МОДУЛЬ 2. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

2.1. Типовые структуры алгоритмов. Понятие об анализе исполнения и правильности алгоритмов

Новые слова

Русский язык	*English*
алгоритм	mechanism, algorithm, routine
анализ	analysis, interpretation
базовый	basic, fundamental
блок	block
блок-схема	flow diagram, block diagram
ветвящийся	branching
виртуальный	virtual
вспомогательный	backup, housekeeping, subsidiary
выбор	choice, decision, selection
выделить	select, insert
выполнить	execute
выход	exit
главный	main
действие	action
детализация	refinement, refining
дискретность	discretion
заказчик	customer
запустить	run
имитационный	simulation
информационный	information
исполнитель	executor
исправить	edit
клавиша	key
количество	amount, quantity
команда	command
конец	end
конечность	definition
линейный	linear
массовость	totality
математический	mathematical
метод	method

модель	model
модуль	module
начало	start
новый	new
объект	object
однозначность	single value
окно	window
определенный	defined, specified
открыть	open
ошибка	error
переведение	translation
перед	before
повторение	iteration
понятие	concept, notion
понятность	understandability
после	after
последовательность	successive
построение	construction
пошаговый	step-by-step, step-type, stepwise
правило	rule
правильность	accuracy
программа	program
программирование	programming
проектирование	design, designing, engineering
простой	simple
реальный	real
результат	result
результативность	result
словесный	verbal
сложный	compound, elaborate
создать	create
сохранить	save
структура	structure
схема	scheme
упрощенный	simplify
условие	condition
файл	file
физический	physical

формальный	formal
формула	formula
формулирование	state
цикл, циклический	cycle
этап	phase, step
ярлык	shortcut

Компьютер дает возможность быстро решать сложные задачи из разных отраслей – математические, экономические, физические, инженерные и другие. Цель изучения программирования – научиться решать разные задачи с помощью компьютера. Существует единый план для решения любой задачи. Он состоит из 6 этапов.

1 этап. Формулирование условия задачи. На этом этапе встречаются заказчик и исполнитель для того, чтобы:

1) рассмотреть исследуемое явление или объект;
2) сформулировать условие задачи;
3) конкретизировать, что дано, а что нужно найти.

Важно, чтобы исполнитель получил от заказчика максимум нужной информации.

Задача. *Вычислите количество литров краски, необходимое для окраски пола прямоугольной формы, длина которого a метров, ширина b метров, если для одного квадратного метра необходимо k литров краски.*

2 этап. Создание модели задачи. ***Модель*** – это формальное описание, отображающее самые существенные свойства объекта или явления и связи между данными. Другими словами, *модель – это упрощенный вариант реального объекта* (например, игрушечная детская машинка является моделью настоящего автомобиля и т.д.). *Такое моделирование называют* ***физическим***. **Информационная модель** – это совокупность свойств, которые описывают некоторый сложный объект или явление. ***Например,*** информационной моделью машины является комплект ее чертежей или совокупность формул, которые описывают ее функционирование.

Разновидностью информационной модели является **математическая модель** – это описание задачи (объекта) с помощью формул. Мы будем создавать математические модели. При их создании можно пренебречь второстепенными свойствами, делая предположения, которые ведут к упрощению модели. Важно, чтобы эти предположения не привели к неправильным результатам. *Процесс создания модели называют* ***моделированием*** *(рис.2.1).*

Имея графическую (чертеж, рисунок, фотография) и математическую (формулы) модели, с помощью компьютера можно создать *виртуальную* (реально несуществующую) модель. Например, так создают модели причесок, одежды, автомобилей и т.д. *Такой способ моделирования называют* ***виртуальным (имитационным)***, потому что он не требует реальных образцов.

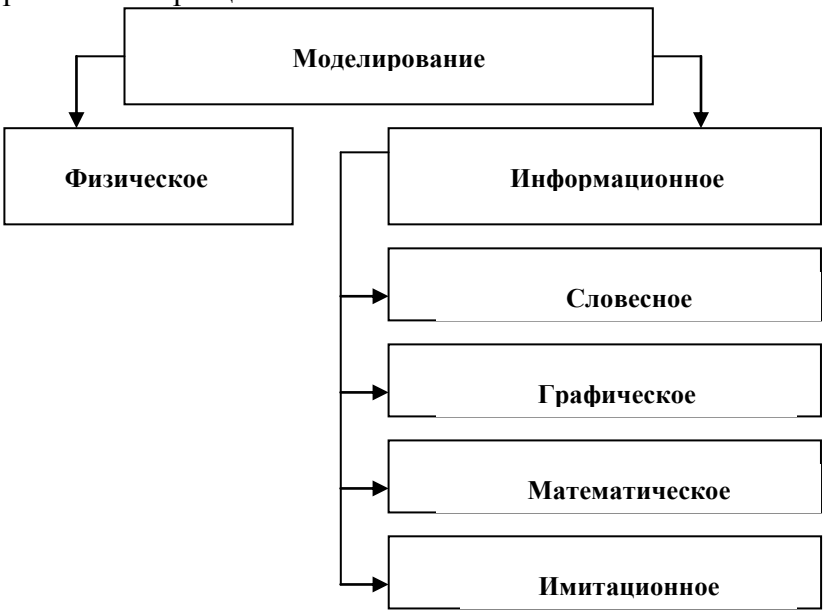


Рис.2.1. Виды моделирования

В нашей задаче будем считать, что пол имеет идеально прямоугольную форму. Даны стороны a и b и количество краски k литров на один квадратный метр. Нужно найти площадь прямоугольника S и общее количество краски m. Все данные и результаты – вещественные числа.

3 этап. Выбор метода решения. Нашу задачу мы будем решать с помощью формул $S = ab$, $m = kS$.

4 этап. Создание алгоритма и переведение его на язык программирования. ***Алгоритм*** *– это последовательность действий со строго определенными правилами выполнения*. Алгоритм может быть записан словесно, графически или последовательностью формул.

Составим алгоритм решения нашей задачи:

1) ввести a, b, k;
2) $S = ab$;
3) $m = kS$;
4) вывести m.

Потом переводят алгоритм на язык программирования (язык, который понимает компьютер) – например, Паскаль. ***Программа*** *– это алгоритм, переведенный на язык программирования.* Перевод нашей задачи на язык программирования Паскаль приводится ниже в ходе выполнения практической работы.

5 этап. Выполнение программы на компьютере. Выбирают компьютер, вводят текст программы, выполняют ее, получают результаты.

6 этап. Анализ результатов. Заказчик должен получить от исполнителя максимум информации о выполненной работе. Если результаты окажутся недостоверными, то процесс повторяют, начиная с этапа, на котором была допущена ошибка.

ПРАКТИЧЕСКАЯ РАБОТА №1

Тема. Работа с программой-образцом.

Цель. Получение элементарных навыков работы в среде Турбо Паскаль.

Ход работы

1. Запустите программу ***TP7.1*** (ярлык на рабочем столе).

2. Откройте новое окно: ***F10 → File → New.***
3. Введите код программы:

```
program pol;
uses crt;
var a, b, k, s, m:real;
    begin
      clrscr;
      writeln('введите a'); readln(a);
      writeln('введите b'); readln(b);
      writeln('введите k'); readln(k);
      S := ab; m := kS;
      writeln('m=',m:9:2); readln;
    end.
```

4. Сохраните программу: ***File → Save as →*** задать имя файла **→ *Ok*** (или нажать клавишу ***F2***).
5. Выполните программу: ***Run → Run*** (или комбинация клавиш ***Ctrl + F9***).
6. Если в тексте программы были допущены ошибки, компьютер выделит соответствующие строки красным цветом. Исправьте ошибки и для сохранения изменений снова нажать клавишу ***F2***.
7. Следуя командам компьютера, введите числовые значения a, b и k, нажимая клавишу ***Enter*** в конце каждой строки.
8. Для возвращения в окно программы нажмите любую клавишу.
9. Выполните программу еще раз, задавая другие данные.
10. Выйдите из среды программирования: ***F10 → File → Exit***.

Алгоритмы

Любой вид деятельности человека – это последовательность процессов принятия решений, выполнения действий, анализа результатов. Поступки людей подчинены выполнению конкретной цели. Их действия

выполняются по конкретному *алгоритму*. Мы составляем такие алгоритмы, обдумывая планы на день, на год и т.д.

Термин *алгоритм* происходит от имени древнего философа и математика из Хорезма, который жил в IX столетии – Аль-Хорезми. В своих трактатах он описал правила (алгоритмы) сложения, вычитания, умножения и деления многозначных чисел, которыми мы пользуемся и сегодня. ***Алгоритм*** *– это конечная последовательность команд, которые нужно выполнить над входными данными для получения результата.*

Пример 1. Вычислите значения выражения $\frac{301+51}{92-32}$.

Решение

Нужно выполнить *алгоритм* $\boldsymbol{A = (A_1, A_2, A_3)}$ или $\boldsymbol{B = (A_2, A_1, A_3)}$, который состоит из команд:

A_1: 92 – 32 и запомнить результат (60).

A_2: 309 + 51 и запомнить результат (360).

A_3: 360 : 60 и запомнить результат (6).

Пример 2. Составьте алгоритм решения квадратного уравнения $ax^2 + bx + c = 0, \quad a \neq 0$.

Решение

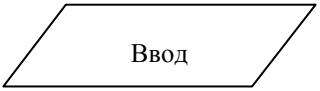
1. Задать *a, b, c.*
2. Вычислить $D := b^2 - 4ac$.
3. Если $D \geq 0$, то перейти к шагу 4, иначе перейти к шагу 7.
4. Вычислить $d = \sqrt{D}$.
5. Вычислить $x_1 = (-b - d)/2a$, $x_2 = (-b + d)/2a$.
6. Записать ответ: x_1, x_2. Перейти к шагу 8.
7. Записать ответ: «Решений нет». Перейти к шагу 8.
8. Конец.

В предыдущих примерах алгоритмы записаны разговорным языком (словами) с нумерацией команд. Это – ***словесно-пошаговая форма описания алгоритмов***. В такой форме заданы многие алгоритмы в математике, повседневной жизни и т.д.

Алгоритм можно описать ***графически***, то есть в виде *схем*, которые состоят из *блоков*, соединенных стрелками. Стрелки указывают направление вычислительного процесса (последовательность шагов (команд)). Блоки обозначают так:

– начало алгоритма. Блок имеет форму овала.

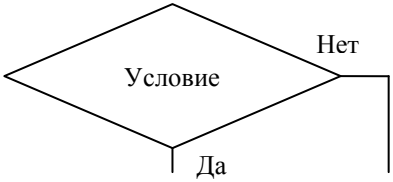
- конец алгоритма. Блок имеет форму овала.



- введение начальных данных. Блок имеет форму параллелограмма.

- вывод результатов работы. Блок имеет форму параллелограмма.

- описание действия (шага). Блок имеет форму прямоугольника.



- логический блок передачи управления (блок проверки условия). Блок имеет форму ромба.

Блоки соединяются линиями, которые описывают *последовательность выполнения команд*. Эти линии называются ***линиями потоков*** передачи информации. Естественны направления потоков ***сверху-вниз (↓)*** и **слева-направо (→).** Если направление потока другое, то линия должна иметь *стрелку*. Блок *«Условие»* имеет два выхода – *вниз, если условие выполняется, и вправо (или влево), если условие не выполняется.*

Алгоритм состоит из отдельных указаний исполнителю для выполнения некоторого законченного действия. Эти указания называются *командами* (или *операторами*). Команды выполняются последовательно, одна за другой. Алгоритм должен быть точным и понятным исполнителю. ***Исполнители алгоритмов*** – это люди, компьютеры, роботы и т.д. ***Допустимые команды исполнителя*** – это команды, которые исполнитель может выполнить.

Пример. Исполнитель может только складывать числа. Составим алгоритм умножения любого числа x на 3:

1. Задать x.
2. Вычислить $a = x + x$.
3. Вычислить $y = a + x$.
4. Записать ответ: y.
5. Конец.

Этому исполнителю команда: «Вычислить $y = 3x$» будет *непонятной.*

Свойства алгоритмов

1. ***Понятность.*** Алгоритм является понятным, если он состоит из допустимых команд исполнителя.
2. ***Конечность.*** Алгоритм является конечным, если он состоит из конечного количества команд.
3. ***Дискретность.*** Алгоритм является дискретным, если каждая команда начинает выполняться после завершения выполнения предыдущей.
4. ***Результативность.*** Алгоритм является результативным, если приводит к получению результатов, которые могут быть и неправильными.
5. ***Правильность.*** Алгоритм является правильным, если его выполнение обеспечивает достижение цели.
6. ***Массовость.*** Алгоритм является массовым, если он обеспечивает решение не только одной задачи, а целого класса задач данного типа.
7. ***Однозначность.*** Алгоритм является однозначным, если все его команды выполняются единым способом.

Базовые структуры алгоритмов – это способы управления процессами обработки данных. Есть три основные базовые алгоритмические структуры: *линейные*, *ветвящиеся* и *циклические*.

1. Линейный (простой) алгоритм – это алгоритм, который состоит из ***последовательности простых команд***. Общий вид ***блок-схемы линейного алгоритма*** изображен на *рис.2.2*. Эта схема значит, что для выполнения задания надо последовательно выполнить несколько простых команд.

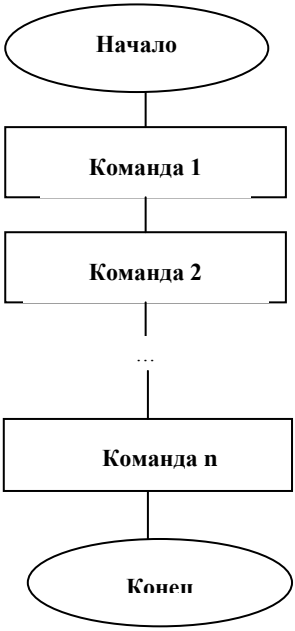


Рис. 2.2. Линейный алгоритм

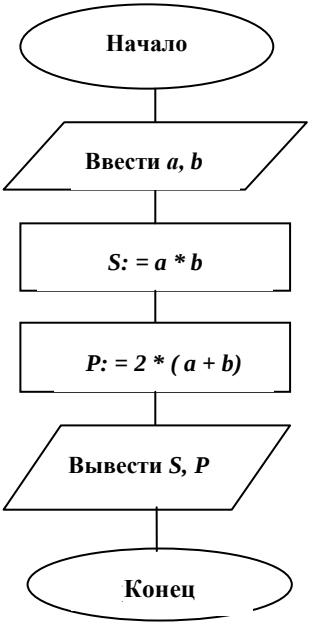
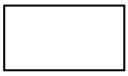


Рис. 2.3. Пример линейного алгоритма

Пример. Составьте алгоритм в виде блок-схемы для вычисления ***площади*** и ***периметра*** прямоугольника со сторонами *a* и *b (рис. 2.4, 2.3).*

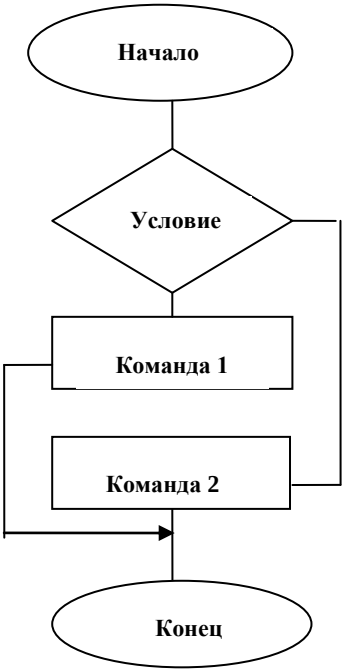


$S = ab$ - ***площадь***

$P = 2 \cdot (a + b)$ - ***периметр***

Рис. 2.4. Прямоугольник

2. Ветвящийся алгоритм – это алгоритм, который содержит команду, включающую какое-либо ***условие*** (*логическое выражение*). Общий вид ***блок-схемы разветвляющегося алгоритма*** изображен на *рис.2.5*. Эта схема значит, что для выполнения задания нужно сначала проверить некоторое условие и в зависимости от его выполнения выполнить команду 1 или команду 2.



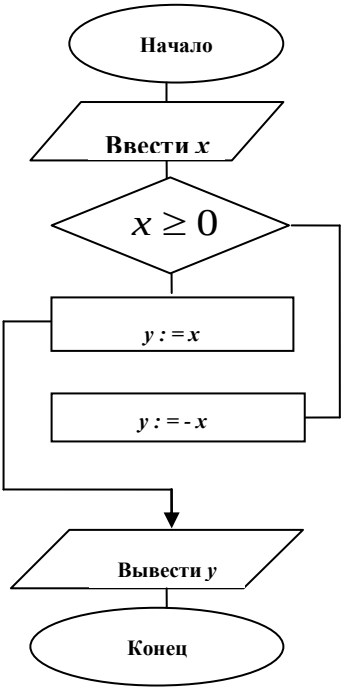


Рис. 2.5. Ветвящийся алгоритм Рис. 2.6. Пример ветвящегося алгоритма

Пример. Составьте алгоритм в виде блок-схемы для вычисления значений функции $y = |x|$ $\left(y = \begin{bmatrix} x, \text{ если } x \geq 0, \\ -x, \text{ если } x < 0. \end{bmatrix} \right)$ (*рис. 2.6*).

3. Циклические алгоритмы (алгоритмы повторения) – это алгоритмы, которые обеспечивают ***повторное выполнение*** некоторых команд конечное количество раз:

а) повторение с предусловием (цикл – „пока”). Общий вид ***блок-схемы алгоритма повторения с предусловием*** изображен на *рис. 2.7*. Эта схема значит, что для выполнения некоторого задания нужно сначала проверить выполнение условия. Если условие выполняется, то выполняем команду 1 и возвращаемся к проверке условия. Если условие не выполняется, то выполнение задания закончено.

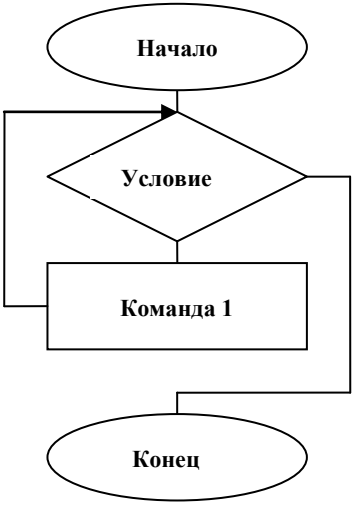


Рис. 2.7. Алгоритм повторения с предусловием (цикл-«пока»)

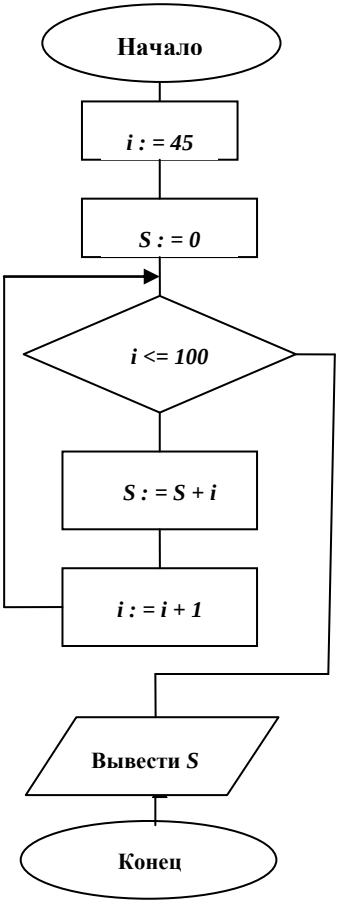


Рис. 2.8. Пример алгоритма повторения с предусловием

Пример. Составьте алгоритм в виде блок-схемы для вычисления суммы всех натуральных чисел с отрезка [45; 100] $(i := 45,46,47,...100;\ S := 45 + 46 + 47 + ... + 100)$ (*рис. 2.8).*

б) повторение с послеусловием (цикл – «до») (блок-схема алгоритма изображена на *рис. 2.9*).

Эта схема значит, что для выполнения некоторого задания сначала нужно выполнить команду 1, а потом проверить выполнение условия. Если условие не выполняется, то опять выполняем команду 1, а если выполняется, то выполнение задания заканчивается.

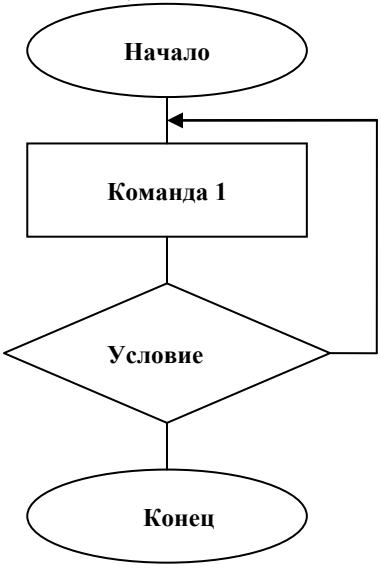


Рис. 2. 9. Алгоритм повторения с послеусловием (цикл – «до»)

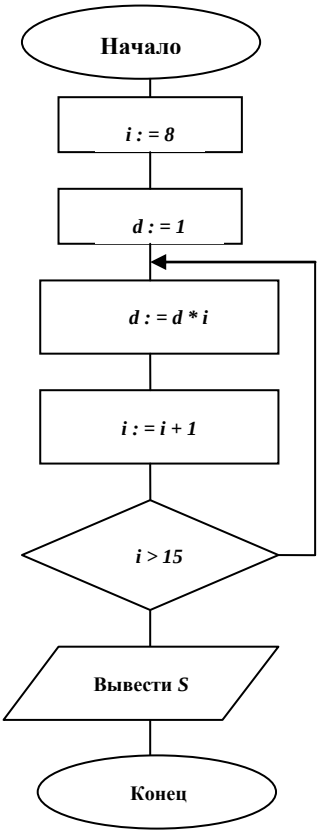


Рис. 2.10. Пример алгоритма с послеусловием

Пример. Составьте алгоритм в виде блок-схемы для вычисления произведения всех натуральных чисел с отрезка [8; 15] ($i := 8,9,10,\ldots,15$, $d := 1 \cdot 8 \cdot 9 \cdot \ldots \cdot 15$) (*рис. 2.10).*

Главный алгоритм - это алгоритм, выполнение которого приводит к достижению основной цели. **Вспомогательный алгоритм** - это алгоритм, предназначенный для достижения промежуточной цели. **Модули –** это части, из которых состоит алгоритм.

Проектирование сложных алгоритмов

1. Анализируют условие задачи, составляют общий план ее решения.
2. Если задача сложная, то разбивают ее на несколько простых заданий.
3. Проектируют модульную структуру алгоритма: описывают предназначение основного и вспомогательных алгоритмов, дают им названия.
4. Детализируют (создают, уточняют) необходимые вспомогательные алгоритмы для решения простых заданий.
5. Составляют основной алгоритм, который состоит из команд вызовов вспомогательных алгоритмов.

Этот метод называется проектированием алгоритма «сверху вниз» с пошаговой детализацией (уточнением) вспомогательных алгоритмов.

ВОПРОСЫ

1. Какие вы знаете виды моделей задач?
2. Какие есть основные этапы решения любой задачи с помощью компьютера?
3. Что такое алгоритм? Что такое программа?
4. Что такое алгоритм?
5. Какие вы знаете свойства алгоритма?
6. Какие есть формы записи алгоритмов?
7. Кто может быть исполнителем алгоритма?
8. Какие команды являются допустимыми для исполнителя?
9. Какие вы знаете три основных вида алгоритмов?

10. Что значат блок-схемы линейного и ветвящегося алгоритма?
11. Какие есть виды циклических алгоритмов?
12. Что значат блок-схемы циклических алгоритмов с предусловием и с послеусловием?
13. Чем отличаются циклические алгоритмы с предусловием и с послеусловием?
14. Что такое главный и вспомогательный алгоритмы?
15. Что такое модуль?
16. В чем заключается метод пошаговой детализации?

ЗАДАНИЯ

1. Приведите примеры линейного, ветвящегося, циклического алгоритмов.
2. Составьте алгоритм для вычисления площади круга и длины окружности, если известен радиус *R*.
3. Составьте алгоритм для вычисления площади равностороннего треугольника, если известна его сторона.
4. Составьте алгоритм для вычисления скорости движения *v*, если известны путь движения *s* и время движения *t*.
5. Составьте алгоритм для решения квадратного уравнения $ax^2 + bx + c = 0, \quad a \neq 0$.
6. Составьте алгоритм для вычисления значения функции $y = \frac{4x-2}{x+5} + \frac{1}{x}$.
7. Составьте алгоритм для вычисления значения функции $y = \begin{cases} 2x+1, & \text{если } x > 4; \\ 2x-1, & \text{если } x \le 4. \end{cases}$
8. Составьте алгоритм для вычисления суммы всех натуральных чисел с отрезка [45; 356].
9. Составьте алгоритм для вычисления произведения всех натуральных чисел с отрезка [12; 23].
10. Составьте алгоритм для вычисления количества всех натуральных чисел с отрезка [17; 100], делящихся на 3.

ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ №4

Задание 1. Составьте алгоритм в виде блок-схемы, 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. Составьте алгоритм в виде блок-схемы 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
$$y = \begin{cases} 6x^3 - x + 2, \text{ если } x \ge 3; \\ 7x^2 + 9x - 1, \text{ если } x < 3. \end{cases}$$

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 $z=\begin{cases}8y^2-4y+9, \text{ если } y\ge -7;\\ y^2+5y-1, \text{ если } x<-7.\end{cases}$

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 $z=\begin{cases}3x^3-4x+6, \text{ если } x\ge -3;\\ x^2+7x-2, \text{ если } x<-3.\end{cases}$

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 $z=\begin{cases}7y^2-y+9, \text{ если } y\ge -5;\\ 4y^2+3y-2, \text{ если } x<-5.\end{cases}$

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

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 $z=\begin{cases}8y^2-4y+9, \text{ если } y\ge -7;\\ y^2+5y-1, \text{ если } x<-7.\end{cases}$

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 $z=\begin{cases}3x^3-4x+6, \text{ если } x\ge -3;\\ x^2+7x-2, \text{ если } x<-3.\end{cases}$

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 $z=\begin{cases}7y^2-y+9, \text{ если } y\ge -5;\\ 4y^2+3y-2, \text{ если } x<-5.\end{cases}$

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 $y=\begin{cases}6x^3-x+2, \text{ если } x\ge 3;\\ 7x^2+9x-1, \text{ если } x<3.\end{cases}$

Вариант 19 $z = \begin{cases} 4x^3 + 2x + 2, \text{ если } x \geq 8; \\ 8x^2 + x - 1, \text{ если } x < 8. \end{cases}$

Вариант 20 $z = \begin{cases} 3y^2 - 4y + 9, \text{ если } y \geq -1; \\ 4y^2 + y - 1, \text{ если } x < -1. \end{cases}$

Вариант 21 $y = \begin{cases} x^2 - x + 5, \text{ если } x \geq 6; \\ 5x^2 + 2x - 9, \text{ если } x < 6. \end{cases}$

Вариант 22 $y = \begin{cases} x^3 - x + 2, \text{ если } x \geq -3; \\ 7x^2 + x - 1, \text{ если } x < -3. \end{cases}$

Вариант 23 $z = \begin{cases} x^3 + 5x + 2, \text{ если } x \geq -8; \\ 8x^2 + 8x - 1, \text{ если } x < -8. \end{cases}$

Вариант 24 $z = \begin{cases} y^2 - 4y + 9, \text{ если } y \geq 7; \\ y^2 + 5y - 2, \text{ если } x < 7. \end{cases}$

Вариант 25 $z = \begin{cases} 9y^2 - y + 3, \text{ если } y \geq -2; \\ 4y^2 + y, \text{ если } x < -2. \end{cases}$

Вариант 26 $y = \begin{cases} 4x^2 - x, \text{ если } x \geq 9; \\ 9x^2 + 2x - 3, \text{ если } x < 9. \end{cases}$

Вариант 27 $y = \begin{cases} 8x^3 - x + 5, \text{ если } x \geq 5; \\ x^2 + 4x - 1, \text{ если } x < 5. \end{cases}$

Вариант 28 $z = \begin{cases} 2y^2 - y + 1, \text{ если } y \geq -5; \\ 6y^2 + 6y - 1, \text{ если } x < -5. \end{cases}$

Вариант 29 $z = \begin{cases} y^2 - 5y + 5, \text{ если } y \geq -9; \\ 7y^2 + 5y - 7, \text{ если } x < -9. \end{cases}$

Вариант 30 $z = \begin{cases} y^2 - 5y, \text{ если } y \geq -3; \\ 9y^2 + y - 3, \text{ если } x < -3. \end{cases}$

Задание 3. Составьте алгоритм в виде блок-схемы для вычисления:

Вариант 1 суммы всех натуральных чисел с [3; 178].

Вариант 2 произведения всех натуральных чисел с [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; 189].
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; 99].

2.2. Основные понятия и системы программирования

Новые слова

Русский язык	***English***
автоматизация	automation
алфавит	alphabet, character set
высокий	high
вычислительный	calculation
грамматический	grammatical
интерпретатор	interpreter
компилятор	compiler routine
машина	machine
низкий	down

объектный	object
отслеживание	tracing, tracking
повторный	reentry
пользователь	user
процедурный	procedure
семантический	semantic
синтаксический	syntactic
система	system
среда	environment
средство	means, medium, arrangement, facility
тип	type
транслятор	translator
универсальный	universal
управление	control
уровень	level
устранение	elimination, removal, removing
шаг	step

Язык программирования – это некоторый алфавит и совокупность грамматических, синтаксических и семантических правил составления программы. **Программа** – ***это алгоритм, переведенный на язык программирования.*** Чтобы перевести алгоритм на любой язык программирования, нужно знать правила этого языка.

Людей, составляющих программы, называют **программистами.** Первой в мире программисткой стала Ада Лавлейс, дочь Джорджа Байрона (известного английского поэта). Она вместе с Чарльзом Беббиджем работала над проектом создания универсальной вычислительной машины. Выдвинула идею программного управления – машина должна выполнять предварительно заданные команды. Идея была реализована через 100 лет. В честь Ады Лавлейс назван один из языков программирования – Ада.

Языки программирования делятся на:

1) языки программирования низкого уровня (были созданы в 40-х годах 20-го столетия). Базируются на

командах-кодах для работы с адресами оперативной памяти и регистрами процессора. Используются для создания быстродействующих программ. Пример языка низкого уровня: Ассемблер;

2) языки программирования высокого уровня (созданы в 50-х годах 20-го столетия). Дают возможность записывать команды в виде предложений, близких к разговорным. Делятся на:

а) **языки процедурного типа** (например, Фортран, Бейсик, Паскаль, Си и т.д.);

б) **языки логического программирования** (например, Пролог);

в) **языки объектного программирования** (например, Object Pascal).

Кроме знания языка программирования, для составления программы необходимо иметь **систему программирования**, которая включает в себя *транслятор* языка (обязательно), библиотеки стандартных программ и средства получения подсказок от компьютера (желательно). **Транслятор** (переводчик) – это специальная программа для перевода программ пользователя, написанных на языке программирования высокого уровня, в машинные коды, понятные процессору. Есть два типа трансляторов:

1. **Интерпретатор** – это транслятор, который переводит команды программы в машинные коды по очереди и сразу их выполняет. Повторный запуск программы сопровождается повторным переводом. Для больших программ это неудобно. Интерпретаторы используются в системах программирования QBasic и других;
2. **Компилятор** – это транслятор, который анализирует команды программы на наличие ошибок и сразу переводит в машинные коды, создавая выполняемый файл, который может сохраняться в оперативной памяти или на диске. В этот файл нельзя вносить

изменения. Он может использоваться много раз. Компиляторы используются в системах программирования Turbo Basic, Turbo Pascal и других.

Для придания дополнительных удобств пользователю была создана *среда программирования.* **Среда программирования** – это программа, обладающая средствами автоматизации процессов подготовки и использования программ пользователя, а именно:

1) редактор текстов программ;
2) справочно-информационная система о языке программирования и среде;
3) библиотеки с полезными процедурами и функциями;
4) компилятор или интерпретатор, который не только констатирует факт ошибки в программе, но и указывает на тип и место ошибки, а в некоторых случаях предлагает пути устранения ошибок (или устраняет их);
5) средства выполнения всей программы или выполнения шаг за шагом с целью выявления семантических ошибок путем отслеживания значений величин, которые входят в программу.

Пример: Turbo Basic, Turbo Pascal, Delphi, Visual Basic и т.д. – это примеры среды программирования.

ВОПРОСЫ

1. Что такое язык программирования, программа?
2. Какие есть два вида языков программирования?
3. Что называется языком программирования низкого уровня?
4. Какие языки программирования являются языками высокого уровня?
5. Что такое транслятор?
6. Какие вы знаете виды трансляторов?
7. Что такое интерпретатор, компилятор?
8. Что называется средой программирования? Приведите пример среды программирования.

2.3. Конструкции и операторы языка программирования

Новые слова

Русский язык	*English*
атрибут	attribute
вершина	top element, top, vertex
вещественный	real
видимый	visible
визуальный	visual
возможность	possibility
вставление	stuffing
выравнивать	to line up, adjust, align, smooth
выражение	expression
высвечивать	highlight
глобальный	global
длина	length
добавить	extension, splitting
единица	one
заглавие	cap, capital
заготовка	pre-built
закладка	beetle, bookmark, tab
зарезервированный	reserved
значение	meaning, sense
идентификатор	identifier
идея	idea
известный	known
инициализировать	initialize
инспектор	inspector
квадратный	square
класс	class
ключевой	key
комплексный	complex
компонент	component
константа	constant
конструирование	construction
координаты	coordinates
латинский	Latin

метка	label, marking
множество	multitude, set
нажатие	press
невидимый	hidden
недопустимый	illegal, intolerable
необходимость	necessity, need
нестандартный	nonstandard
область	area
оболочка	enclosure, frame, framework, wrap
общедоступный	public
объединять	collate, combine crowd, pack
окружность	circle
оператор	operator
операция	operation
описание	description, account
определить	determinate, define
остаток	balance, excess , remainder, rest
оформление	encapsulate
параметр	parameter, quantity, rating
пароль	password
перекрыть	overlap
переменная	variable
перечисленный	listed, enumeration
периметр	perimeter
пиксель	pixel
площадь	area
позиция	position
порядковый	ordinal
появление	appearance
преобразование	transformation
применить	apply
приоритет	priority
пробел	blank, gap
проект	project
произвольный	random, free, unconditioned
прописной	capital letter, uppercase letter
прямоугольник	rectangle
путь	path

радиус	radius
результат	result
ресурс	resource
свойство	property
сетка	grid
символ	symbol
скалярный	scalar
скобки	brackets
скорость	speed
случайный	random
собственный	native, home, on-site
событие	event
совокупность	totality
содержать	contain
соответствовать	conform, fit
состав	composition, compound, repertoire
ссылка	pointer
сторона	side
строчный	string
структурированный	structured
технология	technology
тип	type
традиционный	tradition
треугольник	triangle
усовершенствование	improvement, perfection
учитывать	take into account
фигурные скобки	braces
формат	format
фрагмент	fragment
функция	function
целый	integer
частный	private
шаблон	mask, pattern
щелкнуть	click
элемент	element

2.3.1. Понятие о визуальном программировании

В ***традиционном (процедурном)*** программировании главной целью было *получение результатов* и *выведение их на экран*. В ***визуальном*** программировании важную роль играет *оформление результатов на экране*. В последнее время создаются *графические оболочки* для ***объектно-ориентированных*** языков программирования. Это называется **визуальным программированием.**

Среда визуального программирования Delphi – это графическая автоматизированная оболочка над объектно-ориентированной версией языка Pascal (**Object Pascal**). Если в языке Pascal структурными единицами являются данные и команды, то *в языке Object Pascal такой структурной единицей является* ***визуальный объект***, который называется **компонентом**. ***В этом заключается главное различие процедурных и объектно-ориентированных языков.*** Вообще *идея объектно-ориентированного программирования* заключается в объединении данных и средств их обработки в *тип*, который называется ***классом.*** Конкретной переменной некоторого класса и является ***объект (компонент).***

Для составления программы пользователю предоставляются графические (визуальные) средства *(компоненты*). Компоненты имеют некоторые свойства (атрибуты). Свойства могут принимать значения из некоторого предварительно фиксированного набора или значения, заданные пользователем. Пользователь решает задачи путем выбора компонентов и задания нужных значений их свойствам. Для решения задач составляют подпрограммы-процедуры, как и в традиционном языке Pascal. Такие процедуры называют ***методами объектов***.

Визуальное программирование заключается в конструировании решения поставленной задачи с помощью вставления компонентов в форму, задании

значений их свойствам и в применении или создании методов, необходимых для решения задачи.

Форма – *это компонент, который обладает свойствами окна Windows и предназначенный для размещения других компонентов (текстовых полей, полей редактирования, кнопок и т.д.)*. Компоненты на форме могут быть *видимыми* и *невидимыми*. ***Видимые компоненты*** предназначены для организации диалога с пользователем. ***Невидимые компоненты*** предназначены для доступа до системных ресурсов компьютера.

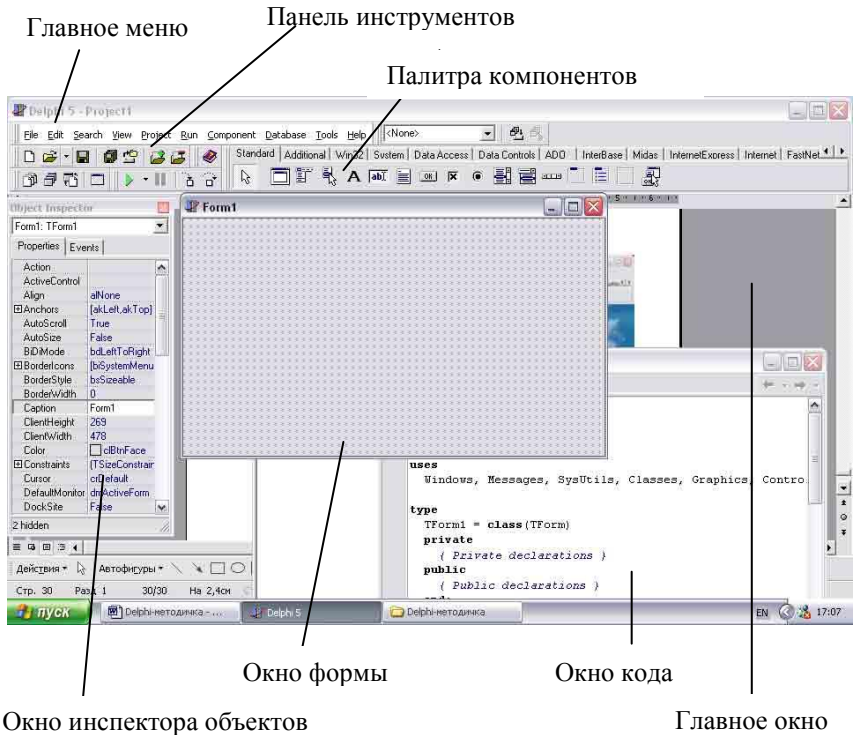


Рис. 2.11. Окно среды программирования Delphi

После запуска среды на экране получим четыре окна (*рис. 2.11*):

1) **окно формы** (Form);
2) **окно инспектора объектов** (Object Inspector) - предназначается для задания начальных значений свойствам объектов и их реакции на стандартные события;
3) **окно кода** (Code Editor) - это окно организовано как многостраничный блокнот открытых в данный момент файлов;
4) **главное окно**, которое содержит:
   а) главное меню (Main Menu);
   б) панель инструментов;
   в) палитру компонентов (Component Palette).

Главное меню

Главное меню активизируется клавишей F10. Включает такие элементы:

1. File (файл) – содержит стандартные команды для работы с файлами проекта:

а) New Application – создание нового проекта;
б) New Form – создание новой формы;
в) New Unit – создание нового модуля;
г) Open – открыть файл;
д) Close – закрыть файл;
е) Close All – закрыть все открытые файлы;
ж) Save – сохранить;
з) Save as – сохранить как;
и) Save All – сохранить все.

2. Edit (редактирование) – содержит команды:

а) Align to Grid – выравнивание компонентов относительно сетки;

б) Align – выравнивание компонентов между собою;

в) Bring to Front – задание порядка отображения пересекающихся компонентов (переместить на передний план);

г) Send to Back (переместить на задний план);

д) Size (изменение размера выбранного компонента);

е) Scale (масштабирование визуальных компонентов).

3. Search (поиск) – содержит стандартные команды поиска и замены фрагмента текста (Find, Replace, Search Again, Incremental Search и др.).

4. View – содержит команды визуализации элементов среды**.**

5. Project – содержит команды компиляции (Compile, Build All) и проверки синтаксиса программы (Syntax Check).

6. Run – содержит команды настройки и запуска программы.

7. Component – используется для создания и инсталляции новых компонентов.

8. DataBase – содержит команды вызова инструментов базы данных.

9. Tools – содержит команды для задания параметров среды.

10. Help – помощь.

Палитра компонентов находится в главном окне и имеет вид многостраничного блокнота. Каждой странице соответствует свой набор компонентов. Для ***размещения*** компонента на форме надо щелкнуть один раз на его пиктограмме и один раз в нужном месте формы. При двойном щелчке на пиктограмме компонента он будет помещен в центре окна формы. Для многоразового вставления одного и того же компонента нужно нажать клавишу *Shift* и щелкнуть на его пиктограмме – теперь можно щелкать в окне форме. Чтобы отказаться от этого режима, нужно нажать на кнопку с изображением стрелки. Выбранный компонент можно перемещать по форме, а также изменять размеры, перетягивая маркеры.

С помощью **инспектора объектов** можно задавать начальные значения свойств объекта и их реакцию на стандартные события. Окно инспектора объектов содержит список компонентов поточной формы, а также две закладки: свойств (*Properties*) и событий (*Events*). Чтобы активизировать окно инспектора объектов, используют клавишу **F11**. Рассмотрим это окно. **Закладка свойств** состоит из двух столбцов: левый содержит названия свойств компонентов, а правый – их значения. Свойства могут быть *простыми* или *комплексными*. ***Комплексные свойства*** состоят из набора других свойств. Такие свойства в инспекторе объектов обозначены символом « + », например, *+Font*. **Закладка событий** также содержит два столбца. В левом отображаются имена ***стандартных событий***, на которые объект может реагировать, а в правом – имена ***обрабатывающих методов (процедур)***, которые будут реализовывать реакцию на событие. Каждому стандартному событию соответствует название метода, которое появляется после двойного щелчка мышей в правом столбце. В этот момент в окно текста программы добавляется шаблон базового кода (процедуры) для соответствующего метода, который надо заполнить.

Для введения значений свойств числового и текстового типа (*Width, Name* и т.д.) используется ***стандартное поле введения***. Значения свойств перечисленного типа (*Align, Cursor* и т.д.) задаются *комбинированным списком*, из которого выбирают необходимое значение. Некоторые комплексные свойства (*Font, Picture, Glyph* и т.д.) используют диалоговые окна, набор управляющих элементов которых зависит от конкретного свойства.

Форма – это окно Windows, которое образуется в одном из возможных для окон стилей. Все внутреннее пространство является *рабочей областью*, которая имеет

сетку выравнивания для удобного размещения компонентов на форме. Для выполнения групповых операций несколько компонентов можно ***объединять***. Для этого необходимо нажать левую клавишу мыши и перемещением указателя охватить все нужные компоненты. В группу включаются компоненты, которые хотя бы частично попадают в охваченную область. Можно также ***добавить*** или ***удалить*** отдельный элемент. Для этого необходимо нажать клавишу *Shift* и, не отпуская ее, выбрать мышей нужный компонент на форме. ***Удаление*** отдельных элементов или группы выполняется клавишей *Delete*. ***Перемещение*** выделенного объекта в пределах формы выполняется мышей. Над компонентами и их группами можно выполнять операции ***вырезания, копирования*** в буфер обмена и ***вставления*** из буфера. ***Выравнивать*** компоненты можно как относительно окна формы, так и один относительно другого. Для этого используется команда ***Edit → Align*** главного меню или палитра выравнивания (команда ***View → Alignment Palette*** главного меню). Другая возможность – непосредственно задать свойства ***Top*** и ***Left*** компонентов. Компоненты в группе выравниваются относительно того элемента, который попал в группу первым.

ПРАКТИЧЕСКАЯ РАБОТА №2

Тема. Работа в среде программирования Delphi. Запуск программ на выполнение.

Цель. Научиться выполнять проекты в среде визуального программирования Delphi.

Ход работы

1. Запустите программу Delphi. Для этого выполните команды: ***Start → Programs → Borland Delphi → Delphi.***
2. Выполните команду: ***File → Open Project***.

3. Выберите указанную преподавателем папку, откройте, выберите нужный проект.
4. Выполните проект: ***Run → Run***.
5. Закройте проект.
6. Откройте любой другой проект, выполните, закройте.

2.3.2. Понятие проекта при объектно-ориентированном программировании

Проект – *это совокупность файлов, из которых Delphi создает готовую для выполнения программу.* В состав каждого проекта обязательно входят файлы:

1) ***файл проекта*** с расширением ***.dpr***. Это небольшой файл с программным кодом на языке *Object Pascal*, который содержит ссылки на все файлы проекта и инициализирует программу;
2) ***файлы описания всех форм***, входящих в проект: файл модуля с расширением ***.pas***, файл формы с расширением ***.dpr***. Каждой форме соответствует собственный модуль.
3) ***файл ресурсов программы*** с расширением ***.res***. В нем описаны ресурсы, которые не входят в форму, например, пиктограмма программы;
4) ***файл параметров проекта*** с расширением ***.dof***;
5) ***файлы параметров среды*** с расширениями ***.drf***, ***.dsk***, ***.dsm***, ***.ddp***, ***.cfg***, ***.dcu***. Эти файлы создаются после компиляции проекта.

Для ***сохранения*** Delphi-проекта необходимо задать имена модулей (автоматически предлагаются имена *Unit1.pas, Unit2.pas,* …) и имя проекта (*Project1.dpr*). Для перемещения Delphi-проекта на другой компьютер необходимо иметь только файлы типов *.dpr, .dfm, .pas, .res*. Остальные файлы создаются автоматически. **Редактор кода** находится в отдельном окне, которое организовано как многостраничный блокнот открытых в данный момент

файлов. Во время открывания нового проекта в модуль *Unit1.pas*, который соответствует форме *Form1*, редактор автоматически заносит ***программный код*** описания этой формы. Во время добавления новых компонентов в окно формы в программу автоматически заносятся коды с описаниями параметров этих компонентов (высота, ширина, размещение, стиль и т.д.). Добавление конкретного объекта или применение к нему метода приводит к появлению заготовки базового кода соответствующей процедуры в окне редактора. *Заготовка (шаблон)* состоит из *заглавия процедуры* и ключевых слов **begin** и **end**. Заготовку заполняет пользователь. Заканчивается модуль собственной командой **end.**

Проект состоит из *форм*. Любой форме соответствует определенный модуль. Файл проекта с расширением *.dpr* их объединяет.

Модуль имеет такой общий вид:

```
Unit Unit1;{имя модуля}
Interface {раздел объявления процедур и функций}
    Uses {список используемых модулей, например}
    Windows, Messages, SysUtils, Classes, Graphics,
    Controls, Forms, Dialogs;
    Type {описание классов и типов объектов,
например}
        TForm = class(TForm)
          Button1: TButton;
          Procedure Button1.Click(Sender: TObject);
          …
      private {частные объявления}
      public {общедоступные объявления}
      end;
    Var {описание глобальных переменных}
      Form1: TForm1;
```

{*форвард-объявления процедур и функций пользователя, например*}
Procedure Information;
Procedure SetPicture;
…
function {*имя функции*};
…
implementation {*раздел описания процедур и функций*}
{$R *.DFM} {*прибавляется файл с описанием формы*}
procedure TForm1.Button1Click(Sender: TObject);
begin
список операторов;
end;
end. {*конец модуля*}

ПРАКТИЧЕСКАЯ РАБОТА №3

Тема. ***Создание проекта «Анкета студента»*** с личными данными и двумя фотографиями, которые перекрывают одна другую и должны появляться в результате нажатия на кнопки.

Цель. Получение элементарных навыков работы в среде Delphi. Познакомиться с такими объектами: форма (**Form**), текстовое поле (**Label**), изображение (**Image**), кнопка (**Button**) и их основными свойствами: подпись (**Caption**), цвет (**Color**), шрифт (**Font**), видимость (**Visible**), ширина (**Width**), высота (**Height**) и другими.

Объекты: форма, текстовое поле, изображение, кнопка.

Теоретические сведения. Объект ***Form*** используют для создания программой нового окна. Рассмотрим такие свойства формы:

Свойство	***Описание свойства***	***Примеры значений***
ActiveControl	Для задания активного объекта (фокуса) в форме	Button1, Edit2

AutoScroll	Наличие в форме полос прокрутки	True, False
BorderStyle	Возможность изменять размеры окна	bsSizeable (окно с произвольными размерами), bsDialog , bsNone (окно с фиксированными размерами)
Width, Height	Ширина и высота окна в пикселях	503, 224 (числовое значение)
Font	Шрифт	Комплексное свойство, задается в диалоговом окне
HorizScrollBar VertScrollBar	Параметры полос прокручивания	Комплексное свойство
Icon	Задание пиктограммы, которая будет в заглавии формы во время выполнения программы	(None) – стандартная пиктограмма для Delphi, или загруженная с конкретного файла *.ico
Name	Имя формы	Form1 (идентификатор)
Caption	Заглавие формы	Любая строка символов
Color	Цвет фона формы	clGreen, clInfoBk (перечислительный тип) или $004525B1 (числовое значение – задается в диалоговом окне)
Cursor	Вид указателя мыши на форме во время выполнения проекта	crDrag, crCross, crHelp, crArrow (перечислительный тип)
Enabled	Доступность для действий объектов в форме во время выполнения	True, False

Left, Top	Координаты левого верхнего угла окна в пикселях	200, 108 (числовое значение)
Position	Размещение и размеры окна в момент запуска программы	poScreenCenter, poDesigned
WindowState	Состояние окна в момент запуска программы	wsNormal, wsMaximized, wsMinimized

Объект ***Label*** используют для создания текстовых полей (надписей) в окне программы. Кроме аналогичных к приведенным в предыдущей таблице свойств ***Width***, ***Height***, ***Font***, ***Color***, ***Name***, ***Caption***, ***Cursor***, ***Enabled***, ***Left***, ***Top***, он обладает еще и такими:

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

Объект Image используют для вставления графических объектов из файлов типа *.bmp, *.emf, *.ico, *.wmf в форму. Кроме известных свойств **Align**, **Width**, **Height**, **Name**, **Cursor**, **Enabled**, **Left**, **Top**, **Visible** используют такие:

Свойство	**Описание свойства**	**Примеры значений**
Center	Выравнивание рисунка к центру относительно поля, которое его содержит	True, False

Picture	Имя графического файла	Задается в диалоговом окне
Stretch	Приведение размера изображения к заданным размерам объекта	True, False
AutoSize	Приведение размера объекта к реальным размерам изображения	True, False

Объект ***Button*** используют для создания кнопок на форме. Кнопки имеют такие свойства: ***Visible***, ***Width***, ***Height***, ***Font***, ***Color***, ***Name***, ***Caption***, ***Cursor***, ***Enabled***, ***Left***, ***Top*** и другие.

Рис. 2.12. Запуск программы Delphi

Ход работы

1. Загрузите среду визуального программирования Delphi. Запуск системы визуального программирования Delphi осуществляют щелканьем на пиктограмме Delphi или с помощью каскадного меню ***Start*** *(Пуск)* → ***Programs*** *(Программы)* → ***Borland Delphi x.0*** → ***Delphi x.0***, где x – версия программы *(рис. 2.12).*

2. Запустите ваш проект *Project1* на выполнение.

Запустить программу можно несколькими способами:

- выполнить команду ***Run*** → ***Run*** главного меню *(рис. 2.13)*;
- щелкнуть на кнопке ***Run*** панели инструментов;
- нажать функциональную клавишу ***F9***.

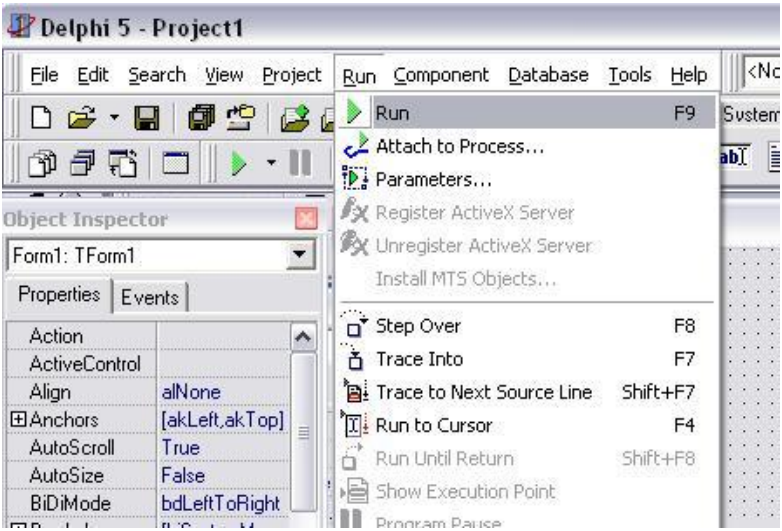


Рис. 2.13. Выполнение проекта

3. Сохраните созданную программу в своей личной папке. Для этого выберите команду главного меню ***File*** → ***Save All*** *(«Сохранить все») (рис. 2.14).*

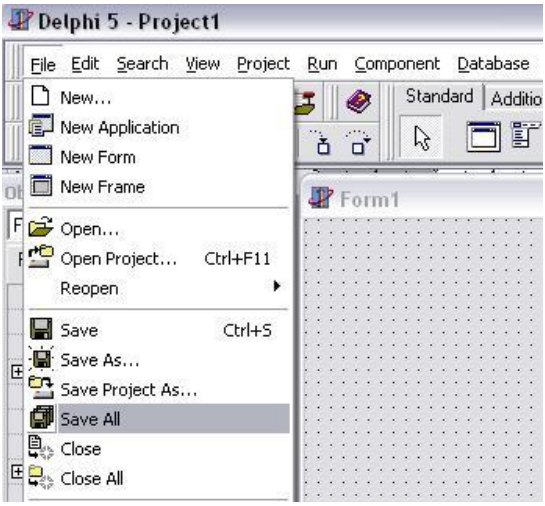


Рис. 2.14. Сохранение проекта

Появится окно *Save Unit As*. В строке под заглавием *"Save in" («Сохранить в»)* с помощью выпадающего меню ***выберите имя рабочего диска***, после чего ***создайте собственную папку, назовите вашим именем*** *(рис. 2.15, 2.16).*

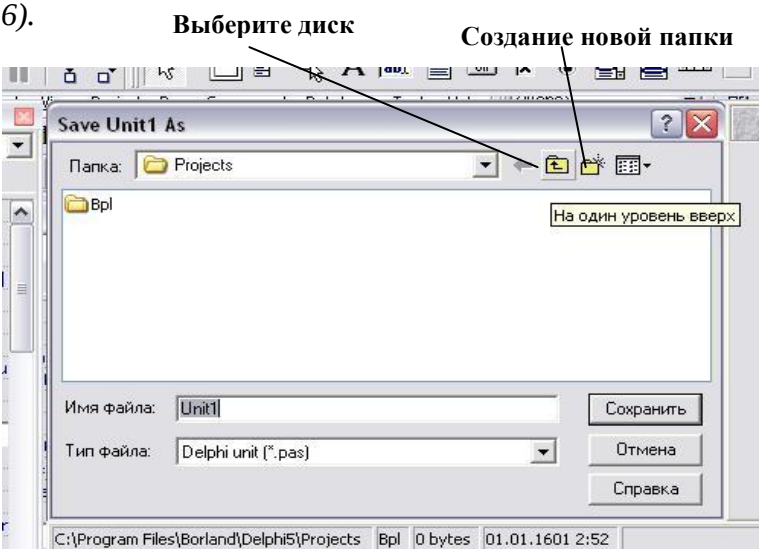


Рис. 2.15. Выбор диска и создание своей папки

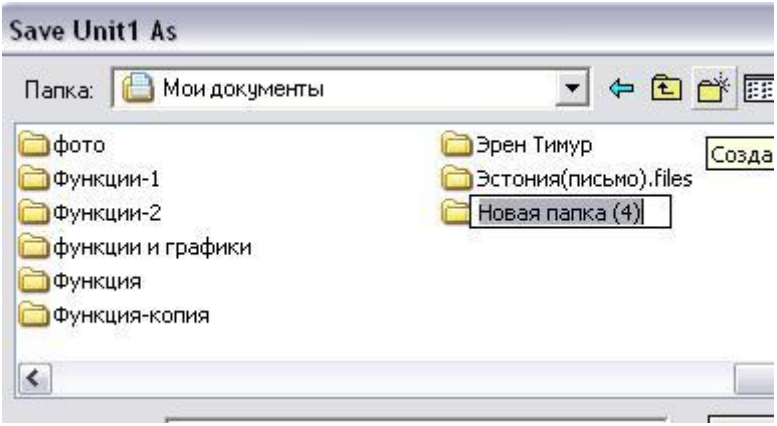


Рис. 2.16. Задание имени папки (ваше имя)

Откройте папку. Сохраните: ***Unit1.pas*** → ***Save (Сохранить)***. *(рис. 2.17)*

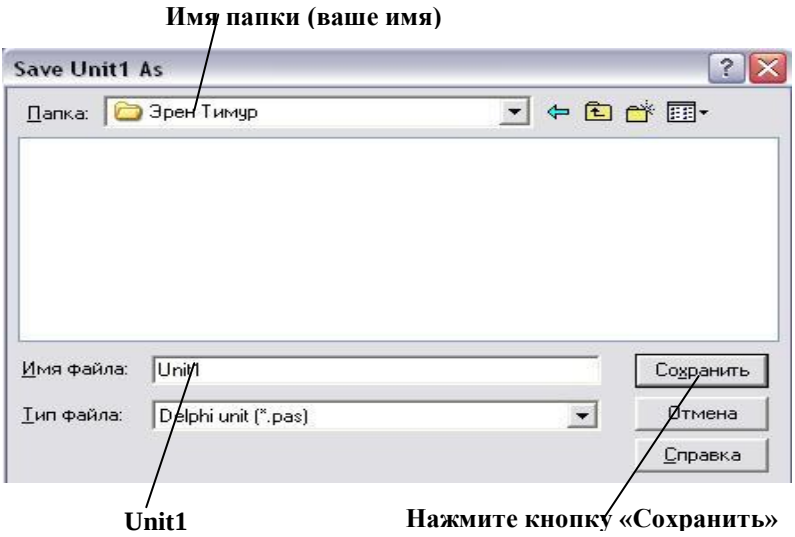


Рис. 2.17. Сохранение модуля Unit1

В следующем окне *"Save Project As"* нажмите кнопку *«Сохранить» ("Save").* Имя файла проекта ***Project1.dpr*** не изменяйте *(рис. 2.18).*

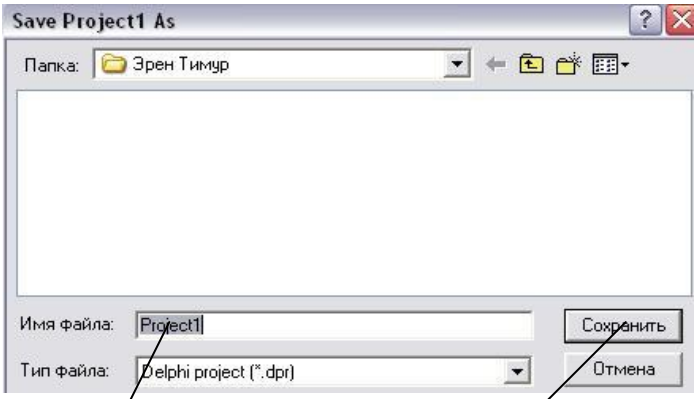


Рис. 2.18. Сохранение проекта Project1

4. Задайте название формы «Анкета». Для этого в окне свойств формы ***Object Inspector*** в строке ***Caption*** напишите *«**Анкета**» (рис. 2.19).*

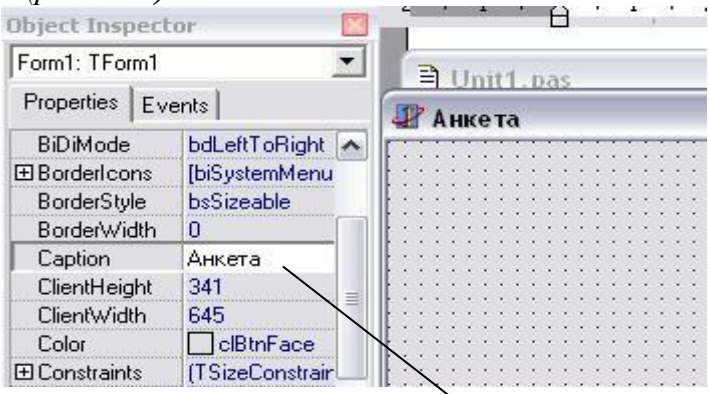


Рис. 2.19. Задание названия формы (Caption) «Анкета»

5. Выберите цвет фона формы. Для этого в окне свойств формы ***Object Inspector*** в строке ***Color*** с помощью выпадающего меню выбрать любой цвет *(рис. 2.20).*

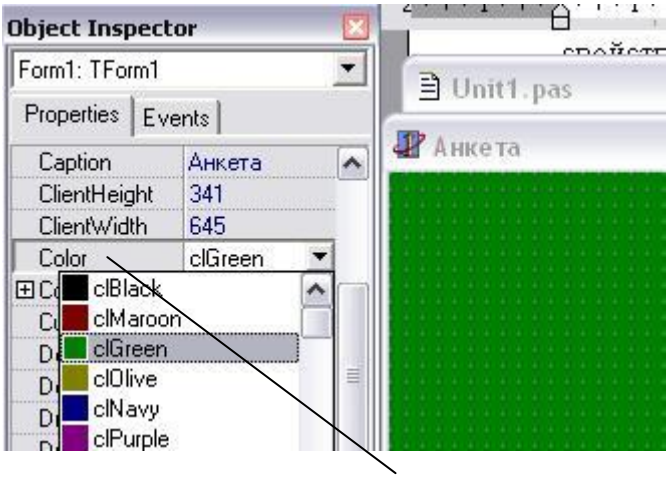


Рис. 2.20. Выбор цвета формы (Color)

6. Задайте пиктограмму формы. Для этого в щелкните в строке ***Icon***, потом щелкните на появившейся кнопке комплексного свойства ... для вызова диалогового окна выбора пиктограммы *Picture Editor (рис. 2.21).*

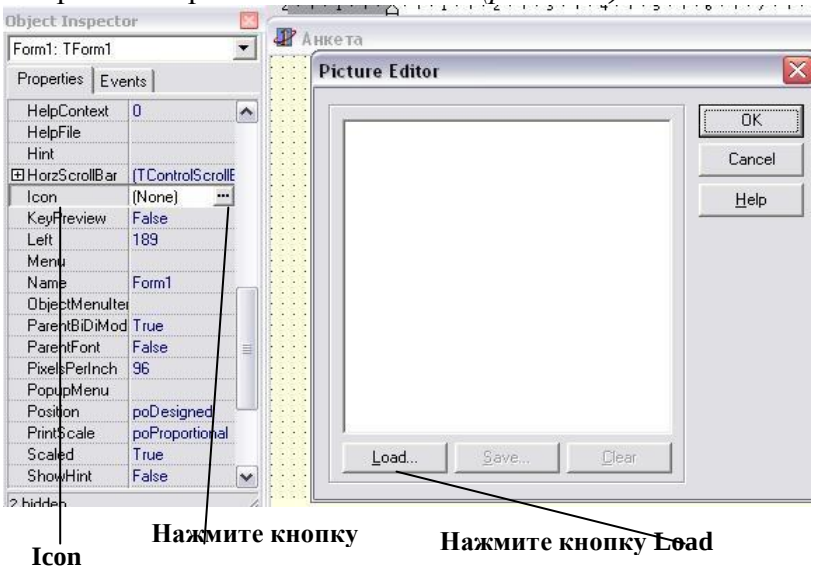


Рис. 2.21. Выбор пиктограммы заглавия формы

Щелкните на кнопке ***Load*** *(загрузить)* и в окне *Load Picture* выберите файл рисунка, который вам нравится, только с расширением ****.ico.*** Нажмите кнопку ***Открыть (Open)*** *(рис. 2.22)*. Подтвердите свой выбор нажатием кнопки ***Ok*** *(рис. 2.23)*. Этот рисунок (***пиктограмма***) будет размещен рядом с названием формы *(«Анкета»)* в ее верхней строке (строке заглавия).

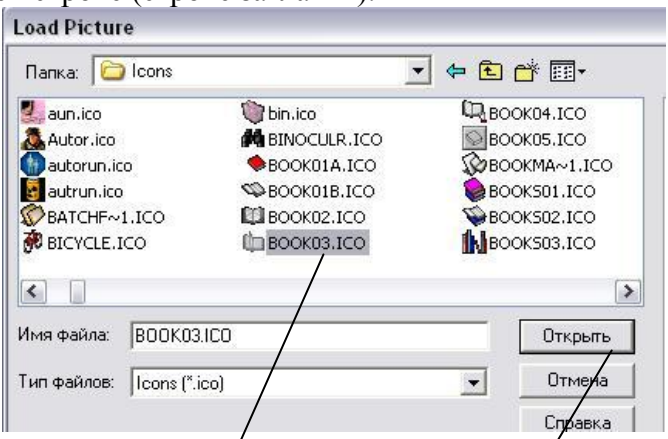


Рис. 2.22. Выбор файла для пиктограммы

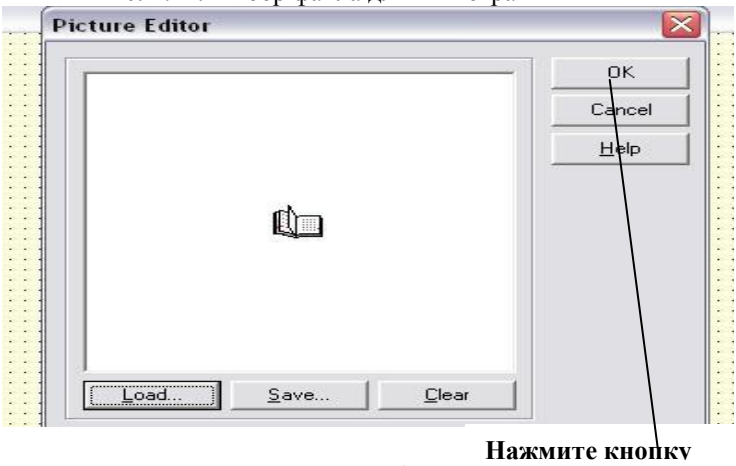


Рис. 2.23. Подтверждение выбора пиктограммы

8. Сохраните форму еще раз (File → Save).

9. Запустите проект на выполнение *(Run → Run).*

10. Вставьте в форму текстовое поле *(Label)* с текстом *„Анкета студента”.* Щелкните мышей на пиктограмме **Label**  на закладке *Standard* палитры компонентов *(рис. 2.24).*

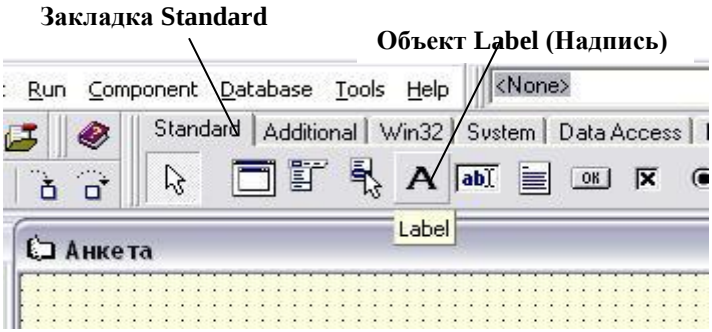


Рис. 2.24. Выбор объекта Label (Надпись)

Разместите выбранный объект на форме, перетягивая его левой кнопкой мыши *(рис. 2.25).* Задайте размеры с помощью мыши.

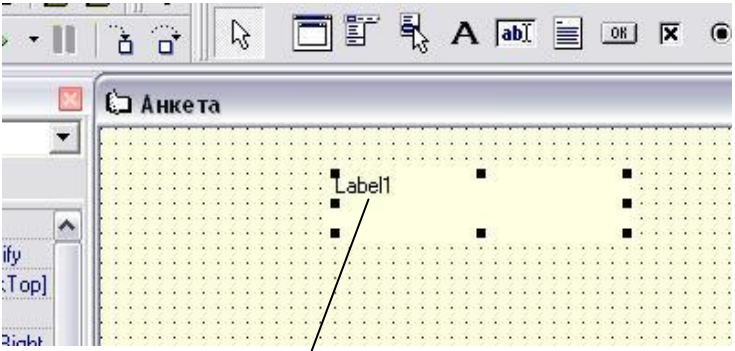


Рис. 2.25. Размещение объекта Label на форме

11. В окне Object Inspector задайте свойство объекта *Label*

Caption: *Анкета студента.*

Примечание. **В окне *Object Inspector* отображается список свойств только выбранного объекта.** Выбранный объект выделяется ***маркерами***.

Выберите объект ***Label*** (щелкните по нему левой кнопкой мыши) и в окне *Object Inspector* задайте свойство ***Caption*** как ***Анкета студента*** *(рис. 2.26).*

Рис. 2.26. Задание свойства Caption для объекта Label

12. В окне *Object Inspector* задайте комплексное свойство *Font* для объекта *Label* так:

Font*: Times New Roman Cyr;*

Font style: *Bold;*

Size: *16;*

Color: *Green (зеленый).*

Для этого в окне ***Object Inspector*** в строке ***Font*** нажмите кнопку комплексного свойства ... и задайте значения свойств объекта *(рис. 2.27).* Если объект *Label1* в данный момент не выбран, сначала *активизируйте* его. Задать параметры шрифта можно также с помощью кнопки ⊞, которая находится рядом с названием свойства *Font.*

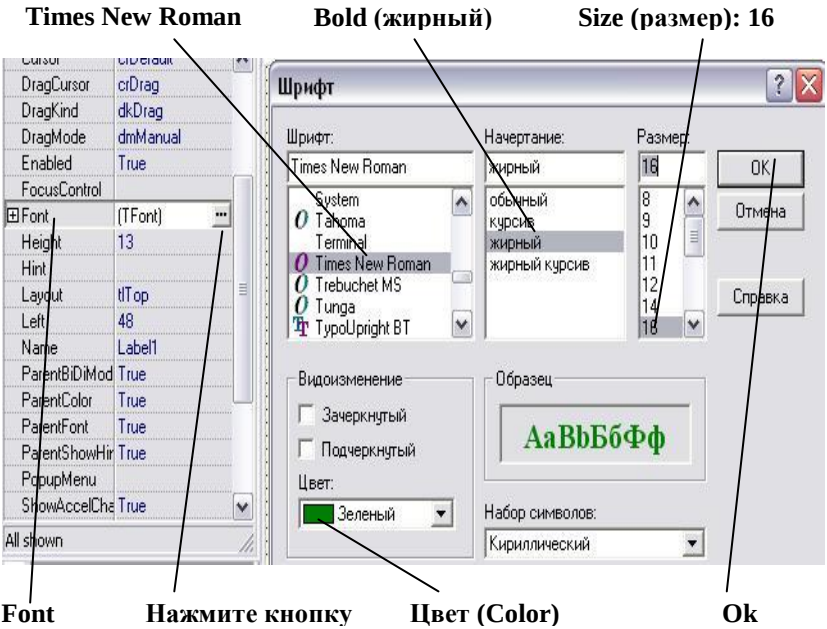


Рис. 2.27. Задание свойства Font для объекта Label

13. Аналогично вставьте в форму еще три текстовых поля (объекта ***Label***). Задайте их свойства ***Caption*** так:

Label1: *Ваши фамилия и имя, номер группы*.

Label2: Студент ФРИС.

Label3: *Ваш год рождения*. ***Например*** *(рис. 2.28)*:

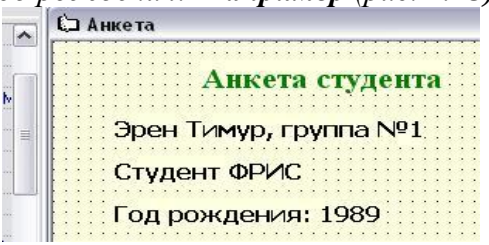


Рис. 2.28. Пример анкетных данных

14. Вставьте в форму объект *Image (изображение).* Для этого щелкните один раз левой кнопкой мыши на пиктограмме ***Image*** *(Изображение)* закладки

Additional *(Дополнительные)* палитры компонентов *(рис. 2.29)* и на форме обведите контуры изображения.

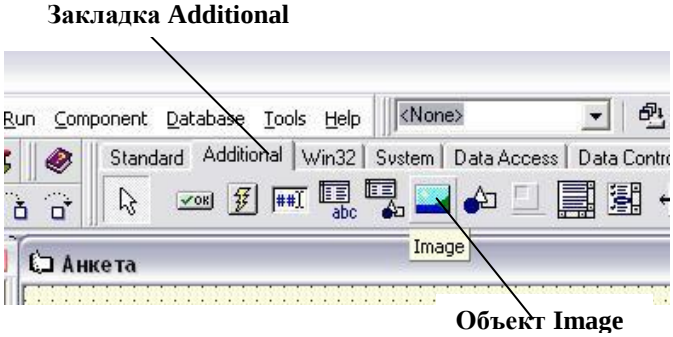


Рис. 2.29. Добавление объекта Image

15. Добавьте *фотографию №1* с помощью свойства *Picture (Рисунок)* объекта *Image1*. Для этого выделите объект ***Image1*** и в окне *Object Inspector* активизируйте строку ***Picture***. Щелкните на кнопке ... для вызова окна выбора рисунка *Picture Editor*. Щелкните на кнопке ***Load*** *(Загрузить)* и в окне *Load Picture* выберите файл с фотографией. Нажмите кнопку ***Open***. Подтвердите свой выбор нажатием кнопки ***Ok***.

Задайте свойства объекта ***Image*:**

Stretch	True
Visible	False

16. Аналогично добавьте еще один объект *Image2* сверху существующего, вставьте фотографию №2 с помощью свойства *Picture*, аналогично задайте свойства *Stretch* *(True)* **и *Visible*** *(False).* Во время накладывания объектов может возникнуть необходимость использовать команды ***Send To Back*** *(Переслать назад)* или ***Bring To Front*** *(Перенести вперед)*, которые есть в ***контекстных меню*** объектов. (вызываются щелканьем правой кнопки мыши по объекте) *(рис. 2.30).*

Рис. 2.30. Команды контекстного меню объекта Image

17. Вставьте в форму кнопки для засвечивания фотографий – два объекта *Button* с названиями *Button1* и *Button2*. Пиктограмма OK объекта *Button* находится на закладке *Standard* палитры компонентов *(рис. 2.31)*.

Рис. 2.31. Объект Button (Кнопка)

Разместите объекты ***Button1*** и ***Button2*** на форме. Задайте для них свойства ***Caption*** так *(рис. 2.32)*:

Button1: *Фотография 1;*

Button2*: Фотография 2.*

Рис. 2.32. Добавление кнопок

18. Запрограммируйте кнопку *«Фотография №1»* так, чтобы после ее нажатия появлялась первая фотография. Для программирования кнопки *Button1* необходимо два раза щелкнуть по ней левой кнопкой мыши. В результате активизируется окно текста программы с заготовкой процедуры *Button1Click*, которая обрабатывает событие щелканья на кнопке *Button1*:

```
procedure Tform1.Button1Click(Sender: TObject);
begin

end;
```

В заготовку необходимо вставить текст программы реакции на это событие. Процедура будет иметь такой вид:

```
procedure Tform1.Button1Click(Sender: TObject);
begin
Image1.Visible:= True;Image2.Visible:= False;
end;
```

С помощью данной процедуры свойство видимости для объекта *Image1* включаем и это же свойство для объекта

Image2 выключаем. Для кнопки «Фотография №2» действия будут обратными. Обратите внимание на использование составных имен типа *Image1.Visible*, в которых ***название объекта*** от его ***свойства*** отделяется ***точкой***. Такие составные имена дают доступ до значения конкретного свойства некоторого объекта.

19. Запрограммируйте кнопку *«Фотография №2»* соответственно ее назначению. Текст процедуры для этой кнопки будет иметь вид:

```
procedure Tform1.Button1Click(Sender: TObject);
begin
Image2.Visible:= True;Image1.Visible:= False;
end;
```

Чтобы создать такую процедуру быстро, можно скопировать две команды присвоения с предыдущей процедуры в новую (команды главного меню ***Edit → Copy*** и ***Edit → Paste***) и поменять выражения справа.

20. Еще раз сохраните программу *(File → Save).*

21. Выполните программу *(Run → Run)* и проверьте действие кнопок.

22. Создайте выполняемый файл программы (файл с расширением *exe*).

Выполните команду главного меню ***Project → Build Project1*** *(Сконструировать проект).* Автоматически будет создан файл ***Project1.exe***. Он будет находиться в той же папке, в которой сохранен файл *Project1.dpr*.

23. Закройте Delphi, откройте свою папку и откройте свой проект. Откройте собственную папку. Выполните файл с именем проекта и пиктограммой 

.

24. Продемонстрируйте созданный проект преподавателю.

Дополнительные задания

1. Вставьте в форму *третью* фотографию и соответствующую ей кнопку с надписью ***«Фотография №3».***
2. Добавьте *четвертую* кнопку с надписью ***«Скрыть все фотографии»***, при щелчке по которой все три фотографии должны становиться *невидимыми*.
3. Создайте проект *«Времена года»,* который содержит 4 кнопки с надписями: ***«Зима», «Весна», «Лето», «Осень».*** При щелчке на каждой кнопке должна появляться фотография соответствующего времени года *(рис. 2.33):*

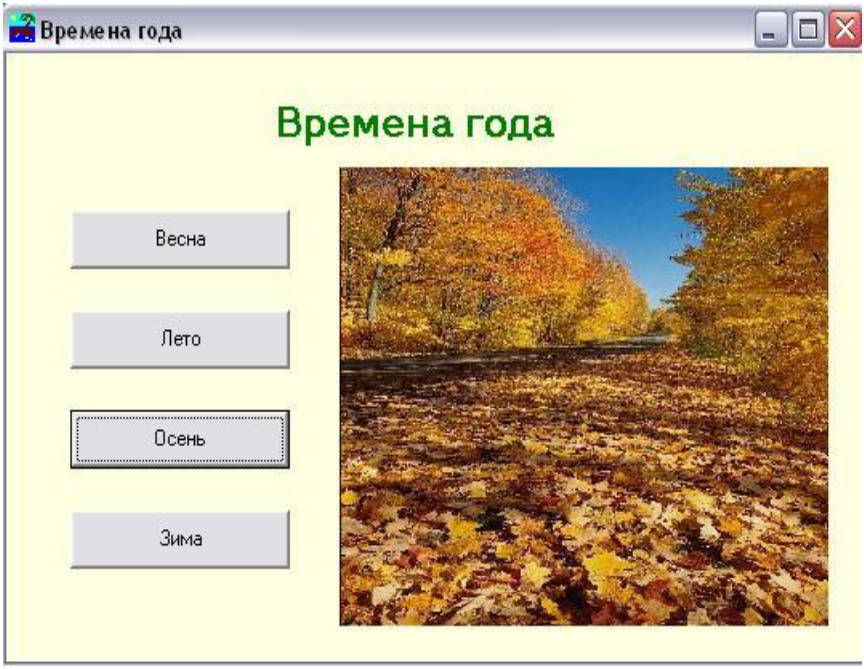


Рис. 2.33. Форма «Времена года»