Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения вычислительной
техники и автоматизированных систем

РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ

Лабораторный практикум

Часть 1

Минск
БНТУ
2010

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ

Лабораторный практикум
для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий», 1-40 01 02 «Информационные системы и технологии»

В 2 частях

Часть 1

Минск
БНТУ
2010

УДК 004.4'236(076.5)
~~ББК 32.97~~
Р 17

Составитель *Гурский Н.Н.*

Рецензенты:
канд. техн. наук, профессор БГАТУ *Р.И. Фурунжиев*;
канд. физ-мат. наук, доцент БНТУ *В.А. Казакевич*

Р 17 Разработка приложений в визуальных средах: лабораторный практикум для студентов специальностей 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 01 02 «Информационные системы и технологии»: в 2 ч. / сост. Н.Н. Гурский. – Минск: БНТУ, 2010. – Ч. 1. – 59 с.

В практикуме представлен комплекс заданий для выполнения лабораторных работ по первой части дисциплины «Разработка приложений в визуальных средах», посвященной изучению основ разработки приложений в визуальной среде Delphi; рассмотрены принципы построения приложений, связанные с использованием основных компонентов; приведен список учебной литературы.

ISBN 978-985-525-455-4 (Ч. 1)
ISBN 978-985-525-456-1

Содержание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

При выполнении лабораторных работ необходимо:

1. В соответствии с целью работы сформулировать задачу, которая должна быть решена с помощью приложения.

2. Разработать алгоритм решения задачи.

3. Разработать приложение, включающее интерфейс, программные модули вычислительных процедур, формы представления результатов.

4. Выполнить компьютерное моделирование.

5. Произвести тестирование алгоритма и приложения.

6. Сделать выводы и обобщения.

7. Составить электронный вариант отчета с результатами выполнения приложения.

Образец оформления титульного листа приведен в приложении. При выполнении работ рекомендуется обратиться к литературе [1–9].

ЛАБОРАТОРНАЯ РАБОТА 1

Разработка простейшего приложения в визуальной среде Delphi

Цель: изучить основы среды Delphi и разработать простейшее приложение.

Краткие сведения

Интегрированная среда разработчика Delphi

Среда Delphi визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске Delphi на экране появляется главное окно, приведенное на рис. 1.

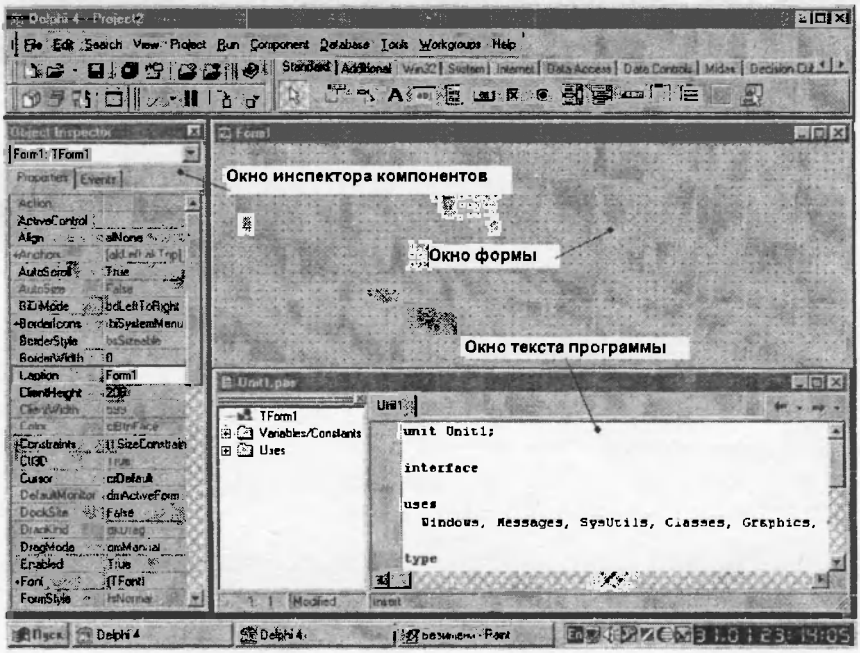


Рис. 1. Главное окно Delphi

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов осуществляется доступ к наборам их свойств, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать. Например, цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора компонентов (вызывается с помощью клавиши F11) предназначено для изменения свойств компонента – закладка *Properties* (свойства) – и создания обработчиков (процедур) при активизации тех или иных событий – страница *Events* (события).

Окно формы представляет собой внешний вид создаваемого Windows-приложения. В это окно в процессе проектирования интерфейса программы помещаются необходимые компоненты. Причем при выполнении программы большинство из помещенных компонентов будут иметь тот же вид, что и на этапе проектирования.

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. В системе Delphi используется язык программирования *Object Pascal*. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-приложения. При размещении некоторого компонента на форме происходит автоматическая фиксация его имени в коде программы.

Программа в среде Delphi представляется набором процедур, вызываемых при наступлении того или другого события и, таким образом, реализует событийно управляемую модель программируемого процесса обработки данных. Для каждого воз-

никающего на форме события с помощью страницы *Events* инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами *begin* и *end* которой необходимо записать на языке Object Pascal требуемый алгоритм.

Переключение между окном формы и окном текста программы осуществляется с помощью клавиши F12.

Структура приложения в Delphi

Приложение в Delphi состоит из файла проекта (*.dpr),* одного или нескольких файлов исходного текста (*.pas),* файлов с описанием компонентов, расположенных на форме (*.dfm*).

В ***файле проекта*** находится информация о модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой Delphi.

Файл исходного текста – программный модуль (Unit) предназначен для размещения текста программы на языке Object Pascal.

Модуль состоит из 2 разделов: интерфейсного (interface) и реализации (implementation). В интерфейсном разделе описываются типы, переменные, заголовки процедур и функций, которые могут быть использованы другими модулями. В разделе реализации располагаются тела процедур и функций, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля.

Структура модуля имеет вид:

```
Unit Unit1;
 interface
   //Раздел объявлений
 implementation
   //Раздел реализации
begin
   //Раздел инициализации
end.
```

При компиляции программы Delphi создает файл с расширением *.dcu,* содержащий в себе результат перевода в машинные коды содержимого файлов с расширением *.pas* и *.dfm.* Компоновщик преобразует файлы с расширением *.dcu* в единый загружаемый файл с расширением *.exe.* В файлах, имеющих расширение *.-df, .-dp, .-pa,* хранятся резервные копии файлов с образом формы, проекта и исходного текста соответственно.

Постановка задачи

Разработать линейное приложение с использованием компонентов TLabel, TEdit, TMemo, TButton в соответствии с индивидуальным вариантом задания.

Задания

1. $t = \dfrac{2\cos\left(x - \dfrac{\pi}{6}\right)}{0{,}5 + \sin^2 y}\left(1 + \dfrac{z^2}{3 - z^2/5}\right).$

2. $u = \dfrac{\sqrt[3]{8 + |x-y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(\operatorname{tg}^2 z + 1\right).$

3. $v = \dfrac{1 + \sin^2(x+y)}{\left|x - \dfrac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(\operatorname{arctg}\dfrac{1}{z}\right).$

4. $w = \left|\cos x - \cos y\right|^{(1+2\sin y)}\left(1 + z + \dfrac{z^2}{2} + \dfrac{z^3}{3} + \dfrac{z^4}{4}\right).$

5. $\alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \dfrac{y}{2}\right) + \sin^2 \operatorname{arctg}(z).$

6. $\beta = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - |x-y|\right).$

7. $\gamma = 5\operatorname{arctg}(x) - \frac{1}{4}\operatorname{arctg}(x)\,\frac{x+3|x-y|+x^2}{|x-y|z+x^2}$.

8. $\varphi = \frac{e^{|x-y|}|x-y|^{x+y}}{\operatorname{arctg}(x)+\operatorname{arctg}(z)} + \sqrt[3]{x^6+\ln^2 y}$.

9. $a = \left|x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right| + (y-x)\frac{\cos y - \frac{z}{(y-x)}}{1+(y-x)^2}$.

10. $a = 2^{-x}\sqrt{x+\sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin(z)}}$.

ЛАБОРАТОРНАЯ РАБОТА 2

Разработка приложения, реализующего разветвляющийся вычислительный процесс

Цель: научиться пользоваться стандартными компонентами организации переключений (TCheckBox, TRadioGroup и др.). Используя компоненты организации переключений разработать интерфейс и программу для заданного разветвляющегося алгоритма.

Краткие сведения

Операторы if *и* case *языка Pascal*

Для программирования разветвляющихся алгоритмов в языке Pascal используются переменные типа ***boolean***, которые могут принимать только два значения **true** и **false** (да, нет), а также операторы ***if*** и ***case***. Оператор ***if*** проверяет результат логического выражения или значение переменной типа ***boolean*** и организует разветвление вычислений. Оператор ***case*** орга-

низует разветвления в зависимости от селектора – значения некоторой переменной, например, n целого типа:

```
case n of
   0: u:= x + y;
   1: u:= x - y;
   2: u:= x * y;
 else u:= 0;
end;
```

В соответствии со значением n вычисляется значение переменной u. При этом, если n = 0, то u = x + y, если n = 1, то u = x – y, если n = 2, то u = x * y и, наконец, u = 0 при любых значениях n, отличных от 0, 1 или 2.

Кнопки-переключатели в Delphi

При создании программ в Delphi для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено/выключено) визуально отражается на форме.

Компонент **TCheckBox** организует кнопку независимого переключателя, с помощью которой пользователь может указать свое решение типа да/нет. В программе состояние кнопки связано со значением булевской переменной, которая проверяется с помощью оператора if.

Компонент **TRadioGroup** организует группу кнопок – зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки отключаются. В программу передается номер включенной кнопки (0, 1, 2, ...), который можно проанализировать с помощью оператора ***case.***

Постановка задачи

Разработать приложение, реализующее разветвляющийся вычислительный процесс в соответствии с индивидуальным заданием.

Задания

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве $f(x)$ использовать по выбору: $\mathrm{sh}(x)$, x^2, e^x. Разработайте вид формы и текст программы, в соответствии с полученным заданием.

1. $$a=\begin{cases}\left(f(x)+y\right)^2-\sqrt{f(x)y}, & xy>0\\ \left(f(x)+y\right)^2-\sqrt{|f(x)y|}, & xy<0\\ \left(f(x)+y\right)^2+1, & xy=0.\end{cases}$$

2. $$b=\begin{cases}\ln\left(f(x)\right)+\left(f(x)^2+y\right)^3, & x/y>0\\ \ln|f(x)/y|+\left(f(x)+y\right)^3, & x/y<0\\ \left(f(x)^2+y\right)^3, & x=0\\ 0, & y=0.\end{cases}$$

3. $$c=\begin{cases}f(x)^2+y^2+\sin y, & x-y=0\\ \left(f(x)-y\right)^2+\cos y, & x-y>0\\ \left(y-f(x)\right)^2+\mathrm{tg}(y), & x-y<0.\end{cases}$$

4. $$d=\begin{cases}\left(f(x)-y\right)^3+\mathrm{arctg}(f(x)), & x>y\\ \left(y-f(x)\right)^3+\mathrm{arctg}(f(x)), & y>x\\ \left(y+f(x)\right)^3+0{,}5, & y=x.\end{cases}$$

5. $e = \begin{cases} i\sqrt{f(x)}, & i - \text{нечетное}, \quad x > 0 \\ i/2\sqrt{|f(x)|}, & i - \text{четное}, \quad x < 0 \\ \sqrt{|if(x)|}, & \text{иначе.} \end{cases}$

6. $g = \begin{cases} e^{f(x)-|b|}, & 0{,}5 < xb < 10 \\ \sqrt{|f(x)+b|}, & 0{,}1 < xb < 0{,}5 \\ 2f(x)^2, & \text{иначе.} \end{cases}$

7. $g = \begin{cases} e^{f(x)}, & 1 < xb < 10 \\ \sqrt{|f(x)+4*b|}, & 12 < xb < 40 \\ bf(x)^2, & \text{иначе.} \end{cases}$

8. $j = \begin{cases} \sin(5f(x)+3m|f(x)|), & -1 < m < x \\ \cos(3f(x)+5m|f(x)|), & x > m \\ (f(x)+m)^2, & \text{иначе.} \end{cases}$

9. $l = \begin{cases} 2f(x)^3+3p^2, & x > p \\ |f(x)-p|, & 3 < x < |p| \\ (f(x)-p)^2, & x = p. \end{cases}$

10. $k = \begin{cases} \ln(|f(x)|+|q|), & |xq| > 10 \\ e^{f(x)+q}, & |xq| < 10 \\ f(x)+q, & |xq| = 10. \end{cases}$

ЛАБОРАТОРНАЯ РАБОТА 3

Разработка приложения, реализующего циклический вычислительный процесс

Цель: изучить средства отладки программ в среде Delphi. Составить и отладить программу выполнения циклического вычислительного процесса.

Краткие сведения

Операторы организации циклов repeat, while, for

Под *циклом* понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме. Для организации повторений в языке Pascal предусмотрены операторы **Repeat**, **While** и **For**.

Оператор **Repeat** имеет форму:

```
Repeat
  <операторы>
until <условие>;
```

и организует повторение операторов, помещенных между ключевыми словами repeat и until, до тех пор, пока не выполнится <условие> = true, после чего управление передается следующему за циклом оператору.

Оператор **While** имеет форму:

```
While<условие >do
begin
  <операторы>
end;
```

и организует повторение операторов, помещенных между begin и end, до тех пор, пока не выполнится <условие> = false. Заметим, что если <условие> = false при первом входе в цикл, то

<операторы> не выполнятся ни разу, в отличие от оператора Repeat, в котором хотя бы один раз они выполнятся.

Оператор **For** имеет форму:

```
For i := i1 to i2 do
begin
  <операторы>
end;
```

и организует повторное вычисление операторов при нарастающем изменении переменной цикла i от начального значения i1 до конечного i2 с шагом, равным единице. Заметим, что если i2 < il, то <операторы> не выполнятся ни разу.

Модификация оператора имеет вид:

```
For i := i2 downto i1 do
begin
  <операторы>
end;
```

и организует повторение вычислений при убывающем изменении i на единицу.

Средства отладки программ в Delphi

В написанной программе после ее запуска, как правило, обнаруживаются ошибки. Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические и/или синтаксические ошибки). При обнаружении ошибки компилятор Delphi останавливается напротив первого оператора, в котором обнаружена ошибка. В нижней части экрана появляется текстовое окно, содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием.

Для получения более полной информации о характере ошибки необходимо обратится к Help нажатием клавиши F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому исправлять ошибки нужно последовательно, сверху вниз, после исправления каждой ошибки необходимо компилировать программу снова.

Ошибки второго уровня – ошибки времени выполнения. Они связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным, либо происходит переполнение (деление на нуль) и др. Поэтому перед использованием отлаженной программы ее необходимо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды Delphi.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить курсор в строке перед проверяемым участком, выделить этот оператор нажатием мыши на полосе слева от текста программы, нажать клавишу F4 (выполнение до курсора). При этом выполнение программы будет остановлено на строке, содержащей курсор. Теперь можно увидеть, чему равны значения интересующих переменных. Для этого нужно поместить на переменную курсор и в качестве подсказки на экране будет высвечено ее значение. В другом варианте требуется нажать комбинацию клавиш Ctrl–F7 и в появившемся диалоговом окне указать интересующую переменную (с помощью данного окна можно также изменить значение переменной во время выполнения программы).

Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если кур-

сор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать мышью на команде <Run> меню Run. Нажимая клавишу F8 можно продолжать отладку не заходя внутрь процедур и функций.

Постановка задачи

Разработать приложение с реализацией циклических вычислений в соответствии с индивидуальным заданием.

Задания

В заданиях (табл. 1) необходимо вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ для x, изменяющихся от x_n до x_k с шагом $h=(x_k-x_n)/n$. Близость значений $S(x)$ и $Y(x)$ во всем диапазоне значений x указывает на правильность вычисления $S(x)$ и $Y(x)$.

Таблица 1

Варианты заданий

№	x_n	x_k	$S(x)$	n	$Y(x)$
1	0,1	1	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	16	$\sin x$
2	0,1	1	$1+\frac{x^2}{2}+..+\frac{x^{2n}}{(2n)!}$	10	$\frac{e^x+e^{-x}}{2}$
3	0,1	1	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	12	$e^{x\cos\frac{\pi}{4}}\cos\left(x\sin\frac{\pi}{4}\right)$

№	x_n	x_k	$S(x)$	n	$Y(x)$
4	0,1	1	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	8	$\cos x$
5	0,1	1	$1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}$	14	$\left(1+2x^2\right)e^{x^2}$
6	0,1	1	$x+\frac{x^3}{3!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$	8	$\frac{e^x-e^{-x}}{2}$
7	0,1	1	$\frac{x^3}{3}-\frac{x^5}{15}+\ldots+(-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$	12	$\frac{1+x^2}{2}\operatorname{arctg}x-\frac{x}{2}$
8	0,1	1	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	10	e^{2x}
9	0,1	1	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	14	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0,1	0,5	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	15	$\operatorname{arctg}(x)$

ЛАБОРАТОРНАЯ РАБОТА 4

Разработка приложения с использованием массивов

Цель: изучить свойства компонента TStringGrid. Написать программу с использованием массивов.

Краткие сведения

Работа с массивами

Массив – упорядоченный набор однотипных элементов, объединенных под одним именем. Каждый элемент массива

обозначается именем, за которым в квадратных скобках следует один или несколько индексов, разделенных запятыми, например: a[1], bb[I], c12[I, j*2], q[l, 1, I*j-l]. В качестве индекса можно использовать любые порядковые типы за исключением LongInt.

Тип массива или сам массив определяются соответственно в разделе типов (Type) или переменных (Var) с помощью ключевого слова **Array** следующим образом:

Array [описание индексов] **of** <тип элементов массива >;

Примеры описания массивов:

```
Const N=20;  // Задание максимального значения индекса;
Type
  TVector = array [1..N] of real;   // Описание типа одномерного массива;
Var
  A : TVector;                      //A – массив типа TVector;
  Ss : array[1..10] of integer;  //Ss – массив из десяти целых чисел;
  Y : array[1..5, 1..10] of char;  //Y – двумерный массив символьного типа.
```

Элементы массивов могут использоваться в выражениях так же, как и обычные переменные, например:

```
F:= 2*a[3] + a[ss[l] + 1]*3;
A[n]:= 1+sqrt(abs(a[n-1]));
```

Компонент TStringGrid

При работе с массивами ввод и вывод информации на экран удобно организовывать в виде таблиц, используя компонент TStringGrid. Последний предназначен для отображения информации в виде двумерной таблицы, каждая ячейка которой представляет собой окно однострочного редактора (аналогично окну TEdit). Доступ к информации осуществляется с помощью свойства Cells[ACol, ARow : integer] : string; где ACol, ARow – индексы элементов двумерного массива. Свойства

ColCount и RowCount устанавливают количество строк и столбцов в таблице, а свойства FixedCols и FixedRows задают количество строк и столбцов фиксированной зоны. Фиксированная зона выделена другим цветом, и в нее запрещен ввод информации с клавиатуры.

Для установки компонента TStringGrid на форму необходимо на странице Additional палитры компонентов щелкнуть мышью по пиктограмме  . После этого щелкните мышью в нужном месте формы. Захватывая кромки компонента, отрегулируйте его размер. С помощью свойств ColCount и RowCount можно задать соответственно число столбцов и строк таблицы. Чтобы разрешить ввод информации в поле таблицы необходимо свойству Options goEditing присвоить значение True.

Постановка задачи

Разработать приложение обработки и представления информации в табличной форме в сооветствии с индивидуальным заданием.

Задания

Во всех заданиях скалярные переменные вводить с помощью компонента типа TEdit с соответствующим пояснением в виде компонента типа TLabel. Скалярный результат выводить в виде компонента TLabel. Массивы представлять на форме в виде компонентов TStringGrid, в которых 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять после нажатия кнопки типа TButton.

1. Задана матрица размером $N \times M$. Получить массив B, присвоив его k-му элементу значение 0, если все элементы k-го столбца матрицы нулевые, и значение 1 в противном случае.

2. Задана матрица размером $N \times M$. Получить массив B, присвоив его k-му элементу значение 1, если элементы k-й строки матрицы упорядочены по убыванию, и значение 0 в противном случае.

3. Задана матрица размером $N \times M$. Получить массив B, присвоив его k-му элементу значение 1, если k-я строка матрицы симметрична, и значение 0 в противном случае.

4. Задана матрица размером $N \times M$. Определить k – количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером $N \times M$. Определить k – количество «особых» элементов матрицы, считая элемент «особым», если в его строке слева от него находятся элементы меньшие его, а справа – большие.

6. Задана символьная матрица размером $N \times M$. Определить k – количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их первых элементов.

8. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию суммы их элементов.

9. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их наибольших элементов.

10. Определить, является ли заданная квадратная матрица n-го порядка симметричной относительно побочной диагонали.

ЛАБОРАТОРНАЯ РАБОТА 5

Разработка приложения обработки строковой информации

Цель: изучить методы программирования с использованием строк и правила работы с компонентами TListBox и TComboBox. Написать программу работы со строками.

Типы данных для работы со строками

Короткие строки типа ShortString и String[N]

Короткие строки имеют фиксированное количество символов. Строка ShortString может содержать 255 символов. Строка String[N] может содержать N символов, но не более 255. Первый байт этих переменных содержит длину строки.

Длинная строка типа String

При работе с этим типом данных память выделяется по мере необходимости (динамически) и может занимать всю доступную программе память. Вначале компилятор выделяет для переменной 4 байта, в которых размещается номер ячейки памяти, начиная с которой будет располагаться символьная строка. На этапе выполнения программа определяет необходимую длину цепочки символов и обращается к ядру операционной системы с требованием выделить необходимую память.

Широкая строка типа WideString

Введена для обеспечения совместимости с компонентами, основанными на OLE-технологии. От типа String отличается только тем, что для представления каждого символа используется не один, а два байта.

Нуль-терминальная строка типа PChar

Представляет собой цепочку символов, ограниченную символом #0. Максимальная длина строки ограничена только доступной программе памятью. Нуль-терминальные строки широко используются при обращениях к *API*-функциям Windows *(API – Application Program Interface* – интерфейс прикладных программ).

Представление строки в виде массива символов

Строка может быть описана как массив символов. Если массив имеет нулевую границу, он совместим с типом PChar.

```
Var
  MasS : array[1..100] of Char;
```

В отличие от нуль-терминальной строки здесь длина имеет фиксированное значение и не может меняться в процессе выполнения программы.

Компонент TListBox

Компонент TListBox представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством Items, методы Add, Delete и Insert используются для добавления, удаления и вставки строк. Для определения номера выделенного элемента используется свойство ItemIndex.

Компонент TComboBox

Комбинированный список TComboBox представляет собой комбинацию списка TListBox и редактора TEdit, поэтому практически все свойства заимствованы у этих компонентов. Для работы с окном редактирования используется свойство Text как в TEdit, а для работы со списком выбора – свойство Items как в TListBox. Существует пять модификаций компонента, определяемых его свойством Style. В модификации csSimple список всегда раскрыт, в остальных он раскрывается после нажатия кнопки справа от редактора.

Компонент TBitBtn

Компонент TBitBtn расположен на странице ***Additional*** палитры компонентов и представляет собой разновидность стандартной кнопки TButton. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством Glyph. Кроме того, имеется свойство Kind, которое задает одну из стандартных разновидностей

кнопок. Нажатие любой из них, кроме bkCustom и bkHelp закрывает модальное окно и возвращает в программу результат mr*** (например, bkOk – mrOk). Кнопка bkClose закрывает главное окно и завершает работу программы.

Обработка событий

Обо всех происходящих в системе событиях таких, как создание формы, нажатие кнопки мыши или клавиатуры и т.д., ядро Windows информирует окна путем посылки соответствующих сообщений. Среда Delphi позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит обработчики сообщений на странице Events инспектора объектов.

Для создания обработчика события необходимо раскрыть список компонентов в верхней части окна инспектора объектов и выбрать необходимый компонент. Затем на странице Events нажатием левой клавиши мыши выбрать обработчик и дважды щелкнуть по его левой (белой) части. В ответ Delphi активизирует окно текста программы и покажет заготовку процедуры обработки выбранного события.

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присущи большинству компонентов. Наиболее часто применяемые события представлены в табл. 2.

Таблица 2

Наиболее часто генерируемые события

Событие	Описание события
OnActivate	Возникает при активации формы
OnCreate	Возникает при создании формы (компонент TForm). В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, например, установка начальных значений

Событие	Описание события
OnKeyPress	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события возвращает через параметр Key:Char ASCII-код нажатой клавиши
OnKeyDown	Обработчик этого события получает информацию о состоянии клавиши Shift, Alt и Ctrl и возвращает через параметр Key:Word номер нажатой клавиши
OnKeyUp	Является парным событием для OnKeyDown и возникает при отпускании ранее нажатой клавиши
OnClick	Возникает при нажатии кнопки мыши в области компонента
OnDblClick	Возникает при двойном нажатии кнопки мыши в области компонента

Постановка задачи

Разработать приложение с использованием компонентов, управляющих представлением строковой информации в соответствии с индивидуальным заданием.

Задания

1. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Найти количество групп с пятью символами.
2. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран самую короткую группу.
3. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.
4. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группы с четным количеством символов.

5. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество единиц в группах с нечетным количеством символов.

6. Дана строка, состоящая из букв, цифр, запятых, точек, знаков «+», «–». Выделить подстроку, которая соответствует записи целого числа (т.е. начинается со знака «+» или «–» и внутри подстроки нет букв, запятых и точек).

7. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков «+» и «–». Выделить подстроку, которая соответствует записи вещественного числа с фиксированной точкой.

8. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков «+» и «–». Выделить подстроку, которая соответствует записи вещественного числа с плавающей точкой.

9. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.

10. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести четные числа этой строки.

ЛАБОРАТОРНАЯ РАБОТА 6

Разработка приложения с использованием записей и файлов

Цель: изучить правила работы с компонентами TOpenDialog и TSaveDialog. Написать программу с использованием файлов и структурированных данных.

Краткие сведения

Программирование с использованием переменных типа запись

Запись – это структура данных, состоящая из полей. Поля – это элементы одного или разных типов. Записи удобны для

создания структурированных баз данных с разнотипными элементами, например:

```
Type
 TStudent = record  //Объявление типа запись
          Fio    : string[20];              //Поле ФИО
          Group : integer;                  //Поле номера студ. группы
          Ocn    : array[1..3] of integer; //Поле массива оценок
  end;
Var
    Student: TStudent;        //Объявление переменной типа запись
```

Доступ к каждому полю осуществляется указанием имени записи и поля, разделенных точкой, например:

```
Student.Fio   := 'Иванов А.И.'; //Внесение данных в поля записи
Student.Group := 107218;
```

Доступ к полям можно осуществлять также при помощи оператора With:

```
With Student do
begin
  Fio    := 'Иванов А.И.';
  Group := 107218;
end;
```

Работа с файлами

Файл – это именованная область данных на внешнем физическом носителе. В Object Pascal различают три вида файлов в зависимости от способа их организации и доступа к элементам: *текстовые, типизированные и нетипизированные.*

Текстовой файл – это файл, состоящий из строк. Примером текстового файла может служить файл исходного текста программы в Delphi (расширение *.pas). Для работы с текстовым файлом должна быть описана соответствующая файловая переменная:

```
var F : TextFile;
```

Типизированный файл – файл, имеющий строго заданную структуру, когда все элементы имеют фиксированный и одинаковый размер. Это свойство типизированных файлов позволяет получить доступ к любому компоненту файла по его порядковому номеру. Элементами такого файла являются, как правило, записи. В описании файловой переменной указывается ее тип:

```
Var F: TStudent;
```

Нетипизированный файл – это файл, в котором данные не имеют определенного типа и рассматриваются, как последовательность байт. Файловая переменная объявляется: Var F: File;

Процедуры работы с файлами

AssignFile(var F; FileName: string) – связывает файловую переменную F и файл с именем FileName.

Reset(var F[: File; RecSize: word]) – открывает существующий файл. При открытии нетипизированного файла RecSize задает размер элемента файла.

Rewrite(var F[: File; RecSize: word]) – создает и открывает новый файл.

Append(var F: TextFile) – открывает текстовый файл для дописывания текста в конец файла.

Read(F, v1[, v2...vn]) – чтение значений переменных, начиная с текущей позиции для типизированных файлов и со строк – для текстовых.

Write(F, v1[, v2,...vn]) – запись значений переменных, начиная с текущей позиции для типизированных файлов и со строк – для текстовых.

CloseFile(F) – закрывает ранее открытый файл.

Rename(var F; NewName: string) – переименовывает неоткрытый файл любого типа.

Erase(var F) – удаляет неоткрытый файл любого типа.

Seek(var F; NumRec: Longint) – для нетекстового файла устанавливает указатель на элемент с номером NumRec.

SetTextBuf(var F: TextFile; var Buf[;Size: word]) – для текстового файла устанавливает новый буфер ввода-вывода объема Size.

Flush(var F: TextFile) – немедленная запись в файл содержимого буфера ввода-вывода.

Truncate(var F) – урезает файл, начиная с текущей позиции.

LoResult: integer – код результата последней операции ввода-вывода.

FilePos(var F): longint – для нетекстовых файлов возвращает номер текущей позиции. Отсчет ведется от нуля.

FileSize(var F): longint – для нетекстовых файлов возвращает количество компонентов в файле.

Eoln(var F: TextFile): boolean – возвращает True, если достигнут конец строки.

Eof(var F)): boolean – возвращает True, если достигнут конец файла.

SeekEoln(var F: TextFile): boolean – возвращает True, если пройден последний значимый символ в строке или файле, отличный от пробела или знака табуляции.

SeekEoln(var F: TextFile): boolean – то же, что и SeekEoln, но для всего файла.

BlockRead(var F: File; var Buf; Count: word[; Result: word])

BlockWrite(var F : File; var Buf; Count: word[; Result: word]) – соответственно процедуры чтения и записи переменной Buf с количеством Count блоков.

Порядок работы с файлами

```
AssignFile(F, 'FileName.txt'); //Связывание файловой переменной F
                               //с именем дискового файла «FileName.txt»
Rewrite(F);                    //Создание нового файла
Reset(F);                      //Открытие уже существующего файла
```

```
Read(F, Student);          //Чтение данных из файла
Write(F, Student);         //Запись данных в файл
CloseFile(F);              //Закрытие файла
```

Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя файла и путь к нему. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержится в свойстве FileName. Для фильтрации файлов, отображаемых в окне просмотра, используется свойство Filter, а для задания расширения файла, в случае, если оно не задано пользователем – свойство DefaultExt. Если необходимо изменить заголовок диалогового окна – используется свойство Title.

Постановка задачи

Разработать приложение, поддерживающее чтение и сохранение информации в файлах в соответствии с индивидуальным заданием.

Задания

В приложении предусмотреть сохранение вводимых данных в файле и возможность чтения из ранее сохраненного файла. Результаты выводить в окно просмотра и в текстовой файл.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес

покупателя и дату постановки на учет. Удалить из списка все повторные записи, проверяя Ф.И.О. и домашний адрес.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 1 000 000 руб.

3. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.

4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести по каждому городу общее время разговоров с ним и сумму.

6. Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовый тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12 % от суммы заработка.

7. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самой молодой, рослой и легкой команде.

8. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

9. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.

10. Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

ЛАБОРАТОРНАЯ РАБОТА 7

Разработка приложения с использованием подпрограмм и модулей

Цель: изучить возможности Delphi для написания подпрограмм и создания модулей. Составить и отладить программу, включающую внешний модуль Unit с подпрограммой.

Краткие сведения

Подпрограмма – это именованная, определенным образом оформленная группа операторов, которая может быть вызвана любое количество раз из любой точки основной программы. Подпрограммы используется в том случае, когда одна и та же, последовательность операторов в тексте программы повторяется несколько раз. Эта последовательность заменяется вызовом подпрограммы, содержащей необходимые операторы. Подпрограммы также применяются для создания специализиро-

ванных библиотечных модулей, содержащих набор подпрограмм определенного назначения, для использования их другими программистами.

Подпрограммы подразделяются на процедуры и функции. Процедура имеет следующую структуру:

```
Procedure <имя процедуры> ([список формальных параметров]);
Const [описание используемых констант];
Type  [описание используемых типов];
Var   [описание используемых переменных];
begin
  <операторы>
end;
```

Процедуры и функции могут быть использованы в качестве формальных параметров подпрограмм. Для этого определяется тип:

Type <имя>= function ([список формальных параметров]):<тип результата>;

или

Type <имя>= procedure ([список формальных параметров]);

Имя процедуры или функции должно быть уникальным в пределах программы. Список формальных параметров необязателен и может отсутствовать. Если же он есть, то в нем перечисляются через точку с запятой имена формальных параметров и их типы. Имеется три вида формальных параметров параметры-значения, параметры-переменные, параметры-константы. При вызове подпрограммы передача данных для этих видов осуществляются по-разному. Параметры-значения копируются, и подпрограмма работает с их копией, поэтому при вызове на месте такого параметра можно ставить арифметическое выражение. При использовании параметров-переменных (в описании перед ними ставится Var) и параметров-констант

в подпрограмму передается адрес, и она работает с самой переменной. С помощью параметров-переменных подпрограмма передает результаты своей работы вызывающей программе.

В функциях используется специальная переменная Result, интерпретируемая как значение, которое вернется в основную программу по окончании работы функции.

В язык Object Pascal встроен ряд наиболее часто употребляемых процедур и функций, которые являются частью языка и вызываются без предварительного определения в разделе описаний.

Использование модулей

Модуль – автономно компилируемая программная единица, включающая в себя процедуры, функции, а также различные компоненты раздела описаний. Структура модуля содержит следующие основные части: заголовок, интерфейсная часть, исполняемая, инициирующая и завершающая.

Заголовок состоит из зарезервированного слова Unit и следующего за ним имени модуля, которое должно совпадать с именем дискового файла. Использование имени модуля в разделе Uses основной программы приводит к установлению связи модуля с основной программой.

Интерфейсная часть расположена между зарезервированными словами Interface и Implementation и содержит объявление тех объектов модуля, которые должны быть доступны другим программам.

Исполняемая часть начинается зарезервированным словом implementation и содержит описание процедур и функций, объявленных в интерфейсной части. Она может также содержать вспомогательные типы, константы, переменные, процедуры и функции, которые будут использоваться только в исполняемой части и не будут доступны внешним программам.

Инициирующая часть начинается зарезервированным словом Initialization и содержит операторы, которые исполняются до передачи управления основной программе.

Завершающая часть начинается зарезервированным словом Finalization и выполняется в момент окончания работы программы. Инициализирующая и завершающая части модуля используются крайне редко.

Создавая модуль, следует обратить внимание на то, что он не должен иметь своей формы. Система Delphi при начальной загрузке автоматически создает шаблон программы, имеющий в своем составе форму, файл проекта и т.д. Так как модуль состоит только из одного файла, то необходимо перед его созданием уничтожить заготовку файла проекта и форму. Для этого в меню File выбрать Close All, файл проекта не сохранять.

Для создания модуля в меню File выбрать File New, и затем в репозитории – пиктограмму 

. В результате будет создан файл с заголовком Unit Unit1. Имя модуля можно сменить на другое, отвечающее внутреннему содержанию модуля, например, Unit MatFunc. Затем необходимо сохранить файл с именем, совпадающим с именем заголовка модуля: MatFunc.pas. Следует обратить внимание на то, что имя файла должно совпадать с именем модуля, иначе Delphi не сможет подключить его к другой программе.

Для того чтобы подключить модуль к проекту, необходимо в меню Project выбрать опцию Add to Project... и указать файл, содержащий модуль. После этого в разделе Uses добавить имя подключаемого модуля – MatFunc. Теперь в проекте можно использовать функции, содержащиеся в модуле.

Постановка задачи

Разработать приложение, состоящее из нескольких Unit в соответствии с индивидуальным заданием.

Задания

По указанию преподавателя выберите вариант задачи из заданий, приведенных в работе 3. Предусмотрите возможность выбора функции, для которой будет рассчитываться таблица. Функции поместите в отдельный модуль. Вызывать выбранную функцию должна процедура, использующая в качестве входного параметра имя соответствующей функции.

ЛАБОРАТОРНАЯ РАБОТА 8

Разработка приложения с выдачей результатов вычислений в виде графиков

Цель: изучить возможности построения графиков с помощью компонента отображения графической информации TChart. Написать и отладить программу построения на экране графика заданной функции.

Краткие сведения

Построение графика с помощью компонента TChart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Система Delphi имеет мощный пакет стандартных программ вывода на экран и редактирования графической информации, который реализуется с помощью визуально отображаемого на форме компонента TChart.

Компонент TChart осуществляет всю работу по отображению графиков, переданных в объект Series: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает результаты вычислений в виде всевозможных графиков или диаграмм. Построение графика (диаграммы) производится с помощью метода Add объекта

Series. При необходимости, с помощью встроенного редактора EditingChart в компоненту TChart передаются данные о толщине, стиле и цвете линий, параметрах шрифта подписей, шагах разметки координатной сетки и другие настройки. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам компонента TChart.

Компонент TChart вводится в форму путем нажатия пиктограммы  , расположенной на закладке Additional палитры компонентов.

Для изменения параметров компонента TChart необходимо дважды щелкнуть по нему мышью в окне формы. Появится окно редактирования EditingChart (рис. 2). Для создания нового объекта Series1 нужно щелкнуть по кнопке Add на закладке Series.

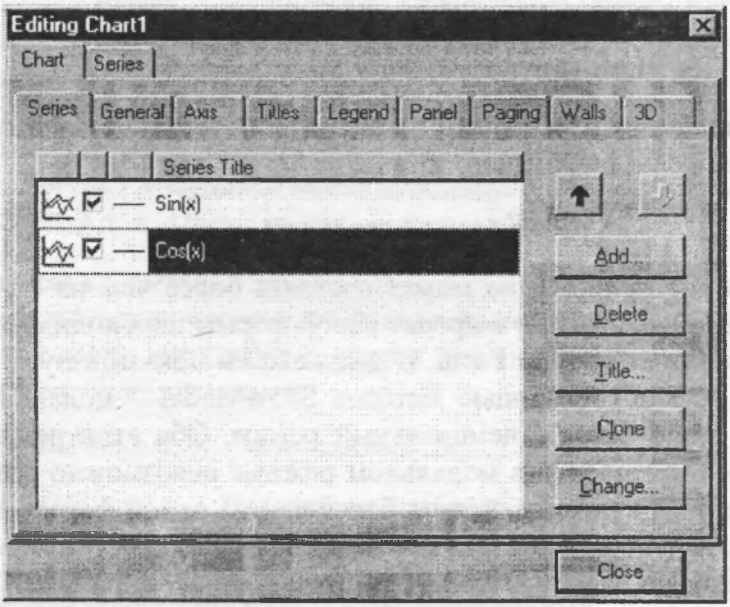


Рис. 2. Задание параметров компонента TChart

Для изменения надписи над графиком используется закладка Titles, а для разметки осей – закладка Axis.

Постановка задачи

Разработать приложение отображения результатов вычислений в виде графика.

Задания

Построить графики функций для соответствующих вариантов из работы 1. Таблицу данных получить путем изменения параметра X с шагом h. Ввод исходных данных организовать через окна TEdit. Самостоятельно выбрать удобные параметры настройки.

ЛАБОРАТОРНАЯ РАБОТА 9

Разработка приложения, состоящего из нескольких форм

***Цель*:** изучить основные свойства и методы, связанные с созданием и активизацией форм.

Краткие сведения

Сложное приложение может состоять более чем из одной форы. Для добавления в проект новой формы необходимо выполнить команду New Form. В дальнейшем показать нужную форму можно с помощью методов ShowModal – модальный режим, либо Show – немодальный режим. Оба этих режима отличаются тем, что в модальном режиме невозможно одновременно выполнять действия более чем на одной форме. Напротив, в немодальном режиме имеется возможность свободно переходить с формы на форму без закрытия остальных.

Задание и представление информации может быть разнесено по отдельным формам.

Для реализации визуальных способов изменения данных в программе используются различные компоненты. Одним из часто используемых компонентов, является компонент TScrollBar.

Компонент TScrollBar

Это управляющий элемент, расположенный на стандартной закладке палитры компонент с пиктограммой  , похожий на полосу скроллинга окна и используемый для изменения числовой величины визуальным изменением положения бегунка компонента.

Свойствами этого компонента являются:

- Kind = (sbHorisontal, sbVertical) – для задания расположения TScrollBar;
- Min, Max : Integer – для установки минимального и максимального значений изменяемой величины;
- Position : Integer – в этом свойстве находится текущее значение числа;
- LargeChange : TScrollBarInc, SmallChange :TScrollBarInc – с помощью этих свойств соответственно можно устанавливать малый и большой сдвиг бегунка.

При изменении положения бегунка возникает событие OnScroll. Чтобы программным образом устанавливать положение бегунка в заданное место используется метод SetParams (Aposition, Amin, Amax : integer).

Смешивание цветов

Для получения составного цвета можно использовать смешивание трех составляющих цветов при вызове функции RGB (red, green, blue), например,

```
Color:= RGB(255, 0, 0); //ярко-красный цвет.
```

Постановка задачи

Разработать приложение, состоящее из нескольких форм и поддерживающее визуальное изменение данных.

Задание

Взять за основу задание к лабораторной работе 7. Добавить к проекту дополнительную форму, на которой реализовать

смешивание цветов. При этом должен быть организован диалог приложения, как показано на рис. 3. Выбранный цвет необходимо применить к одному из компонентов главной формы.

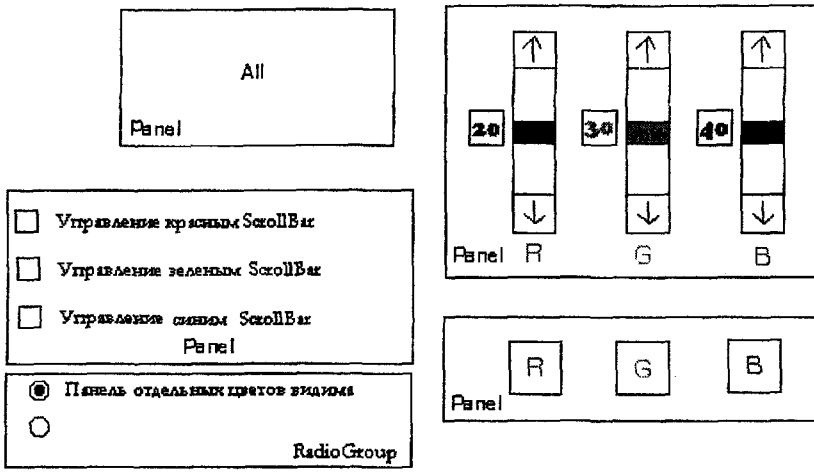


Рис. 3. Форма приложения

ЛАБОРАТОРНАЯ РАБОТА 10

Разработка приложения с сохранением параметров и установок в Ini-файлах

Цель: изучить возможности автоматического сохранения параметров и установок, принятых в программе.

Краткие сведения

Удобным средством запоминания текущих настроек приложения являются ini-файлы. *Ini-файлы* – это текстовые файлы, предназначенные для хранения информации о формах или о настройках различных приложений. Информация в файле логически группируется в разделы, каждый из которых начинается оператором заголовка, заключенным в квадратные скобки,

например, [Desktop]. В строках, следующих за заголовком, содержится информация, относящаяся к данному разделу, в виде:

<ключ>=<значение>.

Любое приложение можно зарегистрировать в системном реестре и зафиксировать там же текущие настройки приложения (в 32-разрядных Windows и выше). Для 32-разрядных приложений Microsoft не рекомендует работать с Ini-файлами. Несмотря на это и 32-разрядные приложения, наряду с реестром, часто используют эти файлы. Да и разработки Microsoft не обходятся без этих файлов.

Ini-файлы, как правило, хранятся в каталоге Windows, который можно найти с помощью функции GetWindowsDirectory.

В Delphi работу с Ini-файлами проще всего осуществлять с помощью создания в приложении объекта типа TIniFile. Этот тип описан в модуле IniFiles, который надо подключать к приложению оператором uses (автоматически это не делается).

Создается объект типа TIniFile методом Create (<имя файла>), в который передается имя Ini-файла, с которым он связывается.

Для записи значений ключей существует несколько методов: WriteString, WriteInteger, WriteFloat, WriteBool и др. Каждый из них записывает значение соответствующего типа. Объявления всех этих методов очень похожи. Например:

```
procedure WriteString (const Section, Ident, Value: string);
procedure WriteInteger(const Section, Ident: string; Value: Longint);
```

где Section – раздел Ini-файла, Ident – имя ключа, Value – значение ключа. Если соответствующий раздел или ключ отсутствует в файле, он автоматически создается.

Имеются аналогичные методы чтения значений ключей: ReadString, ReadInteger, ReadFloat, ReadBool и др. Например:

```
function ReadString(const Section, Ident, Default: string) : string;
function ReadInteger(const Section, Ident: string; Default: Longint) : Longint;
```

В этих примерах методы чтения возвращают значение ключа Ident и раздела Section. Параметр Default определяет значение, возвращаемое в случае, если в файле не указано значение соответствующего ключа.

Проверить наличие значения ключа можно методом ValueExists, в который передаются имена раздела и ключа. Метод DeleteKey удаляет из файла значение указанного ключа в указанном разделе.

Проверить наличие в файле необходимого раздела можно методом SectionExists. Метод EraseSection удаляет из файла указанный раздел вместе со всеми его ключами. Имеется еще ряд методов, которые можно посмотреть во встроенной справке Delphi.

Постановка задачи

Разработать приложение, поддерживающее чтение и сохранение настроек в Ini-файлах.

Задание

Реализовать чтение и сохранение параметров при запуске и завершении программы. В качестве задания использовать лабораторную работу 5.

ЛАБОРАТОРНАЯ РАБОТА 11

Разработка приложения, поддерживающего создание графических изображений

Цель: изучить основные графические компоненты, их свойства и методы.

Краткие сведения

Любая Windows-программа осуществляет вывод информации на экран с помощью GDI (*Graphic Device Interface*). Функ-

ции, реализованные в GDI, являются *аппаратно независимыми*. Эти функции взаимодействуют с конкретным устройством не напрямую, а через специальную программу, которая называется *драйвером устройства*. Для любых устройств (мониторов, принтеров, плоттеров и т.д.) используется соответствующий драйвер.

Функции GDI взаимодействуют с драйвером устройства через специальную структуру, называемую *контекстом устройства (Device Context)*. В качестве контекста в Delphi выступает объект Canvas.

В Delphi имеется несколько *независимых классов*, которые определяют средства создания изображений. К ним можно отнести TCanvas – холст, TPen – перо, TBrush – кисть, TFont – шрифт. Данные классы Delphi иногда называют *классами-надстройками*, т.к. связанные с ними объекты самостоятельно в программе не используются, а выступают как свойства того или иного элемента управления (Form, Edit, ...). Рассмотрим основные свойства этих классов.

Класс TPen

С помощью этого класса производится рисование линий и контуров различных геометрических фигур. Перо характеризуется цветом, стилем и толщиной.

Основные свойства класса:

Color: TColor – для задания конкретного цвета. Цвет в Windows задается в формате RGB, т.е. тройкой чисел, определяющих степени интенсивности трех его цветовых составляющих – красной, зеленой и синей. Для задания конкретного цвета используется тип TColor, описанный в Unit Graphics как:

```
Type TColor = -$7FFFFFFF..$7FFFFFFF,
```

т.е. для задания конкретного цвета выделяется целое число в 4 байта. Самый крайний байт определяет интенсивность крас-

ной составляющей. В шеснадцатиричной системе счисления соответствующие составляющие изменяются в диапазонах:

$00 00 00 00 – $00 00 00 FF – красная составляющая,
$00 00 00 00 – $00 00 FF 00 – зеленая составляющая,
$00 00 00 00 – $00 FF 00 00 – синяя составляющая.

Левый байт задает палитру.

Для наиболее часто используемых цветов определены соответствующие константы. Они разбиваются на 2 группы:

1. Цвета, безотносительно к какому элементу они применяются, например: clBlack .. clWhite, clNone.

2. Цвета, предназначенные для окрашивания каких-либо деталей изображения: полос скроллинга, фона рабочего окна Windows, фона меню и т.д. Это такие цвета как: clWindows, clMenu и т.д.

Цвета второй группы могут меняться в зависимости от настроек Windows.

Замечание. Получить составной цвет, можно также смешав три составляющие при вызове функции RGB:

```
Color:= RGB(255, 0, 0); //ярко-красный цвет.
```

Если требуется выделить из смешанного цвета одну из его составляющих, то это можно сделать функциями GetRValue, GetGValue, GetBValue, например: RedValue:= GetRValue(Color).

Style : TPenStyle – задает тип линии путем использования констант:

psSolid, ————————
psDash, - - - - - - - -
psDot, ...
psDashDot, —·—·—·—
psDashDotDot, ··—··—··—
psClean;

Width: Integer – задает толщину линий.

Класс TBrush

С помощью этого класса задаются характеристики кисти.

Основные свойства, определенные в классе:

Color: TColor – задает цвет кисти. По умолчанию clWhite.

Style: TBrushStyle – определяет стиль кисти. Для задания стиля используются константы:

bsSolid 

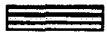
bsClear 

bsBDiagonal 

bsFDiagonal 

bsCross 

bsDiagCross 

bsHorizontal 

bsVertical 

Класс TFont

С помощью этого класса задаются характеристики текста с помощью свойств:

Color: TColor – задает цвет шрифта. По умолчанию clBlack.

Name: TFontName – задает название шрифта, например: 'Arial'.

Size: Integer – задает размер букв.

Style: TFontStyle – задает стиль букв. Для задания стиля используются константы: [fsBold], [fsItalic] [fsUnderline], [fsStrikeOut].

Способы отображения графики

Delphi предоставляет программисту 4 способа отображения графики:

- использование заранее созданных графических изображений;
- создание изображений с помощью графических компонентов;

• создание изображений с помощью примитивов (линия, круг и т.д.) непосредственно во время работы программы;
• представление информации в виде графиков.

1-й способ. Компонент TImage

Если графическое изображение уже создано, например, с помощью графического редактора (например, Paint), то его можно показать с помощью компонента TImage. В Delphi с помощью этого компонента можно отобразить следующие графические изображения:

1) растровое (*.bmp),
2) пиктограммы (*.ico),
3) типа метафайла (*.wmf),
4) курсора (*.cur).

Вместе с тем известны и другие способы хранения изображений (*.pcx, *.gif, *.tiff, *.jpeg, *.dwg). Для того, чтобы включить изображения других форматов их нужно перевести в формат *.bmp.

Основные свойства компонента TImage:

Canvas – содержит канву для прорисовки изображения;

Center – указывает, надо ли центрировать изображение в границах компонента. Игнорируется, если: AutoSize := True; или Stretch := True; и изображение не является пиктограммой (.ico);

Increment – разрешает/запрещает показ большого изображения по мере его загрузки;

Picture – центральное свойство класса. Служит контейнером изображения TPicture;

Proportional – разрешает/запрещает пропорционально уменьшать высоту и ширину изображения, если оно не может целиком уместиться в рабочей зоне компонента;

Stretch – разрешает/запрещает изменять размер изображения так, чтобы оно целиком заполнило клиентскую область компонента;

Transparent – запрещает/разрешает накладывать собственный фон изображения на фон компонента.

Компонент TImage позволяет поместить графическое изображение в любое место на форме. Собственно картинку можно загрузить во время дизайна в редакторе свойств Picture. При проектировании следует помнить, что изображение, помещенное на форму во время дизайна, включается в файл .DPR и затем прикомпилируется к EXE-файлу. Поэтому такой EXE-файл может получиться достаточно большой. Исходная картинка должна храниться в файле в формате BMP (*bitmap*), WMF (*Windows Meta File*) или ICO (*icon*). Как альтернативу можно рассмотреть загрузку картинки во время выполнения программы, для этого у свойства Picture (которое является объектом со своим набором свойств и методов) есть специальный метод LoadFromFile.

Пример. По нажатию кнопки необходимо загрузить в компонент TImage изображение.

Обработчик нажатия кнопки Button1Click выглядит следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if OpenPictureDialog1.Execute then
 begin
  Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
  Image1.Stretch := True;
 end;
end;
```

Если изображение, находящееся в TImage, нужно сохранить в файле, можно применить метод SaveToFile, который также принадлежит свойству Picture.

2-й способ. Компоненты TShape, TBevel

С помощью этого способа имеется возможность рисовать простейшие геометрические фигуры (прямоугольник, квадрат, скругленный прямоугольник, скругленный квадрат, эллипс, окружность). Фигура полностью занимает пространство ком-

понента. Если задан квадрат или круг, а размеры элемента по горизонтали и вертикали отличаются, фигура чертится с размером меньшего измерения. Для создания таких фигур используется компонент **TShape**, расположенный на закладке Additional под пиктограммой  .

Могут быть использованы следующие свойства компонента:

Shape : TShapeType = (stRectangle, stSquare, stRoundRect, stRoundSquare, stEllipse, stCircle) – тип геометрической фигуры, где
stRectangle – прямоугольник,
stSquare – квадрат,
stRoundRect – скругленный прямоугольник,
stRoundSquare – скругленный квадрат,
stEllipse – эллипс,
stCircle – окружность.

Выбранная фигура рисуется на весь экран компонента TShape. Изменение свойства Shape приводит к немедленной перерисовке изображения.
Brush : TBrush – используется для заливки области;
Pen : TPen – используются для изменения параметров рамки.

Пример:

```
Procedure TForm1.FormCreate( );
begin
 with Shape1 do
 begin
   Shape := stRectangle;   //Фигура – прямоугольник
   Brush.Color := clRed;   //Красный цвет заливки
   Pen.Color := Blue;      //Синий цвет рамки
   Brush.Style := bsHorizontal; //Дискретная заливка в виде горизонтальных линий
   Pen.Style := psSolid;   //Сплошной тип линии рамки
   Pen.Width := 2;         //Толщина линии рамки
 end;
end;
```

Bitmap : TBitmap – позволяет в качестве закраски или заливки использовать растровое изображение, например: Shape1.Brush.Bitmap := Image1.Picture.Bitmap;.

Компонент **TBevel**  используется для выделения группы элементов или отделения их друг от друга. Компонент TBevel служит для украшения программ и может принимать вид рамки или линии. Объект предоставляет меньше возможностей по сравнению с TPanel, но не занимает ресурсов.

Изменения внешнего вида компонента осуществляется с помощью свойств:

Shape : TBevelShape = (bsBox, bsFrame, bsTopLine, bsBottomLine, bsLeftLine, bsRightLine) – геометрия компонента;

Style :TBevelStyle = (bsLowered, bsRaised) – вид (вдавленный, выпуклый) компонента.

3-й способ. Поддержка графических операций низкого уровня

Для создания графических изображений в области некоторых компонентов (TForm, TImage, TPaintBox, TPrinter, TListBox, TComboBox, TDrawGrid) используется свойство Canvas. С каждым из перечисленных компонентов связано событие OnPaint. Это событие возникает, когда ядру Windows необходимо перерисовать содержимое компонента (например, при активизации формы, когда один из перечисленных компонентов становится видимым). Чтобы отрисовать графическое изображение внутри рабочей области перечисленных компонентов нужно обработать событие OnPaint, т.е. записать соответствующий обработчик.

Можно воспроизвести на соответствующих компонентах любые графические объекты без использования компонентов TImage, TShape, TLabel.

Класс TCanvas

Класс TCanvas имеет свойства:

Pen: TPen – устанавливает *цвет, толщину, стиль* линий и границ геометрических фигур, например:

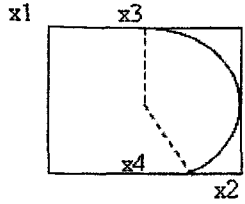
```
with Canvas do
begin
  Pen.Color := clBlue;
  Pen.Width := 2;
  Pen.Style := psDash;
end;
```

Brush: TBrush – позволяет устанавливать цвет и шаблон кисти;
Font: TFont – позволяет устанавливать параметры текста;
PenPos: TPoint – выдает текущую позицию пера;
Pixels : TColor – двухмерный массив, содержащий цвета пикселей, например:

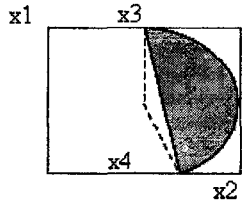
```
Procedure TForm1.Button1Click( );
Var
  i, j : LongInt;
  begin
    Button1.Visible := false;
    with Canvas do
    begin
      for i:=1 to Width do
        for j:=1 to Height do
        Pixels[i,j] := i*j;
        Button1.Visible := true;
    end;
  end;
```

Большое количество **методов класса TCanvas** позволяют отображать различные геометрические фигуры с помощью свойства Pen. Если фигура замкнута, то ее поверхность закрашивается Brush. Все тексты изображаются шрифтом Font.

В процессе работы программы эти характеристики можно изменять. Так:
Arc(x1, y1, x2, y2, x3, y3, x4, y4) – рисует дугу:



Chord(x1, y1, x2, y2, x3, y3, x4, y4) – рисует сегмент из дуги эллипса и хорды:

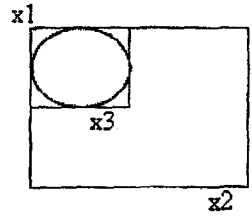


Ellipse(x1, y1, x2, y2) – рисует эллипс;
FillRect(Rect) – закрашивание прямоугольника;
MoveTo (x, y) – перемещает перо в точку с координатами x, y;
LineTo(x, y) – рисует линию из текущего положения пера в точку с координатами x и y;
Pie(x1, y1, x2, y2, x3, y3, x4, y4) – рисует сектор эллипса;
Poligon(Point: array of TPoint) – вычерчивание заданного многоугольника.

Пример:

```
var
  P: array[1..3] of TPoint;
begin
  P[1].x :=  10;   P[1].y := 300;
  P[2].x := 200;   P[2].y := 300;
  P[3].x := 100;   P[3].y :=  20;
  Canvas.Poligon(P);
end;
```

Poliline(Point: array of TPoint) – рисует ломаную;
RoundRect(x1, y1, x2, y2, x3, y3) – вычерчивание и заполнение прямоугольника со скругленными углами



TextOut(x, y, S: String) – осуществляет вывод строки;
TextRec выводит текст только внутри указанного прямоугольника. Длину и высоту текста можно узнать с помощью функций *TextWidth* и *TextHeight;*
Draw(x, y, Graphic: TGraphic) – прорисовка графического объекта Graphic так, чтобы левый верхний угол располагался в (x, y). Объект Graphic может быть типа Bitmap, Icon и Metafile;
StretchDraw(Rect: TRect; Graphic: TGraphic) – вычерчивание и масштабирование объекта Graphic до полного заполнения Rect.

Пример. На форме имеется Image1. С помощью свойства Picture в нее помещена картинка. Требуется переместить эту картинку в другое положение.

```
Procedure TForm1.FormPaint( );
begin
 with Canvas do
 begin
  Draw (0, 0, Image1.Picture.Bitmap);
  StretchDraw (Rect(250,0,350,50), Image1.Picture.Bitmap);
 end;
end;
```

Как правило, все графические операции осуществляются не на форме, а посредством специальных графических компонентов, например компонента Image, который позволяет разместить на экране растровое изображение, пиктограмму, метафайл, либо собственное изображение.

Для более простых графических операций используется компонент TPaintBox.

Постановка задачи

Разработать приложение, поддерживающее основные функции простейшего графического редактора.

Задание

Разработать приложение, содержащее три формы – три способа представления графической информации.

На 1-й форме продемонстрировать отображение графических картинок, созданных в других графических редакторах.

На 2-й форме с помощью кнопочного меню рисовать различные графические фигуры посредством компонента класса TShape.

На 3-й форме реализовать рисование простейшими примитивами, типа линия, прямоугольник, эллипс и т.д.

ЛАБОРАТОРНАЯ РАБОТА 12

Разработка приложения, управляемого с помощью панели инструментов

Цель: научиться подключать инструментальную панель, изучить основные свойства и типы кнопок и использование их для управления вычислительным процессом.

Краткие сведения

Для создания панели инструментов используется компонент TToolBar – инструментальная панель, пиктограмма которой имеет вид  .

Компонент TToolBar – это специальный контейнер для создания инструментальных панелей. В компонент TToolBar можно поместить любые другие компоненты. Как правило, он используется для расположения кнопок, с помощью которых мож-

но оперативно выполнить нужную команду. Кнопки можно группировать и располагать в несколько рядов.

Главная отличительная черта TToolBar – его способность гибкого управления дочерними элементами, которые он может группировать, выравнивать по размерам, располагать в несколько рядов. Компонент может манипулировать любыми вставленными в него дочерними элементами, но все его возможности в полной мере проявляются только при использовании специально для него разработанного компонента TToolButton (инструментальная кнопка). Этот компонент похож на кнопку TSpeedButton, но в палитре компонентов его нет.

Для того чтобы вставить TToolButton в инструментальную панель TToolBar, необходимо правой кнопкой щелкнуть на компоненте TToolBar и в открывшемся окне выбрать NewButton или NewSeparator (новый сепаратор). Сепараторы предназначены для функционального выделения на инструментальной панели групп элементов и представляют собой разновидности кнопок TToolButton.

Хотя компонент TToolButton не имеет свойства, предназначенного для хранения картинки, однако он умеет использовать контейнер TImageList, чтобы извлечь из него нужную картинку и поместить ее на инструментальную кнопку.

Постановка задачи

Разработать приложение, вычислительный процесс которого управляется компонентами, расположенными в инструментальной панели.

Задание

Разработать приложение, в котором предусмотреть управление вычислительным процессом с помощью кнопок, расположенных на инструментальной панели. Задание использовать из лабораторной работы 8.

ЛАБОРАТОРНАЯ РАБОТА 13

Разработка приложения, представленного в виде многостраничного документа

Цель: изучить компоненты TPageControl, TTabSheet.

Краткие сведения

Для создания многостраничных документов используются компоненты TTabControl и TPageControl.

Компонент TTabControl  (на странице Win32) представляет собой контейнер с закладками. Свойство Tabs определяет названия и количество закладок. Событие OnChange возникает при выборе новой закладки и позволяет управлять содержимым окна компонента.

Компонент TPageControl  (на закладке Win32) представляет собой контейнер с закладками, на каждой из которых содержатся панели класса TTabSheet. На каждой панели класса TTabSheet может содержаться свой набор помещенных на нее компонентов.

Для того чтобы добавить новую панель и закладку, нужно щелкнуть правой кнопкой по компоненту PageControl и из локального меню выбрать команду NewPage.

Свойства:

ActivePage: TTabSheet – содержит активную панель. С помощью этого свойства можно установить активной нужную панель.

События:

OnChange – возникает при переключении панелей.

Постановка задачи

Разработать приложение в виде многостраничного документа.

Задание

Разработать приложение, поддерживающее ввод данных, выбор метода расчета, расчет и представление результатов в табличной и графической формах на примере лабораторной работы 8, реализуя отдельные вычислительные шаги на различных закладках многостраничного документа.

ЛАБОРАТОРНАЯ РАБОТА 14

Разработка комплексного приложения

Цель: показать умение создания современного приложения в визуальной среде.

Краткие сведения

При разработке приложения для данной лабораторной работы следует руководствоваться теоретическими сведениями всех предыдущих работ.

Постановка задачи

Разработать современное приложение, содержащее расширенный список компонентов управления программой и представления данных в различных видах.

Задание

Разработать приложение, поддерживающее различные способы управления вычислительным процессом с помощью TPageControl, TTabSheet, TChart, стандартных диалоговых компонентов, TPopupMenu, TMainMenu, TToolBar и других на примере лабораторной работы 8.

ЛИТЕРАТУРА

1. Фаронов, В.В. Delphi 6: учебный курс / В.В. Фаронов. – М.: Изд-во Молгачева С.В., 2001. – 672 с.

2. Тейксейра, Стив. Delphi 6. Руководство разработчика: пер. с англ.: учебное пособие: в 2 т. / Стив Тейксейра, Ксавье Пачеко. – Т. 1: Основные методы и технологии. – М.: Вильямс, 2001. – 832 с.

3. Архангельский, А.Я. Разработка прикладных программ для Windows в Delphi / А.Я. Архангельский. – М.: Бином, 1999. – 256 с.

4. Подольский, С.В. Разработка интернет-приложений в Delphi 6 / С.В. Подольский, С.А. Скиба, О.А. Кожедуб. – СПб.: БХВ-Петербург, 2002. – 452 с.

5. Сван, Том. Delphi 4. Библия разработчика: пер. с англ. / Том Сван. – СПб.: Диалектика, 1998. – 672 с.

6. Бобровский, С. Delphi 5: учебный курс / С. Бобровский. – СПб.: Питер, 2000. – 640 с.

7. Фаронов, В.В. Delphi 2005. Язык, среда, разработка приложений / В.В. Фаронов. – СПб.: Питер, 2005. – 560 с.

8. Сухарев, М.В. Основы Delphi. Профессиональный подход / М.В. Сухарев. – СПб.: Наука и техника, 2004. – 600 с.

9. Марко, Кэнту. Delphi 5 для профессионалов / Кэнту Марко. – СПб.: Питер, 2001, – 944 с.

ПРИЛОЖЕНИЕ

Образец титульного листа

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

О Т Ч Е Т
О ЛАБОРАТОРНОЙ РАБОТЕ 14
«Разработка комплексного приложения в Delphi»

по дисциплине:
«РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ»

Выполнили: студенты: Петров И.И.,
Геращенко С.И.
гр. 107219

Проверил: доцент Гурский Н.Н.

Минск – 2010

Учебное издание

РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ

Лабораторный практикум
для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий», 1-40 01 02 «Информационные системы и технологии»

В 2 частях

Часть 1

Составитель
ГУРСКИЙ Николай Николаевич

Редактор Е.О. Коржуева
Компьютерная верстка Н.А. Школьниковой

Подписано в печать 22.10.2010.
Формат 60×84 1/16. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 3,43. Уч.-изд. л. 2,68. Тираж 100. Заказ 822.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.