2.3.3. Основные элементы языка программирования Object Pascal

Для составления более сложных программ в среде программирования Delphi необходимо знать основные элементы языка программирования *Object Pascal.* Язык **Object Pascal** лежит в основе системы программирования Delphi. Это современный язык, широко использующий технологию объектно-ориентированного программирования. Он создан сотрудниками корпорации Borland как усовершенствование языка Pascal, предложенного швейцарским профессором Н.Виртом для обучения студентов программированию еще в конце 60-х годов. Непосредственным предшественником ***Object Pascal*** был язык ***Turbo Pascal***.

Алфавит языка ***Object Pascal*** включает в себя буквы, цифры, шестнадцатеричные цифры, специальные символы, пробелы и зарезервированные слова. ***Буквы*** – это буквы латинского алфавита от «**a**» до «**z**» и от «**A**» до «**Z**», а также знак подчеркивания (**_**). В языке нет различия между прописными и строчными буквами алфавита, если только они не входят в символьные и строковые выражения. ***Цифры*** – арабские цифры от **0** до **9**.

Специальные символы Object Pascal:

+ - плюс	**<** - меньше
- - минус	**>** - больше
***** - звездочка	**[ ]** – квадратные скобки
/ - прямой слеш	**{ }** – фигурные скобки
= - равно	**( )** – круглые скобки
. - точка	**^** - тильда
‘ - апостроф	**$** - доллар
, - запятая	**@** - коммерческое а
: - двоеточие	**#** - номер
; - точка с запятой	- пробел (не имеет обозначения)

К специальным символам также относятся такие пары символов:

<> - не равно	**: =** - присвоить
<= - меньше или равно	// - два прямых слеша
>= - больше или равно	**..** – диапазон значений

(* *) - можно использовать вместо { }

(. .) - можно использовать вместо [ ]

В программе эти пары символов нельзя разделять пробелами.

В Object Pascal имеются следующие **зарезервированные слова:**

And	Function	Property
Array	Goto	Raise
As	If	Record
Asm	Implemen-tation	Repeat
Begin	In	Resourcestring
Case	Inherited	Set
Class	Initialization	Shl
Const	Inline	Shr
Constructor	Interface	String
Destructor	Is	Then
Dispinterface	Label	Threadvar
Div	Library	To
Do	Mod	Try
Downto	Nil	Type
Else	Not	Unit
End	Object	Until
Except	Of	Uses
Exports	Or	Var
File	Out	While
Finalization	Packed	With
Finally	Procedure	xor
For	Program	

Кроме этих слов, специальное значение имеют слова *at* и *on*. Зарезервированные слова не могут использоваться в качестве идентификаторов.

Идентификаторы в Object Pascal – это имена констант, переменных, меток, типов, объектов, классов, свойств, процедур, функций, модулей, программ и полей в записях. Идентификатор всегда начинается буквой, за которой могут следовать буквы и цифры. Пробелы и специальные символы алфавита не должны входить в идентификатор. Идентификаторы могут иметь произвольную длину.

Константы - это величины, значения которых установлены в описывающей части программы и в процессе выполнения программы не меняются. В качестве констант в Object Pascal могут использоваться целые, вещественные и шестнадцатеричные числа, логические константы, символы, строки символов, конструкторы множеств. Для определения констант служит служебное слово **const.**

Формат: const < идентификатор > = < значение константы >;

Например: **Const** Max = 1000; Min = 1;

Зарезервированные константы:

Идентификатор	Тип	Значение	Описание
true	boolean	true	«Истина»
false	boolean	false	«ложь»
maxint	integer	32767	Максимальное целое

Константам в программе нельзя присваивать новые значения после того, как они были описаны. Компьютер сам определяет тип констант по их значению.

Переменные - это величины, значение которых меняется в процессе выполнения программы. Для описания переменных используется служебное слово **var.**

Формат: **Var** < список идентификаторов > : < тип >;

Например: **Var** sum1, sum2 : real;

Переменная должна быть описана в программе только один раз и принадлежать только к одному типу.

Тип – это множество значений одинаковой природы вместе с набором операций, которые над ними выполняются. Каждая переменная должна быть описана только один раз в начале программы после слова *var*. Программист сам выбирает нужный ему тип в зависимости от возможных значений переменной (с учетом диапазона данных и размера памяти, которая выделяется транслятором для отдельной переменной).

Скалярные (простые) типы данных

Целые типы данных

Название	Длина, байт	Диапазон значений
Byte	1	0…255
ShortInt	1	-128…+127
SmallInt	2	-32768…+32767
Word	2	0…65535
Integer	4	-2 147 483 648…+2 147 483 647
LongInt	4	-2 147 483 648…+2 147 483 647
LongWord	4	0…4 294 967 295
Int64	8	$-2^{63}…+2^{63}-1$
Cardinal	4	0…4 294 967 295

Например: **Var** x, y : **integer**; z : **word**;

На множестве ***целых*** чисел определены такие **операции**:

+ - сложение; ***** - умножение;

- - вычитание; **/** - деление;

div - деление нацело;

mod - остаток от деления нацело.

Математические функции:

Abs (x) - | x |

Cos (x) – cos x

Exp (x) - e^x

Sqr (x) – x^2

Arctan (x) – arctg x

Sin (x) – sin x

Ln (x) – ln x

Sqrt (x) - $\sqrt{x}$

Random (x) – случайное число от *0* до *x – 1*

Odd (x) - возвращает значение *True*, если *x* – нечетное число.

Результат выполнения операций + , - , * , *div, mod, abs(x), sqr(x)* над целыми числами является целым числом.

Результат выполнения операции /, а также всех остальных математических функций на множестве целых чисел, является вещественным числом.

Операции *div* и *mod* можно применять только к целым числам.

Пример: 17 *div* 3 = 5; 17 *mod* 3 = 2.

Вещественные типы

Название	Длина, байт	Диапазон значений
Real	8	$5{,}0*10^{-324}…1{,}7*10^{308}$
Single	4	$1{,}5*10^{-45}…3{,}4*10^{38}$
Double	8	$5{,}0*10^{-324}…1{,}7*10^{308}$
Extended	10	$3{,}4*10^{-4951}…1{,}1*10^{4932}$
Comp	8	$-2^{63}…+2^{63-1}$
Currency	8	±922 337 203 685 477,5807

Например: Var *a, b:**real**; c,d,f:**single**;*

На множестве ***вещественных*** чисел определены такие **операции:**

+	- сложение;	*****	- умножение;
-	- вычитание;	**/**	- деление.

Математические функции

Abs (x)	- | x |	**Arctan (x)**	– arctg x
Cos (x)	– cos x	**Sin (x)**	– sin x
Exp (x)	- e^x	**Ln (x)**	– ln x
Sqr (x)	– x^2	**Sqrt (x)**	- $\sqrt{x}$

Int (x) - задает целую часть числа x

Trunc (x) - отбрасывает дробную часть числа x

Frac (x) - задает дробную часть числа x

Round (x) - округление числа x.

Результатом выполнения функций *trunk (x)* и *round (x)* на множестве вещественных чисел есть целое число.

Результатом выполнения всех остальных операций и функций на множестве вещественных чисел есть вещественное число.

Пример:

Int (3.6) = 3.0;	Trunc (-7.8) = -7;
Int (-5.1) = -5.0;	Frac (-7.8) = -0.8;
Int (0.99) = 0.0;	Round (7.8) = 8;
Trunk (7.8) =7;	Round (-7.8) = -8;
Frac (7.8) = 0.8;	Frac(1) = 0.

Логический (булевый) тип:

Название	Диапазон
boolean	True, False

Например: **Var** let : **boolean**;

Результат выполнения операций сравнения (= , < , > , <= , >= , <>), а так же операции *odd(x)* принадлежит к логическому типу.

Символьный тип:

Название	Диапазон
char	Таблица кодов компьютера

Например: **Var** m : **char**;

Функции для работы с символьным типом:

Ord (S) – определяет порядковый номер (код) символа S в кодовой таблице. Результат данной функции принадлежит к целому типу.

Chr (I) – находит символ, порядковый номер (код) которого равняется I. Результат данной функции принадлежит к символьному типу.

Succ (S) – определяет символ, который находится после символа S в кодовой таблице.

Pred (S) – определяет символ, который находится перед символом S в кодовой таблице.

Upcase (s) – преобразует маленькие буквы английского алфавита в большие.

Результат данных функций принадлежит к символьному типу.

Например: Pred (‘B’) = “A’; Succ (‘B’) = “C’;
Upcase (‘n’) = ‘N’.

Выражение задает порядок выполнения действий над элементами данных и состоит из операторов (констант, переменных, обращения к функциям), круглых скобок и знаков операций.

Примеры выражений:

Математика	Object Pascal
$ax^2 + b$	a*sqr(x)+b
$\frac{a+b}{\lvert c-d \rvert}$	(a+b)/abs(c-d)
$\sqrt{x^2+1}$	sqrt(sqr(x)+1)
$e^x \ln(x+2)$	exp(x)*ln(x+2)
a^b	exp(b*ln(a))

Выполнение каждой операции осуществляется в порядке учета ее приоритета.

Приоритет операций

1. @, not.
2. * , / , div, mod, and, shl, shr.
3. + , - , or, xor.
4. = , <> , < , > , <= , >= , in.

Простые нестандартные типы данных (типы пользователя)

Перечисленный тип задается перечислением тех значений, которые он может получать.

Например: **type** colors = (red, white, blue);
Var a,b:colors;

Тип-диапазон задается границами своих значений внутри базового типа, в качестве которого могут выступать целые типы, логический тип, символьный тип и перечисленный тип. Левая граница диапазона не должна превышать его правую границу.

Пример: **type** digit = ‘0’ .. ‘9’;
dig2 = 48 .. 57 ;
var a,b:digit; x,y:dig2;

Строки

Для обработки текстов в Object Pascal используется тип **string** (строка). Этот тип относится к так называемым **структурированным** типам данных. **Строка** – это последовательность символов кодовой таблицы компьютера. При использовании в выражениях строка окружается с обеих сторон апострофами. Количество символов в строке (максимальная длина строки) может меняться от 0 до 255.

Формат:

Var < идентификатор > : string [максимальная длина строки];

Пример: **Var** R1 : string[10]; R2 : string[4];

Максимальная длина строки для переменной R1 равна 10, для R2 равна 4.

Если длина строки не указана, то она автоматически принимает значение 255 байт.

Строчные величины можно использовать в программе и в виде констант.

Пример: **Const** Name = ‘информатика ‘;

Рассмотрим некоторые процедуры и функции для работы со строковыми величинами.

1. **Процедура Str**

   Формат: **Str** (< число >,< переменная >).

   Процедура *Str* переводит числовое данное в данное типа строка.

   Например: ***Str*** (2000, Kiev);

   Результатом действия процедуры будет значение переменной Kiev = ‘2000’.

2. **Процедура Val**

   Формат: **Val** (R1, S1, S2).

   Процедура Val присваивает числовой переменной S1 числовой образ строки R1. Если это возможно, то переменная S2 получает значение 0, иначе – числовое значение номера первого недопустимого символа заданной строки.

Например: Val (‘1256’, abc, code);

Результатом действия процедуры будет значение переменной abc = 1256, переменная Code примет значение 0.

3. **Функция StrToFloat**

Формат: **StrToFloat** (S1).

Функция *StrToFloat* преобразует символы строки S1 в вещественное число.

Например: S1 : = “123”; N : = ***StrToFloat*** (S1);

Результатом действия функции будет значение переменной N = 123.

4. **Функция StrToInt**

Формат: **StrToFloat** (S1).

Функция *StrToFloat* преобразует символы строки S1 в целое число.

5. **Функция FloatToStr**

Формат: **FloatToStr**(S1).

Функция *FloatToStrt* преобразует вещественное значение S1 в строку символов.

6. **Функция IntToStr**

Формат: **IntToStr**(S1).

Функция *IntToStr* преобразует целое значение S1 в строку символов.

2.3.4. Составление линейных (простых) программ

ПРАКТИЧЕСКАЯ РАБОТА №4

Тема. ***Создание проекта «Периметр и площадь прямоугольника»,*** который позволяет вычислять периметр *P* и площадь *S* прямоугольника, если его длина *a* и ширина *b* заданы пользователем.

Цель. Получение навыков составления линейных программ в среде Delphi. Познакомиться с такими новыми объектами: поле редактирования (**Edit**), кнопка с рисунком и определенным типом действия (**BitBtn**) и их основными свойствами: **Hint** (подсказка**),**

ReadOnly (возможность менять текст)**, Text** (текст в поле редактирования)**, Kind** (тип стандартного действия) и другими.

Объекты: форма, текстовое поле, кнопка, поле редактирования, кнопка с изображением.

Теоретические сведения. Объект ***Edit*** используют для введения пользователем строки символов с клавиатуры. В случае необходимости для преобразования полученной строки (свойство ***Text***) в число и наоборот используют стандартные функции языка *Pascal* ***Val*** и ***Str*** или функции *Delphi* ***StrToFloat*** и ***FloatToStr***. Кроме известных вам свойств, поле редактирования ***Edit*** владеет такими:

Свойство	Описание свойства	Примеры значений
CharCase	Вид символов, которые будут вводиться в поле редактирования	ecNormal (обычные), ecUpperCase (большие буквы), ecLowerCase (маленькие буквы).
Ctl3D	Объемное представление объекта	True, False
PasswordChar	Символ для введения пароля	#0 (прямое отображение текста), * (текст будет отображаться звездочками), 0 (текст будет отображаться нулями)
ReadOnly	Возможность изменить текст (доступность поля)	True (текст нельзя изменить), False (текст можно изменить)
Hint	Текст подсказки, которая высвечивается, если навести курсор мыши	«Введите число» (любая строка символов)
ShowHint	Высвечивать/не высвечивать подсказку	True, False

Text	Текст в поле редактирования	«Об0001» (любая строка символов)

Объект **Edit** находится на закладке **Standard** палитры компонентов *(рис. 2.34).*

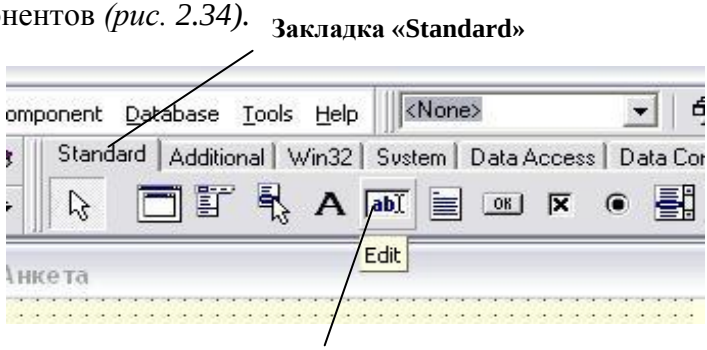

Рис. 2.34. Объект Edit (поле редактирования)

Объект ***BitBtn*** – кнопка с определенным типом действия. Стандартный набор файлов с рисунками для кнопок находится в папке *C:\ Program Files\ Borland\ Delphi x.0\ Images\ Buttons.* Этот объект владеет такими новыми свойствами:

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

Объект **BitBtn** находится на закладке **Additional** палитры компонентов *(рис. 2.35).*

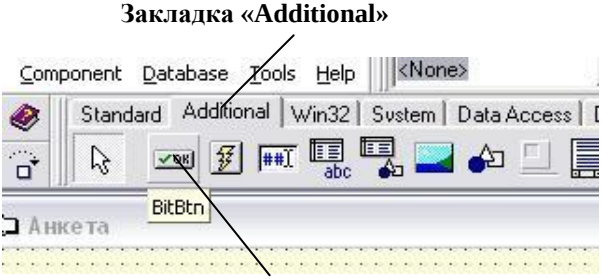

Рис. 2.35. Объект BitBtn (кнопка с действием)

Ход работы

1. **Загрузите среду визуального программирования Delphi.**
2. **Откажитесь от системных кнопок формы, задав комплексному свойству формы *BorderIcons* такие значения:**

   biSystemMenu: False; **biMaximize:** False;
   biMinimize: False; **biHelp:** False.

   Чтобы задать значения комплексного свойства *BorderIcons*, нужно щелкнуть по значку ⊞ рядом с его названием в окне инспектора объектов *(рис. 2.36).*

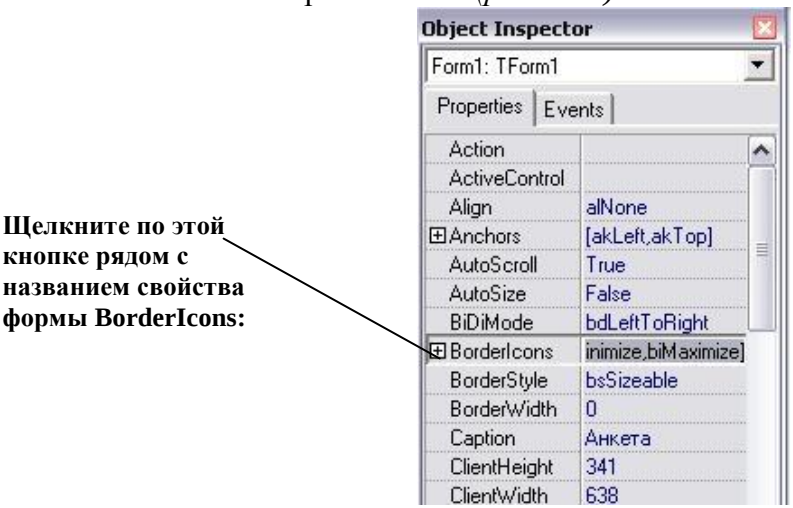

Рис. 2.36. Свойство формы BorderIcons

После щелчка на кнопка изменится с « + » на « - » и откроется список всех позиций комплексного свойства *(рис. 2.37).*

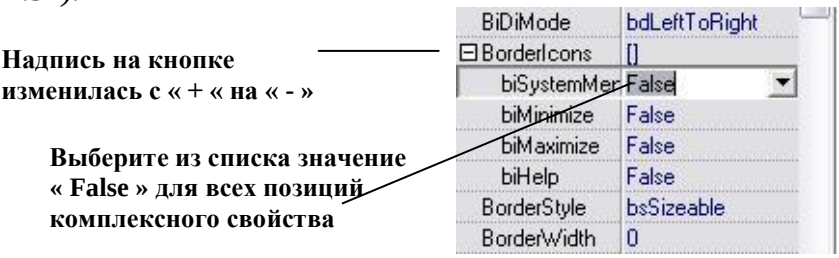

Рис. 2.37. Задание значений комплексного свойства

3. **Разместите объекты на форме так, как показано на *рисунке 2.38*:**

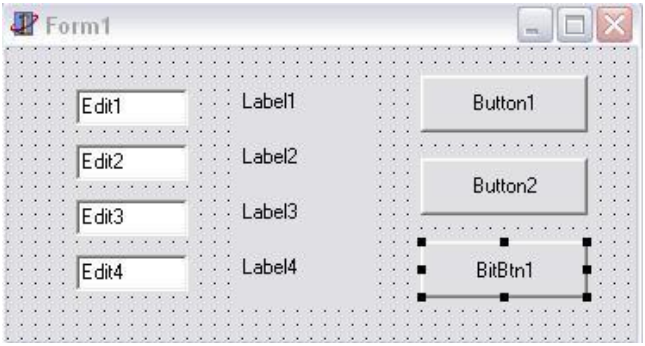

Рис. 2.38. Размещение объектов на форме

4. **Очистите поля редактирования *Edit1, Edit2, Edit3, Edit4*.** В окне ***Object Inspector*** в строке *Text* удалите текст *Edit1, Edit2, Edit3, Edit4 (рис. 2.39).*

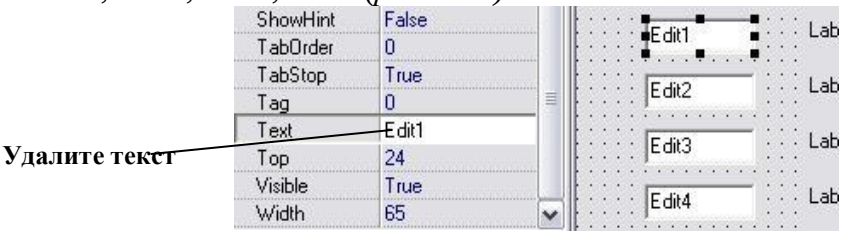

Рис. 2.39. Свойство «Text» объекта Edit

5. Задайте свойствам объектов значения:

Объект **Form**

Caption: Периметр и площадь прямоугольника
Color: ClInfoBk
Icon: любой рисунок из файла с расширением .ico.

Объект **Edit1**

Hint: Введите длину прямоугольника
ShowHint: True
Очистите поле ***Text***

Объект **Edit2**

Hint: Введите ширину прямоугольника
ShowHint: True
Очистите поле ***Text***

Объект **Edit3**

ReadOnly: True
Hint: Периметр прямоугольника
ShowHint: True
Очистите поле ***Text***

Объект **Edit4**

ReadOnly: True
Hint: Площадь прямоугольника
ShowHint: True
Очистите поле ***Text***

Объект **Label1**

Caption**:** *a*; ***Size:***12;
Font: Times New Roman Cyr; ***Color:*** Blue.
Font Style: Bold;

Объект **Label2**

Caption: *b*; ***Size:***12;
Font: Times New Roman Cyr; ***Color:*** Blue.
Font Style: Bold;

Объект **Label3**

Caption**:** *P*; ***Size:***12;
Font: Times New Roman Cyr; ***Color:*** Red.
Font Style: Bold;

Объект **Label4**

Caption**:** *S*; ***Size:***12;
Font: Times New Roman Cyr; ***Color:*** Red.
Font Style: Bold;

Объект **Button1**

Caption**:** Вычислить

Объект **Button2**

Caption**:** Очистить

Объект **BitBtn**

Kind**:** bkClose
Caption: Закрыть

После этого форма примет вид *(рис. 2.40):*

Рис. 2.40. Форма после добавления объектов

6. Запрограммируйте кнопку «Очистить». Для этого дважды щелкните по ней левой кнопкой мыши и в появившейся заготовке процедуры введите программный код:

```
procedure TForm1.Button2.Click(Sender: TObject);
begin
    edit1.clear;edit2.clear;
    edit3.clear;edit4.clear;
end;
```

7. Запрограммируйте кнопку «Вычислить». Для этого дважды щелкните по ней левой кнопкой мыши и в появившейся заготовке процедуры введите программный код:

```
procedure TForm1.Button1.Click(Sender: TObject);
    var a, b, P, S:real;s1,s2:string;code:integer;
begin
    val(edit1.text,a,code); val(edit2.text,b,code);
    P:= 2*(a + b);S:= a*b; str(P:8:2,s1); str(S:8:2,s2);
    edit3.text:=s1; edit4.text:=s2;
end;
```

В записи ***P:8:2*** 8 – это максимальное количество цифр перед запятой в десятичной записи вещественного числа, 2 – максимальное количество цифр после запятой.

Например: $\underbrace{\text{…..656}}_{8}\underbrace{,53}_{2}$

8. **Сохраните проект в своей папке *(File → Save All).***

9. **Выполните программу *(Run → Run).***

10. **Создайте выполняемый файл *(Project → Build Project).***

Дополнительные задания

1. Создайте проект для вычисления *скорости v*, если известны *путь s* и *время t* (формула $v = \frac{s}{t}$).

2. Создайте проект для вычисления *длины окружности l* и *площади круга S*, если задан его *радиус R (рис. 2.41).*

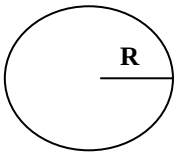

Рис. 2.41. Окружность

Формулы: $P = 2\pi R$,

$S = \pi R^2$, $\pi = 3{,}14$

3. Создайте проект для вычисления периметра и площади прямоугольного треугольника по заданным катету и прилежащему острому углу.

4. Создайте проект для вычисления периметра и площади прямоугольного треугольника по заданным катетам.

5. Создайте проект для вычисления периметра и площади квадрата по заданной диагонали *d.*

6. Создайте проект для вычисления периметра и площади прямоугольника по заданным стороне и диагонали.
7. Создайте проект для вычисления стороны и периметра квадрата по заданной площади S.
8. Создайте проект для вычисления периметра и площади квадрата по заданной стороне a.
9. Создайте проект для вычисления площади треугольника по заданным двум сторонам и углу между ними.
10. Создайте проект для вычисления *площади S* треугольника, если заданы координаты его вершин: *(x1, y1), (x2, y2), (x3, y3) (рис. 2.42).*

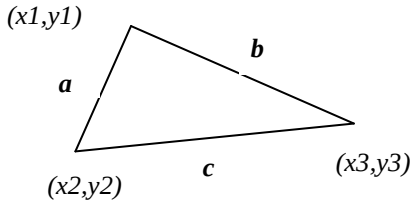

Рис. 2.42. Треугольник

Формулы: $a = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$;

$b = \sqrt{(x3 - x1)^2 + (y3 - y1)^2}$; $c = \sqrt{(x3 - x2)^2 + (y3 - y2)^2}$;

$p = \frac{a + b + c}{2}$; $S = \sqrt{p(p-a)(p-b)(p-c)}$.

ВОПРОСЫ

1. В чем заключаются различия процедурного и визуального программирования?
2. Что есть структурной единицей в процедурном и визуальном программировании?
3. В чем заключается идея объектно-ориентированного программирования?
4. Что такое форма?
5. Какие есть основные окна среды программирования Delphi?
6. Какие вы знаете элементы главного меню и для чего они предназначены?

7. Какие скалярные типы данных используются в языке программирования Pascal?
8. Как описываются данные целого, вещественного, символьного, логического типов?
9. Какие существуют простые нестандартные типы данных?
10. Как описывается перечисленный и интервальный типы?
11. Как вы опишете данные строчного типа?
12. Как определяется порядок действий в арифметических выражениях, записанных на языке Pascal?
13. Какие знаки арифметических операций используются для записи выражений на языке Pascal?
14. Какому типу будет принадлежать результат деления 14 на 4?

ЗАДАНИЯ

1. Определите тип данных для величин:
   а) цена книги;
   б) количество студентов в группе;
   в) имя студента.
2. Запишите числа на языке Pascal:
   а) - 47,14;
   б) $2{,}15 \cdot 10^{6}$;
   в) $6{,}67 \cdot 10^{-11}$;
   г) $7{,}3^{-5}$;
   д) $0{,}6^{4}$.
3. Найдите значение выражений:
   а) 7 div 3 =
   б) -12 div 4 =
   в) 1 div 2 =
   г) 10 mod 3 =
   д) -7 mod 2 =
   е) 2 mod 5 =
4. Найдите значение функций:
   а) odd (16) =
   б) odd (26) =
   в) succ (130) =
   э) round (7.4) =
   г) pred (15) =
   ж) round (-7.4) =
   д) trunk (6.54) =
   з) pred (‘D’) =
   е) trunk (-3.9) =
   и) succ (‘D’) =
5. Запишите выражения на языке Pascal:
   а) $\frac{z^2 + xy - \cos x}{|x - y|} + \ln z$;
   б) $\frac{arctg(a-b) + 5a^2}{ab + 4} - \sqrt{a^2 - c}$.

2.4. Процедуры и функции

Новые слова

Русский язык	***English***
аргумент	argument
глобальный	global
локальный	local
параметр	parameter
подпрограмма	subprogram
пользователь	user
результат	result
стандартный	standard
фактический	factual
формальный	formal

Подпрограммы предназначены для реализации алгоритмов обработки отдельных частей сложной задачи. Различают два вида подпрограмм – ***подпрограммы-процедуры*** и ***подпрограммы-функции***. Существуют подпрограммы *стандартные* и *подпрограммы пользователя*. ***Стандартные подпрограммы*** создавать не нужно – они содержатся в стандартных модулях *System, Crt, Dos, Graph*. ***Подпрограммы пользователя*** – это поименованная группа команд, которую создают и описывают в основной программе в разделах **procedure** или **function** и к которой обращаются из любого места программы нужное количество раз.

1. Процедуры (procedure). Общее описание процедуры:

```
procedure < название > (< список формальных параметров >);
          < разделы описаний и объявлений процедуры>;
        begin
          < раздел команд процедуры>
        end;
```

В списке ***формальных параметров*** перечисляют *переменные вместе с указанием их типов.* Различают ***параметры-аргументы*** (другой термин: *параметры-значения*) – входные данные для процедуры, и ***параметры-результаты*** (другой термин: *параметры-переменные*), через которые можно возвращать результаты работы процедуры в

основную программу. Перед списками параметров-результатов каждого типа записывают слово **var.**

Пример. Рассмотрим процедуру с названием **tangens**, которая вычисляет тангенс заданного угла.

```
procedure tangens (x:real; var y:real);
    begin
        y = (sin(x))/(cos(x));
    end;
```

В приведенном примере x является *формальным параметром-аргументом*, y – *формальным параметром-результатом***.**

К процедуре обращаются из раздела команд основной программы или другой подпрограммы. Процедуру вызывают с помощью команды вызова:

```
<название процедуры>(<список фактических параметров>);
```

Фактические параметры – это параметры, которые записывают в команде вызова процедуры. Фактическими параметрами-аргументами могут быть константы, переменные, выражения, а параметрами-результатами – только переменные. Типы данных здесь не указывают.

Между фактическими и формальными параметрами должно быть соответствие по количеству и типам. Соответствующие фактические и формальные параметры могут иметь разные имена.

Пример. Программа для вычисления значений функции $a = tg(b) + ctg(b) + tg^2(b)$ с использованием созданной нами процедуры **tangens** запишется так:

```
unit Unit1;
…
var a,b:real;code:integer;s1:string;
procedure tangens (x:real; var y:real);
    begin
        y := (sin(x))/(cos(x));
    end;
```

```
procedure TForm1.Button1.Click(Sender: TObject);
begin
    val(edit1.text,b,code); tangens(b,n);
    a:=n+1/n+n^2;
    str(a,s1); edit2.text:=s1;
end;
end.
```

Команда вызова функционирует так: значения фактических параметров присваиваются соответствующим формальным параметрам процедуры, выполняется процедура, определяются параметры-результаты, значения которых присваиваются (возвращаются) соответствующим фактическим параметрам в команде вызова.

Переменные, описанные в разделе описаний основной программы, называются ***глобальными***. Они действуют во всех подпрограммах, составляющих программу. Переменные, описанные в разделе описаний конкретной процедуры, называются ***локальными***. Они действуют только в пределах данной процедуры.

В созданной нами программе глобальными являются переменные a, b и s1, так как они описаны в разделе *var* главной программы. Локальными являются переменные x и y, так как описаны только в подпрограмме.

2. Функции (function). Функция, в отличие от процедуры, может возвращать в место вызова только один результат простого стандартного типа.

Общее описание функции:

```
function <название> (<список формальных параметров>) : <тип
        функции> ;
        < разделы описаний и объявлений функции>;
    begin
        < раздел команд функции, где должна быть такая
        команда: название : = выражение >
    end;
```

Пример. Рассмотрим функцию с названием **tangens**, которая вычисляет тангенс заданного угла.

```
function tangens (x: real):real;
    begin
        tangens:= (sin(x))/(cos(x)) ;
    end;
```

В разделе команд функции должна быть команда присвоения значения некоторого выражения названию функции. Результат функции возвращается в основную программу через ее название. Вызов функции осуществляется так:

```
< название > (< список фактических параметров >
```

Пример. Программа для вычисления значений функции $a = tg(b) + ctg(b) + tg^2(b)$ с использованием созданной нами функции **tangens** запишется так:

```
unit Unit1;
…
var a,b:real;code:integer;s1:string;
function tangens (x:real):real;
    begin
        tangens:= (sin(x))/(cos(x)) ;
    end;
procedure TForm1.Button1.Click(Sender: TObject);
begin
    val(edit1.text,b,code);
    a:= tangens(b)+1/tangens(b)+sqr(tangens(b));
    str(a,s1); edit2.text:=s1;
end;
end.
```

ВОПРОСЫ

1. Какие два вида подпрограмм вы знаете?
2. Чем отличаются процедуры от функций?

3. Чем отличаются формальные параметры от фактических?
4. Какие есть два вида формальных параметров?
5. Какие переменные называются глобальными?
6. Какие переменные называются локальными?
7. Чем отличаются глобальные переменные от локальных?

ЗАДАНИЯ

1. Создайте проект для вычисления значений функции $a = tg(b) + ctg(b) + tg^2(b)$ с использованием процедуры или функции для вычисления тангенса.
2. Создайте проект для вычисления значений функции $m = ctg(n) + tg(n) + ctg^2(n)$ с использованием процедуры или функции для вычисления котангенса.
3. Создайте проект для вычисления площади *S* треугольника, если заданы координаты его вершин *(x1, y1), (x2, y2), (x3, y3),* используя процедуру или функцию для вычисления стороны треугольника.

2.5. Программирование ветвлений

Новые слова

Русский язык	***English***
безусловный	unconditional
ветвление	branch
вспомогательный	auxiliary
выпадающий	out-of-order
двойной	binary, dual, duplicate, pair, twin
комбинированный	combined
конкретный	concrete
конструкция	construction
контекстный	context
короткий	short
очистить	clear
переход	crossing, transfer, transition
перечислить	enumerate
полный	full
предоставленный	vested
привязать	instance, bind, associate, snap

простой	simple
разновидность	subkind, variety, variant
рамка	border, frame
раскрыть	open
совпадать	concur, coincide
составной	composite
список	details
справочник	compendium, manual, directory
телефонный	phone
условный	conditional

Составная команда – это конструкция вида:

```
begin
    < команда 1 >
    …
    < команда n>
end;
```

Составная команда подразумевается как одна команда.

Логическое выражение – это средство записи условий для поиска нужных данных. Логическое выражение может принимать значение **true** (истина) или **false** (ложь). Логические выражения бывают *простыми* и *составными*. **Простое** – это два арифметических выражения, соединенные символом отношения, а **составное** – это простые логические выражения, соединенные названиями логических операций: **not**, **and** и **or**.

Пример. Двойное неравенство *1<x<5* как составное логическое выражение записывают так: *(1<x) and (x<5).*

Команда ветвления if (условная команда) имеет две разновидности:

1. **Полная** команда ветвления имеет вид:

```
if < логическое выражение > then < команда1 > else < команда2 >;
```

Действие команды. Если логическое выражение истинно, то выполняется команда 1, в противном случае выполняется команда 2. Команды 1 и 2 могут быть простыми или составными.

Пример. Для функции $y = \begin{cases} \sin x, \ если \ x < -1, \\ \cos x, \ если \ -1 < x < 1, \\ \ln x, \ если \ x > 1 \end{cases}$

команда ветвления запишется так:

if *x < -1* ***then*** *y : = sin(x)* ***else*** *if (x > = -1)* ***and*** *(x < 1)* ***then*** *y : = cos(x)* ***else*** *y : = ln(x);*

2. **Короткая** команда ветвления имеет вид:

if < логическое выражение > **then** < команда1 >;

Действие команды. Если логическое выражение истинно, то выполняется команда1, иначе выполняется команда, которая находится после данной конструкции.

Команда goto – это команда безусловного перехода, которая меняет последовательность выполнения других команд программы путем перехода к выполнению команды, которая имеет обозначение (**метку**):

goto < метка >;

Метка может стоять перед любой командой в программе. Она отделяется от команды двоеточием (:)

< метка > : < команда >;

Метку нужно предварительно объявить вначале программы в разделе label:

label < список меток >;

Метка может начинаться с буквы или быть числом от 0 до 9999. **Команда выбора (case)** имеет вид:

```
case < выражение > of
    < список значений 1 > : < команда 1 >;
    …
    < список значений n > : < команда n >;
    else < команда n+1 >
end;
```

Здесь выражение – это простая переменная целого, символьного, перечисленного или логического типа; списки значений – постоянные или диапазоны, тип которых

совпадает с типом выражения. Если список значений состоит из нескольких элементов, то они перечисляются через запятую. Составная часть **else** < команда n+1 > может отсутствовать – тогда получим короткую форму команды **case.**

Действие команды. Если значение выражения совпадает с некоторым значением со списка, то выполняется команда, которая соответствует этому значению, а другие команды этой конструкции не выполняются. Если значение выражения не совпадает ни с одним значением из списка, то выполняется команда n+1 или, в случае короткой формы, следующая команда после команды **case.**

Пример. Для функции

$$y = \begin{cases} \sin x, \ \textit{если} \ x = -1, \\ \cos x, \ \textit{если} \ x = 1, \\ \ln x, \ \textit{если} \ x > 1 \ \textit{или} \ x < -1 \ \textit{или} \ -1 < x < 1 \end{cases}$$

команда выбора (**case**) запишется так:

```
case x of
        -1:    y : = sin(x);
         1:    y : = cos(x);
        else   y : = ln(x)
end;
```

ПРАКТИЧЕСКАЯ РАБОТА №5

Тема. ***Создание проекта «Телефонный справочник»***, который позволяет получить сведения о фамилиях, именах и адресах людей по номеру их телефона, выбранному пользователем из предоставленного списка телефонных номеров.

Цель. Получение навыков программирования ветвлений (в частности команды выбора **case**) в среде Delphi. Познакомиться с такими новыми объектами: главное меню (**MainMenu**), содержащее команды «Вывести», «Очистить» («Очистить все» и «Очистить поля выведения») и «Закрыть», контекстное меню

(**PopupMenu**) полей выведения Edit1, Edit2, Edi3 (которое содержит команды «Вывести» и «Очистить»), комбинированный список (**Combobox**) и их основными свойствами.

Объекты: форма, текстовое поле, поле редактирования, главное меню, контекстное меню, комбинированный список.

Теоретические сведения. С помощью объекта **MainMenu** создают 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 ***PopupMenu*** создают контекстное меню некоторого компонента. Для «привязывания» контекстного меню к конкретному объекту необходимо его свойству ***PopupMenu*** задать значение имени ***(Name)*** этого объекта. Рассмотрим некоторые свойства контекстного меню:

Свойство	Описание свойства	Примеры значений
Alignment	Выравнивание меню относительно точки щелканья правой кнопкой мыши	paCenter (по центру), paLeft (слева), paRight (справа)
AutoPopup	Автоматический вызов контекстного меню	True (вызывается щелканьем правой клавишей мыши), False (вызывается с помощью метода Popup)

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

Свойство	Описание свойства	Примеры значений
Break	Разбиение меню в горизонтальном направлении	mbNone (без разбиения), mbBarBreak (разбиение с вертикальной чертой), mbBreak (разбиение без вертикальной черты)
ShortCut	Комбинация «горячих» клавиш для быстрого вызова команды меню	Ctrl+A, F8, Ctrl+F10, Shift+F3, Shift+Ctrl+F11, Ctrl+Del

Объект ***Combobox*** используют для создания выпадающего списка. Его новые свойства:

Свойство	Описание свойства	Примеры значений
DropDownCount	Количество строк в выпадающем списке, которые видны без использования полос прокручивания	7; 3
Sorted	Упорядочивание списка по алфавиту	True (список сортируется), False (список не сортируется)
Style	Стиль оформления и использования списка	csOwnerDrawVariable (заданный программистом), csDropDown (стандартный)

В программном коде используется свойство ***ItemIndex*** объекта ***Combobox***. Значение 0 этого свойства соответствует первому элементу списка, 1 - второму, 2 – третьему и т.д. Значение -1 соответствует пустой строке списка. Для составления списка объекта ***Combobox*** используется его комплексное свойство ***Items***.

Объекты ***MainMenu***, ***PopupMenu***, ***ComboBox*** находятся на закладке ***Standard*** палитры компонентов *(рис. 2.43)*.

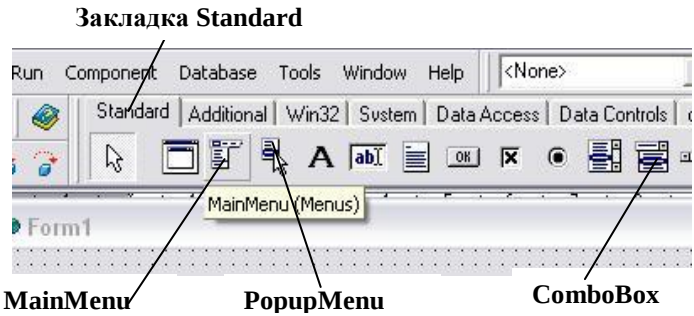

Рис. 2.43. Объекты MainMenu, PopupMenu, Combobox

Ход работы

1. **Загрузите среду визуального программирования Delphi.**
2. **Разместите объекты на форме** *(рис. 2.44)***.**

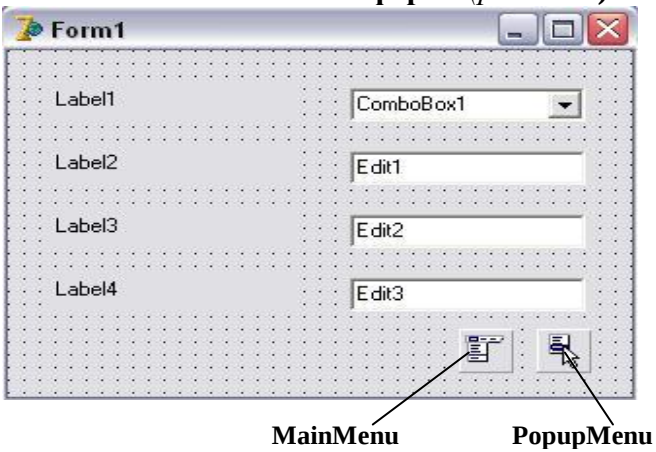

Рис. 2.44. Размещение объектов на форме

3. **Очистите поля Edit1, Edit2, Edit3, Combobox1 (свойство *Text*).**
4. **Задайте свойствам объектов значения:**

Объект **Form**

Caption: Телефонный справочник

Объект **Label1**

Caption: Телефон

Объект **Label2**

Caption: Фамилия

Объект **Label3**

Caption: Имя

Объект **Label4**

Caption: Адрес

Объект **Edit1**

ReadOnly: True
PopupMenu: PopupMenu1
Очистите поле ***Text***

Объект **Edit2**

ReadOnly: True
PopupMenu: PopupMenu1
Очистите поле ***Text***

Объект **Edit3**

ReadOnly: True
PopupMenu: PopupMenu1
Очистите поле ***Text***

Объект **Combobox1**

Очистите поле ***Text***
Items: 24535654
4563767
2763567

5. Введите названия команд главного меню формы.

Для этого дважды щелкните по объекту *MainMenu* (или по его свойству *Items*). В открывшемся окне (*Form1.MainMenu1*) выбирайте с помощью мыши рамку команды и записывайте название команды ***(Вывести, Очистить, Закрыть*)** как значение свойства *Caption* в окне *Object Inspector*. *(рис. 2.45, 2.46, 2.47).* После введения названий всех команд главного меню закройте окно *Form1.MainMenu1*.

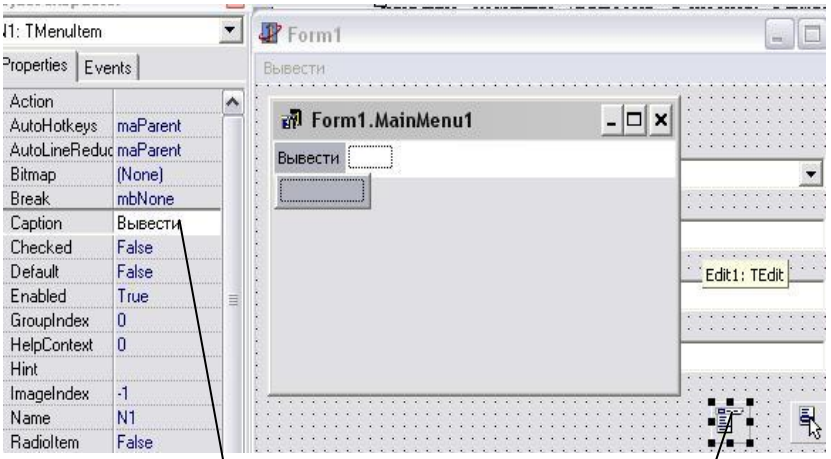

Рис. 2.45. Введение команд главного меню

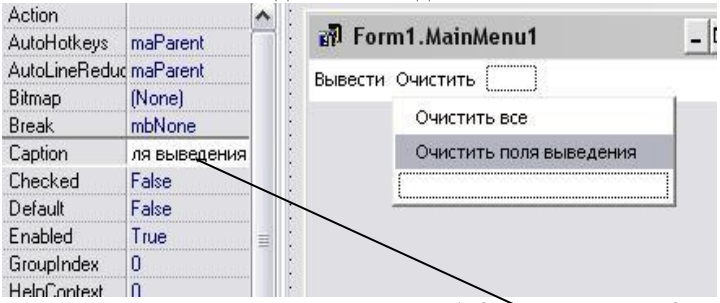

Рис. 2.46. Введение команд низшего уровня («Очистить все», «Очистить поля выведения»)

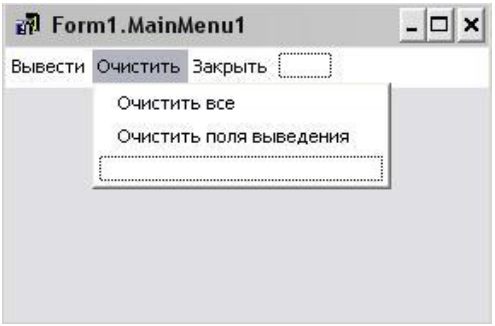

Рис. 2.47. Введение всех команд главного меню

На этапе выполнения программы пиктограммы *MainMenu* и *PopupMenu* будут невидимыми. Видны будут только названия команд главного меню. Чтобы увидеть команды контекстного меню, нужно щелкнуть правой кнопкой мыши по одному из полей введения *Edit1*, *Edit2* или *Edit3*. После задания свойств объектов форма примет вид *(рис. 2.48)*:

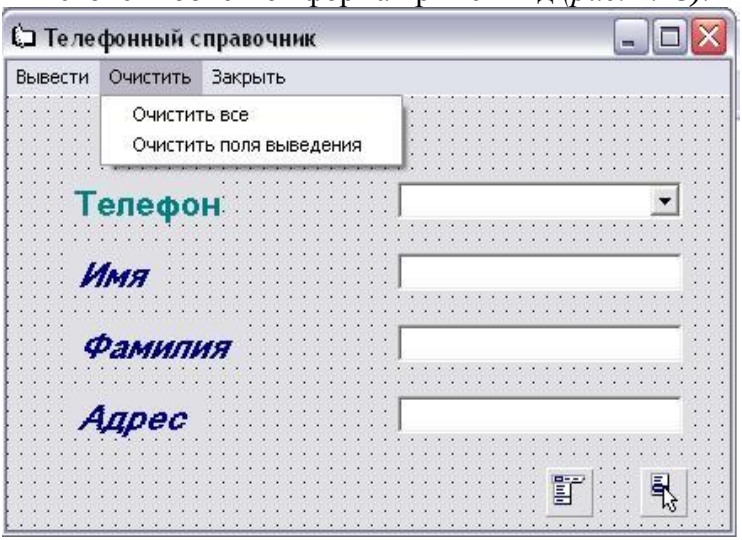

Рис. 2.48. Форма после задания свойств объектов

6. Введите названия команд контекстного меню. Для этого дважды щелкните по объекту *PopupMenu* (или по его свойству *Items*). В открывшемся окне (*Form1.PopupMenu1*) выбирайте с помощью мыши рамку команды и записывайте название команды (***Вывести, Очистить, Закрыть***) как значение свойства *Caption* в окне *Object Inspector (рис. 2.49)*. Можете придумать любую свою команду.

Так как программный код команд контекстного меню *(Popup Menu)* такой же, как и программный код команд главного меню *(Main Menu)*, воспользуйтесь командами копирования и вставки (***Edit → Copy, Edit → Paste***).

После введения всех команд контекстного меню закройте окно *Form1.PopupMenu1*.

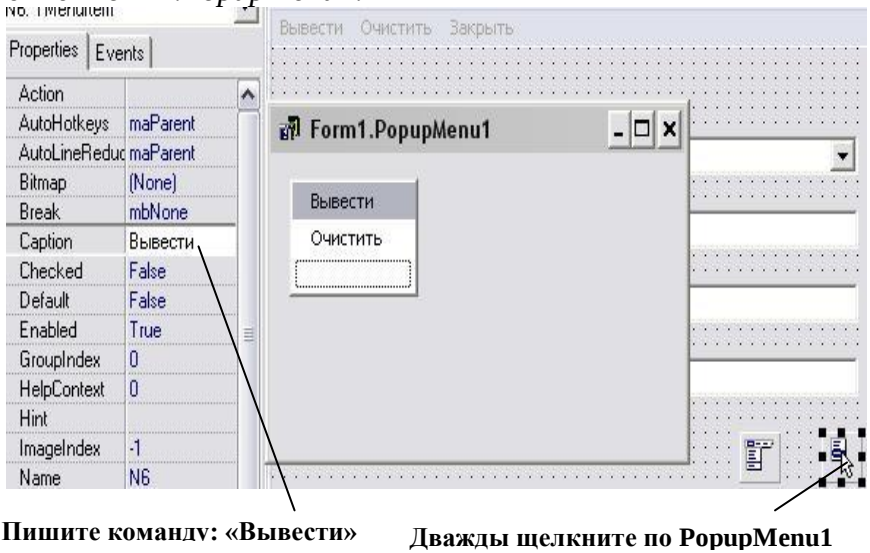

Рис. 2.49. Задание команд контекстного меню

7. Выполните программу (*Run → Run*). Проверьте действие контекстного меню полей выведения. Запустите программу на выполнение. Щелкните правой кнопкой мыши по каждому из полей выведения (*Edit1*, *Edit2* или *Edit3*). Раскроется список команд контекстного меню *(рис. 2.50)*.

8. Сохраните проект в своей папке *(File → Save All).*

9. Запрограммируйте команду главного меню «*Закрыть*». Дважды щелкните по названию команды «*Закрыть*» в главном меню. В появившейся заготовке процедуры напишите *close.*

```
procedure TForm1. N5Click(Sender: TObject);
begin
    close
end;
```

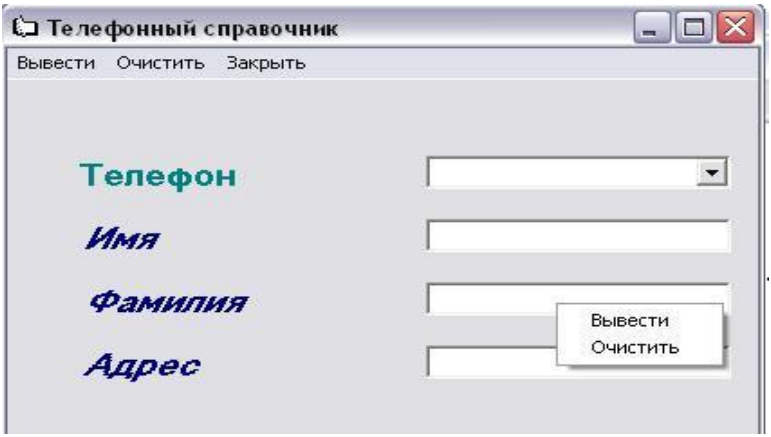

Рис. 2.50. Контекстное меню полей выведения

10. Запрограммируйте команду «Очистить все» главного меню. Дважды щелкните по названию команды «Очистить все» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N3Click(Sender: TObject);
begin
    combobox1.itemindex:= -1;
    edit1.clear; edit2.clear; edit3.clear;
end;
```

11. Запрограммируйте команду «Очистить поля выведения» главного меню. Дважды щелкните по названию команды «Очистить поля выведения» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N4Click(Sender: TObject);
begin
    edit1.clear; edit2.clear; edit3.clear;
end;
```

12. Запрограммируйте команду «Вывести» главного меню. Дважды щелкните по названию команды «Вывести» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N1Click(Sender: TObject);
begin
    case combobox1.itemindex of
        0:  begin
              edit1.text:='Петров'; edit2.text:='Андрей';
              edit3.text:='ул. Жилянская, д.34, кв. 56';
            end;
        1:  begin
              edit1.text:='Волков'; edit2.text:='Дмитрий';
              edit3.text:='ул. Луговая, д.125, кв. 238';
            end;
        2:  begin
              edit1.text:='Ларина'; edit2.text:='Ирина';
              edit3.text:='ул. Космонавтов, д.7, кв. 119';
            end;
      end;
end;
```

13. Аналогично запрограммируйте команды «Вывести» и «Очистить» контекстного меню. Дважды щелкните по объекту ***PopupMenu***, а затем по названию команды в открывшемся окне. В появившейся заготовке процедуры введите программный код.

14. Еще раз сохраните проект *(File → Save).*

15. Выполните проект *(Run → Run)*, убедитесь в действии главного и контекстного меню.

16. **Создайте выполняемый файл *(Project → Build Project1).***

Дополнительные задания

1. Создайте проект для вычисления значения функции

$$y = \begin{cases} 2x+1, & \text{если } x > 4; \\ 2x-1, & \text{если } x \le 4. \end{cases}$$

Поле выведения значений *y* сделайте недоступным для изменения значений пользователем. Для поля введения *x* сделайте *подсказку* с текстом *«Введите значение x»*. Создайте *контекстное меню* для *поля выведения y*. Контекстное поле должно содержать команды *«Вычислить»* и *«Очистить»*.

Указание. Разместите объекты на форме *(рис. 2.51)*:

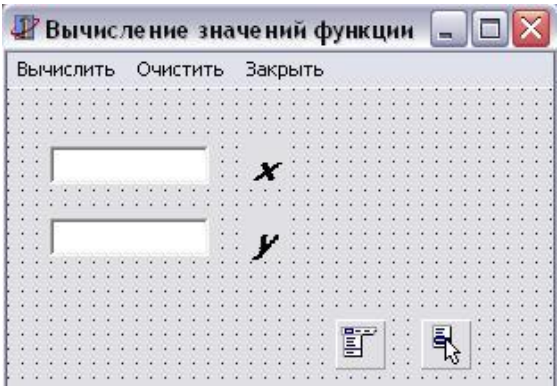

Рис. 2.51. Размещение объектов на форме

Программный код для команды «Вычислить» будет таким:

```
procedure TForm1. N1Click(Sender: TObject);
    var x,y:real;code:integer;s1:string;
begin
    val(edit1.text, x, code);
    if x>4 then y:=2*x+1 else y:=2*x-1;
    str(y:9:2,s1); edit2.text:=s1;
end;
```

2. Создайте проект для вычисления значений функции $y = \frac{x+7}{x-2} + \frac{5}{x}$. *Указание.* Программный код для команды «Вычислить» будет таким:

```
procedure TForm1. N1Click(Sender: TObject);
    var x,y:real;code:integer;s1:string;
begin
    val(edit1.text,x,code);
    if (x=2) or (x=0) then edit2.text:= ‘не существует’
    else
        begin
          y:= (x + 7)/(x – 2)+5/x;
          str(y:9:2,s1); edit2.text:= s1;
        end;
end;
```

3. Создайте проект для решения квадратного уравнения $ax^2 + bx + c = 0, \quad a \neq 0$ по формулам: $D = b^2 - 4ac$;

$x_1 = \frac{-b - \sqrt{D}}{2a}$, $x_2 = \frac{-b + \sqrt{D}}{2a}$.

4. Создайте проект для выведения названия месяца и фотографии этого времени года при выборе его номера из заданного списка *(рис. 2.52)*:

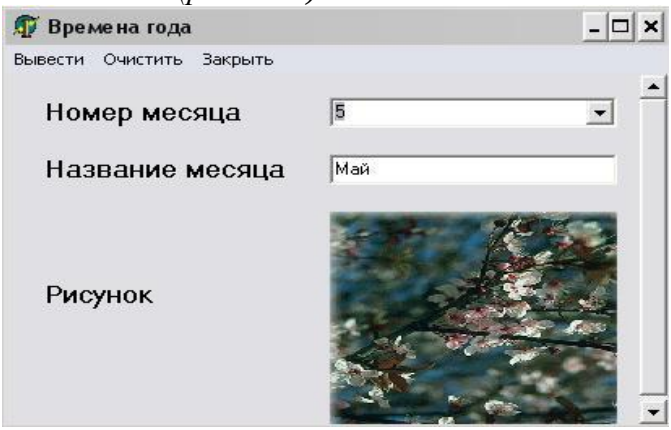

Рис. 2.52. Примерное размещение объектов на форме

5. Выполните предыдущее задание с использованием оператора **if.**

6. Создайте проект для выведения названия дня недели при выборе его номера из заданного списка.

ВОПРОСЫ

1. Что такое логическое выражение?
2. Какими бывают логические выражения?
3. Какое логическое выражение называется простым, какое - составным?
4. Какие две разновидности имеет команда ветвления?
5. Какие вы знаете свойства объекта MainMenu?
6. Какими свойствами обладают объекты PopupMenu и Combobox?
7. На какой закладке палитры компонентов находятся объекты MainMenu, PopupMenu, Combobox?

ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ №5

Задание 1. Создайте проект, который вычисляет

Вариант 1	площадь *S* треугольника, если известны его сторона *a* и соответствующая ей высота *h*. Формула: $S = \frac{1}{2}ah$.
Вариант 2	объем *V* призмы, если известны площадь ее основания *S* и высота *H*. Формула: $V = SH$.
Вариант 3	объем *V* пирамиды, если известны площадь ее основания *S* и высота *H*. Формула: $V = \frac{1}{3}SH$.
Вариант 4	площадь *S* трапеции, если известны ее основания *a* и *b* и высота *h*. Формула: $S = \frac{a+b}{2} \cdot h$.
Вариант 5	площадь *S* треугольника, если известны его стороны *a* и *b* и угол α между ними. Формула: $S = \frac{1}{2}ab\sin\alpha$.
Вариант 6	объем *V* прямоугольного параллелепипеда, если известны его стороны *a, b* и *c* . Формула: $V = abc$.
Вариант 7	высоту *H* пирамиды, если известны площадь ее основания *S* и объем *V*. Формула: $H = \frac{3V}{S}$.
Вариант 8	площадь *S* четырехугольника, если известны его диагонали d_1 и d_2 и угол между ними α. Формула: $S = d_1 \cdot d_2 \cdot \sin\alpha$.
Вариант 9	площадь *S* прямоугольного треугольника, если известны его стороны *a* и *b*. Формула: $S = \frac{1}{2}ab$.
Вариант 10	объем *V* конуса, если известны радиус его основания *R*, число π и высота *H*. Формула: $V = \frac{1}{3}\pi R^2 H$.
Вариант 11	объем *V* куба, если известна его сторона *a*. Формула: $V = a^3$.

Вариант 12	высоту h трапеции, если известны ее основания a и b и площадь S. Формула: $h = \frac{2S}{a+b}$.
Вариант 13	сторону a треугольника, если известны его сторона b, площадь S угол α между этими сторонами. Формула: $a = \frac{2S}{b\sin\alpha}$.
Вариант 14	сторону c прямоугольного параллелепипеда, если известны его стороны a, b и объем V. Формула: $c = \frac{V}{ab}$.
Вариант 15	площадь S пирамиды, если известны его объем V и высота H. Формула: $S = \frac{3V}{H}$.
Вариант 16	диагональ d_1 четырехугольника, если известны его диагональ d_2, площадь S и угол между диагоналями α. Формула: $d_1 = \frac{S}{d_2 \sin\alpha}$.
Вариант 17	сторону a треугольника, если известны соответствующая ей высота h площадь треугольника S. Формула: $a = \frac{2S}{h}$.
Вариант 18	площадь основания S призмы, если известны ее объем V и высота H. Формула: $S = \frac{V}{H}$.
Вариант 19	высоту H конуса, если известны радиус его основания R, число π и объем V. Формула: $H = \frac{3V}{\pi R^2}$.
Вариант 20	основание a трапеции, если известны ее основание b, высота h и площадь S. Формула: $a = \frac{2S}{h} - b$.
Вариант 21	сторону a куба, если известен его объем V. Формула: $a = \sqrt[3]{V}$.
Вариант 22	высоту H призмы, если известны ее объем V и площадь основания S. Формула: $H = \frac{V}{S}$.

Вариант 23	объем V шара, если известен его радиус R и число π. Формула: $V = \frac{4}{3}\pi R^3$.
Вариант 24	площадь боковой поверхности конуса S, если известны радиус основания конуса R, образующая l и число π. Формула: $S = \pi Rl$.
Вариант 25	сторону b треугольника, если известны его сторона a, площадь S угол α между этими сторонами. Формула: $b = \frac{2S}{a\sin\alpha}$.
Вариант 26	радиус основания R конуса, если известны его высота H, число π и объем V. Формула: $R = \sqrt{\frac{3V}{\pi H}}$.
Вариант 27	высоту h треугольника, если известны его площадь S и соответствующая ей сторона a. Формула: $h = \frac{2S}{a}$.
Вариант 28	основание b трапеции, если известны ее основание a, высота h и площадь S. Формула: $b = \frac{2S}{h} - a$.
Вариант 29	радиус шара R, если известны его объем V и число π. Формула: $R = \sqrt[3]{\frac{3V}{4\pi}}$
Вариант 30	площадь сферы S, если известны ее радиус R и число π. Формула: $S = 4\pi R^2$

Задание 2. Создайте проект для вычисления значений функции:

Вариант 1	$y = \begin{cases} 3x^2 - 4x + 5, \text{ если } x \ge 6; \\ 2x^2 + 3x - 6, \text{ если } x < 6. \end{cases}$
Вариант 2	$y = \frac{x+8}{x-6} + \frac{x-4}{x+2}$
Вариант 3	$z = \begin{cases} 4x^3 + 2x + 2, \text{ если } x \ge 8; \\ 8x^2 + x - 1, \text{ если } x < 8. \end{cases}$

Вариант 4	$y=\frac{x+2}{x-6}+\frac{5-x}{x+8}$
Вариант 5	$y=\begin{cases}4z^2-7z+1, \text{ если } z\ge 1;\\ z^2+3z-3, \text{ если } z<1.\end{cases}$
Вариант 6	$y=\frac{x+1}{x+7}+\frac{x-4}{x-9}-3$
Вариант 7	$z=\begin{cases}8x^3+x-4, \text{ если } x\ge -1;\\ 5x^3+6x-9, \text{ если } x<-1.\end{cases}$
Вариант 8	$y=\frac{x+2}{x-6}+\frac{5-x}{x+8}$
Вариант 9	$y=\begin{cases}2x^2-6x+4, \text{ если } x\ge -7;\\ x^2+9x-6, \text{ если } x<-7.\end{cases}$
Вариант 10	$y=\frac{x}{x-7}+\frac{3+x}{x-1}-\frac{3}{x}$
Вариант 11	$y=\begin{cases}8x^3-3x+1, \text{ если } x\ge -90;\\ 6x^2+x-3, \text{ если } x<-90.\end{cases}$
Вариант 12	$y=\frac{x-6}{x-4}+\frac{x}{3}-\frac{5}{x}$
Вариант 13	$y=\begin{cases}4z^2-7z+1, \text{ если } z\ge 1;\\ z^2+3z-3, \text{ если } z<1.\end{cases}$
Вариант 14	$y=\frac{x}{x-4}+\frac{x}{3}-x$
Вариант 15	$z=\begin{cases}8x^3+x-4, \text{ если } x\ge -1;\\ 5x^3+6x-9, \text{ если } x<-1.\end{cases}$
Вариант 16	$y=\frac{x}{x-1}+\frac{x}{5}-\frac{5}{x}$
Вариант 17	$y=\begin{cases}3x^2-4x+5, \text{ если } x\ge 6;\\ 2x^2+3x-6, \text{ если } x<6.\end{cases}$
Вариант 18	$y=\frac{x-9}{x+2}-\frac{5}{x}+4x$
Вариант 19	$z=\begin{cases}4x^3+2x+2, \text{ если } x\ge 8;\\ 8x^2+x-1, \text{ если } x<8.\end{cases}$
Вариант 20	$y=\frac{x-6}{x}+\frac{x}{x-6}$

Вариант 21	$y=\begin{cases}x^2-x+5, \text{ если } x\ge 6;\\ 5x^2+2x-9, \text{ если } x<6.\end{cases}$
Вариант 22	$y=\frac{x-6}{x-4}+\frac{x}{4}-\frac{x-4}{x-6}$
Вариант 23	$z=\begin{cases}x^3+5x+2, \text{ если } x\ge -8;\\ 8x^2+8x-1, \text{ если } x<-8.\end{cases}$
Вариант 24	$y=\frac{x}{7}-\frac{7}{x}+x^2$
Вариант 25	$z=\begin{cases}9y^2-y+3, \text{ если } y\ge -2;\\ 4y^2+y, \text{ если } x<-2.\end{cases}$
Вариант 26	$y=\begin{cases}4x^2-x, \text{ если } x\ge 9;\\ 9x^2+2x-3, \text{ если } x<9.\end{cases}$
Вариант 27	$y=\begin{cases}8x^3-x+5, \text{если } x\ge 5;\\ x^2+4x-1, \text{если } x<5.\end{cases}$
Вариант 28	$z=\begin{cases}2y^2-y+1, \text{если } y\ge -5;\\ 6y^2+6y-1, \text{если } x<-5.\end{cases}$
Вариант 29	$z=\begin{cases}y^2-5y+5, \text{если } y\ge -9;\\ 7y^2+5y-7, \text{если } x<-9.\end{cases}$
Вариант 30	$z=\begin{cases}y^2-5y, \text{если } y\ge -3;\\ 9y^2+y-3, \text{если } x<-3.\end{cases}$

Задание 3. Создайте проект для вычисления значений функции:

Вариант 1	$y=4{,}5\sin^2 x-\lvert\cos x/1{,}6\rvert$
Вариант 2	$y=4{,}8\lvert\cos x/5{,}6\rvert\sin(1{,}5x^2+3)-1$
Вариант 3	$y=5{,}1\sin\lvert x/2\rvert\sin(x^3+8)+2$
Вариант 4	$y=\sin(x+6)\lvert\cos(x/3)\rvert+9x^2+32$
Вариант 5	$y=2\sin x^3\lvert\cos(x-3)\rvert+6x^3+4x-7$
Вариант 6	$y=5\cos^2 x\lvert\sin(x-3)+x\rvert+x^4+9x-1$
Вариант 7	$y=\cos^2 x\sin\lvert x-5\rvert+x^3+7x+4$
Вариант 8	$y=8\cos^4 2x\lvert\sin(6x-10)\rvert-3x^5+9x-1$

Вариант 9	$y = 5\sin^2 x\left\|tg(x-2)\right\| + 4x + 3$
Вариант 10	$y = 7tg^2 x\left\|\ln(x-3) + 5\right\| + x^2 + 9\sin x - 1$
Вариант 11	$y = 3\ln^2 x\left\|arctg(x/8)\right\| + x^4 + 4x - 2$
Вариант 12	$y = 1{,}7\sin^2 x\left\|ctg(4x-9) + 2x\right\| + 8x^3 + 3x - 5$
Вариант 13	$y = 4\cos x + 2ctg(x^2 - 5) - 3x$
Вариант 14	$y = (2x^3 + x - 7)^2 + \sin^2 x - 1/x$
Вариант 15	$y = 2arcctgx + 3x^2 - 4\sqrt{x+2}$
Вариант 16	$y = \dfrac{2\left\|x+7\right\|}{\sin^3(x-3)} + x^4 - 3$
Вариант 17	$y = \sqrt[3]{x-5} + \ln(\left\|x-1\right\| + 2x^2)$
Вариант 18	$y = \sqrt{\dfrac{x^2 - 5x + -7}{\sin x}} + ctg^3(x-7)$
Вариант 19	$y = 5\ln\left\|2x-4\right\| + 3\cos^4(5x-3)$
Вариант 20	$y = 2\sqrt{3x-4} + tg^2 x - 2x^2 + 4$
Вариант 21	$y = \dfrac{\sqrt{4x^2 - 2x - 3}}{\left\|5x-6\right\|} + \sin x$
Вариант 22	$y = 4x^3 - \sin x + \ln(x-2)^2$
Вариант 23	$y = \sqrt{4x\sin x\cos(x-8) + 5} + \cos^2 x$
Вариант 24	$y = \dfrac{5x^3 - 4x + 7}{\sin(x-3)^2} + tg^2 x$
Вариант 25	$y = \sqrt{2x} - \dfrac{x-5}{x+7} + \cos x$
Вариант 26	$y = \sqrt{3x^2 - x + 8} + x^2 - \dfrac{2x-5}{3x+8}$
Вариант 27	$y = \dfrac{3x-7}{2x-9} + \ln^2(x-9) - 7$
Вариант 28	$y = 3x^2 - 4\cos x + 2x\ln(x+6)^3$
Вариант 29	$y = \sqrt{2x-5} + \dfrac{1}{x} - 3x^2$
Вариант 30	$y = \dfrac{4x-5}{x}\cos(x-4)^2 + 6x - 1$

2.6. Программирование циклов

Новые слова

Русский язык	***English***
альтернативный	alternative
отрезок	cut, section
переключатель	selector, switch, toggle, radio button
предыдущий	previous

Цикл – это процесс выполнения некоторого набора команд некоторое количество раз. Цикл реализуют или с помощью конструкции **if – goto** или с помощью команд цикла. Есть три вида команд циклов: *с параметром, с передусловием* и *с послеусловием*.

1. **Команда цикла с параметром (for).** Есть два вида команды **for**. ***Первый***:

for <параметр>: = <выражение1> **to** <выражение2> **do** <команда1>;

Здесь параметр - это переменная целого, символьного, логического или перечисленного типа, а выражения 1 и 2 задают начальное и конечное значение параметра.

Действие команды. Параметру цикла присваивается значение выражения 1. Если это значение меньше или равно значению выражения 2, то выполняется команда 1. После выполнения команды 1 значение параметра автоматически увеличивается на 1 и снова сравнивается со значением выражения 2 и т.д. Когда значение параметра станет больше значения выражения 2, то выполняется следующая после цикла команда.

Пример. Пусть s = 0, d = 1. После выполнения команды цикла ***for*** *i : = 4* ***to*** *6* ***do begin*** *s : = s + i; d : = d*i* ***end;*** переменная s примет значение 0 + 4 + 5 + 6 = 15, а переменная $d = 1 \cdot 4 \cdot 5 \cdot 6 = 120$.

Второй вид команды цикла for:

for<параметр>:=<выражение1>**downto**<выражение2>**do**
<команда1>;

Эта команда действует как предыдущая, но шаг изменения параметра равен -1.

Пример. Пусть s = 0, d = 1. После выполнения команды цикла

for *i : = 7* ***downto*** *5* ***do begin*** *s : = s + i; d : = d*i* ***end;***

переменная s примет значение 0 + 7 + 6 + 5 = 18, а переменная $d = 1 \cdot 7 \cdot 6 \cdot 5 = 210$.

Примечание. Менять значение параметра посреди цикла нельзя.

2. **Команда цикла с предусловием (while)** имеет вид:

while < логическое выражение > **do** < команда 1 >;

Действие команды. Пока значение логического выражения истинно, выполняется команда 1. Истинное логическое выражение описывает условие продолжения выполнения команды цикла.

Пример. Пусть s = 0, i = 1, d = 1. После выполнения команды цикла

while *i <= 4* ***do begin*** *s : = s + i; d : = d*i* ***end;***

переменная s примет значение 0 + 1 + 2 + 3 + 4 = 10, а переменная $d = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

3. **Команда цикла с послеусловием (repeat)** имеет вид:

repeat < команды > **until** < логическое выражение >;

Действие команды. Команды выполняются в цикле, пока значение логического выражения не станет истинным. Истинное логическое выражение задает условие выхода из цикла.

Пример. Пусть s = 0, x = 1, d = 1. После выполнения команды цикла ***repeat*** *s : = s + i; d : = d*i* ***until*** *x > 5;* переменная s примет значение 0 + 1 + 2 + 3 + 4 + 5 = 15, а переменная $d = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$.

ПРАКТИЧЕСКАЯ РАБОТА №6

Тема. ***Создание проекта «Вычисление суммы и произведения целых чисел из заданного отрезка [a;b]».***

Цель. Получение навыков программирования циклов (в частности команды цикла с параметром **for**) в среде Delphi. Познакомиться с таким новым объектом: переключатель (**RadioButton**) и его основными свойствами.

Объекты: форма, текстовое поле, поле редактирования, главное меню, контекстное меню (привязанное к полю выведения), переключатель.

Теоретические сведения. Объект **Radiobutton** используют для создания в форме средства для выбора одной альтернативной возможности среди некоторых. Рассмотрим свойства переключателя **Radiobutton**:

Свойство	Описание свойства	Примеры значений
Checked	Состояние переключателя	True (выбран), False (не выбран)
TabOrder	Порядок выбора объекта клавишей **Tab**	0 (первый), 4 (пятый)
TabStop	Доступ к данному объекту табулятором	True (будет доступным), False (не будет)

Объект **Radiobutton** находится на закладке ***Standard*** палитры элементов *(рис. 2.53).*

Рис. 2.53. Объект Radiobutton

Ход работы

1. **Загрузите программу Delphi.**
2. **Разместите объекты на форме (*рис. 2.54)*:**

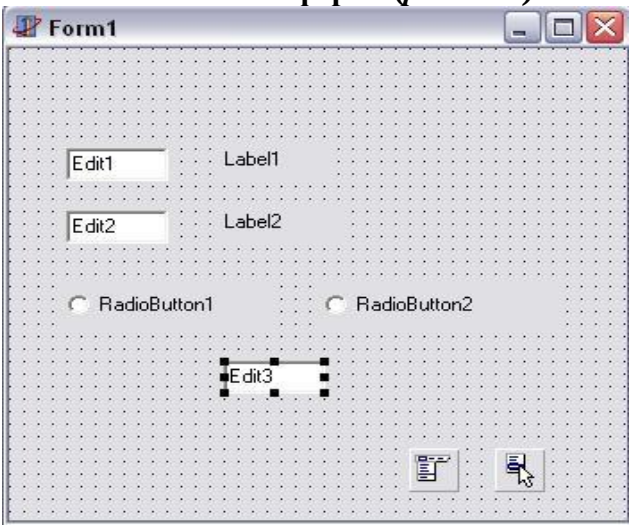

Рис. 2.54. Размещение объектов на форме

3. **Сохраните проект в своей папке *(File → Save All)*.**
4. **Задайте значения свойствам объектов:**

Объект **Form**

Caption: Сумма и произведение чисел

Объект **Label1**

Caption: *a*

Объект **Label2**

Caption: *b*

Объект **Edit1**

Очистите поле ***Text*** ***ShowHint:*** True
Hint: Введите левый конец отрезка ***ReadOnly:*** False

Объект **Edit2**

Очистите поле ***Text*** ***ShowHint:*** True
Hint: Введите правый конец отрезка ***ReadOnly:*** False
PopupMenu: PopupMenu1

Объект **Edit3**

Очистите поле ***Text*** ***ReadOnly:*** True

PopupMenu: PopupMenu1

Объект **Radiobutton1**

Checked: True ***Caption:*** Сумма

Объект **Radiobutton2**

Checked: False ***Caption:*** Произведение

Объект **MainMenu1**

Captions: Вычислить

Очистить (Очистить все, Очистить поля выведения)

Закрыть

Объект **PopupMenu1**

Captions: Вычислить

Очистить

Закрыть

После задания значений свойствам объектов форма примет вид *(рис. 2.55)*:

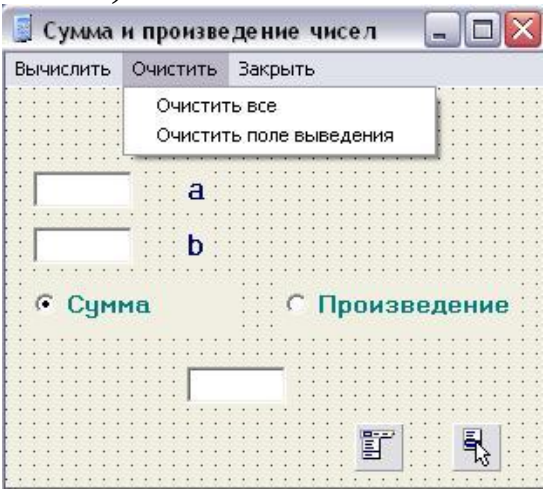

Рис. 2.55. Форма после задания свойств объектов

5. Запрограммируйте команду главного меню «Закрыть». Дважды щелкните по названию команды «Закрыть» в главном меню. В появившейся заготовке процедуры напишите *close.*

```
procedure TForm1. N5Click(Sender: TObject);
begin
    close
end;
```

6. Запрограммируйте команду «Очистить все» главного меню. Дважды щелкните по названию команды «Очистить все» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N3Click(Sender: TObject);
begin
    edit1.clear; edit2.clear; edit3.clear;
end;
```

7. Запрограммируйте команду «Очистить поле выведения» главного меню. Дважды щелкните по названию команды «Очистить поле выведения» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N4Click(Sender: TObject);
begin
    edit3.clear;
end;
```

8. Запрограммируйте команду «Вычислить» главного меню. Дважды щелкните по названию команды «Вычислить» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N1Click(Sender: TObject);
    var a,b,code,s,d,i:integer;s1,s2:string;
begin
    val(edit1.text,a,code); val(edit2.text,b,code);
    s:=0; d:=1;
    for i:=a to b do
          begin s:=s+i;d:=d*i; end;
      str(s,s1); str(d,s2);
      if radiobutton1.checked then edit3.text:=s1
      else edit3.text:=s2;
end;
```

9. Аналогично запрограммируйте команды «Вывести» и «Очистить» контекстного меню. Дважды щелкните по объекту PopupMenu, а затем по названию команды в открывшемся окне. В появившейся заготовке процедуры введите программный код.

10. Еще раз сохраните проект (File → Save).

11. Выполните проект (Run → Run), убедитесь в действии главного и контекстного меню.

12. Создайте выполняемый файл (Project → Build Project1).

Дополнительные задания

1. Выполните предыдущее задание, используя команду цикла с предусловием **while**.
2. Выполните предыдущее задание, используя команду цикла с послеусловием **repeat**.
3. Создайте проект для вычисления количества целых чисел с отрезка [11; 5678], которые делятся на 3.

Указание. Свойству **Text** объекта **Edit1** задайте значение 11, свойству **Text** объекта **Edit2** задайте значение 5678. Программный код для команды «Вычислить» будет таким:

```
procedure TForm1. N1Click(Sender: TObject);
    var a,b,code,n,i:integer;s1:string;
begin
    val(edit1.text,a,code); val(edit2.text,b,code);
    n:=0;i:=12;
    while i<=b do
      begin
          n:=n+1;i:= i+3;
      end;
      str (n,s1); edit3.text:= s1;
end;
```

Переключатели в этой программе не нужны. *n* – это количество искомых чисел. Здесь нужно использовать оператор **while**, так как значение промежуточной переменной i должно меняться не на 1, а на 3. Начальное

значение i принимаем 12, так как ***12 – это первое число из отрезка [11; 5678], которое делится на 3.***

4. Создайте проект для вычисления суммы целых чисел из отрезка [47; 981], которые делятся на 5.
5. Создайте проект для вычисления произведения целых чисел из отрезка [5; 34], которые делятся на 7.
6. Создайте проект для вычисления количества размещений из n по k. Формула: $A_n^k = \frac{n!}{(n-k)!}$, используйте подпрограмму-функцию для вычисления факториала ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$).
7. Создайте проект для вычисления количества комбинаций из n по k. Формула: $C_n^k = \frac{n!}{(n-k)!k!}$, используйте подпрограмму-функцию для вычисления факториала ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$).
8. Создайте проект для вычисления значения выражения $m = 2n! + \frac{n!}{3(n-3)!}$, используйте подпрограмму-функцию для вычисления факториала ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$).
9. Создайте проект для вычисления значения выражения $k = 5(p-2)! + \frac{p!}{7(p-1)!}$, используйте подпрограмму-функцию для вычисления факториала ($p! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot p$).
10. Создайте проект для вычисления значения выражения $h = 4(k+5)! + \frac{5k!}{(k-1)!} - 3(k! + t!)(t+2)!$, используйте подпрограммы-функции для вычисления факториалов ($k! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot k$, $t! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot t$).

2.7. Работа с массивами

Новые слова

Русский язык	***English***
двумерный	flat, two-dimensional
единственный	singular, only, sole, just one
индекс	index
конечный	finite
массив	array, massif
многомерный	many-measure, multidimensional
одномерный	one-dimensional
пересечение	cross, crossing, crossover, intersection
привязанный	instance, attached, binding, associating, snap
прокручивать	scroll
размерность	dimension, dimensionality
слева	left
столбец	column
строка	road
таблица	table
элемент	element
ячейка	cell

Массив (array) – это конечный набор элементов одного (базового) типа, которые сохраняются в последовательно размещенных ячейках оперативной памяти и имеют общее название. Различают **одномерные** и **многомерные** массивы.

Пример одномерного массива *(рис. 2.56):*

0	1	2	3	4	5	6	7	8	9	10
45	30	21	65	32	754	8	3	67	98	5

Рис. 2.56. Одномерный массив

Пусть **A[0..10]** – имя и размерность массива (количество его элементов). Элементы одномерного массива определяются именем массива и номером (который еще называют ***индексом***) элемента. *A*[0] – *нулевой* элемент массива, *a*[1] – *первый* элемент массива и т.д. В нашем примере *a*[0] = 45, *a*[1] = 30, *a*[4] = 32 и т.д.

Двумерный массив данных *(рис. 2.57)* – это таблица, которая состоит из нескольких строк. ***Элементы двумерного***

массива определяются *именем массива* и двумя *индексами* (номерами): первый индекс обозначает номер строки, а второй – номер столбца, на пересечении которых стоит элемент, *например*, *p*[1,2], *cdv*[i, j].

Пример двумерного массива:

	0	1	2	3	4	5	
0	23	876	9	67	9	65	строка
1	6	5	7	34	567	32	
2	76	0	2	76	34	8	столбец
3	45	97	3	6	4	90	

ячейка (элемент массива)

Рис. 2.57. Таблица (двумерный массив)

Пусть ***b*[0..5, 0..3]** – имя и размерность массива (количество его элементов). В нашем примере *b[0,0]* = 23, *b[1,0]* = 876, *b[1,3]* = 97, *b[2,1]* = 7, *b[3,1]* = 34 и т.д.

Общий вид конструкции описания массива :

array [<размерность>] **of** <название базового типа>;

Описать массив можно в разделе описания типов **type**, в разделе констант **const** или в разделе объявления переменных **var**. Названия типов массивов и переменных-массивов задает пользователь.

Например, рассмотрим:

1) описание типа массива (название типа *abcd*):

Type *abcd = array [0..10] of real;*

2) объявление постоянного массива (массива-константы) *massiv1* типа *abcd:*

Const *massiv1: abcd = (5, 3, 7, 4, 8, 90.4, 23, 1, 0.6, 2);*

3) объявление переменных-массивов *a1* и *a2* целого типа, а также массива *d* вещественного типа:

Var *a1, a2: integer;*

d: real;

Над массивами определена единственная команда *присваивания*: *a:=a*1 – все значения массива *a*1 будут присвоены соответственным элементам массива *a*. Все

остальные операции, например, присваивание конкретных значений, сложение, умножение и т.д. определены только над *элементами* массивов. Чтобы обработать все элементы массива, используют команды цикла: ***for, while*** или ***repeat***.

Пример. Создайте массив из первых ста целых чисел и вычислите сумму и произведение всех его элементов.

Решение (фрагмент программы)

```
s := 0 ; d: = 1;
for  i:=1 to 100 do
    begin
      a[i]:= i;  s:= s + a[i]; d:=d * a[i];
    end;
```

ПРАКТИЧЕСКАЯ РАБОТА №7

Тема. ***Создание проекта «Вычисление количества элементов массива b[0..3,0..2], больших заданного числа a»*** *(рис. 2.58).*

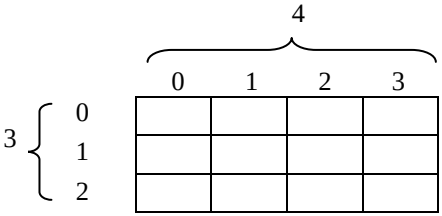

Рис. 2.58. Массив *a*[0..3,0..2]

Цель. Получение навыков составления программ с использованием массивов. Ознакомление с новым объектом: **таблица текстовых строк** (**StringGrid**) и его основными свойствами.

Объекты: форма, текстовое поле, поле редактирования, главное меню, контекстное меню (привязанное к полю выведения), таблица текстовых строк.

Теоретические сведения. Объект **StringGrid** используют для создания в форме двумерной таблицы символьных строк. Кроме рассмотренных раньше, 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 заглавии таблицы, которые не прокручиваются вверх	1; 2

Объект **StringGrid** находится на закладке ***Additional*** палитры элементов *(рис. 2.59).*

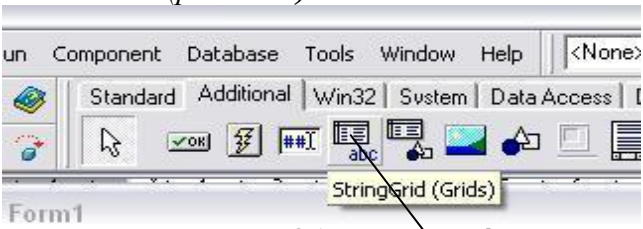

Рис. 2.59. Объект StringGrid

Ход работы

1. **Загрузите среду программирования Delphi.**
2. **Разместите объекты на форме (*рис. 2.60):***

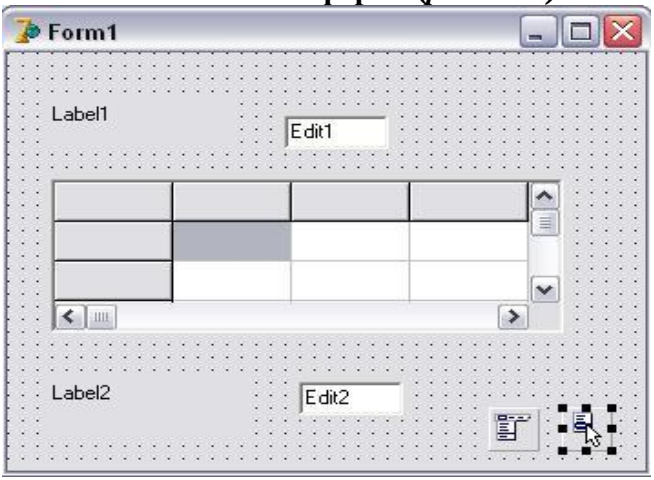

Рис. 2.60. Размещение объектов на форме

3. Задайте значения свойствам объектов *(рис. 2.61)*:

Объект **Form**

Caption: Таблица

Объект **Label1**

Caption: Число a

Объект **Label2**

Caption: Количество чисел n

Объект **Edit1**

Очистите поле ***Text*** ***ShowHint:*** True
Hint: Введите число

Объект **Edit2**

Очистите поле ***Text*** ***PopupMenu:*** PopupMenu1
Hint: Количество чисел ***ReadOnly:*** True
ShowHint: True

Объект **StringGrid1**

ColCount: 4 ***FixedRows:*** 0
RowCount: 3 ***Options (goEditing):*** True
FixedCols: 0

Объект **MainMenu1**

Captions: Вычислить, Очистить (Очистить все, Очистить поле выведения), Закрыть

Объект **PopupMenu1**

Captions: Вычислить, Очистить

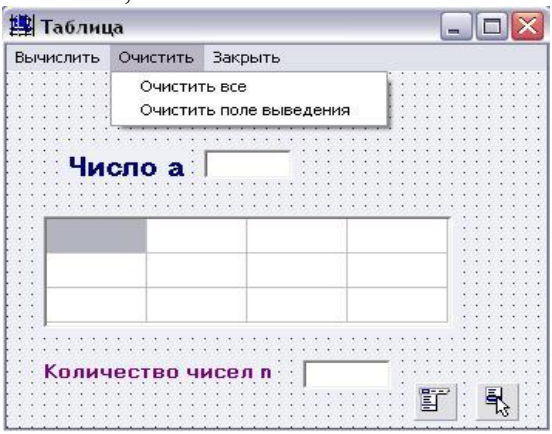

Рис. 2.61. Форма после задания свойств объектов

4. Сохраните проект в своей папке (File → Save All).

5. Запрограммируйте команду главного меню «Закрыть». Дважды щелкните по названию команды «Закрыть» в главном меню. В появившейся заготовке процедуры напишите *close*.

```
procedure TForm1. N5Click(Sender: TObject);
begin
    close
end;
```

6. Запрограммируйте команду «Очистить все» главного меню. Дважды щелкните по названию команды «Очистить все» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N3Click(Sender: TObject);
      var  i,j:integer;
begin
    edit1.clear; edit2.clear;
    for i:=0 to 3 do
        for j:=0 to 2 do
          stringgrid1.cells[i,j]:=' ';
end;
```

7. Запрограммируйте команду «Очистить поле выведения» главного меню. Дважды щелкните по названию команды «Очистить поле выведения» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N4Click(Sender: TObject);
begin
    edit2.clear;
end;
```

8. Запрограммируйте команду «Вычислить» главного меню. Дважды щелкните по названию команды «Вычислить» в главном меню. В появившейся заготовке процедуры напишите:

```
procedure TForm1. N1Click(Sender: TObject);
    var b:array[0..3,0..2] of integer;
        a,code,n,i,j:integer;s1:string;
begin
    val (edit1.text,a,code);n:= 0;
    for i:=0 to 3 do
        for j:=0 to 2 do
      begin
          val(stringgrid1.cells[i,j],b[i,j],code);
           if  b[i, j]>a then n:=n+1;
      end;
      str (n,s1); edit2.text:= s1;
end;
```

9. Аналогично запрограммируйте команды «Вывести» и «Очистить» контекстного меню. Дважды щелкните по объекту PopupMenu, а затем по названию команды в открывшемся окне. В появившейся заготовке процедуры введите программный код.

10. Еще раз сохраните проект (File → Save).

11. Выполните проект (Run → Run), убедитесь в действии главного и контекстного меню.

12. Создайте выполняемый файл (Project → Build Project1).

Дополнительные задания

1. Выполните предыдущее задание, используя команду цикла с предусловием **while**.
2. Выполните предыдущее задание, используя команду цикла с послеусловием **repeat**.
3. Создайте проект для вычисления произведения элементов массива k[0..2,0..1] целых чисел, равных заданному числу m.

Указание. Программный код для команды «Вычислить» может быть таким:

```
procedure TForm1. N1Click(Sender: TObject);
    var k: array[0..2,0..1] of integer;
        m,code,d,i,j:integer;s1:string;
begin
    val (edit1.text,m,code);d:= 1;
    for i:=0 to 2 do
        for j:=0 to 1 do
      begin
          val(stringgrid1.cells[i, j],k[i,j],code);
           if  k[i,j]=m then d:=d*k[i,j];
      end;
      str(d,s1); edit2.text:= s1;
end;
```

4. Создайте проект для вычисления суммы элементов массива d[1..4, 1..2] вещественных чисел, меньших заданного числа t.
5. Создайте проект для вычисления суммы, произведения и количества элементов массива g[1..5, 1..3] целых чисел, меньших заданного числа f.
6. Создайте проект для вычисления суммы элементов главной диагонали массива h[1..5, 1..5] целых чисел (***элементы главной диагонали*** *– это элементы, у которых номер столбца равняется номеру строки*).
7. Создайте проект для вычисления произведения элементов главной диагонали массива h[1..4, 1..4], меньших числа 5.
8. Создайте проект для вычисления количества элементов главной диагонали массива h[1..3, 1..3], равных числу 0.
9. Создайте проект для вычисления суммы элементов главной диагонали массива g[1..3, 1..4], больших числа 7.
10. Создайте проект для вычисления количества элементов главной диагонали массива h[1..3, 1..3], равных числу 0.

2.8. Работа со строками

Кроме рассмотренных на странице 246, существуют и другие процедуры и функции для работы со строками.

1. **Функция Concat**

Формат: **Concat** (R1, R2, R3).

Функция Concat осуществляет склеивание строк R!, R2, R3 в одну строку в том порядке, в котором они записаны.

Например: R1 : = ‘Язык’; R2 : = ‘программирования’;
R3 : = ‘Pascal’; R : = ***Concat*** (R1, R2, R3);

Результатом действия функции будет R = ‘Язык программирования Pascal’.

2. **Функция Length**

Формат: **Length** (R).

Функция Length выдает фактическую длину строки, которая содержится в данной переменной. При подсчете длины строки учитываются все символы, в том числе и пробелы.

Например: R = ‘Turbo Pascal’; N : = ***Length*** (R);

Результатом действия функции будет N = 12.

3. **Функция Copy**

Формат: **Copy** (R, Poz, N).

Функция Copy копирует фрагмент длиной N строки R, начиная с позиции Poz.

Например: R : = ‘Turbo Pascal’; Poz : = 7; N : = 6;
Word : = ***Copy*** (R, Poz, N);

Результатом действия функции будет Word = ‘Pascal’.

4. **Функция Pos**

Формат: **Pos** (Word, R).

Функция Pos находит номер позиции P, с которой начинается первое вхождение слова Word в строку R. Если слово Word в строке R не найдено, то будет выведено число 0.

Например: R : = ‘Севастополь’; Word : = ‘сто’;

P : = ***Pos*** (Word, R);

Результатом действия функции будет P = 5.

5. Процедура Insert

Формат: **Insert**(Word, R, Poz).

Процедура Insert вставляет слово Word в строку R, начиная с позиции Poz.

Например: Poz : = 20; R : = ‘Язык программирования Pascal’; Word : = ‘Turbo’; ***Insert*** (Word, R, Poz);

Результатом действия процедуры будет R = ‘Язык программирования Pascal’.

6. Процедура Delete

Формат: **Delete** (R, Poz, N).

Процедура Delete уничтожает слово, которое начинается с указанной позиции Poz и имеет заданную длину N в строке R.

Например: Poz : = 1;

R : = ‘Язык программирования Turbo Pascal’;

N : = 19; ***Delete*** (R, Poz, N);

Результатом действия функции будет R = ‘Turbo Pascal’.

Задание. Создайте проект, который соединяет две строки в одну (фамилия, имя) и находит длину образованной строки. *Указание.* Разместить объекты на форме можно так, как показано на *рис 2.62.*

Программный код для кнопки *«Объединить»* может быть таким:

```
procedure TForm1.Button1Click(Sender: TObject);
  var d:integer;s:string;
begin
  edit3.text:=concat(edit1.text,' ',edit2.text);
   d:=length(edit3.text);
   str(d,s);
   edit4.text:=s;
end;
```

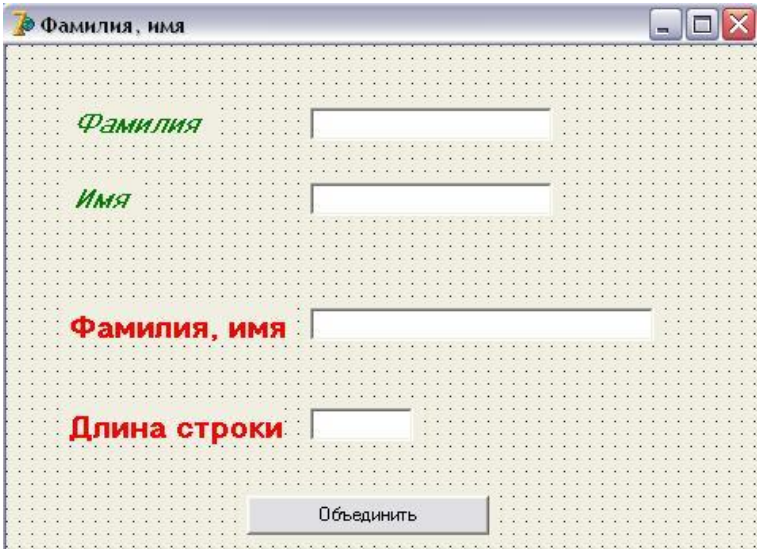

Рис. 2.62. Размещение объектов на форме для задания на ст.296

Пример выполнения программы *(рис. 2.63):*

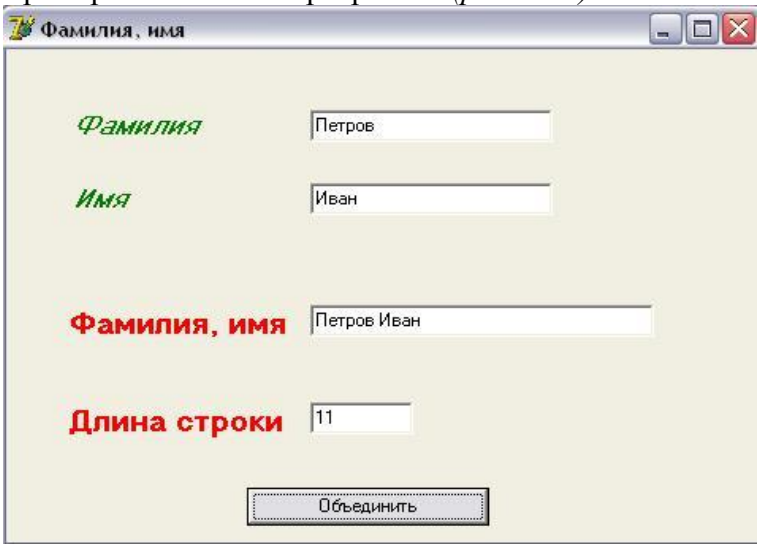

Рис. 2.63. Пример выполнения программы

ЗАДАНИЯ

1. Создайте проект, который подсчитывает количество пробелов во введенном тексте.
2. Создайте проект, который заменяет все символы *‘а’* на символы *‘абв’* во введенном тексте.
3. Создайте проект, который удаляет все символы ‘с’ во введенном тексте.
4. Создайте проект, который везде во введенном тексте заменяет некоторое слово другим словом такой же длины.

2.9. Средства работы с файлами и графикой

2.9.1. Работа с файлами

Часто возникает необходимость обрабатывать информацию, размещенную на внешних носителях (дисках). ***Файл*** *– это совокупность данных, размещенных на внешнем носителе.* Данные в файле называются *элементами.* Количество данных, в отличие от массива, при описывании файла не указывают. Элемент файла не имеет индекса (номера). Тип элемента может быть любым, кроме типа файл.

Файловый тип данных описывают в разделе описи типов так:

type < имя типа > = **file of** < базовый тип >**;**

или непосредственно в разделе объявления переменных

var < список переменных > **: file of** < базовый тип >**;**

Пример.

```
type myfile: file of integer;
var file1: myfile;
    file2, file3: file of string;
```

Действия с файлами

Для того, чтобы найти нужный элемент файла, необходимо последовательно просмотреть все предыдущие. Это называется *последовательным доступом к файлу*.

Для обработки файла его необходимо:

- открыть;
- выполнить необходимые действия;
- закрыть.

Для определения конца файла существует стандартная логическая функция

eof (<имя файла>);

Значением этой функции будет ***true***, если достигнут конец файла.

Для работы с файлами существуют такие команды:

assignfile(<имя файла>, <внешнее имя>) – налаживает связь между именем файла и файлом на внешнем носителе;
reset(<имя файла>) **–** открывает файл для считывания из него данных;
read(<имя файла>, <имя переменной>) – считывает данное из файла в оперативную память;
rewrite(<имя файла>) – открывает файл для записи в него данных;
write(<имя файла>, <имя переменной>) – записывает данное в файл;
closefile(<имя файла>) – закрывает файл.

Здесь

- ***<имя файла>*** - это имя *файловой переменной.* заданное в разделе объявления переменных;
- ***<внешнее имя>*** - это имя файла данных на внешнем носителе, взятое в кавычки, например, *'d:\folder1\file1.txt'. Если не указывать* ***полный путь к файлу****, а указать только* ***имя файла****, то компьютер будет считать, что этот* ***файл находится в той же папке, в которой сохранен проект.***

Кроме файлов *последовательного доступа* можно создавать и обрабатывать файлы *прямого доступа*. Отличие такое: перед использованием команд *read* или *write* нужно

обеспечить доступ к k-му элементу (нумерация с нуля) файла с помощью команды

seek(<имя файла>, k);

Текстовые файлы

Данные в типизированных файлах, описанных выше, некоторым образом кодируются компьютером. Эти файлы нельзя редактировать или просматривать с помощью текстового редактора. Поэтому, кроме типизированных, используют текстовые файлы, у которых нет такого недостатка.

Элементами текстовых файлов являются строки (последовательности символов: букв, цифр, знаков и пробелов). Такой файл можно создать и редактировать с помощью текстового редактора. Разделителем между элементами файла является пробел. Введение каждой строки заканчивается нажатием на клавишу ввода. Для проверки наличие символов в строке используют функцию

eoln(<имя файла>);

которая принимает значение ***true***, если найден конец файла.

Текстовые файлы описывают в разделе описи переменных так:

var <список имен переменных>**: textfile;**

Данные со строки текстового файла можно считать с помощью команд

read(<имя файла>, <список параметров>**);**
readln(<имя файла>, <список параметров>**);**

Разница между этими командами в том, что во время исполнения команды *readln* лишние данные в строке игнорируются и следующая команда *read* или *readln* будет считывать данные со следующей строки.

Строку текстового файла можно создать с помощью обычного текстового редактора или программным способом с помощью команд

write(<имя файла>, <список выражений>**);**

writeln(<имя файла>, <список выражений>**);**

В отличие от обычных файлов в текстовый файл можно добавлять (дописывать) данные. Для этого вместо процедуры rewrite используют процедуру

append(<имя файла>**);**

Задание. В задании на странице 296 добавьте кнопку «Записать в файл», которая записывает результат выполнения программы в файл на внешнем носителе.

Указания к выполнению

1. Разместить объекты на форме можно так *(рис 2.64):*

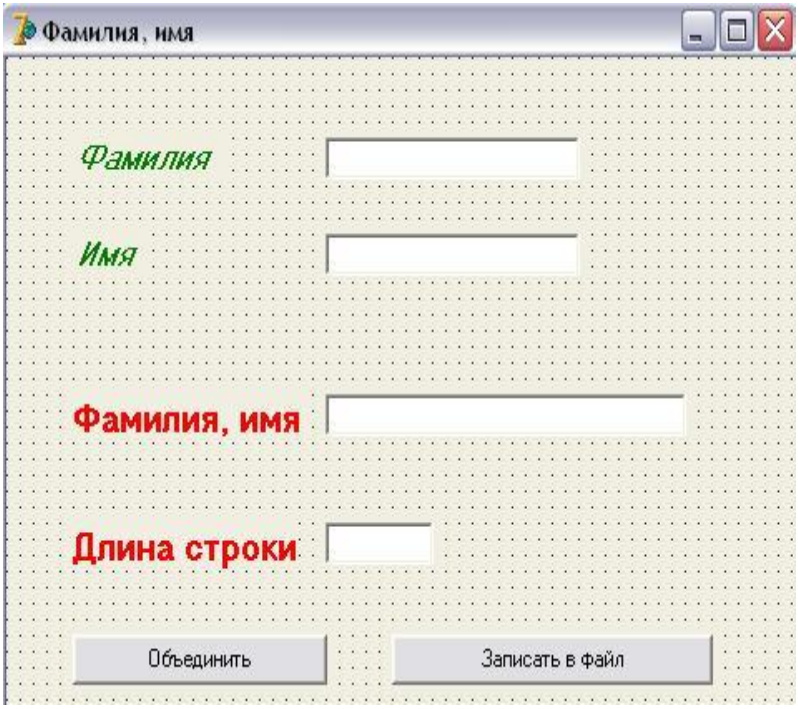

Рис. 2.64. Размещение объектов на форме для задания на ст. 304

2. Программный код для кнопки *«Записать в файл»* может быть таким:

```
procedure TForm1.Button2Click(Sender: Tobject);
  var f:textfile;s:string;d:integer;
begin
  assignfile(f, 'file1.txt');
  edit3.text:=concat(edit1.text,' ',edit2.text);
   d:=length(edit3.text);
   str(d,s); edit4.text:=s;
   rewrite(f);writeln(f, edit3.text,' ', d);
   closefile(f);
end;
```

3**.** Проверьте наличие созданного проектом нового файла *file1.txt* в той же папке, в которой вы сохранили проект. В файле должна появиться запись *(рис. 2.65):*

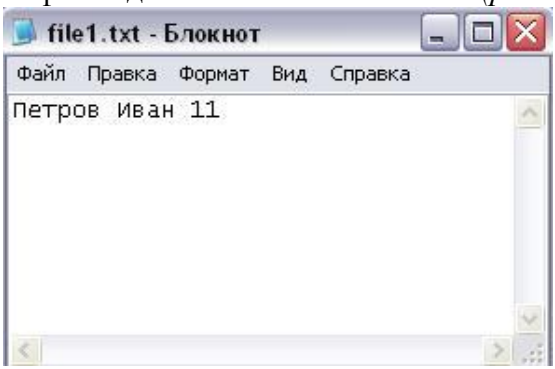

Рис. 2.65. Текстовый файл

2.9.2. Понятие о графике в визуальном программировании

Windows использует для рисования двухмерной графики интерфейс ***GDI (Graphics Device Interface).*** Это самый медленный способ отображения графики из существующих, но самый простой для понимания основ. GDI обычно не

используют для создания сложных графических эффектов, для этого есть *DirectX*, *OpenGL*, или любые графические библиотеки (такие как *DelphiX*, *FastLib*, *DIBUltra*, *Graphics32* и другие). Но для создания простых эффектов с минимальными усилиями GDI вполне подходит.

С GDI тесно связана ещё одна аббревиатура - ***DC*** (***"Device Context"*** - *контекст устройства*). Это то, на чём мы рисуем, и в Delphi контекст устройства представлен как ***TCanvas***. Идея контекста устройства заключается в том, что это универсальное устройство вывода, поэтому можно использовать одинаковые функции как для экрана, так и для принтера.

Предоставляет пользователю окно с канвой для рисования произвольных изображений компонент ***TPaintBox***, который находится на вкладке *System (рис. 2.66)*.

Рис. 2.66. Вкладка “System”

Канва содержится в свойстве *Canvas* компонента, графические инструменты - в свойствах *Font, Pen* и *Brush*, а само рисование осуществляется в обработчике события *OnPaint*.

Все графические функции в Delphi являются надстройками над стандартными *GDI* функциями Windows. Рассмотрим, как устроен *GDI*. Ниже, в таблице, представлены некоторые важные классы:

Имя	***Описание***
Pen	Используется для рисования простых линий. Обычно применяется для функции LineTo или при рисовании рамки для определённой фигуры (например для функции Rectangle).

Brush	Кисть используется для заполнения области определённым цветом. Применяется в функциях Rectangle, FillRect или FloodFill.
Font	Используется для задания шрифта, которым будет нарисован текст. Можно указать имя шрифта, размер и т.д.
Region	Позволяет задать регион (замкнутое пространство). Регионом может быть круг, квадрат или произвольная фигура. Позволяет также создавать отверстия в фигурах.

Рисование линий

Координата *(0,0) - это верхний левый угол экрана*. То есть значения по *оси y увеличиваются вниз экрана*. Самое главное, что надо знать при рисовании линий и фигур, это различие между *пером (Pen)* и *кистью (Brush)*: ***перо (Pen)*** *используется при рисовании линий или рамок, а* ***кисть (Brush)*** *для заполнения фигуры.* Ниже приведены две функции, которые используются для рисования линий и обе принадлежат TCanvas:

Имя	***Описание***	***Пример***
MoveTo	Перемещает точку начала рисования линии в указанные координаты x и y	Canvas.MoveTo(50, 100)
LineTo	Рисует линию, начиная с текущей позиции (см. MoveTo) до указанных координат x и y.	Canvas.LineTo(50, 100)

Эффект перемещения точки начала рисования линии так же достигается при помощи установки свойства PenPos в канвасе. ***Например,*** *"Canvas.PenPos.x:=20", "Canvas.PenPos.y:=50"*, или *"Canvas.PenPos:= Point(20,50)"*.

По умолчанию точка начала рисования установлена в (0,0), то есть если сразу задать команду *"Canvas.LineTo(100,100),* то будет нарисована линия из точки

(0,0) в точку (100, 100). Точка начала рисования автоматически переместится в (100, 100), то есть если выполнить команду *"Canvas.LineTo(200, 100)"*, то следующая линия будет нарисована из точки (100, 100) в (200, 100). Поэтому, если мы хотим рисовать линии, не соединённые одна с другой, то нужно воспользоваться методом *MoveTo.*

Линия, нарисованная при помощи *LineTo*, использует текущее перо канваса (типа *TPen*). ***Основные свойства пера*** - это ***ширина***, например, *"Canvas.Pen.Width:=4"* (при помощи которого можно задавать различную ширину линий), и ***цвет***, например, *"Canvas.Pen.Color := clLime"*.

Рисование фигур

Для рисования фигур в TCanvas предусмотрены функции:

Имя	***Описание***	***Пример***
Ellipse	Рисует эллипс, вписанный в невидимый квадрат с координатами верхнего левого и правого нижнего углов. Если координаты x и y у углов будут совпадать, то получится круг.	Canvas.Ellipse(0,0, 50,50);
FillRect	Заполняет прямоугольник цветом текущей кисти (brush).	Canvas.FillRect (Bounds(0,0,100,100));
FloodFill	Заполняет данную область цветом текущей кисти.	Canvas.FloodFill (10, 10, clBlack, fsBorder);
Rectangle	Рисует прямоугольник (или квадрат), заполненный цветом текущей кисти и обрамлённый цветом текущего пера	Canvas.Rectangle (Bounds(20, 20, 50, 50));

RoundRect	Рисует прямоугольник со скруглёнными углами.	Canvas.RoundRect(20, 20, 50, 50, 3, 3);

Ещё есть очень нужная функция *TextOut*, которая позволяет рисовать текст, используя шрифт, заданный в канвасе:

Имя	***Описание***	***Пример***
TextOut	Рисует строку, начиная (x, y), фон текста заполняется текущим цветом кисти.	Canvas.TextOut(10, 10, 'Some text');

Например, представленный ниже обработчик создаст окно, показанное на *рис. 2.67.*

```
procedure TForm1.PaintBox1Click(Sender: TObject);
  var x,y:integer;
begin
   with paintbox1.canvas do
   begin
            brush.color:=clred;
            ellipse(0,0,width,height);
            font.Name:='arial';
            font.size:=height div 5;
            font.Style:=[fsbold, fsitalic];
            font.Color:=clwhite;
            x:=(width-textwidth('Delphi')) div 2;
            y:=(height-textheight('D')) div 2;
            textout(x,y,'Delphi');
   end
end;
```

Рис. 2.67. Пример использования компонента TPaintBox

Задание. Создайте рисунок *(рис.2.68):*

Рис. 2.68. Рисунок для задания на ст.307

ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ №6

Задание 1. Создайте проект для вычисления:

Вариант 1 суммы всех натуральных чисел с [3; 178].
Вариант 2 произведения всех натуральных чисел с о [5; 34].
Вариант 3 суммы всех натуральных чисел с [7; 211].
Вариант 4 произведения всех натуральных чисел с [25; 74].
Вариант 5 суммы всех натуральных чисел с [7; 233].
Вариант 6 произведения всех натуральных чисел с [6; 27].
Вариант 7 суммы всех натуральных чисел с [19; 238].
Вариант 8 произведения всех натуральных чисел с [28; 72].
Вариант 9 суммы всех натуральных чисел с [26; 115].
Вариант 10 суммы всех натуральных чисел с [7; 211].
Вариант 11 произведения всех натуральных чисел с [7; 22].
Вариант 12 произведения всех натуральных чисел с [7; 22].
Вариант 13 суммы всех натуральных чисел с [7; 233].
Вариант 14 произведения всех натуральных чисел с [6; 27].
Вариант 15 суммы всех натуральных чисел с [19; 238].
Вариант 16 произведения всех натуральных чисел с [28; 72].
Вариант 17 суммы всех натуральных чисел с [5; 109].
Вариант 18 произведения всех натуральных чисел с [7; 33].
Вариант 19 произведения всех натуральных чисел с [4; 13].
Вариант 20 суммы всех натуральных чисел с [2; 169].
Вариант 21 произведения всех натуральных чисел с [7; 19].
Вариант 22 суммы всех натуральных чисел с [24; 89].
Вариант 23 произведения всех натуральных чисел с [5; 11].
Вариант 24 суммы всех натуральных чисел с [35; 239].
Вариант 25 произведения всех натуральных чисел с [9; 23].
Вариант 26 суммы всех натуральных чисел с [76; 199].
Вариант 27 произведения всех натуральных чисел с [8; 14].
Вариант 28 суммы всех натуральных чисел с [54; 187].
Вариант 29 произведения всех натуральных чисел с [3; 12].
Вариант 30 суммы всех натуральных чисел с [4; 999].

Задание 2. Создайте проект для вычисления:

Вариант 1	суммы элементов массива *a*[1..2, 1..3] целых чисел, больших заданного числа *s*.
Вариант 2	произведения элементов массива *b*[1..2, 1..4] целых чисел, меньших заданного числа *z*.
Вариант 3	количества элементов массива *c*[1..2, 1..5] целых чисел, равных заданному числу *a*.
Вариант 4	произведения элементов массива *d*[1..3, 1..2] целых чисел, меньших заданного числа *n*.
Вариант 5	количества элементов массива *n*[1..3, 1..3] целых чисел, меньших заданного числа *t*.
Вариант 6	суммы элементов массива *m*[1..3, 1..4] целых чисел, равных заданному числу *h*.
Вариант 7	суммы элементов массива *p*[1..3, 1..5] целых чисел, больших заданного числа *f*.
Вариант 8	произведения элементов массива *k*[1..4, 1..2] целых чисел, меньших заданного числа *d*.
Вариант 9	количества элементов массива *h*[1..4, 1..3] целых чисел, равных заданному числу *y*.
Вариант 10	произведения элементов массива *t*[1..4, 1..5] целых чисел, меньших заданного числа *w*.
Вариант 11	количества элементов массива *w*[1..5, 1..2] целых чисел, меньших заданного числа *r*.
Вариант 12	суммы элементов массива *x*[1..5, 1..3] целых чисел, равных заданному числу *u*.
Вариант 13	суммы элементов массива *y*[1..5, 1..4] целых чисел, больших заданного числа *p*.
Вариант 14	произведения элементов массива *z*[1..5, 1..5] целых чисел, меньших заданного числа *a*.
Вариант 15	количества элементов массива *g*[1..2, 1..3] целых чисел, равных заданному числу *d*.
Вариант 16	произведения элементов массива *s*[1..2, 1..2] целых чисел, меньших заданного числа *g*.
Вариант 17	количества элементов массива *q*[1..4, 1..4] целых чисел, равных заданному числу *h*.

Вариант 18	суммы элементов массива *u*[1..3, 1..6] целых чисел, меньших заданного числа *k*.
Вариант 19	суммы элементов массива *v*[1..6, 1..2] целых чисел, больших заданного числа *k*.
Вариант 20	произведения элементов массива *f*[1..2, 1..6] целых чисел, меньших заданного числа *n*.
Вариант 21	количества элементов массива *ab*[1..6, 1..3] целых чисел, равных заданному числу *m*.
Вариант 22	суммы элементов массива *kf*[1..2, 1..3] целых чисел, больших заданного числа *d*.
Вариант 23	произведения элементов массива *abc*[1..6, 1..4] целых чисел, меньших заданного числа *t*.
Вариант 24	количества элементов массива *cdf*[1..4, 1..6] целых чисел, равных заданному числу *h*.
Вариант 25	произведения элементов массива *sv*[1..6, 1..2] целых чисел, меньших заданного числа *k*.
Вариант 26	суммы элементов массива *v*[1..6, 1..2] целых чисел, больших заданного числа *k*.
Вариант 27	количества элементов массива *ap*[1..2, 1..2] целых чисел, меньших заданного числа *r*.
Вариант 28	произведения элементов массива *td*[1..3, 1..2] целых чисел, больших заданного числа *s*.
Вариант 29	суммы элементов массива *w*[1..4, 1..2] целых чисел, меньших заданного числа *k*.
Вариант 30	произведения элементов массива *mn*[1..3, 1..3] целых чисел, равных заданному числу *k*.

Задание 3. В задании 1 добавьте к форме кнопку для записи результатов выполнения программы в текстовый файл.

Задание 4. С помощью текстового редактора «Блокнот» создайте текстовый файл, который содержит элементы массива из задания 2. В задании 2 добавьте к форме кнопку для считывания элементов массива из созданного текстового файла.

СЛОВАРЬ

Русский язык	*English*
авиакомпания	air company
автоматический	automatic
агентство	agency
Азербайджан	Azerbaijan
алюминиевый	aluminium
амплитуда	amplitude
анализ	analysis
аналогично	analogy
апрель	April
Арабский	Arabic
балл	mark, grade, point
банк	banc
батарейка	battery
белый	white
беспроводной	wireless
благодарить	thank
больница	hospital
бумага	paper
важность	importance
вариант	variant
вбивать	drive
введение	introduction
вещество	substance
влево	left
вместо	instead
воздух	air
возникновение	rise
возраст	age
волна	wave
вопрос	answer
вправо	right
врач	doctor

время	time
всего	total sum
выдать	display
выразить	express
выход	exit
выяснить	find out
Вьетнам	Vietnam
география	geography
главный	main
говорить	speak
год выпуска	made year
город	city
гостиница	hotel
градуировать	graduate
график	graphic
группа	group
данный	given
двигаться	move
двойственный	double
двусторонний	two-way
день рождения	birthday
дерево	tree
длиться	last
доказательство	proof
документ	document
долгосуществующий	long-term
должна	must
достаточно	considerably
достигнуть	reach
достоверный	authentic, reliable, valid
доступный	accessible
единица	one
жест	gesture

жизнь	life
зависеть	depend
завод, заводской	factory
задание	task
заключаться	consist
занятие	lesson
записать	write
зарплата	pay
зафиксировать	fix
защита	protection
звезда	star
звук	sound
зеркало	mirror
знакомство	meeting
знать	know
иголка	needle
изменить	change
измерять	measure
изображение	image
изучать	study
инженер	engineer
иностранный	foreign
Ирак	Iraq
Иран	Iran
искажать	distort
истолкование	interpretation
исчезнуть	disappear
исчерпывающий	exhaustive
июль	June
июнь	July
каждый	every
картина	picture
керамический	ceramic

Китай	China
клиент	client
коллективный	collective
комбинировать	combine
конечный	finite
контрольная работа	control work
концерт	concert
который	which
коэффициент	coefficient
краска	colour
кремень	flint
кристалл	crystal
круг	circle
лампа	lamp
летчик	aviator, flier, pilot
литература	literature
луч	ray
любой	any
люди	peoples
май	May
максимальный	maximum
маленький	small
март	March
математика	mathematics
материальный	material
машинный	machine
между	between
место работы	work place
месяц	month
метка	label
метод	method
механик	mechanic
мигающий	blink

микроскопический	microscopic
миллион	million
мимика	mimic
многократный	multiple
момент	moment
Монголия	Mongolia
музыка	music
на протяжении	during
называть	name
наименьший	minimal
накопление	accumulation
наличие	presence
намерение	intention, purpose
нанести	bring
например	for example
насколько	how much
наука	science
национальный	national
начальный	opening
недолгосуществующий	of short duration
некоторый	some
неопределенность	indefinite
нести	carry
низкий	low
новый	new
ноготь	nail
ноль	zero
номер	number
ноутбук	notebook
нужный	necessary
нулевой	naught
обеспечить	provide
область	area

обмен	interchange
образовывать	form
общаться	associate
общение	intercourse
общий	general
объявление	declaration
обычный	usual
одновременно	simultaneous
однократный	single
ознакомиться	familiarize
означать	mean
окружающая среда	environment
определенный	definite
определить	determine
организация	organisation
организовать	organise
основание	basis
остров	island
осуществить	realise
отверстие	hole
ответ	answer
отдаленный	remote
отдельный	individually
отклоняться	decline
отражать	repel
отрицательный	negative
отсутствие	absence
оттенок	shade
оценить	mark
ошибка	error
пакет	packet
парк	park
пароль	password

пауза	pause
перевод	translation
переводчик	translator
перейти	cross
переменный	variable
пересылать	send
перечислить	enumerate
период	period
персональный	personal
печать	print
питание	board
пластиковый	plastic
пластина	plate
повседневный	everyday
поглощать	absorb
подать	give
подпись	signature
подтверждать	confirm
позиция	position
полезный	useful
получатель	recipient
получить	get
пользоваться	use
помещение	room
помощь	help
понимать	understand
понятный	understandable
поощрительный	encourage
попадать	get
после	after
пособие	textbook, manual
постоянный	constant
почта	mail

правильность	rightness
правильный	right
превращаться	transform
предложить	prepositional
предмет	object
предназначать	destine
предоставить	grant
предприятие	business
представить	show
преобразовать	transform
привод	drive
прикладной	applied
примечание	note
принимать, принять	accept
природа	nature
проверить	control
проводной	wire, cable
программный	program
производительность	productivity, efficiency
произвольный	arbitrary
происходить	happen
промежуточный	intermediate
профессия	profession
прохождение	passing
псевдографика	pseudo graphic
пусть	let
пылесос	vacuum cleaner
пятно	spot
работа	work
разговор	talk
разделить	divide
разместить	distribute
разный	different

разряд	category
расписание	timetable
распространение	diffusion
рассеивать	disperse
расстояние	distance
расчет	calculation
реализовать	realise
регистр	register
режим	mode
режим	mode
результат	result
речь	speech
решение	solution
решить	decide
рисунок	picture
родился	was born
роман	roman
рукописный	manuscript
свет	light
светильник	lighter
своевременно	timely
секунда	second
сенсорный	sensor
сентябрь	September
серый	grey
сигнал	signal
сила	strength
скорость	speed
сложение	addition
слой	layer
служебный	official
смешивать	mix
смещение	displacement

снабжение	provision
сначала	at first
совершенный	absolute
совокупность	totality
современный	modern
содержание	table of contents
содержать	contain
содержимое	consist
соединение	connection
соответствовать	correspond
сопровождать	accompany
сорт	sort
составлять	compose
состоит	consist
специальный	special
способность	quality
справка	information
среднее образование	secondary education
средство	means
стадион	stadium
статья	item
стойкий	firm
страна	country
студент	student
существовать	exist
считывать	read
такт	time
текстовый	text
телевизор	TV set
телефонный	telephone
терять	lose
технический	technical
тип	type

типографский	printing
товар	ware
ток	current
транспорт	transport
требовать	demand
три	three
Турция	Turkish
углубление	deepening
удобный	comfortable
указывать	point out
Украина	Ukraine
умывальник	wash-basin
университет	university
упорядочить	arrange
упражнение	exercise
ускорение	acceleration
условие	condition
установить	establish
устройство	arrangement
утюг	iron
участок	section
учреждение	office
фаза	phase
фамилия	surname
февраль	February
фиксировать	fix
филиал	filial
фирма	firm
фон	background
фотоаппарат	photo camera
функциональный	functional
характеристика	characteristic
химия	chemist

холодильник	fridge
хороший	good
хранить	save
художественный	artistic
цвет, цветной	colour
целый	whole
ценность	value
цифра	figure
часть	part
человек	man
чередовать	alternate
черный	black
чертеж, черчение	drawing
четкий	distinct
число	number
ширина	width
экзамен	examination
экономист	economist
электрический	electrical
электроэнергетика	electrical energetic
эфир	ether
явление	event
язык	language
январь	January
яркий	vivid

ПОСЛЕСЛОВИЕ

Учебный материал, изложенный в настоящем пособии, является хорошей базой как для изучения основ информатики и вычислительной техники, а также смежных и специальных дисциплин в высшей школе, так и для практического использования средств компьютерной техники и современных информационных технологий в научно-познавательной деятельности студентов.

Так как в системе образования компьютер является не только естественным объектом учебного процесса, но и ценным техническим средством обеспечения общего процесса образования, то знания, умения и навыки, полученные во время изучения данной дисциплины, будут использованы при изучении других дисциплин.

Преподаватели физики и химии, математики и биологии, языка и литературы, географии, истории и других общеобразовательных и специальных дисциплин все шире и шире используют компьютер в своей повседневной работе со студентами. Неоценимую пользу оказывают компьютеры и как средство изучения иностранных языков.

Также студенты используют компьютер как вспомогательное средство при подготовке домашних заданий, рефератов и других работ, при самостоятельном изучении учебного материала.

Таким образом, внедрение информационных технологий в систему образования позволяет повысить качество учебного процесса в высшем учебном заведении. Применение компьютеров при изучении практически всех дисциплин постепенно переходит из статуса эксперимента в режим постоянного функционирования как неотъемлемый элемент системы образования.

СПИСОК ЛИТЕРАТУРЫ

1. *Навчальні* програми (довузівська підготовка іноземних громадян). – К.: Політехніка, 2005. – 48-59 с.
2. *Информатика: Базовый курс* / С.В.Симонович и др.- СПб.: Питер, 2001.- 640 с.
3. *Симонович С.В., Евсеев Г.А., Алексеев А.Г.* Общая информатика. – М.: АСТ-ПРЕСС Книга, 2004. – 592 с.
4. *Фабричев В.*А., *Труш О.І., Чижевський Й.Ф.* Основи інформатики: Навч. посібник. – К.: Книжкове видавництво НАУ, 2006. – 352 с.
5. *Фаронов В.* Delphi 6. Учебный курс. – С.-П: Питер, *2002*.- 507 с.
6. *Глинський Я.М., Анохін В.Є., Ряжська В.А.* Паскаль. Turbo Pascal і Delphi.- Львів: Деол, 2002. – 142 с.
7. *Следзінський І..Ф., Василенко Я.П.* Основи інформатики. Посібник для студентів. – Тернопіль: Навчальна книга – Богдан, 2003. – 160 с.
8. *Колисниченко Д.Н.* Англо-русский толковый словарь компьютерных терминов / Под ред. М.В. Финкова. Серия «Просто о сложном». – СПб.: Наука и Техника, 2006. – 288 с.
9. *Єфименко В.В., Онищенко С.М.* Опрацювання табличних даних засобами Microsoft Excel. Лабораторний практикум: Навчальний посібник. – К.: Логос, 2005. - 167 с.
10. *Кравченко С.М., Єфименко В.В., Онищенко С.М.* Операційна система Microsoft Windows. Лабораторний практикум: Навчальний посібник. – К.: Логос, 2005. - 76 с.
11. *Зарецька І.Т., Колодяжний Б.Г., Гурджій А.М., Соколов О.Ю.* Інформатика 10-11. – К.: Навчальна книга, 2002. – 495 с.
12. *Глинський Я.М.* Практикум з інформатики. – Львів: Деол, 2002. – 224 с.
13. *Глинський Я.М.* Інформатика: В 2-х ч. – Львів: Деол, 2002. – 255 с.
14. *Симонович С., Евсеев Г.* Практическая информатика. – М.: АСТ-ПРЕСС, 2001. – 480 с.
15. *Матвеев М.*Д., *Юдин М.В., Куприянова А.В.* Microsoft Windows XP. Все об использовании и настройках. Изд. 3-е, перераб. и доп. – СПб.: Наука и техника, 2008. – 624 с.

Навчальне видання

Основи інформатики та обчислювальної техніки

Навчальний посібник для іноземних студентів
підготовчого відділення
факультету по роботі з іноземними студентами
(Російською мовою)

Укладач Бедренко Валентина Іванівна

Основы информатики и вычислительной техники

Учебное пособие для иностранных студентов
подготовительного отделения
факультета по работе с иностранными студентами

Составитель Бедренко Валентина Ивановна

Підп. до друку . Формат 60×84/16. Папір офс.
Офс. друк. Ум. друк. арк.. Обл.-вид. Арк..
Тираж 100 пр. Замовлення № . Вид. №20/III.

Видавництво НАУ
03680. Київ – 680, проспект Космонавта Комарова, 1.

Свідоцтво про внесення до Державного реєстру ДК № від.