графики и файлы шрифтов, и восстанавливает текстовый режим работы экрана.

**Ellipse(X,Y:ineger; U1,U2,XR,YR:word)** Рисует дугу эллипса текущим цветом; X, Y - координаты центра эллипса; U1, U2 - углы до начльной и конечной точек дуги эллипса (см. процедуру Arc); XR, YR - горизонтальная и вертикальная полосы эллипса.

**FillEllipse(X,Y:ineger; XR,YR:word)** Рисует заштрихованный эллипс, используя X,Y как центр и XR, YR как горизонтальную и вертикальную полосы эллипса.

**FillPoly(N:word;Vwr PolyPoints)** Рисует и штрихует многоугольник, содержащий N вершинс координатами в PolyPoints.

**InitGraph(Var Driver, Mode:integer;Path:String)** Организует переход в графический режим. Переменные Driver и Mode содержат тип графического драйвера и его режим работы. Третий параметр определяет маршрут поиска графического драйвера. Если строка пустая (т.е. ранва""), считается что драйыер находдится в текущем каталоге.

**Line(X1,Y1,X2,Y2:Integer)**Рисует линию от точки X1,Y1 до точки X2,Y2.

**LineTo(X,Y:Integer)**Рисует линию от текущего указателя к точке X1,Y1.

**MoveTo(X,Y:Integer)**Смещает текущий указатель к точке X,Y.

**OutTextXY(X,Y:Integer;TextString:String)**Выводит текст в заданное место экрана

**PieSlice(X,Y:Integer;U1,U2,Radius:Word)**Строит сектор круга, закрашенный текущей штриховкой и цветом заполнения. X,Y - координаты центар сектора круга; U1 и U2 - начальный и конченый кглы сектора, отсчитываемые против часовой стрелки от горизонтальной оси, напрвленной вправо; Radius - радиус сектора.

**PutPixel(X,Y:Integer;Color:Word)** Выводит точку цветом Color с координатами X,Y.

**Rectangle(X1,Y1,X2,Y2)** Рисует контур прямоугольника, используя текущий цвет и тип линии. X1,Y1 - координаты левого верхнего угла прямоугольника, X2,Y2 - координаты правого нижнего угла прямоугольника.

**Sector(X,Y:Integer; U1,U2,XR,YR:word)** Рисует и штрихует сектор эллипса радиусами XR и YR с центром в X,Y от начального угла U1 к конечному углу U2.

**SetBkColor(Color:Word)**Устанавливает цвет фона.

**SetColor(Color:Word)**Устанваливает основной цвет, которым будет осуществляться рисование.

**SetFillStyle(Pattern,Color:Word)**Устанавливает образец штриховки и цвет.

**SetLineStyle(LineStyle,Pattern,Thickness:Word)**Устанавливает толщину и стиль линии.

**SetTextStyle(Font,Direction,CharSize:Word)**Устанавливает текущий шрифт, направление (горизонтальное или вертикальное) и размер текста.

**SetViewPort(X1,Y1,X2,Y2: Integer; ClipOn:Boolean)** Устанавливает прямоугольное окно на крафическом экране. Параметр ClipOn определяет "отсечку" элементов изображния, не умещающихся в окне.

Функции

**GetMaxX** и **GetMaxY** Возвращает значения максимальных координат экрана в текущем режиме работы, соответственно, по горизонтали и вертикали.

**GraphResult** Возвращает значение GrOk, соответствующее коду 0, если все графические операции программы выполнились без ошибок.ю или возвращает числовой код ошибки (от -1 до -14).

# **Практическая часть**

## Основные графические операторы для построения изображений:

PutPixel (X, Y, цвет) - вывод точки на экран, где X, Y  - координаты точки ;  
Line ( X1, Y1, X2, Y2 ) - проводит линию из точки с координатами (X1, Y1 ) в точку с координатами ( X2, Y2 );  
Rectangle (X1, Y1, X2, Y2 ) - прямоугольник со сторонами, параллельными осям координат;  (X1, Y1) и (X2, Y2)  - координаты, определяющие одну из диагоналей прямоугольника ;  
Bar ( X1, Y1, X2, Y2 ) - закрашенный прямоугольник (без окантовки);  
Circle ( X, Y, радиус ) – на экран выводится окружность с центром в точке  ( X, Y )(тип integer) ;  
Arc (X, Y, начальный угол, конечный угол, радиус) - на  экран  выводится  дуга окружности с центром в точке (X, Y ); углы задаются в градусах; дуга рисуется ПРОТИВ часовой стрелки;  
Ellipse ( X, Y, начальный угол, конечный угол,  горизонтальный радиус, вертикальный радиус) -  на экран выводится эллиптическая дуга с центром в точке с координатами ( X, Y ) (тип integer);  
SetFillStyle (заполнение, цвет) – определение вида и цвета заполнения области;  
FloodFill (x, y, цвет границы) – заливка замкнутой области.

## Процесс  построения изображений с помощью графических процедур разбивается на отдельные этапы:

1. Переключить монитор в графический режим с помощью оператора InitGraph (Driver, mode, <путь к драйверу>).  
2. Установить разрешающую способность экрана по умолчанию режимом Detect или процедурой SetGraphMode. Режим Detect устанавливает разрешающую способность экрана 640\*480 пикселей, т.е. координата  Х может принимать значения от 0 по 639, а У от 0 по 479.  
3. Очистить  и инициализировать графический экран процедурой ClearDevice.  
4. Установить цвет фона  оператором  SetBkColor и цвет изображения  оператором SetColor.  
5. Вывести на экран точки,  отрезки, прямоугольники, дуги, окружности, эллипсы.  
6. Вывести на экран закрашенные фигуры.  
7. Вывести тексты и подписи на экран. Для использования операторов Write и Writeln в графическом режиме необходимо выполнить следующую операцию присваивания : DirectVideo := FALSE;  Или использовать процедуру Outtextxy(x,y,st), которая выводит строку st,  начиная с позиции  x, y

## Пример программы построения  графика функции.

program graphic;   
uses graph;   
var driver, mode, errorcode : integer; xm,ym,i,j : integer;   
pi,pi300,x1,y1,x2,y2, sc : real;   
st1,st2,st3 : string;   
function f(x:real) : real;  
begin  
f:=sin(x)+sin(2\*x)+sin(3\*x)-1-cos(x)-cos(2\*x);{ функция для построения}  
end; {графика}   
begin   
st1:='x';st2:='y';   
st3:=' Press ENTER';   
sc:=50;   
driver:=9; {egavga}   
mode:=2; {640х480 пикселей}   
initgraph(driver,mode,'d:\bp\bgi'); **{инициализация графического режима }**  
errorcode:=graphresult;  
if errorcode<>grok then {ошибка }  
begin  
writeln('Error init Graph');  
closegraph;   
halt;  
end;  
xm:=getmaxx div 2;  
ym:=getmaxy div 2;  
{ xm=320;ym=240;центр экрана}  
line(xm,20,xm,460);{ось y}  
line(20,ym,620,ym);{ ось x}  
outtextxy(630,ym,st1); {маркировка оси х}  
outtextxy(xm,10,st2); {маркировка оси у}  
pi:=3.1415926; pi300:=pi/300;  
x1:=-pi;  
for i:=0 to 24 do {разметка оси х вертикальными черточками}  
begin  
line(xm+round(80\*x1),230,xm+round(80\*x1),250);  
x1:=x1+pi300\*25;  
end;  
x1:=-pi; {собственно построение графика отрезками прямых}  
while x1<pi do  
begin  
y1:=f(x1);x2:=x1+pi300;  
y2:=f(x2);  
line(xm+round(80\*x1), ym-round(sc\*y1),  
xm+round(80\*x2), ym-round(sc\*y2));  
x1:=x2;  
end;  
outtextxy(270,470,st3);  
readln;  
closegraph;  
end.

Программа компилируется и выдает график функции.

*Задачи для дополнительного решения:*

* 1. Представьте, что вы встретились с хранителем времени и он открыл вам тайну звезд. Почему мерцают звезды на небе ? Потому, что каждая звезда - жизнь одного человека. Вам нужно вовремя погасить старую и зажечь новую. Помогите хранителю времени решить задачу Решение задачи начинается с построения модели: пусть на экране каждая точка обозначает звезду и всего горит 20 (выбранных случайным образом) звезд; теперь необходимо погасить первую зажженную точку и нарисовать ее в любом другом месте, затем перейти ко второй и т.д. Когда погаснет двадцатая, необходимо вновь перейти к первой и т.д.
  2. Напишите программу "дождь" - на экране появляется (случайным образом) рисуется 50 точек и они движутся сверху вниз. В тот момент, когда одна из точек достигнет нижней границы экрана, точка появляется в верхней части.
  3. Измените программу "дождь" так, чтобы некоторые точки двигались быстрее, другие медленнее.
  4. Измените программу "дождь" так, чтобы точки начинали движение в левом нижнем углу и заканчивали в правом верхнем.

**Контрольные вопросы:**

* 1. Какие размеры имеет экран в графическом режиме ЯП Паскаль?
  2. Где находится точка начала координат?
  3. Как изменяются координаты х и у в Паскале?
  4. Какого типа должны быть координаты объектов?
  5. Какие стандартные функции позволяют перевести действительное числа в целые?
  6. Какие команды рисования объектов вы знаете?
  7. Как установить графический режим?

# **Практическое занятие №22 - 23 Создание библиотеки подпрограмм. Использование библиотеки подпрограмм.**

# **Цель работы: научить студентов работать с подпрограммами**

# **Теоретическая часть**

## Основные понятия о подпрограммах

При разработке сложных программ используют так называемый структурный подход к программированию и нисходящее проектирование программ, когда сложная программ разбивается на более (или менее) функционально-законченные части, каждая из которых проще исходной программы. Такие программы легче отлаживать и использовать. Отдельные части программы называют подпрограммами. Использование подпрограмм наиболее эффективно в тех случаях, когда одна и та же подпрограмма может использоваться в программе не один раз, возможно с различными параметрами. Это позволяет экономить память компьютера.

Подпрограммы, в свою очередь, могут разбиваться на более мелкие части, реализуемые также в виде подпрограмм более низкого уровня.

В языке Турбо Паскаль используют подпрограммы двух типов: процедуры (Procedure) и функции (Function). Подпрограммы по структуре сходны с программой, но они обязательно имеют оригинальное имя, которое указывается в заголовке. Подпрограммы описываются в разделе описаний, использующих (вызывающих) их программ (или подпрограмм).

***Процедура*** — это независимая именованная часть программы, которую можно вызвать по имени для выполнения определенных действий. Структура повторяет структуру программы. Процедура не может выступать как операнд в выражении. Упоминание имени процедуры в тексте программы приводит к активизации процедуры и называется ее вызовом.

***Функция***аналогична процедуре, но имеются два отличия: функция передает в точку вызова скалярное значение; имя функции может входить в выражение как операнд. Например, функция Sqr(X) — возведет в квадрат значения целого или вещественного X и возвратит в точку вызова вычисленное значение квадрата числа X.

Итак, ***отличие подпрограмм-процедур от подпрограмм-функций состоит в том****,*что процедуры служат для задания совокупности действий, направленных на изменение внешней по отношению к ним программной обстановки, а функции, являясь частным случаем процедур, отличаются от них тем, что они обязательно возвращают в точку вызова основной программы единственный результат как значение имени этой функции.

Все процедуры и функции языка Турбо Паскаль делятся на две группы: встроенные (стандартные) и определенные пользователем. Первые входят в состав языка я вызываются для выполнения по строго фиксированному имени. Вторые разрабатываются и именуются самим пользователем.

Выделим глобальные переменные, которые используются в главной программе: М1, М2, К1, К2, Мах1, Мах2.

Uses crt;

Type Tmas=array[1..1000] of integer;

Var M1, M2: Tmas;

K1, K2, Max1, Max2 : integer;

{ Процедура ввода длины массива и самого массива }

Procedure Vvod(Var K:integer; Var M:Tmas);

Var i:integer;

Begin

Write(' Введитедлинумассива');

Readln(K);

Writeln(' Введите элементы массива целых чисел, через пробел');

For i:=1 to K do

Read(M[i]); readln;

End; { конец процедуры ввода}

{ Процедура поиска максимального элемента в массиве}

Procedure Poisk\_max(K:integer; M:Tmas; Var Max:integer);

Vari:integer;

Begin

Max:=M[1]; { За максимум принимаем первый элемент}

For i:=2 to K do

If M[i]>Max then Max:=M[i]; {Запоминаем новый максимум}

End;

{ Начало основной программы}

Begin

Clrscr;

Writeln(' Ввод первого массива');

Vvod(K1, M1);

Writeln(' Ввод второго массива');

Vvod(K2,M2);

Poisk\_max(K1, M1, Max1);

Poisk\_max(K2, M2, Max2);

If Max1>Max2 then writeln(' Max1 большеионо = ',Max1)

Elsewriteln(' Max2 больше и оно = ',Max2);

Readkey; { Останов для просмотра результатов}

End.

# **Практическая часть**

*Задачи для самостоятельного решения:*

1. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая проверяет их сумму, третья выводи результат. Используйте эти процедуры в основной программе. Используйте X и Y как глобальные переменные.

1. Напишите программу вычисления площади поверхности и длины экватора на основе известного радиуса планет солнечной системы. Форму планет будем считать шаром. Вычисление площади поверхности и длины экватора оформите отдельными функциями.
2. Напишите программу поиска большего из четырех чисел с использованием подпрограммы поиска большего из двух чисел.
3. Даны координаты вершин многоугольника (x1, y1,x2,y2,…x10,y10). Напишите программу для вычисления его периметра (вычисление расстояния между вершинами оформить подпрограммой).
4. Напишите программу вычисления суммы: 1! + 2! + 3! + … + n!, используя функцию вычисления факториала числа k.
5. Напишите программу для вычисления числа сочетаний из N по M. Число сочетаний определяется по формуле N!/(M!\*(N-M)!, где N – количество элементов перебора. Используйте подпрограмму вычисления факториала.
6. Напишите программу для определения НОД трех натуральных чисел.
7. Даны действительные числа s,t. Составить программу вычисления выражения f(t, -2s, 1.17) + f(2.2, t, s-t), где f(a,b,c) = (2a – b – sin(c)) / (5 + |c|).
8. Даны натуральные m и n (m<n). Составить программу, сокращающую дробь m/n.
9. Напишите программу вычисления суммы квадратов простых чисел, лежащих в интервале (M,N).
10. Напишите программу подсчета числа четных цифр, используемых в записи N-значного числа M.
11. Составьте программу вычисления суммы трехзначных чисел, в десятичной записи которых нет четных цифр.
12. Составьте программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр.
13. Составьте программу нахождения наименьшего натурального N-значного числа X (X>=10), равного утроенному произведению своих цифр.
14. Составьте программу подсчета числа всех натуральных чисел, меньших М, квадрат суммы цифр которых равен X.

**Контрольные вопросы:**

1. Данные каких типов используются в ЯП ПАСКАЛЬ?
2. Как происходит ввод данных в программу?
3. Как вывести результаты работы программы?
4. Какие стандартные функции вы знаете?

**Практическое занятие №24. Изучение интегрированной среды разработчика.**

ЗАДАЧА ОБ АНКЕТЕ

Объекты: форма, текстовое поле, изображение, кнопка. Программирование кнопок

Цель работы. Создать форму «Анкета студента» с данными о себе и двумя фотографиями (портретной и художественной), ко­торые перекрывают друг друга и должны появляться в результате нажатия на кнопки (рис. 4).

Ознакомиться с такими объектами: форма (Form), текстовое поле (Label), рисунок (Image), кнопка (Button) и их основными свойствами: подпись (Caption), цвет (Color), шрифт (Font), види­мость (Visible), ширина (Width), высота (Height) и другими.

Теоретические часть. Объект Formиспользуют длясоздание окна программы-приложения. Рассмотрим такие свойства:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| ActiveControl | Для задачи активно­го объекта (фокуса) в форме | Buttonl, Edit2 |
| AutoScroll | Наличие в форме полос прокручива­ния | True, False |
| BorderStyle | Возможность менять размеры окна | bsSizeable (окно с произволь­ными размерами), bsDilog, bsNone (окно с фиксирован­ными размерами) |
| Width, Height | Ширина и высота окна в пикселях | 503, 224 (числовые значе­ния) |
| Font | Шрифт | Комплексное свойство, зада­ется в диалоговом окне |
| HorizScrollBar VertScrolIBar | Параметры полос прокручивания | Комплексное свойство |
| Icon | Задаем пиктограмму, которая будет раз­мещена в заголовке формы при выполне­нии программы | (None) - стандартная пикто­грамма для Delphi, или за­груженная с определенного файла \*.ico |
| Name | Имя формы | Forml (идентификатор) |

|  |  |  |
| --- | --- | --- |
| Caption | Заголовок формы | Произвольная строка символов |
| Color | Цвет фона формы | clGreen, clInfoBk (перечис­лимый тип)или $004525В1 (числовое значе­ние - задается в диалоговом окне) |
| Cursor | Вид курсора на сво­бодном месте окна на этапе выполнения | crDrag, crCross, crHelp, crArrow (перечислимыйтип) |
| Enabled | Доступность для действий над объек­тами в форме во время выполнения | True, False |
| Left, Top | Координаты левого верхнего угла окна в пикселях | 200, 108 (числовые значе­ния) |
| Position | Размещение и раз­меры окна в момент запуска программы | poScreenCenter, poDesigned |
| WindowState | Состояние окна в момент запуска про­граммы | wsNormal, wsMaximized, wsMinimized |

Объект текстовое поле (Label) используют для создания тек­стов Надписей) в окне программы. Кроме аналогичных к выше приведенным в предшествующей таблице свойств Width, HeightFont, Color, Name,Caption, Cursor, Enabled, Let, Top, он владеет еще и такими:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Align | Выравнивание поля относительно объекта, который его содержит (формы) | alBottom, al- Client, alLeft, alNone, alTop |
| Alignment | Выравнивание текста в границах поля | taCenter, taLeft- Justify, taRight- Justify |
| AutoSize | Приведение границ поля к границам текста | True, False |
| Visible | Видимость объекта | True, False |
| WordWrap | Перенесение слов текста в новую строку | True, False |

Объект картинка (Image, изображение) используют для встав­ки графических объектов из файлов типа \*.bmp, \*.emf, \*.ico, \*.wmf в форму. Кроме известных свойств Elign, Width, Height, Name, Cursor, Enabled, Left, Top, Visible, он обладает еще таки­ми:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значение |
| Center | Выравнивание рисунка по центру относительно содержащего его поля | True, False |
| Picture | Имя графического файла | Задают в диалоговом окне |
| Stretch | Приведение размера изображения к задан­ным размерам объекта | True, False |
| AutoSize | Приведение размера объекта к реальным размерам изображения | True, False |

Объект кнопка (Button) используют для размещения элемента управления — кнопки на форме. Кнопкиимеюттакиесвойства: Visible, Width, Height, Font, Color, Name, Caption, Cursor, En­abled, Left, Top идругие.

**Практическая часть**

Ход работы

1. Загрузите систему визуального программирования Delphi.

Для запуска системы визуального программирования Delphi щелкают на пиктограмме Delphiили с помощью каскадного меню выбирают Start (Пуск) Programs (Программы) ==>BorlandDelphi х.О=>Delphi х.О, где х - версия программы. В ре­зультате на экране откроются четыре окна. 2. Исследуйте способы активизации четырёх окон Delphi:

* главного окна Delphi х.О — Projectl, где находятся па­нель инструментов, палитра компонентов и главное меню;
* окна инспектора объектов ObjectInspectorсо значения­ми свойств активного объекта;
* окна формы (Forml), в котором будут приведены резуль­таты работы будущей программы;
* окна текста программы (Unitl.pas).

Замечание. Окно текста программы может частично перекры­ваться окном формы. Активизировать окна, а также менять их размеры или расположение можно с помощью мыши или ис­пользуя функциональные клавиши на клавиатуре: F10 - для активизации главного меню (после этого нажмите

на клавишу Esc); F11 - для активизации окна инспектора объекта; F12 - для перехода между окнами формы и кода программы. 3. Запустите программу Projectl на выполнение и рассмотрите окно пустой пока что формы. Поупражняйтесь с окном фор­мы.

Запустить программу можно несколькими способами:

* Используя команду Run=>Run главного меню;
* Щелкнув на кнопке Run.Ипанели инструментов;

• Нажав функциональную клавишу F9. Выполните такие действия: максимизируйте окно, восстанови­те его предыдущий размер, минимизируйте и снова разверните окно, передвиньте на рабочем столе и измените его размеры, вы­зовите системное меню (Alt+ пропуск). .Выполните те же дейст­вия с помощью команд Move, Size и других и клавиатуры.

Вывод: окно формы обладает всеми свойствами стандартного окна операционной системы Windows.

1. Закройте окно программы Forml, минимизируйте главное окно Delphi и создайте на рабочем диске папку с названием группы, а в ней папку, названную вашей фамилией. Снова активизируйте окно Delphi.
2. Сохраните созданную программу в своей папке.

Для этого выполните команду главного меню Fi)eSaveAll (Сохранить Все) или нажмите на кнопку SaveAll на панели инструментов. В поле «Savein:» (Сохранить в:) появившегося ок­на («SaveUnitlAs») с помощью списка выберите название ра­бочего диска, после чего найдите и откройте свою папку. Задайте название для файла текста программы, предварительно удалив предложенное компьютером название Unitl.pas =>Save. В сле­дующем окне «SaveProjectlAs» введите название файла проекта, удалив предложенное компьютером название Projectl. dpr =>Save. Обратите внимание: файлы проекта и текста программы должны иметь разные названия.

6. Визуально ознакомьтесь со свойствами формы Left, Тор,  
Width и Height.

Переместите форму Forml с помощью мыши. Обратите внима­ние, что изменение расположения формы ведет к изменению ее I свойств Left и Тор - координат левого верхнего угла формы в ок­не ObjectInspector. Поменяйте размеры формы. Убедитесь, что теперь меняются свойства Width (ширина) и Height (высота) фор­мы в окне инспектора объектов.

7. Исследуйте, как изменение значений свойств Left, Тор,  
Width или Height формы в окне ObjectInspector ведет к изменению расположения или размера формы.

Для этого введите некоторое значение в пикселях и нажмите на клавишу Enter.

8. Измените цвет фона формы.

Для этого в окне свойств формы ObjectInspector в строке Color выберите значение цвета фона одним из двух способов:

* вызовите окно выбора цвета двойным щелчком мыши по текущему значению свойства Color. Выберите один из ба­зовых цветов (Basiccolors) или установите свой (DefineCustomColors) цвет. Подтвердите выбор цвета (Ok).
* с помощью списка поупражняйтесь с разными значе­ниями свойства Color. Задайте начальное значение цвета - clBtnFace.

1. Выполните программу еще раз (см. пункт 3).
2. Вставьте в форму текстовое поле (объект типа Label) с тек­стом «Анкета студента».

Два раза щелкните мышью по пиктограмме Label на заклад­ке Standard палитры компонентов главного окна Delphi. Распо­ложите вставленный объект, например, так, как показано на рис. 3, перетягивая его мышью. Если объект Labell не выделен­ный, активизируйте его и в окне ObjectInspector, измените зна­чение свойства Caption с Labell на текст «Анкета студента» (без кавычек). Измените значения свойства Font (шрифт) этого тек­стового поля на такие:

Font : New Roman Cyr;

Font style : Bold;

Size : 16;

ColorTimes : Purple.

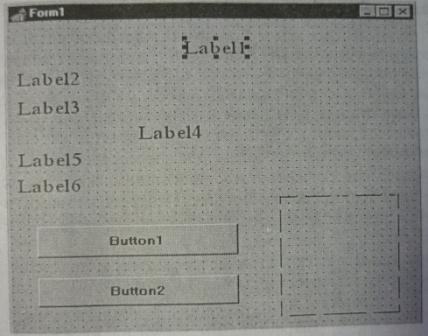


Рис. 3

Замечание. В окне ObjectInspector отображается список свойств лишь активного на данный момент объекта.

11. Аналогично вставьте в форму еще несколько текстовых по­лей, где укажите свои биографические данные.

Один из вариантов расположения текстовых полей показан на рис. 4

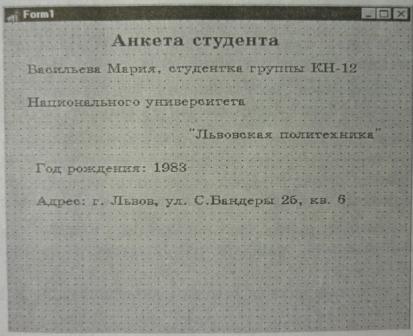


Рис. 4

12. Вставьте в форму объект типа Image (картинка).

Для этого щелкните один раз левой клавишей мыши на пикто­грамме Image закладки Additional (дополнительные) палитры компонентов и, например, в нижнем правом углу формы обведите контур для будущего изображения (фотографии). Если нужно, из­мените размер формы или вставленного объекта и добейтесь наи­лучшего расположения на ней созданных прежде объектов. Менять размеры объекта можно методом их «растягивания» за маркеры (черные габаритные квадратики). Запомните название, которое сре­да присвоит этому объекту (значение свойства Name) или замените его по своему усмотрению. По умолчанию этот объект будет иметь стандартное название Imagel.

13. Вставьте свою портретную фотографию с помощью свойства  
Picture (иллюстрация) объекта Imagel.

Для этого выберите объект Imagel и активизируйте свойство Picture в окне ObjectInspector. Щелкнув на кнопке Ш, вызовите диалоговое окно выбора рисунка PictureEditor. Щелкните на кнопке Load (загрузить) и в окне Loadpicture укажите путь к файлу с фотографией. Если такого файла нет, воспользуйтесь лю­бой картинкой из библиотеки 16со1ог, которая по умолчанию на­ходится в папке

[C:\Program](file://C:/Program) Files\Borland\Delphi x.0\Images\Splash\16color.

Выберите любой файл Open. Подтвердите свой выбор в окне PictureEditor нажатием на кнопку Ок. Задайте значение свойства Stretch для объекта Imagel - True.

14. Наложите свою художественную фотографию на портретную, вставив в форму еще один объект типа Image.

Один из вариантов расположения фотографии показан на рис.5. Пусть этот объект имеет название Image2.

Замечание. При наложении объектов может возникнуть необ­ходимость менять порядок их отображения. Для этого исполь­зуют команды Send То Back (переслать назад) или Bring То Front (перенести вперёд), находящиеся в контекстном меню.

15. Поэкспериментируйте со свойством Visible (видимость)  
двух изображений, каждый раз выполняя программу (см.  
пункт 3).

После этого установите значение свойства Visible в False для обоих изображений.

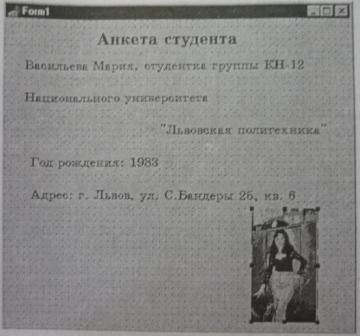


Рис. 5

16. Вставьте в форму кнопки для поочередного отображения на­ложенных одна на другую фотографий - два объекта типа Button с названиями Buttonl и Button2.

Пиктограмма объекта типа Button (кнопка) находится на закладке Standard палитры компонентов главного окна Delphi. Поменяйте подписи на кнопках (измените значения свойства Caption) на «Портретная фотография» и «Художественная фото­графия» соответственно. Выберите наилучший, на Ваше усмотре­ние, кириллизированный шрифт для подписей. Если Вы приме­нили картинки из стандартной библиотеки Delphi, задайте подписи для кнопок на своё усмотрение. Один из вариантов рас­положения кнопок показан на рис. 6.

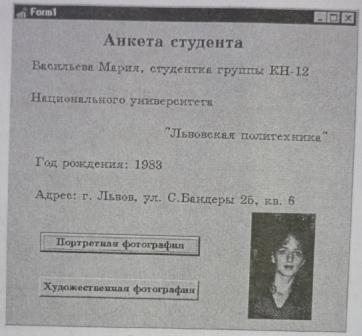


Рис. 6

17. Запрограммируйте кнопку «Портретная фотография» так, чтобы после ее нажатия в форме появлялась портретная фотография.

Для программирования кнопки Buttonl необходимо два раза щелкнуть на ней левой клавишей мыши. В результате активизи­руется окно текста программы с заготовкой процедуры ButtonlClick, которая будет обрабатывать событие щелканья на кнопке Buttonl:

procedureTforml.ButtonlClick(Sender: Tobject); begin

end;

В заготовку необходимо вставить текст программы реакции на это событие. Процедура будет иметь такой вид:procedureTForml.ButtonlClick(Sender:TObject); begin

{Портретная фотография становится видимой) Imagel.Visible: =True;

{Художественная фотография становится невидимой) Image2 .Visible: =False; end;

С помощью данной процедуры свойство видимости для объекта Imagel включаем, а это же свойство для объекта Image2 выклю­чаем. Для кнопки «Художественная фотография» действия будут противоположные. Обратите внимание на использование состав­ных имен типа Imagel.Visible, в которых название объекта от его свойства отделяется точкой. Такие составные имена дают доступ к значению конкретного свойства некоторого объекта.

18. Запрограммируйте кнопку «Художественная фотография»  
соответственно ее назначению (см. пункт 17).

Текст процедуры для этой кнопки будет иметь вид:

procedureTForml.Button2Click(Sender: TObject); begin

{Портретная фотография становится невидимой) Imagel.Visible:=False;

end;

Чтобы создать такую процедуру быстро, можно скопировать две команды присваивания из предыдущей процедуры в новую и поменять выражения справа.

1. Запустите программу и убедитесь, что кнопки выполняют свои функции. Закройте окно программы «Анкета студента».
2. Сохраните созданную программу в своей папке. Выберите элемент главного меню File =>SaveAll (Сохранить Все) или нажмите кнопку SaveAll°Ана панели инструментов.
3. Создайте ехе-файл проекта

Выполните команду главного меню Project =>BuildAll (Скон­струировать Все).

22. Закройте Delphi, выполните созданную программу и поупражняйтесь с работой кнопок.

Запустите ехе-файл с именем проекта и пиктограммой т$ из своей папки.

23. Продемонстрируйте созданную форму преподавателю. За-  
кончите работу.

Задача 1.1. Вставьте в форму третью фотографию (вашего дома или машины) и еще одну кнопку с соответствующей подписью. Если файл с такой фотографией не существует, воспользуйтесь любым файлом из библиотеки 16со1ог (см. п. 13).

Задача 1.2. Поменяйте подписи на кнопках на такие: «Сме­нить фотографию» и «Забрать фотографию», перепрограммировав кнопки. Выполните программу и убедитесь в правильности ее ра­боты.

Подсказка. В тексте процедур, описывающих работу кнопок, можно воспользоваться командами вида:

ifImagel. Visible = True then ...

{Если видимость = True или равносильной командой} ifImagel.Visiblethen{Здесь условие истинное, }

{если видимость включена}

Задача 1.3. Поменяйте сценарий работы программы для задачи 1.2 на следующий:

• сразу после запуска программы фотографий на форме не видно, есть две кнопки «Портретная фотография» и «Спрятать фотографию». Доступной является первая кнопка;

после щелчка на кнопке «Портретная фотография» долж­но появиться портретное фото, текст на первой кнопке -поменяться на «Художественная фотография», кнопка «Спрятать фотографию» стать доступной;

• после щелчка на кнопке «Художественная фотография» фотография в форме должна поменяться на художествен­ную, а подпись на этой кнопке - на «Третья фотография»;

после щелчка на кнопке «Третья фотография» фотогра­фия в форме должна поменяться на третью, а подпись на этой кнопке - на«Портретная фотография»; • после щелчка на кнопке «Спрятать фотографию» фото­графия должна исчезнуть, а эта кнопка стать недоступ­ной.

Запишите фрагменты программного кода в отчет. Выполните программу и убедитесь в правильности ее работы.

Подсказка. В тексте процедур, описывающих работу кнопок, можно воспользоваться командами, которые меняют их свойства: Caption (подпись), Visible (видимость), Enabled (доступность).

Задача 1.4. См. условие задачи 1.3. Запрограммируйте кнопку «Спрятать фотографию» так, чтобы, щелкнув на ней, она стано­вилась не только недоступной, но и невидимой.

Задача 1.5. Измените, программный код решения задачи 1.4 так, чтобы, отключая фотографии, надпись на первой кнопке все­гда соответствовала фотографии, которая должна появиться после ее нажатия.

Задача 1.6. Исходя из условия задачи 1.5, сделайте так, чтобы последовательность переключения фотографий не нарушалась вследствие их отключения, а также добавьте текстовую надпись с названием фотографии, видимой в текущий момент.

# **Практическое занятие №25 Создание простого проекта.**

ЗАДАЧАОБОБМЕНЕВАЛЮТЫ

Объекты: поля редактирования, переключатели и их свойства. Программирование ветвлений

Цель работы. Создать форму с названием «Обмен валюты» для моделирования соответствующих операций обменного пункта. Применить поля редактирования (Edit) и переключатели (RadioButton, радиокнопка), а также кнопки для выполнения вы­числений и выхода из программы (см. образец формы на рис. 8).

Теоретические часть

Объект поле редактирования (Edit) используют для ввода строки символов с клавиатуры. В случае необходимости для преобразования полученной строки (свойствоText) в число и наоборот применяют стандартные функции языка PascalValи Str, или функции среды DelphiStrToFloatи FloatToStr. Кроме уже известных вам свойств, поля редактирова­ния Edit обладают такими:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| CharCase | Вид символов, кото­рые будут вводить в поле редактирования | ecNormal (обычные), ecUpperCase (прописные буквы), ecLowerCase (строчные буквы) |
| CtI3D | Объемное изображе­ние объекта | True, False |
| Pass- wordChar | Символ для ввода пароля | #0 (обычное отображение тек­ста), \* (текст будет отображать­ся звездочками), 0 (текст будет отображаться нулями) |
| ReadOnly | Возможность менять текст (доступность поля) | True (текст нельзя менять), False (текст можно менять) |
| Hint | Текст подсказки, ко­торый появляется при наведении указателя мыши | «Введите сумму» (произвольная строка символов) |
| ShowHint | Показывать/Не пока­зывать подсказку | True, False |
| Text | Текст в поле редактирования | «0,0001» (произвольная строка символов) |

Объект переключатель (RadioButton) используют для выбора некоторого действия среди нескольких альтернатив. Рассмотрим такие свойства переключателей:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Checked | Состояние пе­реключателя | True (выбранный), False (не выбранный) |
| TabOrder | Порядок вы­бора объекта клавишей Tab | 0 (первый), 4 (пятый) |
| TabStop | Доступ к дан­ному объекту табулятором | True (доступный), False (не доступный) |

Практическая часть

Ход работы

Загрузите среду визуального программирования Delphi.

Откажитесь от возможности изменять размеры окна про­граммы, указав значение свойства формы BorderStyle — bsDialog.

Задав это значение, выполните программу и убедитесь, что Нельзя изменить размер формы. Обратите внимание на отсутствие кнопок для минимизации и максимизации окна, а также систем­ного меню. Завершите работу программы.

1. Вставьте в форму два объекта типа RadioButton (переключатель), как показано на рис. 7.

Для этого щелкните на пиктограмме Шобъекта типа RadioButton (переключатель), которая находится на закладке Standard палитры компонентов главного окна Delphi. После этого щелкните в точке вставки на форме. Повторите эти действия, чтобы вставить вторую радиокнопку.

2. Задайте начальное значение второго переключателя как  
активное.

Для этого щелкните на правом переключателе и задайте значе­ние его свойства Checked (контроль выбора) - True.

3. Вставьте в форму два поля редактирования — объекты Editl  
и Edit2.

Для этого щелкните па пиктограмме объекта типа Edit (редак­тирование), которая находится на закладке Standard палитры ком­понентов, а потом щелкните в нужном месте на форме. Вставьте вто­рой объект (рис. 7). Запустите программу и поупражняйтесь со вставленными объектами: щелкните в поле редактирования, введите некоторое число, удалите его. Закройте окно программы.

1. Разместите на форме два текстовых поля — объекты Labell и Label2 (рис. 7).
2. Вставьте в форму два поля редактирования — объекты Edit3 и Edit4 (рис. 7).
3. Вставьте две кнопки — объекты типа Button (рис. 7).
4. Вставьте в форму еще два текстовых поля — объекты Label3 и Label4 (рис. 7).

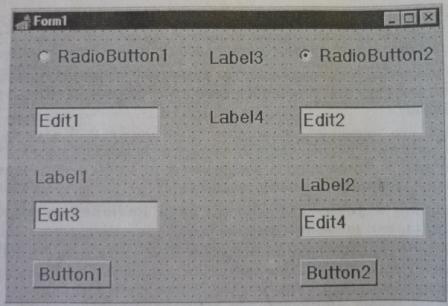


Рис. 7

8. Сохраните созданную на данный момент форму в своей папке.

File =>SaveAll. Файлы текста программы и проекта назовите разными именами. Имена запишите в отчет. В дальнейшем пе­риодически, в частности перед очередными запусками проекта на

выполнение, сохраняйте файлы программы (File =>SaveAll, вво­дить имена файлов уже не нужно).

9. Поменяйте название формы с «Forml» на «Обмен валюты».

Для этого замените значения свойства Caption формы. Щелк­ните на свойстве Caption окна ObjectInspector. Введите название формы без кавычек. Обратите внимание на то, что для объектов многих типов (в частности, Label, Button, Form, RadioButton, CheckBox и других) значения свойств Caption и Name совпадают.

10. Измените подписи Caption на объектах типа RadioButton,  
Label и Button так, как показано на рис. 8.

Для этого поочередно выбирайте объекты (щелкайте на них) и меняйте значения свойства Caption.

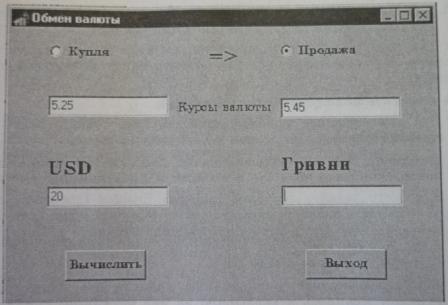


Рис. 8

11. Задайте одинаковые размеры для всех текстовых полей, по­лей редактирования и кнопок и выровняйте их на форме.

Для этого одновременно выделите пять объектов в левом столб­це одним из способов:

* удерживая нажатой клавишу Shift, поочередно активизи­руйте объекты, щелкая на них левой клавишей мыши;
* обведите вокруг этих объектов контур, удерживая нажа­той левую клавишу мыши.

В окне ObjectInspector задайте общие для этих объектов зна­чения в пикселях свойств Width (ширина), Height (высота) и Left (отступ от левой границы окна). После выбора некоторого свойст­ва группы объектов его значением является значения соответствующего свойства первого объекта в группе. Можете изменить стиль, цвет или размер шрифта одновременно для всех выделен­ных объектов (свойство Font). Снять выделение можно, щелкнув на свободном месте формы. Аналогично выполняется выравнива­ние правого столбца объектов.

Выровняйте вставленные поля попарно в горизонтальном на­правлении. Для этого поменяйте значение свойства Тор (отступ от верхней границы окна в пикселях) для соответствующих групп объектов. Сохраните работу (SaveAll).

12. Задайте значения курсов купли-продажи валюты, а также  
сколько валюты кантор покупает или продает.

Для этого задайте значение свойства Text объекта Editl, на­пример, 5.25. Повторите это для объекта Edit2 (значение 5.45) и Edit3 (20).

13. Очистите поле редактирования Edit4.

Для этого удалите значение свойства Text объекта Edit4. Не путайте значения свойств Name и Text этих объектов.

14. Заблокируйте возможность ввода данных для поля Edit4,  
поскольку в это поле будет выводиться результат.

Для этого задайте значение свойства Enabled (доступность) -False. Изменять значение определенного свойства можно, дважды щелкнув на нём левой клавишей мыши. Сохраните работу (SaveAll). Выполните программу и убедитесь, что нельзя ввести или редактировать данные в поле Edit4.

15. Запрограммируйте радиокнопки так, чтобы направление  
стрелки менялось и указывало на вид операции: купля или  
продажа. Сделайте активным поле Edit3.

Щелкните дважды на правом переключателе RadioButton2 (Продажа). Получите заготовку процедуры RadioButton2Click. В теле этой процедуры опишите действия, которые должны быть выполнены в результате щелчка на правом переключателе RadioButton2:

procedure TForml.RadioButton.2Click(Sender: Tobject); begin

{Меняем направление стрелки) Label3.Caption := '=>';

Edit3.SetFocus {Активизируемполе Edit3) end;

Аналогично запрограммируйте событие Click для переключате­ля RadioButtonl. Учтите, что стрелка должна показывать на ле­вое поле ('<=').

Фрагмент программного кода созданной процедуры, запишите в отчет.

16. Запустите программу и убедитесь, что переключатели вы­полняют предписанные им функции.

17. Запрограммируйте кнопку «Выход».

Воспользуйтесь процедурой закрытия окна программы Close:

procedure TForml.Button2Click(Sender: TObject); begin

Close {Закрываем окно программы]

end;

18. Запрограммируйте кнопку «Вычислить».

Данное в поле редактирования - это значение свойства Text типа string. Для преобразования этого данного в числовой дейст­вительный тип (real) воспользуйтесь процедурой Val, а наоборот — процедурой Str. Опишите соответствующие основные (kurs, suma) и дополнительные (cod, ed) переменные в разделе var.

procedure Tforml.ButtonlClick(Sender: TObject); var kursl, kurs2, summa : real; cod : integer; ed : string[9];

begin

{Получаем значения курса покупки)

Val (Editl.Text, kursl, cod);

{Получаем значения курса продажи)

Val (Edit2.Text, kurs2, cod);

{Получаем числовое значение суммы в USD)

Val (Edit3.Text, summa, cod);

ifRadioButton2.Checked = True then

summa: = summa \* kurs2

else

summa := summa \* kursl;

Str(summa:9:2, ed);

Edit4.Text := ed; {Полученное число суммы в гривнах преобразовываем в текстовый формат и результат присваиваем свойст­ву Text поля вывода Edit4)

end;

1. Сохраните работу (Save All).
2. Выполните программу и поупражняйтесь с разными денеж­ными суммами и операциями купли - продажи. Закройте окно программы «Обмен валюты».

Для прерывания работы программы в случае неправильного

ввода входных данных выполните пункт главного меню Run =>Program Reset.

1. Измените размеры и цвета символов (в частности объектов Label3 и Edit3), расположение объектов, фон формы (свой­ство Color) так, чтобы форма выглядела как можно лучше.
2. Обеспечьте появление подсказки «Введите сумму в долла­рах» после перемещения указателя мыши к полю Edit3.

Выберите объект Edit3 и установите значение True для свойства ShowHint, а в поле значения свойства Hint введите текст подсказки. Сохраните работу, запустите программу и убедитесь, что подсказка появляется.

23. Поменяйте вид стрелки с => на а <= на

Для этого выберите объект Label3 и в поле значения свойства Caption введите русскую букву р, после чего, активизировав свой­ство Font, выберите название шрифта Wingdings. Дважды щелк­ните на правом переключателе и в его процедуре введите русскую букву р вместо =>. В процедуре для левого переключателя симво­лы <= замените буквой п. Сохраните работу, запустите программу и убедитесь, что стрелка изменила свой вид.

24. Создайте ехе-файл Вашей программы.

Выполните пункт главного меню Project =>BuildAll.

25. Закройте среду Delphi, запустите созданную программу и  
выполните вычисления для разных входных данных.

Запустите ехе-файл с именем проекта и пиктограммой ^ из своей папки.

26. Продемонстрируйте созданную форму преподавателю. За-  
кончите работу.

Замечание. Обратите внимание на использование запятой или точки во входных данных. В числах, которые указывают кур­сы валют, сумму гривней или долларов, для десятичной точки используйте символ, предусмотренный операционной системой вашего компьютера (см. Start (Пуск) =>Settings (Настройка)

=>ControlPanel (Панель управления) =>RegionalSettings

(Язык и стандарты) => закладка Number (Числа), строка

Decimalsymbol (Разделитель целой и дробной частей числа)).

Дополнительные задания:

Задача 2.1. Вставьте в созданную форму ещё одну кнопку для очистки полей денежных сумм. Выполните программу и убеди­тесь в правильности ее работы.

Подсказка. Для объектов Edit3, Edit4 в процедуре обработки события нажатия на кнопку очистки используйте команду при­сваивания их свойствам Text пустой строки (‘’’)/

Задача 2.2. Обеспечьте появление подсказки «Введите курс купли» и «Введите курс продажи» после перемещения указателя мыши к полям Editl и Edit2 соответственно (см. п. 24).

Задача 2.3. В процедуре для кнопки «Вычислить» предусмот­рите 1% сбора в пенсионный фонд от операции продажи.

Задача 2.4. Модифицируйте программу, введя дополнитель­ную возможность: после щелчка мышью на стрелке изменяется тип операции (купля, продажа). Запишите в отчет фрагмент про­граммного кода, реализующий эту возможность. Выполните про­грамму.

Подсказка. Для этого дважды щелкните на текстовом поле  
стрелки. Откроется окно программного кода с заготовкой проце-  
дуры Label3Click (описание действий в случае щелканья на объек-  
те Label3). В теле этой процедуры можно воспользоваться коман-  
дами вида:

ifRadioButtonl.Checked=Truethenbegin

• • • {Меняем направление стрелки на => для

Label3.CapUon)

• • • {Устанавливаем переключатель в правое положение,

меняя значение свойств Checked объектов

• • • RadioButlonl и RadioButton2)

end

else

begin

* • • {Устанавливаем направление стрелки <=}
* • • {Устанавливаем переключатель в левое положение,

меняя значения свойств Checked объектов

end;

Задача 2.5. Упростите форму (удалите лишние объекты) и из­мените код кнопки «Вычислить» так, чтобы ее можно было ис­пользовать для перевода миль в километры или наоборот в зави­симости от положения переключателя к(1миля = 1,609344 километров).

Задача 2.6. В созданную для задачи 2.5. форму вставьте объект группу переключателей (RadioGroup) для выбора типа мили из двух возможных значений: морской или обычной (1 морская ми­ля = 1,852 километров).

Подсказка. Чтобы ввести подписи к переключателям исполь­зуйте свойство Items группы переключателей RadioGroup, а для контроля выбора определенного переключателя воспользуйтесь свойством Itemlndex (равняется -1, если ни один не выбран, 0 -если выбран первый переключатель группы, 1 - если второй и т.д.)

# **Практическое занятие №26 Объявление класса, создание экземпляров класса.**ЗАДАЧАТАБУЛИРОВАНИЯФУНКЦИИ

Объекты Memo, MainMenu, PopupMenu, CheckBox, GroupBox. Программирование циклов

Цель работы. Создать форму для решения задачи табулирова­ния функции. Создать главное меню (объект типа MainMenu), содержащее команды: выйти из программы, табулировать функцию, очистить поля вывода результатов, а также кон­текстное меню (PopupMenu) с командой для очистки поля выво­да. Результаты табулирования вывести в многострочное поле редактирования (объект типа Memo). Предусмотреть возможность вывода результатов на экран, в файл, в массив. Направление вы­вода задать с помощью трёх флажков (объектов типа CheckBox), расположенных на панели группы объектов (типа GroupBox) (см. рис. 9).

Теоретические часть.

Объект Memoприменяют для созда­ния многострочного редактора текста. Кроме обычных свойств, поле редактирования Memo обладает еще такими:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| HideSelection | Сохранение выделения фрагмента текста в мо­мент потери фокуса | True (выделение не со­храняется), False (со­храняется) |
| Lines | Ввод начального текста в поле редактирования | Комплексное свойство (задается в диалоговом окне) |
| MaxLength | Максимально возможное количество введенных символов | Например, 50 - пятьде­сят символов, 0 - без ограничений |
| ScrollBars | Наличие полос прокрутки | ssNone(отсутствуют), ssHorizontal(горизон­тальная), ssBoth(обе) |

Объект CheckBoxиспользуют для создания независимого двух или трёхпозиционного флажка: включен/выключен(/серый). Для этого объекта определены такие два новых свойства:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свой­ства | Примеры значений |
| AllowGrayed | Наличие третьей позиции | True (трёхпозиционный флажок), False (двухпозиционный флажок) |
| State | Состояние флажка | cbGrayed(серый), cbUnchecked(отключен), cbChecked(включен) |

Панель группы объектов GroupBoxпредназначена для разме­щения на ней группы из нескольких объектов. Панель группы используют для размещения на ней группы из нескольких объектов.Панель группы используют для улучшения дизайна окна программы. Свойства этого объекта аналогичны описанным выше.

С помощью объекта MainMenuсоздают главное меню програм­мы. Вот некоторые свойства главного меню:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Items | Команды меню | Комплексное свойство (зада­ется в диалоговом окне) |
| Tag | Вспомогательная пе­ременная, использует­ся в тексте программы | 0; 8 (целое число) |

С помощью объекта PopupMenuсоздают контекстное меню не­которого компонента. Для «связывания» контекстного меню с конкретным объектом необходимо свойству PopupMenu этого объ­екта присвоить значение имени (Name) соответствующего контек­стного меню. Рассмотрим некоторые свойства контекстного меню:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Alignment | Выравнивание меню относительно точки щелчка по правой кла­више мыши | paCenter(по центру), paLeft(слева), paRight(справа) |
| AutoPopup | Автоматический вы­зов контекстного меню | True (вызовется при щелчке по правой клавише), False (вызовется с помощью мето­да Popup) |

Конкретная команда меню (главного или контекстного) может иметь такие свойства:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Break | Разбивка меню в гори­зонтальном направле­нии | mbNone (без разбивки), mbBarBreak (разбивка с вер­тикальной черточкой), mbBreak (разбивка без верти­кальной черточки) |
| Shortcut | Комбинация "горячих" клавиш для вызова ко­манды меню | Ctrl+A, F8, Ctrl+F10, Shift+F3, Shift+Ctrl +F11, Ctrl+Del |

Практическая часть

Ход работы

1. Загрузите среду визуального программирования Delphi.
2. Поменяйте заголовок (Caption) формы с «Forml» на «Табу­лирование функции» (без кавычек) и увеличьте размеры формы в вертикальном направлении.
3. Смените пиктограмму в левом Верхнем углу формы, указав конкретный файл с рисунком пиктограммы в поле значения свойства Icon (пиктограмма) формы.

Щелкните в строке Icon на Ш, а потом на кнопке Load окна PictureEditor, чтобы получить окно Loadpicture. Откройте папку С: \ ProgramFiles \ Borland \ Delphi х.О \ Images \ Icons, выбери­те графический файл с любой пиктограммой =>Open =>Ok.

Сохраните выполненную на данный момент форму в своей папке (File=»Save АН).

Расположите в форме поля редактирования Editl, Edit2, Edit3 и соответствующие им текстовые поля «Левая грани­ца», «Правая граница», «Шаг», а также текстовое поле для задания вида данной функции y=sin(x) +1 (см. рис. 9).

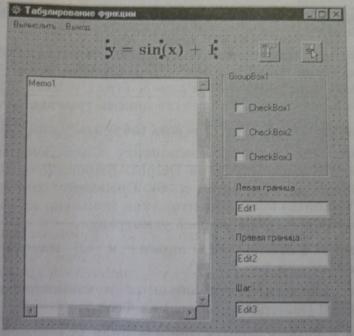


Рис. 9

Замечание. Чтобы быстро вставить в форму несколько одно­типных объектов, щелкните на пиктограмме этого объекта, удерживая нажатой клавишу Shift, а потом - на форме необ­ходимое количество раз. Если случайно вставите лишний объ­ект, то удалить его можно с помощью клавиши Delete. Для от­каза от режима вставки щелкните на изображении стрелки на палитре компонентов. Размер, стиль и цвет шрифта выберите на своё усмотрение так, чтобы форма выглядела как можно лучше. Saveall.

первого объекта и отцентрируйте текстовое поле с ука­занием функции, воспользовавшись окном выравнивания Alignment.

Выделите группу полей редактирования вместе с подписями к ним и выполните команды Edit (редактировать) =>Align (выров­нять) =>Horisontal - Leftsides (горизонтально - левые границы) =Ф Vertical - Spaceequally (вертикально - равномерно) =>Ok. Вы­делите текстовое поле с выражением для функции и выполните команды Edit =>Align =>Horisontal - CenterinWindow (горизон­тально - к центру окна) =>Ok. SaveAll.

7. Вставьте в форму панель группы объектов (объект типа  
GroupBox).

Компонента GroupBox находится на закладке Standard. По­меняйте значение свойства Caption (подпись) этого объекта на слово «Вывод» (без кавычек). Размер, стиль и цвет шрифта выбе­рите на своё усмотрение. Увеличьте панель группы.

8. Вставьте на панель три флажка (объекты типа CheckBox).

Для этого используйте компоненту CheckBoxГ\* закладки Standard палитры компонентов Delphi. Выровняйте флажки, вы­делив их заблаговременно (см. п. 6). Поменяйте значения свойст­ва Caption (подпись) этих объектов, как показано на рис. 9. Стиль и цвет шрифтов выберите на своё усмотрение.

9. Установите флажки «На экран» и «В массив» в режим  
«включен».

Для этого выделите эти объекты и смените значения их свойств Checked (контроль выбора) на True.

10. Вставьте в форму многострочное поле редактирования (объект типа Memo).

Компонента Memo находится на закладке Standard. Уве­личьте размеры поля. Значение свойства ScrollBars (наличие по­лос прокрутки) этого объекта задайте как ssBoth (наличие двух полос - вертикальной и горизонтальной). File =>SaveAll.

11. Задайте начальные значения для полей редактирования ле-  
вой и правой границ аргумента функции и для шага изме-  
нения аргумента, как на рис. 9.

Для этого поменяйте значения свойства Text этих объектов. Для ввода символа десятичной точки примените символ, преду­смотренный операционной системой компьютера.

12. Удалите из многострочного поля редактирования слово  
Memol.

Для этого в окне ObjectInspector смените значение свойства Lines (строки) объекта Memol. Вызовите окно редактирования этого свойства (окно Stringlisteditor - редактор многострочного поля), нажав на кнопку "3.Удалите слово Memol и для заверше­ния работы с этим окном щелкните на кнопке Ок.

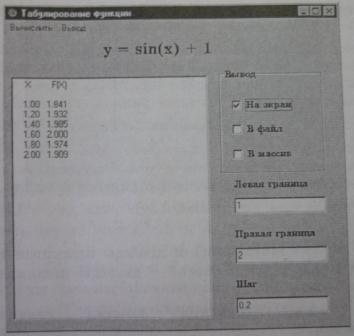


Рис. 10

13. Вставьте в форму главное и контекстное меню (объекты ти-  
па MainMenu и PopupMenu).

Компоненты MainMenuff'lи PopupMenuЧ\* находятся на за­кладке Standard. Расположите их пиктограммы в произвольном месте формы (на этапе выполнения программы они будут невиди­мыми).

14. Введите названия команд главного меню формы (см. рис. 11  
и рис. 12).

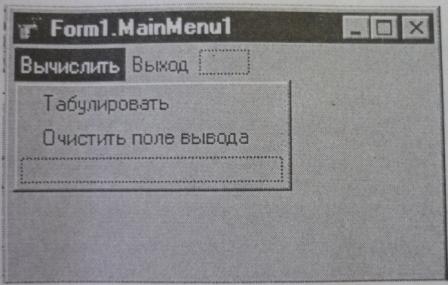


Рис. 11

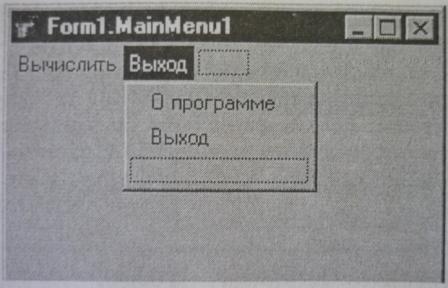


Рис. 12

Выберете объект MainMenu 1 и дважды щелкните на значении его свойства Items. Другой способ - дважды щелкните на самом объекте.

Закройте окно создания команд главного меню Form 1. MainMenu 1.

15. Введите названия команд контекстного меню формы.

Для этого аналогично как для главного меню поменяйте значения свойства Items объекта PopupMenul в окне Forml.PopupMenu 1 (как вызвать это окно см. п. 14). В окне ObjectIspector введите значение свойства Caption - «Очистить поле вывода» (без кавычек, рис.13). Можете придумать и задать еще несколько команд. Закройте окно Forml.PopupMenul. Сохраните форму (SaveAll).

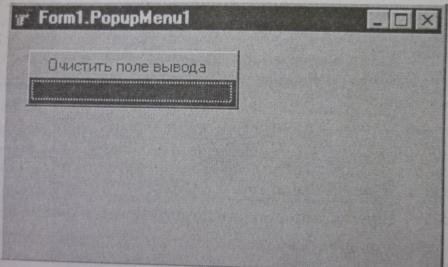


Рис. 13

16. Запрограммируйте команду «Очистить поле вывода» главно­го меню с помощью метода Clear объекта Memol.

Методы объекта - это набор процедур и функций, которые ана­логично свойствам применяют к объекту. Как и в случае свойства, имя объекта от имени метода отделяется точкой. Щелкните по ко­манде меню «Очистить поле вывода». Появится заготовка процеду­ры реакции на событие вызоваэтой команды. В нее запишите ко­манду вызова метода Clear для очистки поля вывода объекта Memol:

procedureTForml.N3Click(Sender: Tobject);

{Здесь N3 - имя команды «Очистить поле вывода», у Вас может быть другой номер, его не исправляйте)

begin

{Вызовем метод объекта Memol, очищающий)  
Memol.Clear{многострочное поле редактирования)  
end; {Теперь щелкните на форме)

17. Запрограммируйте команду «Выход» главного меню,  
воспользовавшись стандартной процедурой Close.

procedureTForml.N6Click(Sender: Tobject); {Здесь N6 - имякоманды «Выход») begin

Close {Закрываем окно программы)

end; {Теперь щелкните на форме)

Сохраните выполненную на данный момент форму в своей пап­ке (File=>SaveAll).

18. Запрограммируйте команду «Очистить поле вывода» контек-  
стного меню.

Дважды щелкните на команде контекстного меню «Очистить поле вывода» в окне Forml.PopupMenul (вызов этого окна см. в п. 14). Текст процедуры очистки такой же, как и для команды главного меню: Memol.Clear (см. п. 16).

19. «Свяжите» контекстное меню PopupMenul с формой Forml.

Щелкните на форме и задайте свойство формыPopupMenu как PopupMenul.

Замечание. Значение свойства формы Menu автоматически ус­танавливается как MainMenul в момент создания главного ме­ню (п. 13).

20. Запустите созданную на этот момент программу и исследуй-  
те ее работу.

Поупражняйтесь с многострочным полем редактирования Memol, вводя и корректируя в нем некоторый текст. Обратите внимание на то, что в этом окне можно выполнять такие же дей­ствия с текстом, как и в текстовом редакторе: выделять фрагмент текста, копировать, переносить или удалять этот фрагмент. Выте-рите текст с помощью команды главного меню «Очистить поле вывода». Еще раз введите текст и вытерите его с помощью кон­текстного меню (для вызова контекстного меню формы нужно щелкнуть правой клавишей мыши на свободном месте формы). Закончите работу программы, щелкнув на кнопке «Выход».

21. Запрограммируйте команду «Табулировать».

Свойство Lines объекта типа Memo комплексное, то есть оно также является объектом со своими свойствами и методами. Ре­зультат работы данной программы - это таблица, которая состоит из нескольких строк. Чтобы добавить в поле Memol новую строку таблицы, надо поменять значения комплексного свойства Lines (строки) с помощью его метода Add (добавить) с одним аргумен­том - символьной строкой: Memol.Lines.Add('cTpoKaсимволов'). Щелкните на команде «Табулировать» из главного меню один раз. Появится заготовка к процедуре, которую заполните так:

procedureTforml.N2Click(Sender: TObject); var x, у, h, a, b : real;

cod : integer;

strl, str2 : string;

begin

{В объект Memol вставляем строку с подписями столбцов

значений аргумента и функции)  
Memo 1. Lines.Add(' X F(X)');

{Получаем числовое значение левой границы) Val(Editl.Text, a, cod);

{Получаем числовое значение правой границы)

Val(Edit2.Text, Ъ, cod);

{Получаем числовое значение шага)

Val(Edit3.Text, h, cod);

х:=а; {Табулируем функцию)

whilex<=b+h/2 do{пока аргумент х не превысит )  
begin {правую границу с определенным запасом)

y:=sm(x)+l; {Вычисляем значения функции) {Формируем символьные строки из значений ) Str(x:8:2,strl); { аргумента и функции) Str(y:8:3, str2); ifCheckBoxl.Checkedthen

{Вставляем строку в поле Memol)

Memol.hmes.Add(strl+str2); x:=x+h; {Увеличиваем аргумент на величину шага) endend;

1. Выполните программу и поупражняйтесь с разными значе­ниями левой, правой границы и шага аргумента. Закройте окно программы «Табулирование функции».
2. Сохраните созданную программу в своей папке.
3. Создайте ехе-файл программы.
4. Закройте Delphi, запустите созданную программу и выпол­ните вычисления для различных входных данных.
5. Продемонстрируйте созданную форму преподавателю. За­кончите работу.

Задача 3.1. Добавьте к контекстному меню команду «Выход» и запрограммируйте ее.

Задача 3.2. Создайте и запрограммируйте еще одно контекст­ное меню, содержащее команды "Убрать все флажки", "Вклю­чить все флажки", "Включить инверсивно". Такое меню должно появляться после щелчка правой клавишей мыши в любой точке панели с флажками. Не забудьте "привязать" это меню (PopupMenu2) к панели с флажками (см. пункт 19).

Задача 3.3. Модифицируйте программу, предусмотрев возмож­ность табулирования функции и ее производной. Выбор варианта (с производной или без неё) осуществить с помощью дополнитель­ного флажка.

Подсказка. Выполните такие действия: • вставьте в форму объект типа CheckBox (флажок), его свойству Caption присвойте значение «Производная», вы­берите для подписи кирилизованный шрифт 12-го разме­ра, выровняйте вставленный объект.

• поменяйте программный код кнопки «Табулировать», ис­пользовав в тексте процедуры ButtonlClick такие коман­ды:

{Объявите переменные strl, str2, str3 как string в начале процедуры}

ifCheckBox.Checkedthen{если флажок установлен}  
Memol.Lines.Add(' X F(X) F "(X)')

{Строка содержит подписи для столбцов  
значений аргумента, функции и ее производной}  
else{Иначе - если флажок не установлен}  
{Строка содержит лишь подписи для столбцов  
значений аргумента и функции}  
Memol. Lines.Add(' X F(X)');

• • •

{А в цикле задаем:}

y:=sin(x)+l; {Вычисляем значение функции}

yl:=cos(x); {Вычисляем значение производной}

if CheckBox.Checked then {Еслифлажокустановлен} begin

Str(x:8:2, strl); Str(y:8:2, str2); Str(yl:8:2, str3); str3 := strl + str2 + str3

end; {Формируем символьную строку из значений аргумента, функ­ции и ее производной, между которыми вставляем несколько пропус­ков}

else{Если флажок не установлен} begin

Str(x:8:2, strl); {Формируем символьную строку} Str(y:8:2, str2); {из значений аргумента и функции) str3 :== strl + str2 end;

• • •

Замечание. Номера объектов'в Вашей программе могут отли­чаться от приведённых выше.

Задача 3.4. Определите количество элементов массива больших 0,5 и меньших 1.

Задача 3.5. Предусмотрите в созданной программе дополни­тельную возможность для определения максимального и мини­мального значений функции.

Подсказка. В теле процедуры ButtonlClick, описывающей про­граммный код кнопки «Табулировать», воспользуйтесь такими командами:

{В начале процедуры:}

max:=sin(a)+l;

• • •

{Вцикле:}

if max<y then max:=y;

• • •

{Послецикла:}

Str(max:8:3, strl); Memol.LinesAdd( 'max=' + strl);

Задача 3.6. Поменяйте процедуру команды «Табулировать» так, чтобы для включенного флажка «В массив» вывод результа­тов выполнялся в одномерный массив.

Задача 3.7. Вставьте в форму объект ChartFX из закладки OCX для построения графика функции. Дважды щелкните на нем и на закладке Appearance из списка Gallery Туре выберите пиктограм­му нужного графика. На закладке 3DView уберите флажок 3D. На закладке DataValues задайте ожидаемые (ориентировочно) макси­мальное и минимальное значения функции. Создайте дополни­тельный пункт меню или вставьте кнопку «Нарисовать график» для получения графика. Запрограммируйте кнопку так, например:

procedureTForml.ButtonlClick(Sender: TObject); vari, n : integer;

x : real;

s : string;

begin

x:=0;

n:=10; {n- количествоточек} ChartFXl.OpenDataEx(COD\_VALUES, 1, n); for i := 0 to n-1 do begin

ChartFXl.Value[i].:= sin(x)+l ; {Формируемлегендуоси OX} str(x:5:2,s);

ChartFX 1. Legend[i]:=s;

x:=x+0.5;

end;

ChartFXl .CloseData(COD\_VALUES);

end;

Задача 3.8. Совместите процессы табулирования функции и построения графика.

# **Практическое занятие №27 Создание наследованного класса.**

ЗАДАЧАОБИРЖЕЦЕННЫХБУМАГ

Таблица строк StringGrid и ее свойства. Кнопки BitBtn и SpeedButton. События и их обработка

Цель работы. Создать бизнес-игру «Биржа ценных бумаг», мо­делирующую ежегодное инвестирование капитала в акции трёх разных предприятий (рис. 15). На момент инвестирования извес­тно, что на конец года рынок может находиться в одном из трех возможных с некоторой вероятностью состояний, от чего будут зависеть прибыли или убытки инвестора. В таблице 3x3 приведе­ны значения прибылей или убытков на каждый вложенный дол­лар для трёх возможных состояний рынка и трех предприятий (например, Hundai, Philips, Bosh). Конкретное состояние рынка проясняется уже после инвестиции. Нужно, учитывая все три во­зможных состояния рынка, инвестировать капитал во все три предприятия с целью добиться максимальной прибыли. Инвести­ции надо делать на протяжении нескольких лет. Если это делать удачно, можно ожидать прибыль (увеличение капитала), иначе -предприятия ждут убытки.

Научиться применять таблицу текстовых строк StringGrid, а также кнопки типа BitBtnи SpeedButtonдля выполнения вы­числений или стандартных действий.

Ознакомиться с понятием события в Delphi.

Указание. Все фрагменты программы данной работы рекомен­дуется заранее разместить в файле и разрешить студентам копи­ровать их в свою программу через буфер обмена.

Теоретические часть

Объект StringGridпредназначен для создания в форме двумерной таблицы символьных строк. Кроме изученных свойств, таблица символьных строк имеет еще и та­кие:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Свойство | | Описание свойства | | Примеры значений | |
| ColCount | | Количество столбцов таблицы | | 3; 7 | |
| RowCount | | Количество строк таблицы | | 5; 8 | |
| FixedCols | Количество фиксированных столбцов таблицы, которые не прокручиваются слева | | 0; 3 | |
| FixedRows | Количество строк в шапке табли­цы, которые не прокручиваются вверх | | 1; 2 | |

Объект BitBtn - кнопка с рисунком и определенным типом действия. Стандартный набор файлов с рисунками для кнопок находится в папке С:\ ProgramFiles\ Borland\ Delphix.0\ Images\ Buttons. Этот объект владеет такими новыми свойствами:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Glyph | Рисунок из файла на кнопке | Адрес файла задается в диалоговом окне |
| Kind | Тип стандартного действия | bkClose (закрывает окно), bkCancel (кнопка "Отменить" диалогового окна), bkNo (кнопка "Нет" диало­гового окна) |

Объект SpeedButtonприменяют для создания кнопки панели инструментов. Рассмотрим такие свойства объекта:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Margin | Расстояние в пикселях между рисунком и бли­жайшим к нему краем кнопки | -1 - рисунок с текстом центрируется, 8 - восемь пикселей между краем кнопки и рисунком |
| Layout | Взаимное расположение рисунка и текста на кнопке | blGlyphLeft (рисунок воз­ле левой границы, текст справа), blGlyphRight (наоборот) |
| Spacing | Расстояние в пикселях между рисунком и тек­стом | 6 - шесть пикселей между рисунком и текстом на кнопке |

Практическая часть

Ход работы

1. Загрузите среду визуального программирования Delphi.

2. Поменяйте заголовок (Caption) формы с « Forml\* на «Биржа ценных бумаг» и увеличьте размеры формы

4. Откажитесь от возможности менять размеры окна програм­мы, указав значение bsDialog свойства BorderStyle формы.

4. Сохраните выполненную на данный момент форму в своей папке.

5. Расположите в нижней части формы две пиктографических кнопки типа SpeedButton и командную кнопку окончания работы программы BitBtn (см. макет на рис. 14 и окно рабо­чей программы на рис. 15).

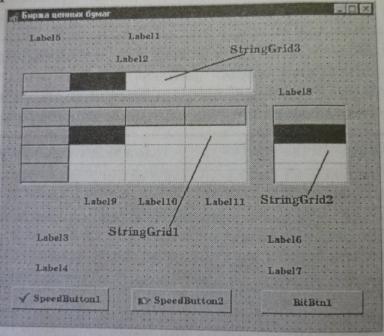


Рис. 14

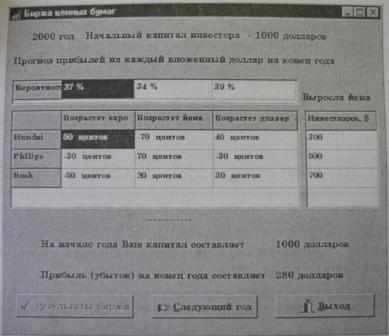


Рис. 15

Для этого примените компоненты SpeedButtonШи BitBtn из закладки Additional палитры компонентов. На кнопках SpeedButton можно расположить пиктограмму из специального набора пиктограмм - графических файлов формата bmp. Кнопки типа BitBtn владеют свойством Kind (стандартное действие) с возмож­ными значениями bkClose (Закрыть), bkCancel (Отменить), bkYes (Да), bkNo (Нет), bkHelp (Помощь) и другими. На них также мо­гут быть расположены пиктограммы. Если задать стандартное действие, то данную кнопку программировать не нужно. Задайте следующие значения свойств вставленных объектов:

|  |  |  |  |
| --- | --- | --- | --- |
| Свойство | Значение | | |
| Name | SpeedButton1 | SpeedButton2 | BitBtnl |
| Caption | &Биржевые ре­зультаты | &Следующий год | &Выход |
| Glyph (файл с пиктограм­мой) | С:\ Program Files \ Borland \ Delphi х.О \ Im­ages \ Buttons \ check.bmp | С:\ Program Files \ Borland \ Delphi х.О \ Im­ages \ Buttons \ arrow3r.bmp |  |
| Kind |  |  | BkClose |

Замечание. Символ & в заголовке кнопки указывает на то, что эту кнопку можно нажать также и с помощью «горячей» ком­бинации клавиш Alt + буква заголовка, перед которой стоит этот символ.

Вставьте в форму три таблицы текстовых строк StringGrid (см. рис. 14).

Воспользуйтесь компонентой StringGridна закладке Additional. Задайте следующие значения свойств этих объектов:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Свойство | Значение | | | | | |
| Name | StringGrid1 | | StringGrid2 | | StringGrid3 | |
| ColCount  (количество столбцов) | 4 | | 1 | | 4 | |
| FixedCols | |  | |  | |  | |
| (количество фик­сированных столбцов) | | 1 | | 0 | | 1 | |
| RowCount | |  | |  | |  | |
| (количество строк) | | 4 | | 4 | | 1 | |
| FixedRows | |  | |  | |  | |
| (количество фик­сированных строк) | | 1 | | 1 | | 0 | |
| Options\* goEditing  (возможность ре­дактировать) | | False (отсутствует) | | True (имеется) | | False (отсутству­ет) | |

\* Замечание. Чтобы открыть список свойства Options (Пара­метры) объекта StringGrid2, надо дважды щелкнуть на этом слове в окне ObjectInspector данного объекта.

7. Разместите на форме текстовые поля так, как показано на рис. 14.

Задайте следующие значения свойств этих объектов: Для Label5, Label6, Label8:

Caption : (удалитесодержимое)

Visible : True.  
Для Label9, LabellO, Labell:

Caption : (несколько знаков «минус»)

Visible : False.  
Для Label 7:

Caption : (удалите содержимое)

Visible : False.

Все другие поля - видимые, значения их свойства Caption по­казаны на рис. 15.

8. В окне текста программы перед разделом implementation введите описание констант и переменных программы:

const

N=5;

{Массив прогноза прибыли}  
Р : array[0..N-1,1..3,1..3] ofinteger =  
(((40,-40,30), (-20, 10, 10), (-10,40,-30)), {r=(0-l)}

((50,-70,40), (-30, 70,-30), (-50,20, 30)), {r=(l-2)}

((30,-70,70), (-30, 70,-30), (20,10,-50)), {r=(2-3)}

((-50,-30,70), (50,-50, 70), (30,50,-70)), {r=(3-4)}

((30,-70,70), (-30, 70,-30), ( 40, 0,-50))); {r=(4-5)}

{Массив вероятностей} Prob :array[0..N-1,1..3] ofinteger= ((22,41,37),(27,34,39),(47,33,20),(10,53,37),(41,34,25)); var

Forml :TForml; {Эта запись уже есть в заготовке модуля} capital, year, profit, г : integer; {Инвестированные суммы} money : array[0..2] ofinteger;

9. Запрограммируйте кнопки SpeedButtonlи SpeedButton2 следующим образом:

procedure TForml.SpeedButtonlClick(Sender: Tobject);

{Кнопкарезультатовгода} var first, second, variant, i, code : integer;

s : string;

begin

randomize;

{Случайныйвыборвозможногосостояниярынка} variant := random(lOO);

{Первый и второй разделители отрезка вероятностей) first := Probfr, 1 j; second :=first + Prob[r, 2];

{Начальное значение прибыли перед суммированием}

profit := 0;

for i := 0 to 2 do

vaI(StringGrid2.Cells[0,i + 1], money[i], code); {Получаемчисловыезначениядляинвестицийи

заполняем ими массив money} {Вычисление прибыли для первого варианта) ifvariant<firstthenbegin

for i := 0 to 2 do {Суммированиепопредприятиям} profit := profit + moneyfi] \* P[r, i+1, 1];

{Подчеркивание выбранного варианта} Label9.Visible := True;

{Информационная строка} Label8.Caption := 'Возросло евро' end;

{Вычисление прибыли для второго варианта состояния рынка} if(variant>= first) and(variant<second) thenbegin

fori := 0 to 2 do

{Суммирование no предприятиям} profit := profit + money[i] \* P[r, i+1, 2];

{Подчеркивание выбранного варианта} LabellO.Visible := True;

Label8.Caption := 'Выросла йена'{Информационная строка}

procedureTForml.SpeedButton2Click(Sender: Tobject); {Кнопка перехода на следующий год} begin

year := year + 1; {Следующийгод}

capital := capital + profit; {Обновляемкапитал}

{Прибыль к началу года равна нулю}

profit := 0;

{Вызываем процедуру ежегодного обновления формы) Refresh;

{Выключаем подчеркивания варианта состояния} Label9.Visible := False; {рынка} LabellO.Visible := False; Label 11.Visible := False; {Гасим информацию о прибыли} Label7.Visible := False; end;

10. В окне текста программы после раздела implementation вве­дите описание метода Refresh формы.

procedureTForml.Refresh; var i, j : integer; s : string;

begin

Str(capital, s);

Forml.Label6.Caption := s + долларов'; Str(year, s);

Forml.Label5.Caption := s + ' год'; Str(profit, s);

Forml.Label7.Caption := s + долларов'; Forml.Label8.Caption := 'Введитесумму'; Forml.SpeedButton2.Enabled := False; Forml.SpeedButtonl.Enabled := False; randomize;

r := random(N); {Выборинвестиционногоклиматаг} for i:=l to 3 do for j := 1 to 3 do begin

Str(P[r, j, i], s);

Forml.StringGridl.Cellsfi, j] := s + ' центов'; end;

for i := 1 to 3 do begin

Str(Prob[r, i], s);

Forml.StringGrid3.Cells[i, 0] := s + ' %'; Forml.StringGrid2.Cells[0, i] := "; end end;

Строку

procedure Refresh;

вставьтевлюбомместераздела type TForml = class(TForm) ... вокнемодуля.

11. Запрограммируйте процедуру создания формы FormCreate.

Для этого дважды щелкните на форме и полученную заготовку процедуры FormCreate заполните следующим образом:

procedure TForml.FormCreate(Sender: TObject); begin

capital := 1000; year := 2000; profit := 0; Refresh;

StringGridl.Cells[l, 0] := 'Возрастетевро'; StringGridl.Cells[2, 0] := 'Возрастетйена'; StringGridl.Cells[3, 0] := 'Возрастетдоллар';

StringGridl.Cells[0, 1] StringGridl.Cells[0, 2] StringGridl.Cells[0, 3] StringGrid2.Cells[0, 0] StringGrid3.Cells[0, 0] end;

1. Сохраните выполненную на данный момент работу в своей папке.
2. Запрограммируйте событие ввода/редактирования текста в таблицу StringGrid2 так, чтобы в момент ввода денежных сумм в первую и вторую ячейки в третьей ячейке отобра­жался остаток капитала (поскольку каждый год инвестиру­ют весь имеющийся капитал).

Активизируйте объект StringGrid2 и в окне ObjectInspector откройте закладку Events этого объекта. Здесь следует выбрать событие, для которого вы хотите описать последовательность действий - реакцию на событие. Выберите событие OnSetEditText (ЕслиТекстРедактируется) и дважды щелкните в поле справа от неё. Откроется окно программы с заготовкой процедуры StringGrid2SetEditText. Заполните ее следующим образом:

procedureTForml .StringGrid2SetEditText(Sender:

TObject; ACol, ARow: Longint; const Value: string); var i, code : integer; s : string; begin

{Считываем числа с первой и второй ячеек таблицы инвестиций) fori := 0 to1 do

val(StringGrid2.Cells[0, i + 1], money[i], code); {Вычисляем остаток капитала и инвестируем его в третье предприятие) money[2] := capital - money[0] - money[l]; Str(money[2], s); StringGrid2.Cells[0, 3] := s; SpeedButtonl.Enabled := True; end;

Замечание. Заготовки для событий-процедур каждого объекта можно вызвать, дважды щелкая по объекту на макете формы, как это делалось для кнопок. Для большинства объектов это событие - OnClick (ЕслиЩелкнуть), в том числе и для StringGrid. Поэтому мы были вынуждены «вручную» выбрать нужный для нашего случая тип события - OnSetEditText.

14. Запустите созданную программу и исследуйте ее работу.

Введите денежные суммы инвестиций в ячейки для первого и второго предприятий и наблюдайте за изменением третьей суммы. Убедитесь в ее правильности. Щелкните на кнопке «Биржевые ре­зультаты», просмотрите результаты финансового года, щелкните на кнопке «Следующий год». Сыграйте в эту игру несколько раз и за­кончите работу программы, щелкнув на кнопке «Выход».

1. Создайте ехе-файл программы.
2. Закройте Delphi, запустите созданную программу, сыграйте и попробуйте достичь хороших результатов.
3. Продемонстрируйте созданную форму преподавателю. За­кончите работу.

Задача 4.1. Информационную надпись «Прибыль (убыток) на конец года составляет» поменяйте на «Прибыль на конец года составляет» или «Убыток на конец года составляет» в зависимо­сти от того возрос или уменьшился капитал игрока в конце года по сравнению с начальным.

Задача 4.2. Обеспечьте вывод денежной суммы прибыли зеле­ным цветом, а убытка - красным.

Задача 4.3. Не дайте возможности игроку ввести отрицатель­ные денежные суммы в таблицу инвестиций.

Задача 4.4. Вычислите и выведите в форме все денежные сум­мы с точностью до цента.

# **Практическое занятие №28 Перегрузка методов.**

ЗАДАЧАОТЕЛЕФОННОМСПРАВОЧНИКЕ

Новые объекты: многостраничный блокнот, комбинированный список. Создание новых окон и стандартных диалогов. Работа с файлами записей в ObjectPascal.

Цель работы. Создать программу для работы с базой данных -телефонным справочником, сформированным в виде файла запи­сей. В каждой записи определены поля: фамилия и имя (строки длиной по 20 символов каждая), адрес и номер телефона (строки длиной 100 и 15 символов соответственно). Необходимо реализо­вать следующие операции: создание и удаление записи, сохране­ние и считывание файла записей с диска, редактирование, поиск и сортировку данных в алфавитном порядке, просмотр записей и навигацию по базе.

Ознакомиться с новыми объектами: многостраничным блокно­том (PageControl), страницей блокнота (TabSheet), комбиниро­ванным списком (ComboBox), диалоговым окном (AboutBox), ста­ндартными диалогами открытия и сохранение файлов (OpenDialogи SaveDialog).

**Теоретические часть.**

Многостраничный блокнот (PageControl) находится на особом виде формы TabbedPages (об­ъект PagesDlg, заголовок "TabbedNotebookDialog"). Форму соз­дают командами File =>New...=> Закладка Forms =>TabbedPages =>• Ok. После её создания ненужную форму "Forml" в случае не­обходимости можно удалить с помощью кнопки JiJ (Removeunit) окна менеджера проекта (ProjectManager). Вызов окна менеджера проекта осуществляется командами View =>ProjectManager. В данной работе активизацию объектов PageControl и PagesDlg мо­жно осуществлять с помощью инспектора объектов. Свойства объ­екта PagesDlg совпадают со свойствами обычной формы Form. Ра­ссмотрим новые свойства блокнота PageControl:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| ActivePage | Имя активной страни­цы | TabSheet7 |
| MultiLine | Возможность размещать закладки страниц в не­сколько строк | True (закладки в не­сколько строк), False(за­кладки в одну строку) |

Свойства отдельной страницы блокнота (объекта TabSheet) аналогичны описанным в предыдущих разделах.

Объект ComboBoxпредназначен для создания выпадающего списка. Его новые свойства такие:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| DropDown Count | Количество строк в выпадающем списке, которые видимы без использования полос прокручивания | 7; 3 |
| Sorted | Сортировка списка по алфавиту | True(список сортируется), False(не сортируется) |
| Style | Стиль оформления и использование списка | csOwnerDrawVariable(за­данный программистом), csDropDown(стандартный) |

Стандартные диалоги открытия (объект OpenDialog) и со­хранение файлов (объект SaveDialog), - это невидимые на форме объекты, предназначенные для создания стандартных в Windows9х окон: "Openfile" и "SaveFileas" - в момент вызова их из про­граммного кода ' проекта командой OpenDialogl .Execute или SaveDialogl .Execute. Имя выбранного файла возвращается в программу свойством FileName этих объектов.

**Практическая часть**

Ход работы

1. Загрузите среду визуального программирования Delphi.
2. Откройте новую форму с многостраничным блокнотом.

File => New..-^Закладка Forms => Tabbed Pages => Ok.

1. Выделите и удалите клавишей Delete панель (Panel) с кноп­ками Ok, Cancel и Help в нижней части открытой формы.
2. Поменяйте заголовок формы с «TabbedNotebookDialog\* на «Телефонный справочник» и поменяйте стиль границы окна (BorderStyle) на Sizeable (способный менять размер).

Для этого сначала в окне ObjectInspector в списке компонен­тов текущей формы выберите объект PagesDlg (окно формы с многостраничным блокнотом). Только теперь можно поменять его заголовок и стиль.

5. Выделите (щелкнув внутри страницы мышью) и удалите  
(Delete) третью страницу блокнота TabSheet3.

Будьте внимательны и не удалите весь многостраничный блок­нот - объект PageControll. В случае необходимости Вы можете отказаться от неправильного действия командой Edit =>Undelete. Добавить новую страницу в блокнот можно командой NewPage контекстного меню объекта PageControl (блокнот) или TabSheet (страница блокнота).

6. Замените заголовки (свойство Caption) первой и второй  
страниц блокнота (объектов TabSheetl и TabSheet2) на  
«Список всех записей» и «Редактирование и поиск записей»  
соответственно.

7. Заполните обе страницы формы визуальными объектами так, как это показано на рис. 16 и рис. 17.

|  |  |
| --- | --- |
| Свойство | Значения |
| ColCount | 4 |
| FixedCols | 0 |
| RowCount | 100 |
| FixedRows | 1 |
| Options =>goEditing | True |

Для создания объекта Комбинированный Список воспользуй­тесь кнопкойИ(ComboBox) на закладке Standard палитры ком­понентов. Кнопки «Предыдущий» - «Следующий» называются Buttoml - Buttom4 соответственно. Очистите поля свойства Text объектов Editl - Edit5 и ComboBoxl. Установите свойство Style объекта ComboBoxl в csDropDownList (выпадающий список). За­дайте такие значения свойств объекта StringGridl:

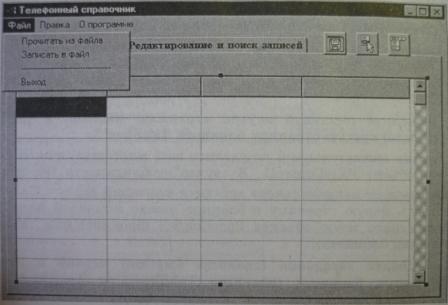


Рис. 16

8. Сохраните выполненную на данный момент форму в своей папке.

9. С помощью мыши задайте ширину столбцов таблицы, на-  
пример так, как показано на рис. 16.

Для этого мышью передвиньте границу между двумя столбца­ми в первой (серой) строке таблицы StringGridl. Не допускайте появления горизонтальной полосы прокручивания в таблице.

10. Добавьте на форму главное меню и введите названия его  
команд, как показано на рис. 16 и рис. 17.

Для того, чтобы провести линию-разделитель, между команда­ми меню «Записать в файл» и «Конец» создайте дополнительный пункт с заголовком «-» (символ «минус»).

Замечание. Подменю «А->Я» и «Я->А» команды «Упорядо­чить по алфавиту» создают с помощью комбинации клавиш Ctrl-Кстрелка вправо> или используют контекстное меню CreateSubmenu этой команды. Удаление ненужной команды выполняют клавишей Delete.

11. Создайте окно формы «О программе» (см. рис. 18).

Прибавьте к проекту новое окно формы типа Aboutbox (File =>New... =>Закладка Forms=>Aboutbox=>Ok). Поменяйте тексты и заголовок формы, указав свою фамилию, например, так, как показано на рис. 18. Поменяйте изображения в окне по своему усмотрению, сменив свойство Picture объекта Programlcon (см. п.13 практической работы № 1).

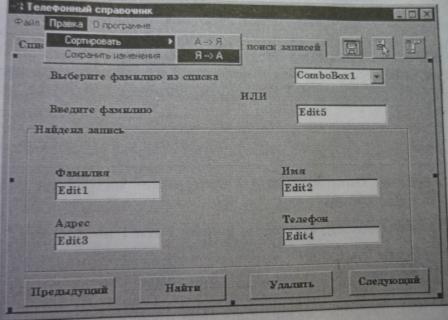


Рис. 17

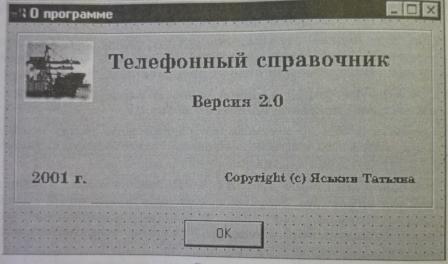


Рис. 18

12. Запрограммируйте команду «О программе» главного меню окна «Телефонный справочник» так, чтобы в момент ее вы­полнения на экране появлялось только что спроектирован­ное окно.

В заготовку процедуры обработки этой команды введите строку для вызова метода Show (показать) объекта AboutBox:

AboutBox .Show

13. Запрограммируйте кнопку Okокна «О программе».

Для этого возвратитесь к форме «О программе» и в процедуре OKButtonClick кнопки Ok введите команду закрытия активной формы:

Close

Замечание. Окно формы «О программе» находится под други­ми окнами Delphi.

14. Удалите ненужную пустую форму Forml из проекта.

Для этого вызовите менеджер проекта командой View =>ProjectManager (см. рис. 19). Выделите строку со значением Forml в поле Form инажмите клавишу Delete. Не сохраняйте эту форму (No). Закройте окно менеджера проекта.

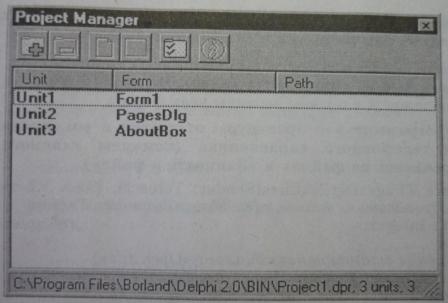


Рис. 19

1. Запрограммируйте команду Файл => Выход главной формы.
2. Сохраните работу и запустите проект на выполнение.

Убедитесь в правильности работы закладок, комбинированного списка, наличия возможности редактировать поля таблицы и пе­ремещаться по ним с помощью полосы прокручивания. Вызовите окно «О программе» и закройте его клавишейОк. Закончите ра­боту командой Файл => Выход.

17. Опишите глобальные переменные и типы данных програм-  
мы: тип записи телефонного справочника (info), файл запи-  
сей (F), переменную типа info (г) и целую переменную коли-  
чества записей (N).

Для этого перед разделом реализации implementationдобавьте такие команды:

info = record {Типзаписи} familia, imja : string[20]; adres : string[100]; tel : string[15]; end;

{Эта строка уже есть в заготовке модуля}

varPagesDlg : TPagesDlg;

F : file of info; {Файлзаписей}

г :info; {Одна запись}

N :integer; {Количество записей}

18. Добавьте на форму два объекта - стандартные диалоги  
Windows для открытия и сохранения файлов.

Кнопки Ш (OpenDialog) иВ(SaveDialog) находятся на заклад­ке Dialogs панели компонентов Delphi. На рис. 16 и 17 эти объек­ты расположены в правом верхнем углу формы.

19. Запрограммируйте процедуры открытия и сохранения фай-  
лов телефонного справочника (команды главного меню  
«Прочитать из файла» и «Записать в файл»).

procedureTPagesDlg.N3Click(Sender: Tobject); {Здесь N3 - номер ко­манды «Прочитать из файла», у Вас может быть другой номер)

vari : integer;

begin

{Выполнениестандартногодиалога «Ореп file»} if OpenDialogl.Execute then begin

{F — названиевыбранногодиалогомфайла (свойство FileName объекта OpenDialogl)} AssignFile(F, OpenDialogl .FileName); Reset(F);

N := 0; {Читаем с начала файла} {Считываем все записи из файла) whilenoteof(F) dobegin

{В переменную г считываем текущую запись) Read(F, г);

{Фамилия — в первый столбец таблицы)

StringGridl.Cells[0, N+l] := r.familia;

{Имя - во второй столбец таблицы)

StringGridl.Cellsfl, N+l] := r.imja;

{Адрес - в третий столбец таблицы)

StringGridl.Cells[2, N+l] := r.adres;

{Телефон - в четвертый столбец таблицы)

StringGridl.Cells[3, N+l] := г. tel;

N := N + 1; {Увеличиваем общее число записей N)

end;

CloseFile(F); {Закрываем файл) {Описанная ниже процедура заполняет поля редактирования второй страницы формы.)

NewEdit(ComboBoxl); {Описанная ниже процедура заполняет комбинированный список второй страницы формы)

NewComboBox; endend;

procedureTpagesDlg.N4Click(Sender: TObject); {Здесь N4 - номер ко­манды «Записать в файл», у Вас может быть другой номер)

vari : integer;

begin

if SaveDialogl.Execute then begin

AssignFile(F, SaveDialogl .FileName); Rewrite(F); i := 1;

while StringGridl.Cells[3, i] <> " do begin

r.familia := StringGridl.Cells[0, i];

r.imja := StringGridl.Cells[l, i];

r.adres := StringGridl.Cells[2, i];

r.tel := StringGridl.Cells[3, i];

Write(F, r); i := i + 1;

end; CloseFile(F); end end;

20. Напишите процедуру создания формы.

В окне инспектора объектов активизируйте объект PagesDlg (окно формы с многостраничным блокнотом) и дважды щелкните в строке OnCreate (МоментСоздания) закладки Events этого объ­екта. Сформируйте процедуру создания главной формы програм­мы:

procedureTpagesDlg.FormCreate(Sender: TObject); begin

{Заполняем заголовок таблицы) StringGridl.cellsfO, 0] := 'Фамилия'; StringGridl.cells[l, 0] := 'Имя'; StringGridl.cells[2, 0] := 'Адрес'; StringGridl.cells[3, 0] := 'Телефон'; {Создаем пустую строку в комбинированном списке) ComboBox 1.Items.Add(«);

{Активизируем эту строку на случай создания нового справочника} ComboBox 1. Itemlndex:=0; end;

21. Напишите процедуру заполнения полей редактирования  
NewEdit.

Вызов этой процедуры осуществляется, в частности, в момент выбора фамилии для поиска, то есть в момент смены активной строки в комбинированном списке. Поэтому для инициализации заготовки этой процедуры активизируйте на закладке Events окна ObjectInspector объекта ComboBox 1 строку OnChange (МоментЗа-мены) и в правом столбце укажите название процедуры реакции на это событие: NewEdit. Дважды щелкните мышью на введенном названии и заполните заготовку этой процедуры следующим обра­зом:

procedureTPagesDlg.NewEdit(Sender: Tobject); var i : integer; begin

{i - номер строки данных в таблице} i •:= ComboBox 1.Itemlndex + 1; ifi> 0 thenbegin

{Для значащих позиций комбинированного списка}

Editl.Text := StringGridl.Cells[0, i];

{копируем данные i-го записи с}

Edit2.Text := StringGridl.Cells[l, i];

{первойстраницыформыв}

Edit3.Text := StringGridl.Cells[2, i];

{соответствующие поля на второй странице}

Edit4.Text := StringGridl.Cells[3, i];

end

end;

22. Напишите процедуру заполнения комбинированного списка  
NewComboBox.

Сначала к описанию типа класса (объекта) TPagesDlg (много­страничной формы) в разделе interfaceдобавьте декларацию:type

{Эта запись уже есть в заготовке модуля} TPagesDlg = class(TForm)

{Необходимо вставить} procedureNewComboBox;

В раздел implementationдобавьте описание этой процедуры:

procedureTPagesDlg.NewComboBox; var i, j : integer; begin

{Запоминаемактивнуюпозициюкомбинированного} j := ComboBox 1.Itemlndex; {списка} {Очищаемкомбинированныйсписок} ComboBoxl .Clear;

fori := 1 to N do {Длявсех N записей...}

{В обновленный комбинированный список добавляем все фамилии из первого столбца таблицы} ComboBox 1.. .Items.Add(StringGridl.Cells[0, I]); {Переходим на позицию j} ComboBoxl.Itemlndex := j; end;

1. Сохраните работу, запустите программу, заполните таблицу данными и убедитесь в том, что программа правильно запи­сывает на диск и считывает с диска файлы телефонных справочников.
2. Запрограммируйте команду главного меню «Сохранить из­менения в списке записей».

procedureTPagesDlg.N6Click(Sender: Tobject); {Здесь N6 - номер команды «Записать в файл», у Вас может быть другой}

vari : integer;

begin

{Определяем помер активной записи}

i :'= ComboBoxl.Itemlndex + 1;

ifPageControll.ActivePage=TabSheet2 then

begin \_ {Если активная вторая страница,}

StringGridl.CellsfO, i ] := Editl.Text;

{то вносим изменения в первую }StringGridl.Cells[l, i] := Edit2.Text; StringGridl.Cells[2, i] := Edit3.Text; StringGridl.Cells[3, i] := Edit4.Text; end

{Иначе — вносим изменения во вторую страницу) elseNewEdit(ComboBoxl);

{В обоих случаях обновляем комбинированный список)

NewComboBox;

end;

25. Запрограммируйте кнопку перехода на предыдущую запись  
« Предыдущий ».

procedureTpagesDlg.ButtonlClick(Sender: Tobject); begin

ifComboBoxl.Itemlndex = 0 then

{Зацикливаемсписок) ComboBoxl.Itemlndex := N-l

else

{В середине списка показатель уменьшается на единицу) ComboBoxl.Itemlndex := ComboBoxl.Itemlndex - 1;

{Обновляем панель «Найдена запись»)

NewEdit(ComboBox 1);

end;

26. Аналогично запрограммируйте кнопку «Следующий».

Программный код созданной процедуры запишите в отчет.

27. Сохраните программу и убедитесь в правильности ее работы.

Запустите программу, считайте с диска файл телефонного спра­вочника, выберите какую-то фамилию в комбинированном списке и поменяйте ее в поле редактирования «Фамилия». Убедитесь в том, что синхронные изменения происходят и в комбинированном списке. Сохраните изменения и используйте кнопки «Предыдущий» и «Следующий» для навигации по записям, убе­дитесь в зацикливании списка. Удалите одну запись кнопкой «Стереть», сохраните изменения и перейдите на первую страницу. Отредактируйте данные в таблице, сохраните справочник в файле на диске, просмотрите окно «О программе», закончите работу с программой.

Задача 5.1. Запрограммируйте кнопку «Удалить» так, чтобы после нажатия на нее удалялись данные из панели «Найдена за­пись» и из комбинированного списка.

Задача 5.2. Запрограммируйте кнопку «Найти» так, чтобы на панели «Найдена запись» отображалась информация о лице, фа­милия которого указана в поле редактирования Edit5 «Введите фамилию». Если такой фамилии нет, то выведите сообщения об этом в отдельном окне.

Задача 5.3. Запрограммируйте команды подменю «А->Я» и «Я—>А» пункта главного меню «Сортировать» так, чтобы фами­лии в таблице и комбинированном списке были отсортированы в алфавитном или обратном порядке.

Задача 5.4. Удалите команду «Сохранить изменения в списке записей» из главного меню и обеспечьте сохранение изменений в момент редактирования таблицы и полей на панели «Найдена за­пись».

Задача 5.5. В случае попытки открыть несуществующий файл телефонного справочника обеспечьте появление окна с соответст­вующим сообщением.

Задача 5.6. В момент выхода из программы обеспечьте появ­ление окна для сохранения файла телефонного справочника в слу­чае, если в нем были сделаны изменения.

# **Практическое занятие №29 Создание проекта с использованием кнопочных компонентов.**

ИЗУЧАЕМАНГЛИЙСКИЙ

Работа с фигурами, таймером, мультимедийным проигрывате­лем, двойными кнопками и индикатором состояния. Перетяги­вание объектов в окне программы

**Цель работы.** Разработать программу для проверки знаний конкретных английских слов. Пользователь должен за ограни­ченное время методом перетягивания расположить три (или четы­ре серии по три) рисунка под соответствующими английскими словами (см. рис. 21). Применить индикатор времени выполнения задачи, звуковые эффекты и привести анализы-итоги тестирова­ния.

Приобрести навыки работы с такими объектами: геометриче­ская фигура (Shape), таймер (Timer), мультимедийный проигры­ватель (MediaPlayer), двойная кнопка с полем редактирования (SpinEdit), индикаторы состояния (Gaugeи ProgressBar).

**Теоретические часть**

Рассмотрим назначения и свойства некоторых новых объектов. Геометрическая фигура (Shape, пик­тограмма \Ш\ на закладке Additional) предназначена для изобра­жения элементарных геометрических фигур и имеет, в частности, такие свойства:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| Brush | Характеристики цвета (Color) и стиля (Style) заливки | Brush-Color: clMaroon  Brush-Style: bsSolid (сплошной), bsVertical |
| Shape | Форма фигуры | stRoundRect(прямо­угольник с округленны­ми краями), stEllipse, stSquare |
| Реп | Характеристики границы фигуры | Комплексное свойство |

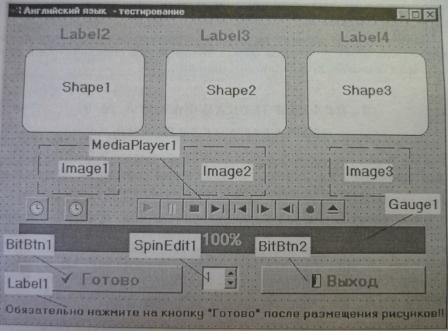


Рис. 20

Таймер (Timer, пиктограмма Я на закладке System) исполь­зуют для повторения фрагмента программы с определенной пе­риодичностью. Соответствующий фрагмент располагают в теле процедуры обработка события OnTimer таймера. Периодичность включения таймера в милисекундах задают свойством Interval.

Мультимедийный проигрыватель (MediaPlayer, пиктограмма на закладке System) предназначен для проигрывания видео и аудиофайлов. Управлять проигрывателем, можно как с помощью традиционных кнопок Play, Pause, Stop, Next и т.п. на этапе вы­полнения программы, так и из программного кода путем выпол­нения методов этого объекта, например:

MediaPlayerl.FileName :=

'полное имя видео- или аудиофайла';

MediaPlayerl .Open; MediaPlayerl .Play;

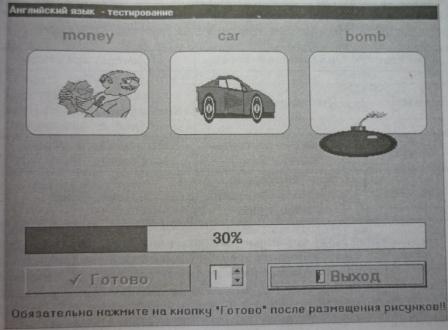


Рис.21

Двойная кнопка с полем редактирования (SpinEdit, пикто­грамма на закладке Samples) предназначена для ввода и кор­ректировки целочислового значения некоторой величины во вре­мя работы программы. Свойства кнопки:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значений |
| EditorEnabled | Возможность прямого редактирования | True, False |
| Enabled | Возможность любого редактирования | True, False |
| Increment | Шаг изменения | 1; 4 (целое число) |
| MaxValue | Максимальное значение | целое число |
| Min Value | Минимальное значение | целое число |

Индикатор состояния (Gauge, пиктограмма на закладке Samples, ProgressBar, пиктограмма на закладке Win95) исполь­зуют для наглядной демонстрации состояния выполнение некото­рого процесса. Рассмотрим три свойства индикатора Gauge:

|  |  |  |
| --- | --- | --- |
| Свойство | Описание свойства | Примеры значение |
| Kind | Тип индикатора | gkHorizontalBar (гориз. строка), gkPie (круговая диаграмма), gkText (про­центное отображение) |
| Progress | Отображает состояние индикатора | Целое число между MaxValue и MinValue |
| ShowText | Дополнительно отображает состояние в процентах | True, False |

Замечание. Для создания сокращенной версии программы (лишь для одной серии из трёх рисунков и без показа результа­тов тестирования), выполните пункты хода работы, обозначен­ные одной звездочкой (\*). Чтобы получить полную версию про­граммы, выполните пункты с двумя звездочками (\*\*). Пункты хода работы без звездочек выполните в обоих случаях.

**Практическая часть**

Ход работы

1. Загрузите среду визуального программирования Delphi.
2. Поменяйте заголовок (Caption) формы с «Forml» на «Анг­лийский язык - тестирование».
3. Откажитесь от всех системных кнопок формы, задав значе­ния False во всех позициях свойства Borderlcons: biSystemMenu, biMinimize, biMaximize и biHelp.

Работа с программой будет завершаться нажатием на кнопку Выход.

4. Расположите на форме объекты так, как показано на рис. 20. Задайте такие свойства объектов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Объект | | Свойство | Значение | |
| Timer 1 | | Interval | 500 | |
| Timer2 | | Interval | 100 | |
| MediaPlayerl | | Visible | False | |
| Gaugel | | BackColor | clInfoBk | |
| Color | clBtnFace | |
| Font | MS Sans Serif, Bold, 14, Yellow | |
| ForeColor | clGreen | |
| Kind | gkHorizontalBar | |
| MaxValue | 100 | |
| MinValue | 0 | |
| Progress | 100 | |
| inEditl | | EditorEnabled | False | |
| MaxValue | 10 | |
| MinValue | 1 | |
| Value | 5 | |
| BitBtnl | | Glyph | C:\ Program Files\ Borland\ Delphi x.O \IMAGES \BUTTONS\ check.bmp | |
| Caption | Готово | |
| BitBtn2 | | Glyph | C:\ Program Files\ Borland\  Delphi x.O \IMAGES \BUTTONS\ dooropen.bmp | |
| Caption | Выход | |
| Shapcl, Shape2, | | Brush-Color | clInfoBk | |
| Shape3 | Shape | | stRoundRect |
| Imagel, Image2, Image3 | Stretch | | True |
| Labell | Font-Color | | Red |
| Font-Size | | 10 |
| Caption | | Обязательно нажмите на кнопку "Готово" после раз­мещения рисунков!! |
| Label2, label3, Label4 | Font-Color | | Olive |
| Font-Size | | 16 |

Замечание. Размеры фигур должны быть больше, чем размеры рисунков, поскольку последние следует располагать строго в середине фигур. Если во время наложения рисунка на фигуру рисунок исчезает, переместите его на передний план командой контекстного меню Bring То Front.

5. Дважды щелкните на первом таймере и запрограммируйте мигание сообщения «Обязательно нажмите на кнопку «Гото­во» после размещения рисунков!!» так:

procedure TForml.TimerlTimer(Sender: Tobject); begin

{Еслиполемигает}

if Labell.Visible = True then

Labell.Visible := False {полегасится,}

else

{иначе - полезасвечивается} Labell.Visible := True;

end;

6\*. Введите описание глобальных переменных программы:

varForml : Tforml

{Эта строка уже есть в заготовке модуля} shiftX, shiftY :integer;

6\*\*. Введите описания глобальных констант и переменных программы:

const к = 4; {Количество обновлений содержания теста}

a: array [l..k, 1..3] of string = ( {Массиврисунков} ('car.wmf, 'money.wmf', 'bomb.wmf'),

('coins.wmf', 'clock.wmf, 'tennis.wmf),  
('brick.wmf, 'building.wmf, 'champgne.wmf),  
('dice.wmf, 'donkey.wmf, 'door.wmf) );  
b: array [l..k, 1..3] of string = ( {Массивслов}

('money', 'car', 'bomb'),

('tennis', 'clock', 'coins'),

('building', 'champagne', "brick'),

('dice', 'door', 'donkey')

);

check: array [l..k, 1..3] of integer = (

(2, 1, 3), {Таблица отображения} (3, 2, 1), {массива рисунков а в }(3, 1, 2), {массив слов Ъ. Здесь число-} (1, 3, 2) ); {номер рисунка к слову}

var

Forml : TForml; {Этастрокаужеестьвзаготовкемодуля} i, score : integer; shiftX, shiftY : integer; leftl, left2, left3 : integer; topi, top2, top3 : integer;

7\*.Запрограммируйте кнопку BitBtn2 как кнопку закрытия окна командой close.

Вставьте рисунки bomb.wmf, tennis.wmf и money.wmf, из папки С:\ ProgramFiles \ MicrosoftOffice \ Clipart \ Popular в объекты Imagel, Image2 и Image3 соответственно.

Свойствам Caption текстовых полей Label2, Label3 и Label4 присвойте значения "money", "bomb" и "tennis" соответственно.

7\*\*.Запрограммируйте процедуры: создания формы FormCreate, вставки в неё рисунков SetPicture, вывод информации о ре­зультате теста Informationи кнопку BitBtn2:

procedure Tb1orml.FormCreate(Sender: TObject); begin

i := 0; {Номер обновления содержания теста) score := 0; {Набранные пользователем баллы) {Сохраняем начальные координаты)

{левых верхних углов) {трех рисунков в форме)

leftl ;= Imagel.Left left2 := Image2.Left left3 := Image3.Left topi := Imagel.Top; top2 := Image2.Top;

top3 := Image3.Top;

SetPicture; {Процедура i-ойвставкирисунков)

end;

procedure Information; begin

{ОстанавливаемобатаймераПрекращаеммигание Labell} Forml.Timer 1.Enabled := False; {Останавливаемдвижениеиндикаторавремени) Forml.Timer2.Enabled := False; {Результаттестаприводимк 100 баллам) score := (score \* 100) div к;

{Если результат > 90 - считываем файл громкого

аплодирования)

if score >90 then

Forml.MediaPlayerl.FileName :— 'applause.wav';

{Если результат между 70 и 90 - считываем файл

легкогоаплодирования.)

if (score >70) and (score <90) then

Forml.MediaPlayerl.FileName := 'clap.wav'; {Иначе файл не считываем и аплодирования не будет)

Forml.MediaPlayerl.Open; {Открываем мульт. плеер)

Forml.MediaPlayerl.Play; {Воссоздаём звук }

{Открываем окно сообщения"Вы набрали ## баллов с 100}

MessageDlg('Bbi набрали '+IntToStr(score)+' баллов из 100', mtInformation,[mbOk], 0); {После щелчка на кнопке "Ok" информационного окна:} {Закрываем окно программы. - заканчиваем работу}

Forml.close;

end;

procedure TForml.BitBtn2Click(Sender: TObject); begin

Information;

end;

Процедуры Information и SetPicture не являются методами класса Forml. Поэтому изменение свойств любых объектов фор­мы происходит в них путем явного указания имени формы Forml перед именем соответствующего объекта, например, Forml.Timer1.Enabled. Декларации процедур Information и

SetPicture вставьте между описанием типа TForml и разделом подсоединенных модулей - uses:

procedureInformation; procedureSetPicture;

8. Обеспечьте перетягивания первого рисунка на форме, запрограммировав обработки таких трех событий для объек­та Imagel: OnMouseDown (ЕслиНажатьМышь), OnMouseMove (ЕслиПеретягиватьМышь) и OnMouseUp (ЕслиОтпустить-Мышь). Для заполнения заготовок приведенных процедур воспользуйтесь закладкой Events окна инспектора объектов первого рисунка.

procedureTForml.ImagelMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

{Временная переменная Tag объекта Imagel=1, если) {клавиша мыши в этот момент нажата) Imagel.Tag := 1;

shift := Y; {Запоминаем координату (X;Y) точки) shift := X; {щелчка мыши в середине рисунка) end;

procedureTForml.lmagelMouseMove(Sender: TObject;Shift: TshiftState; X,Y: Integer);

begin

ifImagel.Tag = 1 then {Еслинажатаклавишамыши} begin

{Меняем, координаты рисунка на величину изменения ко­ординаты указателя мыши (X;Y) с учетом его смещений в середине рисунка shiftX, shiftY}

Imagel.Top := Imagel.Top + Y - shiftY;

Imagel.Left := Imagel.Left + X - shiftX;

end;

end;

procedureTForml.ImagelMouseUp(Sender : Tobject;

Button : TMouseButton; Shift : TShiftState; X, Y : Integer);

begin

{Запоминаем, что клавиши мыши отпущены.} Image 1.Tag := 0 end;

Обратите внимание, что программа не проверяет, какую имен­но клавишу мыши нажал пользователь, поэтому перетягивать картинки можно любой клавишей. Проверить клавиши мыши можно, проанализировав аргумент Button (типа TMouseButton) приведенных процедур: Button = {mbLeft (левая клавиша), mbRight (правая), mbMiddle (средняя)}. Координаты указателя мыши в пикселях передаются в процедуры с помощью аргументов X и Y типа integer.

1. Аналогично запрограммируйте соответствующие события для перетягивания двух других рисунков.
2. Создайте еще один метод формы TForml — функцию ImagelnShape для проверки расположения рисунка (объекта Image) в середине некоторой геометрической фигуры (объ­екта Shape).

functionTForml.ImagelnShape (Image : TImage;

Shape : TShape) : Boolean;

begin

if(Image.Left >= Shape.Left) and

(Image.Left+Image.Width <= Shape.Left+Shape.Width) and (Image.Top >= Shape.top) and (Image.Top + Image.Height) <= (Shape.Top + Shape.Height) then

ImagelnShape :=True{Рисунок в середине фигуры}

else

ImagelnShape := False; {Рисуноквнефигуры}

end;

Функция ImagelnShape является методом класса Forml, поэто­му обращение к свойствам любых объектов формы Forml осуще­ствляется непосредственно с помощью имени этих объектов, например, Image.Тор. Декларацию функции вставьте в раздел описания формы typeTForml в раздел деклараций общедоступ­ных процедур после служебного слова public:functionImageInShape(Image:TImageShape:TShape):Boolean;

11\*. Запрограммируйте кнопку "Готово", проверяющую пра­вильность расположения рисунков в середине фигур и вы­водящую информационное окно с сообщением "Правильно!" или "Неправильно!":

procedure TForml.BitBtnlClick(Sender: TObject); begin

{Останавливаемобатаймера}

Timer 1.Enabled := False; {Останавливаеммигания Labell}

{Останавливаем индикатор времени}

Timer2.Enabled := False;

if ImageInShape(Imagel,Shape2) and

ImageInShape(Image2,Shape3) and

ImageInShape(Image3,Shapel) then begin

{Считываемфайлгромкогоаплодирования}

MediaPlayerl.FileName := 'applause.wav";

MediaPlayerl.Open; {Открываемплеер}

MediaPlayerl.Play; {Воссоздаем звук}

{Открываем, окно сообщения "Правильно!"}

MessageDlg('11равильно!', mtlnformation, [mbOk], 0);

end else begin

MediaPlayerl.FileName := 'ding.wav';

MediaPlayerl.Open;

MediaPlayerl.Play; {Подаемзвук "ДИНГ"} MessageDlg('He правильно!', mtlnformation, [mbOk], 0);

{Включаем оба таймера и индикатор времени)

Timerl.Enabled := True;

Timer2.Enabled := True;

Gaugel.Progress := 100;

end;

11\*\*.Запрограммируйте кнопку "Готово", проверяющую пра­вильность расположения рисунков в середине фигур:

procedureTForml.BitBtnlClick(Sender : Tobject);

varcheck 1, check2, check3 : integer;

begin

{Где находится первый рисунок?){В 1-ойфигуре)

if ImageInShape(Imagel,Shape 1) then checkl := 1; {Во 2-ойфигуре)

if ImageInShape(Imagel,Shape2) then checkl := 2; {В 3-ейфигуре)

if ImageInShape(Imagel,Shape3) then checkl := 3; {Гденаходитсявторойрисунок?) {В 1-ойфигуре)

if ImageInShape(Image2,Shape 1) then check2 := 1; {Во 2-ойфигуре)

if ImageInShape(Image2,Shape2) then check2 := 2; {В 3-ейфигуре)

if ImageInShape(Image2,Shape3) then check2 := 3; {Гденаходитсятретийрисунок?) {В 1-ойфигуре)

if ImageInShape(Image3,Shapel) then check3 := 1; {Во 2-ойфигуре)

if ImageInShape(Image3,Shape2) then check3 := 2; {В 3-ейфигуре)

if ImageInShape(Image3,Shape3) then check3 := 3;

{Еслирисункирасположеныправильно)

if (check[i, 1] = checkl) and (check[i, 2] = check2)

and (check[i, 3] = check3) then {считаемудачныепопытки)

score := score + 1; SetPicture; {Устанавливаемновыерисунки)

end;

12\*.Запрограммируйте второй таймер, обеспечивающий инди­кацию времени выполнения от 100 до 0% со скоростью, об­ратно пропорциональной числу в SpinEditl.

Если пользователь не успел расположить рисунки в прямо­угольниках и нажать на кнопку "Готово" в заданное время, пода­ется звук "ДИНГ" и открывается информационное окно с сообще­нием "Попробуйте ещё раз!".

procedure TForml.Timer2Timer(Sender: TObject); begin

{Задаем скорость движения индикатора) timer2.interval := SpinEditl.value \* 100;

{Работает индикатор времени) gaugel.progress := gaugel.progress - 1; ifgaugel.progress — 0 then{Если время исчерпано) begin

Timer 1.Enabled := False; Timer2.Enabled := False; MediaPlayerl.FileName := 'ding.wav'; MediaPlayerl.Open;

MediaPlayerl.Play; {Подаемзвук "ДИНГ") MessageDlg('Попробуйтеещераз!', mtlnformation,

[mbOk], 0);

Timer 1.Enabled := True; Timer2.Enabled := True; Gaugel.Progress := 100; end;

end;

12\*\*.Запрограммируйте второй таймер, обеспечивающий дос­тупность кнопки "Готово" лишь в случае правильного раз­мещения всех рисунков в отведенных прямоугольниках и показывающий время со скоростью, обратно пропорцио­нальной числу к SpinEditl.

В случае, если пользователь не успел расположить рисунки в прямоугольниках и нажать на кнопку "Готово" за надлежащее время, подается звук "ДИНГ", попытка не засчитывается, и про­исходит переход к следующей серии рисунков.

procedure TForml .Timer2Timer(Sender: TObject); begin

{Задаемскоростьдвиженияиндикатора }  
timer2.interval := SpinEditl.value \* 100;  
{Еслиодинизрисунковнаходитсявпервомпрямоугольнике)  
if ( ImageInShape(Imagel,Shapel) or

ImageInShape(Image2,Shape 1) or  
ImageInShape(Image3, Shape 1)) and  
{иодинизрисунковнаходитсявовторомпрямоугольнике)  
( ImageInShape(Imagel,Shape2) or

ImageInShape(Image2,Shape2) or  
I m age I n S liape(Image3 ,Shape2)) and  
{иодинизрисунковнаходитсявтретьемпрямоугольнике)  
( ImageInShape(Imagel ,Shape3) or

ImageInShape(Image2,Shape3) or

ImageInShape(Image3,Shape3)) {токнопка "Готово" становитсядоступной:} then BitBtnl.Enabled := True; {Работаетиндикаторвремени} gaugel.progress := gaugel.progress - 1; if gaugel.progress = 0 then {Есливремяисчерпано} begin

MediaPlayerl.FileName := "ding.wav'; MediaPlayerl .Open;

MediaPlayerl.Play; {Подаемзвук "ДИНГ"}  
SetPicture; {Меняемсериюрисунков}

end;

end;

1. Сохраните программу и проект в своей папке.
2. Скопируйте музыкальные файлы ding.wav, applause.wav и clap.wav в свою папку из папок [C:\WINDOWS\MEDIA\](file://C:/WINDOWS/MEDIA/) Office97 и [C:\WINDOWS\MEDIA](file://C:/WINDOWS/MEDIA) соответственно.

15\*\*.Скопируйте рисунки car.wmf, money.wmf, bomb.wmf, coins.wmf, clock.wmf, tennis.wmf, brick.wmf, building.wmf, champgne.wmf, dice.wmf, donkey.wmf, door.wmf в свою пап­ку из папки [C:\Program](file://C:/Program)Files\MicrosoftOffice\Clipart\ Popular.

1. Запустите программу. Перетягивайте мышью рисунки на середину соответствующих фигур.
2. Создайте ехе-файл программы и поупражняйтесь с ней.

Задача 6.1. Обеспечьте возможность перетягивания рисунков лишь левой клавишей мыши (см. пункт 8).

Задача 6.2. Увеличьте вдвое частоту мигания текста "Обяза­тельно нажмите на кнопку "Готово" после размещения рисун­ков!!".

Задача 6.3. Поменяйте тип (свойство Kind) индикатора состоя­ния на gkPie, gkNeedle и т.п., скорректировав форму и располо­жение объектов.

Задача 6.4. Замените в программе индикатор состояния типа Gauge на индикатор типа ProgressBar.

Задача 6.5.\*\* Обеспечьте (кроме подсчета баллов) индикацию оценки за тест. Диапазону 100-86 баллов поставьте в соответствие оценку "5", диапазону 78-85 баллов - оценку "4", 51-77 - оценку "3", меньшее 51 балла - оценку "2". Подберите звуковые эффек­ты для разных оценок.

Задача 6.6. В версии программы, предназначенной для кон­троля знаний, подберите самую удачную скорость движения ин­дикатора времени и сделайте объект SpinEditl недоступным для внесения изменений.

Задача 6.7. В версии программы, предназначенной для кон­троля знаний, удалите объект SpinEditl и задайте самую удачную скорость движения индикатора времени непосредственно в проек­те.

Задача 6.8.\*\* Увеличьте количество серий рисунков и подпи­сей к ним.

# **Практическое занятие №30 Создание проекта с использованием компонентов для работы с текстом.**

**Разработка проекта «Тук-тук»**

# **Цель работы: научиться создавать проект с использованием компонентов для работы с текстом.**

**Теоретические сведения**

Для выполнения настоящего проекта необходимо использовать следующие компоненты:

командная кнопка Button, окно редактирования Edit и метка Label;

а также следующие свойства этих компонентов:

Caption (заголовок), Color (цвет), Font (шрифт), Visible (видимость).

В программе используется условный оператор:

**if***условие***then***действия1***else***действия2*

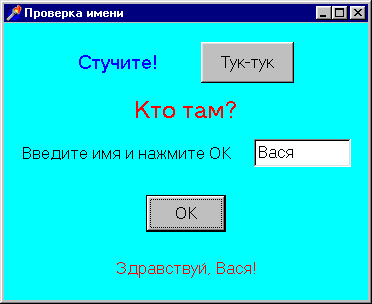
где *условие* - равенство или неравенство;

*действия1*и *действия2* - оператор или операторы (в этом случае их надо заключить между **begin** и **end**).

При выполнении условного оператора проверяется *условие*, если оно истинно (верно), то выполняются *действия1*, а *действия2* - нет (игнорируются). Если же *условие* ложно, то напротив, выполняются *действия2*, а *действия1* - нет.

Постановка задачи

Компьютер должен запросить имя пользователя (пароль). Если пользователь дает правильный ответ (вводит знакомое имя, например, Вася), то компьютер приветствует его (рис.1). В случае ввода любого другого слова, компьютер должен реагировать иначе, например, выводить запрос: "А где Вася?"



# Рис.1

**Практическая часть**

Ход работы

1. Откройте новый проект.
2. На новой форме Form1 разместите необходимые компоненты: 2 командные кнопки Button, окно редактирования Edit и 4 метки Label (страница Standard) (рис. 2).

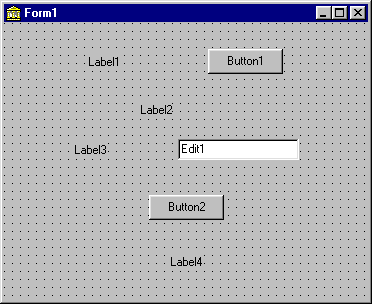


Рис. 2

1. Используя ObjectInspector, задайте необходимые заголовки (свойство Caption) форме, всем меткам и командным кнопкам, а также подберите для них подходящие цвета и размер (см. рис.1). Заголовок четвертой метки, а также текст в окне редактирования (свойство Text) можно сделать пустыми. Выполните необходимое выравнивание всех элементов по горизонтали и по вертикали (Edit → Align…).
2. Сразу после запуска программы метки Label2 и Label3, окно редактирования Edit1 и кнопка Button2 должны оставаться невидимыми до нажатия на кнопку Button1. Для этого свойство Visible каждого из этих компонентов должно иметь значение False (ложь).
3. Сохраните проект (File Save → Project As…) в отдельной папке Name, заменив предлагаемое название модуля unit1.pas на nam.pas, а предлагаемое название проекта Project1.dpr - на Name.dpr. Проверьте работоспособность программы на этом этапе и только после этого продолжайте ее разработку. Далее обязательно проверяйте правильность своих действий пробным запуском программы (в конце каждого этапа). При отсутствии ошибок сохраняйте все файлы проекта (File → Save All).
4. Теперь нужно сделать так, чтобы нажатие на кнопку Button1приводило к появлению невидимых до этого компонентов (меток Label2 и Label3, окна редактирования Edit1 и кнопки Button2). Для этого выполните двойной щелчок на кнопке Button1. В тексте программы будет создана процедура TForm1.Button1Click(Sender:TObject), в которую впишите следующие строки:

Label2**.**Visible **:**= True**;**

Label3**.**Visible **:**= True**;**

Edit1**.**Visible **:**= True**;**

Button2**.**Visible **:**= True**;**

1. Сначала упростим нашу задачу. Пусть компьютер приветствует по имени любого пользователя. Для этого пустой заголовок метки Label4 после нажатия на кнопку Button2 должен принять значение 'Здравствуй, ' + имя, набранное в строке редактирования Edit1. Дважды щелкните на кнопке Button2 и впишите следующую строку в созданную процедуру TForm1.Button2Click(Sender:TObject):

Label4**.**Caption **:**= 'Здравствуй, ' + Edit1**.**Text + '**!**'**;**

1. Теперь сделаем так, чтобы заголовок метки Label4 принимал значение 'Здравствуй, Вася!', если в строку редактирования введено имя Вася, а в любом другом случае заголовком метки Label4 становилось значение 'А где Вася?'. Для этого из процедуры TForm1.Button2Click(Sender:TObject) удалите ранее написанную строку и впишите вместо нее следующий условный оператор:

if Edit1**.**Text = 'Вася' then Label4**.**Caption **:**= 'Здравствуй, Вася!'

else Label4**.**Caption **:**= 'АгдеВася!'**;**

1. Еще раз сохраните все файлы проекта и запустите вашу программу.

## Упражнения. Совершенствование проекта

1. Обеспечьте удаление кнопки Button1 (а также и метки Label1) после ее нажатия. Для этого добавьте в процедуру TForm1.Button1Click(Sender:TObject) строки, задающие значение False свойствам Visible этих компонентов.
2. Засекретим текст, вводимый в строку редактирования Edit1. Для этого в окне ObjectInspector укажите для этого компонента символ (свойство PasswordChar), который будет печататься в строке при наборе имени вне зависимости от набираемых букв (обычно используется \*).
3. Сделайте так, чтобы цвет формы менялся в зависимости от правильности введенного имени. Для этого добавьте вусловный оператор выражения вида (не забудьте использовать **begin** и **end**):

Form1.Color:=clBlack;

# **Практическое занятие №31 Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.**

**Калькулятор**

**Цель работы: научиться создавать проект с использованием компонентов ввода и отображения чисел, дат и времени.**

**Теоретическая часть**

Создадим новый проект приложения. К примеру, в Delphi 7 для этого следует выполнить команду меню **File → New → Application**.

Теперь разработаем программу калькулятора. Это будет не инженерный, а обычный калькулятор, поэтому над оформлением формы сильно мучаться не придётся – она будет содержать всего лишь одно текстовое поле и 16 кнопок.

Конечный вид размещения элементов управления на форме показан на рис. 1.5.

**Практическая часть**

Ход работы

Приступим к разработке этой формы. Для начала присвоим значения некоторым свойствам формы Form1 согласно табл. 1.2.



Рис. 1.5. Конечный вид формы программы калькулятор

Таблица 1.2. Свойства формы Form1

|  |  |  |
| --- | --- | --- |
| Свойство | Значение | Пояснение |
| Caption | Калькулятор | Заголовок формы |
| BorderStyle | bsDialog | Граница формы не позволяет изменять ее размеры – как у диалоговых окон |
| Height | 170 | Высота формы |
| Width | 173 | Ширина формы |

Теперь разместим на форме компонент **Edit** (текстовое поле) со вкладки **Standart**палитры компонентов.



Присвоим свойствам компонента Edit1 значения согласно табл. 1.3

Таблица 1.3. Свойства компонента Edit1

|  |  |  |
| --- | --- | --- |
| Свойство | Значение | Пояснение |
| Text |  | Очищаем содержимое текстового поля |
| Left | 4 | Отступ слева |
| Top | 4 | Отступ сверху |
| Width | 157 | Ширина поля |

Настал черед кнопок. Для этого можно было бы использовать обычный компонент **Button**со вкладки **Standart**, но вместо него мы воспользуемся компонентом **BitBtn**со вкладки **Additional**, чтобы надписи на кнопках можно было сделать цветными.



Разместим вначале на форму кнопку, соответствующую цифре 7, и присвоим ее свойствам значения согласно табл. 1.4.

Таблица 1.4. Свойства кнопки BitBtn1

|  |  |  |
| --- | --- | --- |
| Свойство | Значение | Пояснение |
| Caption | 7 | Надпись на кнопке |
| Left | 4 | Отступ слева |
| Top | 32 | Отступ сверху |
| Width | 33 | Ширина кнопки |
| Font.Size | 10 | Делаем шрифт надписи крупнее |
| Font.Color | clBlue | Выделяем надпись синим цветом |

Разместим на форме остальные кнопки, соответствующие цифрам калькулятора. Для этого можно просто выделить уже созданную кнопку, скопировать ее в буфер обмена (например, нажав комбинацию клавиш <Ctrl+C> или выполнив команду меню **Edit → Copy**), а затем вставить из буфера обмена требуемое количество раз, нажав комбинацию клавиш <Ctrl+V> или выполнив команду меню **Edit → Paste**. Присвоим свойству Caption каждой кнопки соответствующее значение, а значения свойств Left и Top установим, руководствуясь табл. 1.5.

Таблица 1.5. Координаты цифровых кнопок калькулятора

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Caption | 8 | 9 | 4 | 5 | 6 | 1 | 2 | 3 | 0 |
| Left | 40 | 76 | 4 | 40 | 76 | 4 | 40 | 76 | 4 |
| Top | 32 | 32 | 60 | 60 | 60 | 88 | 88 | 88 | 116 |

Разместим на форме кнопки, соответствующие сбросу (“CE”) и знаку равно (“=”). Изменим из свойства Caption , а так же Left (для кнопки **CE** – 40, для кнопки **=** - 60) и Top (116). Свойству Font.Color Присвоим значение clRed (надписи будут выделены красным цветом).

Напоследок разместим на форме ещё четыре кнопки **BitBtn**, соответствующие арифметическим действиям сложения (свойство Caption = +), вычитания (Caption = -), деления (Caption = /) и умножения (Caption = \*). Свойству Width каждой из кнопок присвоим значене 37, свойству Left – значение 124, а свойству Font.Color – значене clNavy. Значения свойства Top соответсвуют цифровым кнопкам калькулятора (32, 60, 88 и 116).

*Объявление переменных*

Теперь перейдем в редактор кода. Первым делом объявим все переменные, которые будем использовать в программе. Переменные объявляются в разделе var:

|  |
| --- |
| …  Var  Form1: TForm1;  …  **Num1, num2: integer;** // для хранения первого и второго числа  **Doit: string**; // действие, которое надо выполнить с числами |

*Событие OnClick кнопок калькулятора*

После того как в раздел var внесены изминения, создадим обработчик события для каждой цифровой кнопки калькулятора. Для этого можно просто дважды щелкнуть мышью на каждой из кнопок в конструкторе формы и ввести необходимые операторы в процедуре обработки события OnClick

Для кнопки 0 это будет:

|  |
| --- |
| Procedure TForm1.BitBtn10Click (Sender: TObject);  Begin  Edit1.Text:= Edit1.Text + ‘0’;  End; |

Для кнопки 1:

|  |
| --- |
| Procedure TForm1.BitBtn7Click (Sender: TObject);  Begin  Edit1.Text:= Edit1.Text + ‘1’;  End; |

И так далее…

При нажатии кнопки **CE**должен происходить сброс:

|  |
| --- |
| Procedure TForm1.BitBtn11Click (Sender: TObject);  Begin  Edit1.Clear;  Num1:=0;  Num2:=0;  Doit:=’ ‘;  End; |

Здесь очищается текстовое поле (метод Clear) и обнуляются все переменные.

Теперь обработаем события нажатия кнопок арифметических действий.

Для сложения:

|  |
| --- |
| Procedure TForm1.BitBtn13Click (Sender: TObject);  Begin  Num1:= StrToInt (Edit1.Text);  Doit:= ‘плюс’;  Edit.Clear;  End; |

Для вычитания:

|  |
| --- |
| Procedure TForm1.BitBtn14Click (Sender: TObject);  Begin  Num1:= StrToInt (Edit1.Text);  Doit:= ‘минус’;  Edit.Clear;  End; |

Для умножения:

|  |
| --- |
| Procedure TForm1.BitBtn16Click (Sender: TObject);  Begin  Num1:= StrToInt (Edit1.Text);  Doit:= ‘умножить’;  Edit.Clear;  End; |

Для деления:

|  |
| --- |
| Procedure TForm1.BitBtn15Click (Sender: TObject);  Begin  Num1:= StrToInt (Edit1.Text);  Doit:= ‘разделить’;  Edit.Clear;  End; |

В этих процедурах сначала запоминается число, введенное пользователем (num1), поскольку, если он выполняет какое либо действие, то это означает, что первое число уже введено. Затем запоминается действие (doit), которое пользователь хочет выполнить над переменной num1. После этого текстовое поле очищается (Edit1.clear), чтобы пользователь мог ввести второе число.

Осталась только одна кнопка =, которая выводит результат:

Для сложения:

|  |
| --- |
| Procedure TForm1.BitBtn12Click (Sender: TObject);  Begin  Num2:= StrToInt (Edit1.Text);  If doit:= ‘плюс’  Then Edit1.Text:= IntToStr (num1 + num2);  If doit:= ‘минус’  Then Edit1.Text:= IntToStr (num1 - num2);  If doit:= ‘умножить’  Then Edit1.Text:= IntToStr (num1 \* num2);  If doit:= ‘разделить’  Then Edit1.Text:= IntToStr (num1 div num2);  Num1:=0;  Num2:=0;  Doit:= ‘ ‘;  End; |

Пользователь нажимает кнопку = после того как уже ввел второе число, поэтому число сохраняется в переменной num2. Затем мы проверяем, какое действие хотел выполнить пользователь. Результат операции заносится в текстовое поле (свойство Edit1.Text), после чего все переменные обнуляются для использования в последующих расчетах.

*Компиляция и запуск программы*

Теперь можно откомпилировать и запустить программу на выполнение. При первом запуске программы потребуется сохранить файлы с исходным кодом программы. Сохраним файл, содержащий код программы, под именем Calc1.pas, а файл проекта Delphi – под именем Calc.dpr. Программа Калькулятор в действии показана на рис 1.6.

**Примечание:**

Возможно, у учителя возник вопрос: “А где же различные проверки, например, деления на ноль?” Об этом можно не беспокоиться, поскольку в Delphi это все учитывается, и при неккоректных значениях на экране появляется соответствующее сообщение (рис. 1.7).

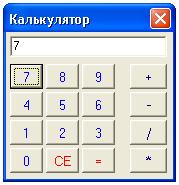


Рис.1.6. Программа калькулятор в действии Рис. 1.7. Сообщение о делении на ноль

# **Практическое занятие №32 Создание проекта с использованием компонентов стандартных диалогов и системы меню.**

Разработка проекта Goroscop

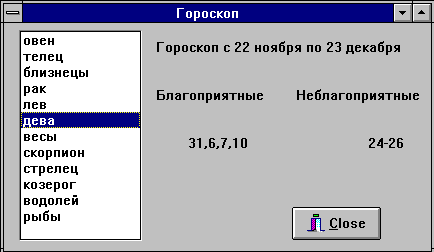
# **Цель работы: научиться создавать проект использованием компонентов стандартных диалогов и системы меню.**

Теоретическая часть

**Постановка задачи.**

Разработать программу, с помощью которой пользователь мог бы выполнить следующее. Он запускает программу, выбирает знак зодиака с помощью мышки или стрелок и нажимает клавишу Enter. На экране появляется список благоприятных и неблагоприятных дней этого знака. Программа заканчивает свою работу по нажатию клавиши Close.

**Пояснения.** Новым в этой программе является компонента ListBox (список).



*Рисунок 1.*

**Практическая часть**

**Ход работы.**

1. Открыть новый проект.

2. Разместить на форме компоненты в соответствии с рис 2. Свойству Kind кнопки BitBtn1 придать значение Close.

3. Сохранить проект под именем gor.pas и goros.dpr.

4. Запустить программу, затем закрыть окно проекта кнопкой Close.

5. Выделить Label1, найти в Object Inspector’e свойство Caption и справа от него вместо слова Label1 написать “Гороскоп с 22 ноября по 23 декабря”.

6. Выделить Label2, свойству Caption придать значение “Благоприятные Неблагоприятные”, то есть справа от свойства Caption вместо слова Label2 набрать указанные слова.

7. Выделить Label3 и справа от свойства Caption убрать слово “Label3”.

8. Выделить ListBox1, в Object Inspector’e найти свойство Items, справа от него нажать на кнопку с тремя точками. В появившемся окне встроенного редактора ( см. рис. 3 ) ввести названия знаков зодиака, каждый на новой строке. Нажать клавишу OK или, если вы в Delphi 1, то сохранить текст, нажав клавишу Save, под именем znak.txt, затем нажать клавишу OK

9. В Object Inspector’е найти событие OnKeyPress, справа от него в поле сделать двойной щелчок левой кнопкой мыши. В том месте Unit1, где будет мигать курсор, ввести следующий код:

if key=#13 then

case Listbox1.ItemIndex of

0: Label3.Caption:='20 24-13 ';

1: Label3.Caption:='26-4 14,15,22 ';

2: Label3.Caption:='12-19 ';

3: Label3.Caption:='24,30,2,6,12,16 26,5,18 ';

4: Label3.Caption:='14,15,16,19-22 27,2,3 ';

5: Label3.Caption:='31,6,7,10 24-26 ';

6: Label3.Caption:='29-31,6,12,20 26 ';

7: Label3.Caption:='24,29,31,2,6,12,20 26,28,5,18 ';

8: Label3.Caption:='24-30 5,12 ';

9: Label3.Caption:='23-22 ';

10: Label3.Caption:='23-22 ';

11: Label3.Caption:='20 1,2 ';

end;

10. Сохранить проект, запустить и протестировать его.

**Упражнения. Совершенствование проекта.**

1. Изменить интерфейс программы: изменить шрифты, цвет экрана и букв.

Подсказка. Возможно придется в коде подкорректировать количество пробелов.

2. Сделать так, чтобы когда курсор мыши находится в поле ListBox1, появлялась бы подсказка о том, что надо сделать.

Подсказка. Воспользуйтесь свойствами Hint Showhint компонента ListBox1.

3. Сделать так, чтобы при нажатии клавиши Close появлялось бы окно сообщения “Вы уверены?”

Подсказка. Применить функцию MessageDlg.

**Практическое занятие №33 Разработка оконного приложения. Создание новых форм с меню, геометрическими фигурами, палитрой и со шкалой.**

**Цель работы: научиться разрабатывать оконные приложения**

**Теоретическая часть**

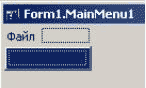
Палитра компонентов. Палитра компонентов расположена в главном окне и имеет вид многостраничного блокнота. Каждая страница имеет свой набор компонентов (рис. 1). Чтобы поместить компонент в центре окна формы, дважды щелкают на его пикто­грамме. Если нужно разместить компонент где либо на форме, щелкают один раз на его пиктограмме и один раз в нужном месте формы. Для того, чтобы вставить один и тот же компонент не­сколько раз, нужно нажать клавишу Shift и щелкнуть на его пиктограмме - теперь можно щелкать в окне формы. Чтобы отка­заться от этого режима, надо нажать на кнопку палитры компо­нентов с изображением стрелки. Выбранный компонент можно перемещать на форме, а также изменять его размеры, перетягивая маркеры.

**Практическая часть**

Ход работы

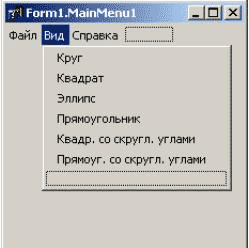
1. Создайте новое приложение. Форму назовите Paint.

2. Добавьте на форму меню в любое место (*Standard ––> MainMeny*).



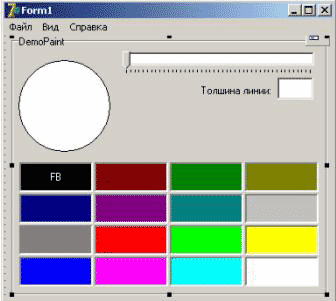
3. На форме появится значок меню, кликните по нему 2 раза.

4. Появится окно; в свойстве – *caption* напишите “Файл”, нажмите Enter, перейдите курсором мыши на строку ниже; в свойстве – *caption* напишите “Выход”; Enter; перейдите курсором на строку справа в свойстве – *caption* напишите “Вид”; и т.д. создайте меню как на рисунке. В меню “Справка” добавьте один пункт: “Об Авторе”;



5. Закройте эту форму (Form1.MainMenu1);

6. Создание GroupBox (*Standard ––> GroupBox*); разместите его по всему периметру формы. В свойстве Caption напишите текст (DemoPaint).



7. Добавьте в левый угол формы геометрическую фигуру (*Additional ––> Shape*) прямоугольного, не квадратного размера!; в свойстве *Shape* укажите stCircle; прямоугольник станет кругом.

8. Вставьте палитру цветов (*Samples ––> Colorgrid*) в нижнюю часть формы.

9. Добавьте на форму шкалу для изменения толщины линий фигуры (*Win32 ––> TrackBar*); в свойстве *max* укажите 45, в свойстве *min* укажите 1.

10. Добавьте поле (*Standard ––> Edit*) в свойстве *Enabled* укажите false (это запретит изменение данного поля), укажите в свойстве *text* значение 1; добавьте текст “Толщина линии:” (*Standard ––> Label*).

11. Кликните 2 раза на шкалу изменения ширины линий; после **begin** введите:

**shape1.Pen.Width:=TrackBar1.Position;  
Edit1.Text:=IntToStr(TrackBar1.Position);**

12. Кликните 2 раза по палитре цветов; после **begin** введите:

**Shape1.Brush.Color:=ColorGrid1.ForegroundColor;  
Shape1.Pen.Color:=ColorGrid1.BackgroundColor;**

(Это дает возможность менять цвет заливки фигуры (левой клавишей мыши) и цвет линий фигуры (правой клавишей мыши).

13. Кликните на форме по пункту меню “Выход” (Файл ?Выход); после **begin** введите **close;** (теперь кнопка выход в меню закрывает форму).

14. Кликните по пункту Круг; после **begin** введите: **Shape1.Shape:=stCircle;**

15. Кликните по пункту Квадрат; после **begin** введите: **Shape1.Shape:=stSquare;**

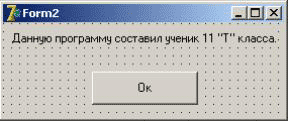
16. Кликните по пункту Эллипс; после **begin** введите: **Shape1.Shape:=stEllipse;**

17. Кликните по пункту Прямоугольник; после **begin** введите: **Shape1.Shape:=stRectangle;**

18. Кликните по пункту Квадрат.со скруг углами; после **begin** введите: **Shape1.Shape:=stRoundSquare;**

19. Кликните по пункту Прямоуг. со скруг. углами; после **begin** введите: **Shape1.Shape:=stRoundRect;**

20. Создайте новую форму (*Файл  ––> Новый ––> Форма*); назовите её “Об Авторе”; напишите на ней о себе; сделайте кнопку “Ок”.



21. Кликните по пункту “Об авторе”; после **begin** введите: **Form2.show;** (это позволит откроет вторую форму); после запуска программы появится сообщение о присоединении второй формы, нажмите Да.

22. На второй форме кликните 2 раза по кнопке “Ок” и после **begin** введите: **Form2.Close;**

23. Протестируйте программу, исправьте все недочёты, добавьте что-нибудь своё.

# **Практическое занятие №34 Разработка оконного приложения с несколькими формами.**

**Разработка проекта Halfbottle.**

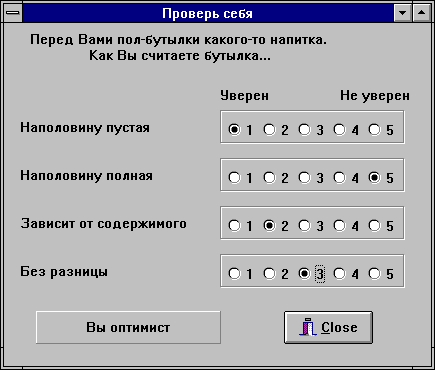
**Цель работы: научиться создавать оконное приложение с несколькими формами**

**Теоретическая часть**

**Постановка задачи.**

Разработать программу, с помощью которой пользователь мог бы выполнить следующее. После запуска программы появляется изображение, аналогичное рис.1. Пользователь по своему усмотрению выбирает один переключатель в группе. Каждому переключателю соответствует определенный балл. В зависимости от суммы набранных баллов появляется одно из сообщений “Вы пессимист”, “Вы реалист” или “Вы оптимист”.

**Пояснения.** Новыми в этом пректе являются группа переключателей RadioGroup и оператор вывода сообщений ShowMessage.



*Рисунок 1.*

**Практическая часть**

**Ход работы**

1. Разместить на форме компоненты согласно рис.2 и присвоить заголовки меткам и панелям.

2. Выделить компонент RadioGroup1, найти в Object Inspector’e свойство Caption и удалить заголовок.

Свойству Columns, определяющему количество колонок, в оторые будут отображаться переключатели, присвоить значение 5.

Вызвать String List Editor, дважды щелкнув мышкой рядом со свойством Items, ввести 5 строк со значениями 1, 2, 3, 4 и 5 соответственно. Сохранить эти строки под именем, например, t1.txt .

3. Аналогичные действия проделать с остальными компонентами RadioGroup.

4. Чтобы суммировать набираемые пользователем баллы, вобработчик события RadioGroup1.OnClick вставить код:

with RadioGroup1 do

if ItemIndex>=0 then sum:=sum+ItemIndex+1;

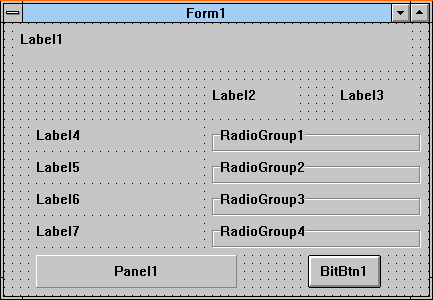
Единицу необходимо прибавлять, т.к. индекс первого переключателя равен 0, но соответствует 1 баллу. Целочисленную переменную sum необходимо описать в разделе var.

5. Вставить в обработчики событий RadioGroup2.OnClick, RadioGroup3.OnClick и RadioGroup4.OnClick аналогичные коды, но без обнуления переменной sum, т.к. оно необходимо лишь один раз перед началом суммирования.

6. Выведем на контрольную панель итоговое сообщение в зависимости от набранной суммы баллов. Для этого вобработчик события RadioGroup4.OnClick добавим код

casesumof

4..9: Panel1.Caption:=‘Выпессимист’;



*Рисунок 2.*

10..15: Panel1.Caption:=‘Вы реалист’;

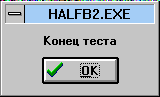
16..20: Panel1.Caption:=‘Выоптимист’;

end;

7. Вывести сообщение об окончании тестирования, добавив вобработчик события RadioGroup4Click код

ShowMessage(‘Конец теста’);

В результате выполнения этого оператора появится информационное окно со словами ‘Конец теста’ и единственной кнопкой Ok (cм. рис.3).



*Рисунок 3*

**Упражнения. Совершенствование проекта.**

1. После нажатия кнопки Ok в информационном окне сделать недоступными все RadioGroup, а все переключатели в них - невыбранными.

Подсказка. Свойствам Enabled всех RadioGroup присвоить значение False, а свойствам ItemIndex - значение -1.

2. Для контроля правильности работы программы вывести на панель набранную пользователем сумму баллов.

Подсказка. Заголовку соответствующей панели присвоить значение IntToStr(sum).

3. Запустить программу и убедиться, что верная сумма баллов получается лишь при последовательном выборе переключателей сначала из RadioGroup1, затем из RadioGroup2 и т.д. Если порядок был нарушен, или пользователь, изменив решение, выбрал другой переключатель в одной и той же группе, то результат будет неверным. Чтобы этого не случалось, сделать доступной только ту группу переключателей, в которой необходимо сделать выбор.

Подсказка. Первоначально свойству Enabled всех RadioGroup, кроме RadioGroup1, присвоить значение False. Вобработчик события RadioGroup1Click добавить код, присваивающий свойству Enabled RadioGroup2 значение True и т.д.

4. Сделать возможным повторный запуск программы.

Подсказка. Разместить на форме кнопку Button1, свойству Caption которой присвоить значение Retry, а вобработчик события Button1Click вставить код, делающий доступной RadioGroup1.

# **Практическое занятие №35 Разработка многооконного приложения.**

**Проект “Худой или полный?”.**

**Цель работы: научиться разрабатывать приложения**

**Теоретическая часть**

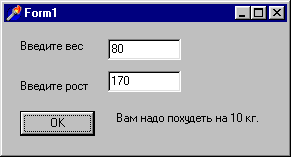
**Типы переменных**. Целочисленный и Действительный типы переменных:**integer и real**.

**Преобразование типов** переменных из строкового в целочисленный или в дествительный. Функции StrToInt( ); FloatToStr(); Format();

**Алгоритм выбора**. Оператор условия **if – then – else.**

Известно, что оптимальный вес человека определяется так: рост человека минус 100. Если твой фактический вес меньше оптимального, то ты худой, и наоборот, если больше, то надо похудеть.

Надо разработать такую программу, чтобы пользователь, введя свой рост и фактический вес, мог бы с помощью программы определить, худой он или полный, и насколько надо поправиться или похудеть (см. рис1).

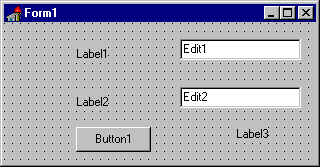


*Рисунок 1.*

**Практическая часть**

Ход работы

1. В соответсвии с рис.2 расположите необходимые компоненты. В Edit1 будем вводить вес в кг., а в Edit2 – рост в см.



*Рисунок 2*

2. Сохраните оба файла проекта в папке Weight.

3. Введите в раздел VAR переменные для сохранения значений фактического веса (faktW), оптимального веса (optW), роста (Rost) и разницы между оптимальным весом и фактическим (Delta). Пусть значения этих переменных будут округленными, в этом случае тип этих переменных можно объявить как Integer.

VAR

factW, optW, Rost, Delta : integer;

4. Образуйте процедуру, реагирующую на щелчок по кнопке Button1, и заполните ее следующим кодом.

**procedure TForm1.Button1Click(Sender: TObject);**

**begin**

**factW := StrToInt(Edit1.text);**

**Rost := StrToInt(Edit2.Text);**

**OptW :=Rost - 100;**

**Delta := abs(factW - OptW);**

**if OptW = factW then Label3.caption := 'Вашвесоптимален'**

**else**

**if OptW > factW then Label3.caption := 'Вамнадопоправитьсяна '+IntToStr(Delta)+' кг.'**

**else**

**Label3.caption := 'Вамнадопохудетьна '+IntToStr(Delta)+' кг.'**

**end;**4. В предыдущем пункте мы договорились вводить округленные исходные данные. Но это не всегда приемлимо. Усовершенствуйте программу так, чтобы можно было бы вводить десятичные величины. Для этого в разделе Var назовите тип переменных не Integer, а Real.

**VAR**

**factW, optW, Rost, Delta : real;**

Кроме этого, преобразование действительных чисел встроковый будет выполняться с помощью функции: FloatToStr( ).

В целом процедура Button2Click будет выглядеть следующим образом:

procedure TForm1.Button1Click(Sender: TObject);

begin

**factW := StrToFloat(Edit1.text);**

**Rost := StrToFloat(Edit2.Text**);

OptW :=Rost - 100;

**Delta := factW - OptW**;

if OptW = factW then Label3.caption := 'Вашвесоптимален'

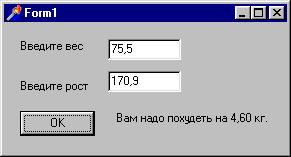
else

if OptW > factW then Label3.caption := 'Вамнадопоправитьсяна '+FloatToStr(**Delta)**+' кг.'

else

Label3.caption := 'Вамнадопохудетьна ‘ + FloatToStr(Delta)+' кг.';

end;



**Упражнения**

1. Для форматирования и преобразования действительных чисел вместо функции FloatToStr() примените функцию Format(‘ %f ’,[пeременная]). Например, Format(‘%f ‘, [Delta]).

2 Теперь для форматирования и преобразования примените эту же функцию в таком виде Format(‘Строка %f’,[пeременная]). Например, Format(‘Вам надо похудеть %f[Delta]) .

1. Сделайте к программе заголовок, замените кнопку Button1 на кнопку BitBtn1, введите кнопку “Close”,

**СООБЩЕНИЯ И КОДЫ ОШИБОК**

Среда Турбо Паскаля дает исчерпывающую информацию о характере и месте обнаруженной компилятором ошибки. При обнаружении ошибки среда автоматчики загружает в окно редактора исходный файл и помещает курсор около того места, где в исходном тексте обнаружена ошибка. При этом в верхней строке редактора появляется диагностическое сообщение. После нажатия на любую клавишу (кроме F1) верхняя строка восстанавливает свой первоначальный вид, и среда переходит к режиму редактирования. Если после появления сообщения об ошибке нажать F1, на экране появится окно справочной службы с легальной информацией об ошибке и рекомендациями по ее устранению. Некоторые ошибки в исходном тексте обнаруживаются не сразу, а в ходе продолжающегося контекстного разбора. Например, несоответствие типов в операторе присваивания не может быть обнаружено до тех пор, пока не будет вычислено целиком выражение в правой части этого оператора. В таких случаях ищите ошибку слева от курсора или в предыдущей строке текста.

Ниже приводятся сообщения об ошибках в том виде, в каком они появляются в верхней строке редактора, а также перевод сообщений справочной службы.

**1. Out of memory (Выход за границы памяти).** Компилятору не хватает памяти. Имеется ряд возможных решений этой проблемы:

* если в опции COMPILE / DESTINATION установлено значение MEMORY, замените эту опцию на DISK;
* если в опции OPTIONS / LINKER / LINK BUFFER установлено значение MEMORY, замените эту опцию на DISK;
* если Вы используете резидентные обслуживающие программы, такие как SIDEKICK, NORTON, удалите их из памяти;
* если Вы используете интегрированную среду TURBO.EXE, то попробуйте воспользо-ваться компилятором ТРС.ЕХЕ - он занимает меньше памяти.

Если ни одна из рекомендаций не помогает, то, возможно, Ваша программа просто слишком велика, чтобы компилировать ее в таком объеме памяти. В этом случае Вы должны разбить ее на два или более модулей.

**2. Identifier expected (He указанидентификатор).** В том месте должен находиться идентификатор. Возможно, Вы пытаетесь использовать в качестве идентификатора зарезервированное слово.

**3. Unknown identifier (Неизвестный идентификатор).** Этот идентификатор не был описан.

**4. Duplicate identifier (Двойной идентификатор).** Попытка дважды описать один и тот же идентификатор.

**5. Syntax error (Синтаксическая ошибка).** В исходном тексте найден недопустимый символ. Возможно, Вы забили заключить в апострофы строковую константу.

**6. Error in real constant (Ошибка в вещественной константе).** Синтаксис констант вещественного типа

**7. Error in integer constant (Ошибка в целой константе.).** Учтите, что после целых действительных чисел, превышающих диапазон представления целых чисел (-214748З648…+2147483647), должны ставиться точка и ноль, например, 12345678912.0.

**8. String constant exceeds line (Строковая константа превышает допустимые размеры).** Вероятно, Вы забыли поставить апостроф в конце строковой константы.

**9. Too many nested files (Слишком много вложенных файлов).** Компилятор допускает не более пяти уровней вложенности включаемых файлов (т.е. в исходном файле есть ссылка на включаемый файл, в котором есть ссылка на еще одни включаемый файл и т.д.).

**10. Unexpected end of file (He найденконецфайла).**Вы могли получить это сообщение об ошибке по одной из следующих причин:

* Ваш исходный файл закончился перед последним END основного раздела операторов;
* вероятно, в Вашей программе неодинаковое количество операторов BEGIN и END;
* включаемый файл заканчивается в середине раздела операторов;
* каждый раздел операторов должен целиком помещаться в одном файле;
* Вы не закончили комментарий.

**11. Line too long (Слишком длинная строка).**Максимальная длина строки, обрабатываемой компилятором, равна 126 символам (обратите внимание: редактор среды может обрабатывать строки практически неограниченной длины).

**12. Type identifier expected (Здесь нужен идентификатор типа).** Не указан тип идентификатора.

**13. Too many open files (Слишком много открытых файлов) .**Появление такой ошибки означает, что конфигурационный файл CONFIG.SYS операционной системы не включает параметр FILES=xx или этот параметр указывает слишком мало файлов. Увеличьте число файлов до нужного значения, например, до 80.

**14. Invalid file name (Неверное имя файла).** Имя файла неверно или указан несуществующий путь.

**15. File not found (Файлненайден).**Файл не был найден в просмотренных каталогах.

**16. Disk full (Диск заполнен).** Удалите некоторые файлы или воспользуйтесь новым диском.

**17. Invalid compiler directive (Неправильная директива компилятора).** Неверная буква в директиве компилятора, один из параметров директивы компилятора неверный, или Вы пользуетесь глобальной директивой компилятора, когда компиляция тела программы уже началась.

**18. Too many files (Слишком много файлов).** В компиляции программы или программного модуля участвуют слишком много файлов. Попытайтесь не использовать так много файлов, например, объединяя включаемые файлы или делая короче имена файлов.

**19. Undefined type in pointer definition (Неопределенный тут в объявлении указателя).** Попытка объявить типизированный указатель, связанный с ранее не объявленным типом данных.

**20. Variable identifier expected (Отсутствует идентификатор переменной).** На этом месте должен быть идентификатор переменной.

**21. Error in type (Ошибка в объявлении типа).** Объявление типа не может начинаться с этого символа.

**22. Structure too large (Слишком большая структура).** Максимально допустимый размер любого структурного типа составляет 65520 байт.

**23. Set base type of range (Базовый тип множества нарушает границы).** Базовый тип множества должен представлять собой тип-диапазон с границами в пределах от 0 до 255 или перечисляемый тип с не более чем 256 значениями.

**24. File components may not be files (Компонентамифайланемогутбытьфайлы).** Конструкции типа file of file (файл файлов) или file of object (файл объектов) не допускаются. Нельм объявлять любые структурные типы, которые используют в качестве компонентов объекты или файлы.

**25. Invalid string length (Неверная длина строки).** Объявляемая длина строки должна находиться в диапазоне от 1 до 255.)

**26. Type mismatch (Несоответствие типов).** Это сообщение может быть вызвано следующими причинами:

* несовместимые типы переменной и выражения в операторе присваивания;
* несовместимые типы фактического и формального параметров в обращении к процедуре или функции;
* тип выражения не совместим с типом индекса при индексировании массива;
* несовместимые типы операндов в выражении.

**27. Invalid subrange base type (Неправильный базовый тип для типа-диапазона).** Допустимыми базовыми типами являются все порядковые типы.

**28. Lower bound greater than upper bound (Нижняя граница больше верхней).** Описание типа-диапазона содержит неправильные границы.

**29. Ordinal type expected (Нужен порядковый тип).** Вещественные, сороковые, структурные, процедурные типы и указатели в данном месте программы не допускаются.

**30. Integer constant expected (Нужнацелаяконстанта).**

**31. Constant expected (Нужнаконстанта).**

**32. Integer or real constant expected (Нужнацелаяиливещественнаяконстанта).**

**33. Type identifier expected (Нуженидентификатортипа)**

**34. Invalid function result type (Неправильный тип результата функции).** Правильными типами результата функции являются все простые типы, строковые типы и указатели.

**35. Label Identifier expected (Нуженидентификаторметки).**Meтка не обозначена с помощью идентификатора, как это требуется из контекста программы.

**36. BEGIN expected (Нужен BEGIN).**

**37. END expected (Нужен END).**

**38. Integer expression expected (Нужновыражениетипа INTEGER).**

**39. Ordinal expression expected (Нужновыражениеперечисляемоготипа).**

**40. Boolean expression expected (Нужновыражениетипа BOOLEAN).**

**41. Operand types do not match operator (Типы операндов не соответствуют операции).** Данная операция не может быть применена к указанным операндам, например, ' А' div ' 2 ' .

**42. Error in expression (Ошибкаввыражении).**Данный символ не может участвовать в выражении указанным образом. Возможно, Вы забыли указать операцию между двумя операндами.

**43. Illegal assignment (Неверное присваивание).** Файлам и нетипизированным переменным нельзя присваивать значения. Идентификатору функции можно присвоить значение только внутри раздела операторов данной функции

**44. Field identifier expected (Нуженидентификаторполя).** Попытка использовать запись целиком в том месте, где требуется ссылка на какое-либо поле записи.

**45. Object file too large (Объектный файл слишком большой).** Турбо Паскаль не может компоновать файлы .OBJ больше 64 Кбайт.

**46. Undefined external (Неопределенная внешняя процедура).** Внешняя процедура или функция не имеет соответствующего определения PUBLIC в объектном файле. Убедитесь, что Вы указали все объектные файлы в директивах {$L <имя .OBJ файла>} и проверьте написание идентификаторов процедуры или функции в файле. ASM.

**47. Invalid object file record (Неправильная запись объектного файла).** Файл OBJ содержит неверную объектную запись. Убедитесь, что данный файл является действительно файлом .OBJ.

**48. Code segment too large (Сегмент кода слишком большой).** Максимальный размер кода программы или программного модуля равняется 65520 байтам. Разбейте Вашу программу или модуль на два или более модулей.

**49. Data segment too large (Сегмент данных слишком велик).** Максимальный размер сегмента данных программы равен 65520 байтам, включая данные, используемые программными модулями. Если Вам нужно большее количество глобальных данных, опишите большие структуры с помощью указателей и выделяйте для них память динамически с помощью процедуры NEW.

**50. DO expected (Нуженоператор DO).**

**51. Invalid PUBLIC definition (Неверное PUBLIC-определение).** Возможные причины сообщении:

* данный идентификатор получил тип PUBLIC с помощью соответствующей директивы языка ассемблера, но не cooтветствует описанию EXTERNAL в программе или программном модуле Паскаля;
* две или более директивы PUBLIC языка ассемблера определяют один и тот же идентификатор;
* файлы.OBJ определяют символы PUBLIC, не находящиеся в сегменте CODE.

**52. Invalid EXTRN definition (Неправильное EXTRN-определение).** Возможные причины сообщения:

* программа на ассемблере ссылается с помощью директивы EXTRN на идентификатор, который не описан в программе на Паскале и не был описан в интерфейсных секциях используемых программных модулей;
* ассемблерная программа ссылается на идентификатор, обозначающий абсолютную переменную (т.е. определенную словом ABSOLUTE);
* ассемблерная программа ссылается на идентификатор процедуры или функции типа INLINE.

**53. Too many EXTRN definition (Слишкоммного EXTRN-определений).** Турбо Паскаль не может обрабатывать файлы. OBJ при более чем 256 определениях EXTRN.

**54. OF expected (Требуется OF).**

**55. INTERFACE expected (Требуется интерфейсная секция).**

**56. Invalid relocatable reference (Неправильная перемещаемая ссылка).** Возможные причины сообщения:

* файл .OBJ содержит данные и перемещаемые ссылки в сегментах, отличных от CODE. Например, Вы пытаетесь описать инициализированные переменные в сегменте DATA;
* файл .OBJ содержит ссылки с размерами в байтах на перемещаемые символы. Такая ошибка происходит в случае, если Вы используете, операторы HIGH и LOW с перемещаемыми символами или если Вы ссылаетесь в директивах DB на перемещаемые символы;
* операнд ссылаемся на перемещаемый символ, который не был определен в сегменте CODE или а сегменте DAТА;
* операнд ссылается на процедуру EXTRN или функцию EXTRN со сдвигом, например, CALL SortProc+8,

**57. THEN expected (Требуется THEN).**

**58. TO or DOWNTO expected (ТребуетсяТОили DOWNTO).**

**59. Undefined forward (Неопределенное опережающее описание).** Возможные причины сообщения:

* была описана процедура иди функция в интерфейсной секции программного модули, но ее определение отсутствует в секции реализации;
* процедуры или функции были описаны с помощью опережающего описания, но их определение не кайлено.

**60. Too many procedures (Слишкоммногопроцедур).** Турбо Паскаль допускает не более 512 процедур или функций в одном модуле. Если Вы компилируете программу, то поместите некоторые процедуры или функции в модули. Если Вы компилируете модуль, то разбейте его на два или несколько модулей.

**61. Invalid typecast (Неверное преобразование типа).** Возможные причины сообщения:

* Вы пытаетесь разместить в памяти, занимаемой некоторой переменной, значение выражения другого типа в случае, когда размер размещаемого значения не равен размеру переменной;
* Вы пытаетесь осуществить преобразование типа выражения, когда разрешается только ссылка на переменную, процедуру или функцию.

**62. Division by zero (Деление на ноль).** Предшествующая операция пытается выполнить деление на ноль.

**63. Invalid file type (Неверный файловый тип).** Данный файловый тип не обслуживается процедурой обработки файлов. Например, процедура READLN используется для типизированного файла или процедура SEEK - для текстового файла.

**64. Cannot Read or Write variables of this type (Нет возможности считать или записать переменные данного типа).** Нарушены следующие ограничения:

* процедуры READ и KEADLN могут считывать переменные символьного, целого, действительного и строкового типов;
* процедуры WRITE и WRITELN могут выводить переменные символьного, целого, действительного, булевского и строкового типов.

**65. Pointer variable expected (Нужно использовать переменную-указатель).** Предыдущая переменная должна быть указателем.

**66. String variable expected (Нужна строковая переменная).** Предшествующая переменная должна иметь строковый тип.

**67. String expression expected (Нужно выражение строкового типе).** Предшествующее выражение должно иметь строковый тип.

**68. Circular unit reference (Перекрестная ссылка модулей).** Два модуля ссылаются друг на друга:

Unit A; Unit В;

Uses В; Uses A;

**69. Unit name mismatch (Несоответствие имен программных модулей).** Имя программного модуля, найденное в файле .TPU, не соответствует имени, указанному в предложении USES.

**70. Unit version mismatch (несоответствие версий модулей).** Один или несколько программных модулей, используемых данной программой, были изменены после их компиляции. Воспользуйтесь опцией COMPILE/MAKE или COMPILE/BUILD в интегрированной среде или опциями /М или /В в компиляторе ТРС, что позволит автоматически скомпилировать программные модули, нуждающиеся в перекомпиляции.

**71. Duplicate unit name (Повторное имя программного модуля).** Вы уже указали этот программный модуль в операторе USES.

**72. Unit file format error (Ошибкаформатафайламодуля).**Файл .TPU не соответствует стандарту Турбо Паскаля.

**73. IMPLEMENTATION expected (Отсутствует исполняемая часть модули).**

**74. Constant and case types do not match (Типы констант и тип выражения CASE не соответствуют друг другу).** Тип константы оператора CASE не совместим с выражением в операторе варианта.

**75. Record variable expected (Нужна переменная типа запись).** Предшествующая переменная должна иметь тип запись.

**76. Constant out of range (Константа нарушает границы).** Возможные причины сообщения:

* Вы пытаетесь указать индекс массива, выходящий за его границы;
* Вы пытаетесь присвоить переменной значение, выходящее за границы, допустимые для переменной;
* Вы пытаетесь передать в качестве фактического параметра процедуре или функции константу, выходящую за границы, допустимые для типа соответствующего формального параметра.

**77. File variable expected (Нужна файловая переменная).** Предшествующая переменная должна иметь файловый тип.

**78. Pointer expression expected (Нужно выражение типа указатель).** Предшествующее выражение должно иметь тип указателя.

**79. Integer or real expression expected (Нужно выражение вещественного или целого типа).** Предшествующее выражение должно иметь тип REAL или INTEGER.

**80. Label not within current block (Метка не находится внутри текущего блока).** Оператор GOTO не может ссылаться на истку, находящуюся вне текущего блока.

**81. Label already defined (Метка уже определена).** Данная метка уже помечает оператор.

**82. Undefined label in processing statement part (Неопределенная метка в предшествующем разделе операторов).** Данная метка была описана, и на нее осуществлялась ссылка в предшествующем разделе опера она не указана в тексте программы.

**83. Invalid @ argument (Неправильный аргумент операции @).** Правильными аргументами являются идентификаторы переменных, процедур и функций.

**84. Unit expected (Нужно кодовое слово UNIT).**

**85. “;” expected (Нужно указать «;»).**

**86. “:” expected (Нужно указать”:”).**

**87. “,” expected (Нужно указать «,»).**

**88. “(” expected (Нужно указать “(”).**

**89. «)» expected (Нужно указать «)»).**

**90. «=» expected (Нужно указать “=”).**

**91. “:=” expected (Нужно указать":=”).**

**92. “[” or “(.” expected (Нужно указать “[” или “(.”).**

**93. “]” or “.)” expected (Нужно указать"]” или «.)»).**

**94. “.” expected (Нужно указать «.»).**

**95. “..” expected (Нужно указать «..»).**

**96. Too many variables (Слишком много переменных).** Нарушены следующие ограничения:

* общин размер глобальных переменных, описанных в программе или программной модуле, не может превышать 64 Кбайт;
* размер локальных переменных, описанных в процедуре или функции, не может превышать 64 Кбайт.

**97. Invalid FOR control variable (Неправильный параметр цикла оператора FOR).** Параметр цикла оператора FOR должен быть переменной порядкового типа, определенной в разделе описаний текущей подпрограммы.

**98. Integer variable expected (Нужна переменная целого типа).** Предшествующая переменная должна иметь целый тип.

**99. File and procedure types are not allowed here (Здесь не могут использоваться файлы или процедурные типы).** Типизированная константа не может иметь файловый или процедурный тип.

**100. String length mismatch (Несоответствие длины строки).** Длина строковой константы не соответствует количеству элементов символьного массива.

**101. Invalid ordering of fields (Неверныйпорядокполей).**Поля в константе типа запись должны записываться в порядке их описания.

**102. String constant expected (Нужнаконстантастроковоготипа).**

**103. Integer or real variable expected (Нужнапеременнаятипа INTEGER или REAL).**Предшествующая переменная должна иметь целый или вещественный тип.

**104. Ordinal variable expected (Нужна переменная порядкового типа).** Предшествующая переменная должна иметь порядковый тип.

**105. INLINE error (Ошибка в операторе INLINE).** Оператор «<» не допускается в сочетании с перемещаемыми ссылками на переменные. Такие ссылки всегда имеют размер в слово.

**106. Character expression expected (Предшествующее выражение должна иметь символьный тип).**

**107. Too many relocation items (Слишком много перемещаемых элементов).** Размер таблицы перемещения файла .ЕХЕ превышает 64 Кбайта, что является верхним пределом в Турбо Паскале. Если Вы обнаружили эту ошибку, то это значит, что программа просто слишком велика для обработки компоновщиком Турбо Паскаля. Возможно также, что она слишком велика для выполнения в MS-DOS. В таком случае нужно выделить в программе основной раздел, который выполнял бы обращение к двум или более вспомогательным разделам с помощью процедуры ЕХЕС из модуля DOS.

**108. Overflow in arithmetic operator (Переполнение при выполнении арифметического оператора).** Результат предыдущей арифметической операции не лежит в диапазоне -2146483648… +2147483647. Исправьте операцию или используйте вещественные типы вместо целочисленных.

**109. No enclosing FOR, WHILE or REPEAT statement (Нетоператоров, заканчивающихоператоры FOR, WHILE или REPEAT).**Процедуры BREAK и CONTINUE не могут вызываться вне тела оператора цикла.

**110. Debug information table overflow (Переполнение информационной таблицы отладки).** Возможно, программа содержит более 65536 имен или 65536 строк. Отключите генерацию таблиц отладки директивой компилятора {SD-} или исправьте один или более модулей.

**111.** Ошибка с этим кодом не описана в версии 7.0 системы Турбо Паскаль.

**112. CASE constant out of range (Константа CASE нарушает допустимые границы).** Целочисленные константы оператора CASE должны находиться в диапазоне от -32768 до 32767.

**113. Error in statement (Ошибкавоператоре).** Данный символ не может быть первым символом в операторе.

**114. Cannot call an interrupt procedure (Невозможно вызвать процедуру прерывания).** Вы не можете непосредственно вызвать процедуру обработки прерывания.

**115.** Ошибка с этим кодом не описана в версии 7.0 системы Турбо Паскаль.

**116. Must be in 8087 mode to compile this (Длякомпиляциинеобходимрежим 8087).**Данная программа может быть скомпилирована только в режиме {SN+}. В состоянии {SN-} операции с типами SINGLE, DOUBLE EXTENDED и COMP не допускаются.

**117. Target address not found (Указанный адрес не найден).** Команда COMPILE / FIND ERROR в среде Турбо Паскаля (или поиск с помощью опции /F в командной строке компилятора ТРС.ЕХЕ) не обнаружила оператор, соответствующий заданному адресу.

**118. Include files are not allowed here (Здесьнедопускаютсявключаемыефайлы).**Раздел операторов должен целиком размешаться в одном файле.

**119. No inherited methods are accessible here (В этом месте программы нет унаследованных методов).** Вы используете зарезервированное слово INHERITED вне метода объекта или в методе, который не унаследован от родительского объекта.

**120.** Ошибка с этим кодом не описана в версии 7.0 системы Турбо Паскаль.

**121. Invalid qualifier (Неверный квалификатор).** Возможные причины сообщения:

* Вы пытаетесь индексировать переменную, которая не является массивом;
* Вы пытаетесь укачать поля в переменной, которая не является записью;
* Вы используете в качестве указателя переменную, которая не является указателем.

**122. Invalid variable reference (Недействительная ссылка на переменную).** Предыдущая конструкция удовлетворяет синтаксису ссылки на переменную, но она не указывает адрес памяти. Возможно, Вы вызываете функцию-указатель, но забываете сослаться на результат с помощью знака.

**123. Too many symbols (Слишком много символов).** Программа или программный модуль содержат более 64 Кбайт символов. Если Вы компилируете программу с директивой {SD+}, то попробуйте отключить эту директиву или разбейте программу на несколько модулей.

**124. Statement part too large (Слишком большой раздел операторов).** Турбо Паскаль ограничивает размер раздела операторов, но величины примерно 24 Кбайта. Если Вы обнаружили эту ошибку, поместите части раздела операторов в одну или несколько процедур и вообще сделайте Вашу программу более структурированной.

**125.** Ошибка не этим колом не описана в версии 7.0 системы Турбо Паскаль.

**126. Piles must be war parameters (Файлы должны передаваться как параметры-переменные).** Вы пытаетесь передать процедуре или функции параметр-значение файлового типа. Параметры файлового типа должны быть параметрами-переменными.

**127. Too many conditional symbols (Слишком много условных символов) .**Недостаточно памяти для определения условных символов (слов, управляющих командами условной компиляции). Попытайтесь удалить некоторые символы или уменьшить их длину.

**128. Misplacedconditionaldirective (Пропущена условная директива).**Компилятор обнаружил директиву {$ELSE} или {$ENDIF} без соответствующих директив {$IFDEF},. {$1FNDEF) или {$IFOPT}.

**129. ENDIF directive missing (Пропущена директива EHDIF).** Исходный файл закончился внутри конструкции условной компиляции. В исходном файле должно быть равное количество директив {$IFxxx} и {$ENDIF}.

**130. Error in initial conditional defines (Ошибкавусловныхопределениях).**Исходные условные символы, указанные в опции OPTIONS/COMPILER / CONDITIONAL DEFINES являются недействительными. Турбо Паскаль требует нуля или более идентификаторов, разделенных пробелами, запятыми или точками с запятой.

**131. Header does not match previous definition (Заголовок не соответствует предыдущему определению).** Возможные причины сообщения:

* заголовок процедуры или функции, указанный в интерфейсной секции, не соответствует заголовку в исполняемой части.
* заголовок процедуры или функции, указанный с помощью опережающего описания FORWARD, не соответствует заголовку найденной далее одноименной процедуры или функции.

**132. Critical disk error (Критическая ошибка диска).** Но время компиляции произошла критическая ошибка диска (например, дисковод находится в состоянии “не готов”).

**133. Cannot evaluate this expression (Нельзявычислитьданноевыражение).**В выражении-константе или в отладочном выражении Вы пытаетесь использовать неподдерживаемые средства, например, я описании константы пытаетесь использовать функцию SIN или вызвать в отладочном выражении определенную пользователем функцию.

**134. Expression incorrectly terminated (Некорректное завершение выражения).** Контекстуально в данном месте программы должен быть конец выражения или оператора.

**135. Invalid format specifier (Неверный спецификатор формата) .**Используется неверный спецификатор формата или числовой аргумент спецификатора формата выходит за допустимые границы.

**136. Invalid indirect reference (Недопустимая косвенная ссылка).** Оператор пытается осуществить недопустимую косвенную ссылку. Например, Вы используете абсолютную переменную, базовая переменная которой в текущем модуле неизвестна, или используете программу типа INLINE, в которой делается ссылка на переменную, неопределенную в текущем модуле.

**137. Structured variable are not allowed here (Здесь нельзя использовать переменную структурного типа).** Делается попытка выполнить нал переменной структурного типа неподдерживаемую операцию. Например. Вы пытаетесь перемножить две записи.

**138. Cannot evaluate without System unit (Нельзя вычислить выражение без модуля SYSTEM).** Чтобы отладчик смог вычислить выражение, в файле TURBO.TPL должен содержаться модуль SYSTEM.

**139. Cannоt access this symbol (Нет доступа к данному символу).** Как только Вы скомпилируете программу, все множество ее символов становится доступным. Однако к отдельным символам (например, к переменным) нельзя получить доступ, пока Вы не запустите программу.

**140. Invalid floating-point operation (Недопустимая операция с плавающей запятой).** При выполнении операции с плавающей запятой произошло переполнение или деление на ноль.

**141. Cannot compile overlay to memory (Нельзя выполнить компиляцию оверлейных модулей в память).** Программа, использующая оверлейные модули, должна компилироваться на диск.

**142. Procedure or function variable expected (Должна использоваться переменная процедурного типа).** В этом контексте оператор получения адреса @ может использоваться только с переменной процедурного типа.

**143. Invalid procedure or function reference (Недопустимаяссылканапроцедуруилифункцию).**Возможные причины сообщения:

* Вы пытаетесь вызвать процедуру в выражении;
* процедура или функция, использующаяся в качестве параметра вызова другой процедуры или функции, должна компилироваться в состоянии {$F+} и не может описываться с помощью зарезервированных слов INLINE или INTERRUPT.

**144. Cannot overlay this unit (Этот модуль не может использоваться в качестве оверлейного).** Попытка использовать в качестве оверлейного модуля, который не был скомпилирован с директивой {$O+}.

**145. Too many nested scopes (Слишком много вложений).** В программе не может быть больше 512 вложений с не более чем 128 вложениями в каждом модуле. Вложениями считаются:

* каждый модуль в предложении USES;
* каждая вложенная запись в типе RECORD;
* каждый вложенный оператор WITH.

**146. File access denied (Отказано в доступе к файлу).** Возможные причины:

* Вы пытаетесь использовать файл с атрибутом "только для чтения” в качестве выводного файла;
* Вы используете имя каталога вместо имени выводного файла.

**147. Object type expected (Здесьдолженбытьтип OBJECT).**Этот идентификатор должен принадлежать к типу OBJECT.

**148. Local object types are not allowed (Нельзяобъявлятьлокальныеобъекты).**Нельзя объявить объект в процедуре (функции).

**149. VIRTUALexpected (Пропущено слово VIRTUAL).**

**150. Method identifier expected (Пропущен идентификатор инкапсулированного правила).**

**151. Virtual constructor are riot allowed (Конструктор не может быть виртуальным).**

**153. Destructor identifier expected (Пропущен идентификатор деструктора).**

**154. Fail only allowed within constructor (Обращение к стандартной процедуре FAIL может содержаться только в конструкторе).**

**155. Invalid combination of opcode and operands (Недопустимая комбинация хода команды и операндов).** Код ассемблерной команды не может иметь такие операнды. Причины ошибки:

* указано слишком много или слишком мало операндов для данной команды, например, INC АХ,ВХ или MOV АХ;
* количество операндов правильное, но их типы или порядок следования не соответствуют данной команде, например, DEC 1, MOV AX, CL или MОV 1,АХ.

**156. Memory reference expected (Отсутствуетадрес).**Операнд ассемблерной команды не содержит адрес. Вероятно, Вы забыли выделить квадратными скобками индексный регистр.

**157. Cannot add or subtract relocatable symbols (Нельзя складывать или вычитать перемещаемые символы).** В ассемблерных выражениях обоими операндами в операциях сложения или вычитания могут быть только ссылки на константу. Идентификаторы переменных, процедур, функций и меток являются переметаемыми символами и не могут использоваться в качестве двух операндов одновременно в таких операциях. Если VAR -идентификатор переменной, a CONST - константы, то предложения MOV АХ, CONST+CONST и MOVE AХ, VAR+CONST будут правильными, в то время как выражение MOV AX, VAR+VAR Недопустимо.

**158. Invalid register combination (Недопустимая комбинация регистров) .**Допустимыми комбинациями индексных регистров являются [ВХ] , [ВР] , [SI], [DI] , [BX+SI], [BX+DI], [BP+SI] и [BP+DI] . Другие комбинации (например, [AХ] , [ВР+BХ] , [SI + DX]) недопустимы. Заметим, что локальные переменные размещаются на стеке и доступ к ним организуется через регистр BP. Ассемблер автоматически добавляет [ВР] в ссылках на такие переменные, поэтому в операндах типа LOCAL [BX], где LOCAL - локальная переменная, образуется недопустимая ссылка LOCAL[BP+BX] .

**159. 286/287 instructions are not enabled (Недоступен набор команд микропроцессоров 286/287).** Используйте директиву {SG+}, но учтите, что такую программу можно исполнять только на ПК, оснащенных микропроцессором Intel 80286/80287 или более поздним.

**160. Invalid symbol reference (Недопустимая ссылка на символ).** К указанному символу нельзя обратиться в ассемблерной программе. Это может быть следствием таких причин:

* Вы пытаетесь использовать ссылки на стандартные процедуры (функции) или специальные массивы MEM, MEMW, MEML, PORT я PORTW в ассемблерных операндах;
* Вы обращаетесь к строковой, вещественной константе или константе типа SET в операнде ассемблерной команды;
* в ассемблерном операнде Вы обращаетесь к процедуре или функции, написанной я режиме INLINE;
* Вы пытаетесь получить с помощью операции @Result доступ к результату, возвращаемому функцией;
* Вы пытаетесь использовать короткую форму команды JMP для передачи управления внепомеченный оператор программы.

**161. Code generation error (Ошибка генерации кода).**

**162. ASM expected (Отсутствует зарезервированное слово ASM).**

**ОШИБКИ, ВОЗНИКАЮЩИЕ**

**ВО ВРЕМЯ ВЫПОЛНЕНИЯ ПРОГРАММ**

Некоторые ошибки, обнаруженные но время выполнения программы, приводят к появлению на экране сообщения вида **Runtime error nnn at хххх: уууу (Ошибка периода исполнения nnn по адресу хххх: уууу)**, где nnn - номер ошибки; хххх:уууу - адрес (сегмент и смещение). После этого сообщения программа завершает свою работу.

Ошибки периода исполнения делятся на четыре категории: ошибки, обнаруживаемые ДОС (коды ошибок 1 до 99), ошибки ввода-ввывода (с 100 по 149), критические ошибки (с 150 по 199) и фатальные ошибки (коды ошибок с 200 до 255).

**ОШИБКИ, ОБНАРУЖИВАЕМЫЕ ДОС**

**1. Invalid function number (Неверный номер функции).** Вы пытаетесь вызвать несуществующую функцию ДОС.

**2. File not found (Fie найденфайл).** Ошибка генерируется процедурами RESET, APPEND, RENAME или ERASE в случае, если имя, присвоенное файловой переменной, указывает несуществующий файл.

**3. Path not found (Путь не найден).** Ошибка генерируется процедурами:

* RESET, REWRITE, APPEND или ERASE в случае, если имя, присвоенное файловой переменной, является недействительным иди указывает на несуществующий подкаталог;
* CHDIR, MKDIR или RMDIR в случае, если путь является недействительным или указывает несуществующий подкаталог.

**4. Too many open files (Слишком много открытых файлов).** Ошибка генерируется процедурами RESET, REWRITE или APPEND в случае, если программа имеет слишком много открытых файлов, ДОС не позволяет использовать более 15 открытых файлов для каждого процесса. Если ошибка возникла при наличии менее 15 открытых файлов, то она может указывать на то, что файл CONFIG.SYS не содержит параметра FILES=xxx или что этот параметр задает слишком мала файлов. Увеличьте параметр FILES=xxx до какого-либо подходящего значения, например, до 80.

**5. File access defined (Отказано в доступе к файлу).** Данная ошибка генерируется процедурами:

* RESET или APPEND в случае, когда имя.присвоенное файловой переменной, указывает каталог или файл, доступный только для чтения, в то время как параметр F1LEMODE файловой переменной содержит указание на запись данных;
* REWRITE и случае, когда каталог заполнен, или если имя, присвоенное файловой переменной, задает каталог или существующий файл, доступный только для чтения;
* RENAME в случае, если имя, присвоенное файловой переменной, указывает каталог или если новое имя указывает существующий файл;
* ERASE в случае, если имя, присвоенное файловой переменной, указывает каталог или файл, доступный только для чтения;
* MKDIR в случае, если файл с тем же именем уже существует в порождающем каталоге, если нет места в порождающем каталоге или если путь к каталогу содержит имя логического устройства;
* RMDIR в случае, если каталог не является пустым, если путь не определяет каталог или если пул залает корневой каталог;
* READ или BLOCKREAD в случае типизированного или нетипизированного файла, если файл не открыт для чтения;
* WRITE или BLOCKWRITE для типизированного или нетипизированного файла в случае, если этот файл не открыт для записи.

**6. Invalid filе handle (Недопустимый файловый канал).** Данная ошибка генерируется в случае, когда системному вызову ДОС передается недопустимый файловый канал. Эта ошибка не должна возникать в правильно работающей программе. Ее появление является свидетельством того, что файловая переменная каким-либо образом испорчена.

**12. Invalid file access code (Недействительный код доступа к файлам).** Ошибка генерируется процедурами RESET или APPEND, если значение параметра FILEMОDE файловой переменной не является допустимым.

**15. Invalid drive number (Недопустимый номер дисковода).** Ошибка генерируется процедурой CETDIR в случае, если номер дисковода не является допустимый.

**16. Cannot remove current directory (Нельзяудалитьтекущийкаталог).**Ошибка генерируется процедурой RMDIR в случае, если путь указывает текущий каталог.

**17. Cannot rename across drives (Нельзя при переименовании указывать разные дисководы).** Генерируется процедурой RENAME в случае, если оба файла не находятся на одном и том же диске.

**ОШИБКИ ВВОДА-ВЫВОДА**

Если один из операторов компилировался с директивой {$I+}, то ошибка ввода-вывода приводит к прекращению выполнения программы. В состоянии {$I-} программа продолжает выполняться, а ошибка возвращается функцией IORESULT.

**100. Disk read error (Ошибка чтения с виска).** Генерируется процедурой READ в типизированном файле в случае, если Вы пытаетесь осуществить считывание из полностью прочитанного файла.

**101. Disk write error (Ошибка записи на диск).** Ошибка генерируется процедурами CLOSE, WRITE, WRITELN, FLUSH в случае, если диск заполнен.

**102. File not assigned (Файлунеприсвоеноимя).**Ошибка генерируется процедурами RESET, REWRITE, APPEND, RENAME и ERASE в случае, если файловой переменной не было присвоено имя файла с помощью обращения к процедуре ASSIGN.

**103. File not open (Файлнеоткрыт).**Ошибкагенерируетсяпроцедурами CLOSE, READ, WHITE, SEEK, EOF, FILEPOS, F1LESIZE, FLUSH, BLOCKHEAD, BLOCKWR1TE вслучае, еслифайлнеоткрыт.

**104. File not open for input (Файлнеоткрытдляввода).** Ошибка генерируется процедурами READ, READLN, EOF, EOLN, SEEKEOF или SEEKEOLN в текстовом файле в случае, если файл не открыт для ввода.

**105. File not open for output (Файл не открыт для вывода).** Ошибка генерируется процедурами WRITE или WR1TELN в текстовом файле в случае, если файл не открыт для вывода.

**106. Invalid numeric format (Неверный числовой формат).** Генерируется процедурами READ или READLN в случае, если числовое значение, считанное из текстового файла, не соответствует правильному числовому формату.

**КРИТИЧЕСКИЕ ОШИБКИ**

**150. Disk is write protected (Диск защищен от записи).**

**151. Unknown unit (Неизвестный модуль).**

**152. Drive not ready (Дисковод находится в состоянии “не готов”).**

**153. Unknown command (Неопознанная команда).**

**154. CRC error in data (Ошибка в исходных данных).**

**155. Bad drive requiest structure length (При обращении к диску указана неверная длина структуры).**

**156. Disk seek error (Ошибка при операции установки головок на диске).**

**157. Unknown media type (Неизвестный тип носителя).**

**158. Sector not found (Сектор не найден).**

**159. Printer out of paper (Кончилась бумага на принтере).**

**160. Device write fault (Ошибка при записи на устройство).**

**161. Device read fault (Ошибка при чтении с устройства).**

**162. Hardware failure (Сбой аппаратуры).**

**ФАТАЛЬНЫЕ ОШИБКИ**

Эти ошибки всегда приводят к немедленной остановке программы.

**200. Division by zero (Деление на ноль).**

**201. Range check error (Ошибка при проверке границ).** Ошибка генерируется операторами, скомпилированными в состоянии {$R+}, при возникновении одной из следующих ситуаций:

* индексное выражение массива находилось вне допустимого диапазона;
* была осуществлена попытка присвоить переменной значение, находящееся вне диапазона переменной;
* была осуществлена попытка передать значение, находящееся вне допустимого диапазона, а качестве параметра процедуре или функции.

**202. Stack overflow error (Переполнение стека)** Эта ошибка генерируется на входе в процедуру или функцию, скомпилированную в режиме {$S+}, случае, если нет достаточной области для размещения локальных переменных подпрограммы. Увеличьте размер стека, используя директиву компилятора {$М}

**203. Heap overflow error (Переполнение кучи).** Эта ошибка генерируется процедурами NEW или GETMEM в случае, если в куче нет свободной памяти требуемого размера.

**204. Invalid pointer operation (Недействительная операция с указателем).** Эта ошибка генерируется процедурами DISPOSE или FREEMEM в случае, когда указатель имеет значение NIL или указывает на адрес, лежащий за пределами динамически распределяемой области памяти.

**205. Floating point overflow (Переполнение при операции с плавающей запятой).** В результате выполнения операции с плавающей запятой получено слишком большое вещественное число.

**206. Floating point underflow (Исчезновение порядка при операции с плавающей запятой).** Эта ошибка генерируется только в том случае, если используется сопроцессор 8087/80287/80387 с управляющим словом, которое демаскирует ошибку исчезновения порядка. По умолчанию исчезновение порядка приводит к возвращению результата, равного нулю.

**207. Invalid floating point operation (Недопустимая операция с плавающей запятой).** Возможные причины сообщения:

* аргумент функций TRUNC или ROUND не может быть преобразован в целое число, находящееся внутри диапазона типа LONGINT (от -2147483648 до -12147483647);
* отрицательный аргумент функции SQRT (извлечение квадратного корня);
* аргумент функции LN (логарифм) равен нулю или имеет отрицательное значение;
* произошло переполнение стека сопроцессора.

**208. Overlay manager not installed (He установлена подсистема управления оверлеем).** Ваша программа вызывает оверлейную процедуру или функцию, а подсистема управления оверлеем не инициирована. Вероятнее всего, в программе отсутствует обращение к процедуре OVRINIT или обращение к этой процедуре завершилось с ошибкой. Нужно иметь в виду, что если в каком-либо из оверлейных модулей содержится раздел инициации, то в программе необходимо создать дополнительный или использовать имеющийся неоверлейный модуль, вызывающий процедуру OVRINIT в своем разделе инициализации, и указать этот модуль в предложении USES перед любым из оверлейных модулей.

**209. Overlay file read error (Ошибка чтения оверлейного файла).** Ошибка чтения произошла, когда подсистема управления оверлеем пыталась считать оверлейный модуль из оверлейного файла.

**210. Object not initialized (He инициированобъект).**Вы обращаетесь к виртуальному правилу применительно к неинициированному объекту (до вызова конструктора).

**211. Call to abstract method (Вызовабстрактногоправила).** Эта ошибка генерируется правилом ABSTRACT модуля OBJECT при работе в среде Turbo Vision в случае обращения к абстрактному правилу, т.е. к виртуальному правилу, которое разработано специально для его замены в объектах-потомках.

**212. Stream registration errata (Ошибка в регистрируемом потоке).** Эта ошибка оперируется правилом REGISTERTYPE модуля OBJECT при работе в среде Turbo Vision в случае возникновения одной из следующих ошибок:

* запись регистрируемого потока не содержит сегмента данных;
* поле OBJTYPE записи регистрируемого потока нулевое;
* указанный гип уже был зарегистрирован;
* существует другой тип с таким же полем OBJTYPE.

**213. Collection index out of range (Набираемый индекс выходит из границ диапазона).** Индекс, передаваемый объекту TCOLLECTION при работе в среде Turbo Vision, выходит за границы диапазона.

**214. Collection overflow error (Переполнение коллекции).** Эта ошибка генерируется объектом TCOLLECTION при работе в среде Turbo Vision в случае, если делается попытка добавить элемент к коллекции, которую нельзя расширять.

**Литература**

***Основная***

1. Бобровский, С., Delphi 5 [Текст]: учебный курс / С. Бобровский. - "[Питер](http://www.colibri.ru/pubinfo.asp?cod=1436)", 2009. – с. 640.
2. Глинский, Я.Н., Turbo Pascal 7.0 и Delphi [Текст] / Я.Н. Глинский и др. - "ДиаСофт", 2010. – 208 с.
3. Голицына, О.Л., Основы алгоритмизации и программирования [Текст]: Учебное пособие / О.Л. Голицына, И.И. Попов – М.: Форум: Инфра-М, 2010.
4. Гофман, В.Э. Delphi 6. В подлиннике [Текст] / В.Э. Гофман, и др. – "[BHV-Санкт-Петербург](http://www.colibri.ru/pubinfo.asp?cod=12)", 2007. – 1152 с.
5. Грызлов, В.И., Турбо Паскаль 7.0. [Текст] / В.И. Грызлов, Т.П. Грызлова – М.: ДМК, 2008.
6. Климова, Л.М. Pascal 7.0. Практическое программирование. [Текст]: Решение типовых задач. 3-е изд., доп. / Л.М. Климова – "[КУДИЦ-Образ](http://www.colibri.ru/pubinfo.asp?cod=931)", 2008. – 528 с.
7. Культин, Н. Turbo Pascal в задачах и примерах [Текст] / Н. Культин. – "[BHV-Санкт-Петербург](http://www.colibri.ru/pubinfo.asp?cod=12)", 2010. – 256 с.
8. Меняев, М.Ф.Информатика и основы программирования[Текст]/ М. Меняев – « Омега-Л», 2007
9. Немнюгин, С.Н. Pascal [Текст]: учебник / С.Н. Немнюгин – "[Питер](http://www.colibri.ru/pubinfo.asp?cod=1436)", 2001. – 496 с.
10. Фаронов, В.В. Turbo Pascal 7.0 [Текст]: практика программирования. 7-е издание / В.В. Фаронов – "[Нолидж](http://www.colibri.ru/pubinfo.asp?cod=1324)", 2010
11. Фаронов, В.В. Delphi 6. [Текст]: учебный курс / В.В. Фаронов - "[Нолидж](http://www.colibri.ru/pubinfo.asp?cod=1324)", 2010. – 608 с.
12. Фаронов, В.В. Программирование баз данных в Delphi 7. [Текст]: учебный курс / В.В. Фаронов –«Питер», 2006- 728с.
13. Фаронов, В.В. Искусство создания компонентов Delphi [Текст]: учебный курс / В.В. Фаронов –«Питер», 2005.

***Дополнительная***

1. Альфред, В. А, Моника С. Лам, Рави Сети, Джеффри Д. Ульман Компиляторы. Принципы, технологии и инструментарий [Текст]: учебный курс /В.А. Альфред, Моника С. Лам, Рави Сети, Джеффри Д. Ульман- «Вильямс», 2008 – 786 с.
2. Зеленяк, О.П. Решение задач по планиметрии. Технология алгоритмического подхода на основе задач-теорем. Моделирование в среде Turbo Pascal [Текст]:\ О.П. Зеленяк – « ДиаСофтЮП, ДМК пресс», 2008 – 346 с.
3. Тейксейра, С., Delphi 5. [Текст]: руководство разработчика. Т. 1. Основные методы и технологии программирования. / С. Тейксейра, К. Пачеко – «Вильямс», 2000. – 832 с.
4. Тейксейра, С., Delphi 5. [Текст]: руководство разработчика Т. 2. Разработка компонентов и программирование баз данных + (CD) / С. Тейксейра, К. Пачеко – "Вильямс", 2000. – 992 с.
5. Фленов, М Библия Delphi, 2-ое издание[Текст]: руководство разработчика\М. Фленов-« БХВ-Петербург», 2008- 658 с.

***Интернет-источники:***

1. [Электронный ресурс] http://www.codenet.ru
2. [Электронный ресурс ]<http://www.chemisk.narod.ru/html/algoritm01.html>
3. Хант, Э Томас, Д Программист-прагматик. Путь от подмастерья к мастеру [Текст]: руководство разработчика \ Э. Хант, Д. Томас « Лори, Питер Пресс», 2007
4. Хомоненко , А.Самоучитель Delphi .NET[Текст]: руководство разработчика./А. Хоменко- «БХВ-Петербург», 2006-604 с.

Учебное издание

Татьяна Николаевна Коренькова

Елена Николаевна Запасник

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Учебное пособие для студентов специальности

230401 – Информационные системы (по отраслям)

*Технический редактор:* Иванова Н.И.

*Компьютерная верстка:* Коренькова Т.Н.

Подписано к печати \_\_\_\_\_\_\_\_\_\_\_\_\_\_Бумага для множительной техники

Формат \_\_\_\_ \_\_\_\_\_Усл.печ.лист.\_\_ \_\_\_\_\_\_\_Тираж\_\_\_\_\_\_\_\_экз. Заказ\_\_\_\_\_\_\_\_\_

Отпечатано с авторского оригинала в отделе оперативной печати

Старооскольского технологического института.

Старый Оскол, микрорайон Макаренко, 40.