МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пермский государственный национальный исследовательский университет»

Колледж профессионального образования

ОСНОВЫ ПРОГРАММИРОВАНИЯ

Методические рекомендации

для практических работ по изучению дисциплины

для студентов Колледжа профессионального образования

специальности

09.02.03 Программирование в компьютерных системах

Утверждено на заседании ПЦК
Информационных технологий
Протокол № 9 от 23.05.2018
председатель _____Н.А. Серебрякова

Пермь 2018

Пермь 2018

Составитель:
Серебрякова Н.А., преподаватель ФГБОУ Пермский государственный национальный исследовательский университет, Колледж профессионального образования

Основы программирования: методические указания по выполнению практических заданий по дисциплине для студентов Колледжа профессионального образования специальности 09.02.04 Информационные системы (по отраслям)/ сост. Н.А.Серебрякова; Колледж проф. образ. ПГНИУ. – Пермь, 2018. -67 с.

Методические указания «Основы программирования» разработаны на основе требований Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.03 Программирование в компьютерных системах для оказания помощи студентам по дисциплине «Основы алгоритмизации и программирования». Содержат практические и лабораторные задания и примеры их выполнения по дисциплине «Основы программирования». Предназначены для студентов Колледжа профессионального образования ПГНИУ специальности 09.02.03 Программирование в компьютерных системах (СПО) всех форм обучения.

СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Основное содержание методических указаний по выполнению практических работ ориентировано на обучение структурной методике программирования. Поэтому первый практические работы предназначены для освоения программирования основных алгоритмических конструкций: линейной, ветвлению, циклам, а также способам составления подпрограмм. Последующие практические работы посвящены работе с разными видами структур данных: массивы, записи, множества, файлы.

Практические работы проводятся после изучения соответствующих разделов и тем учебной дисциплины в соответствии с учебным планом специальности. Целью практических работ является закрепление теоретических знаний и приобретение практических умений и навыков:

1) использования базовых алгоритмических структур;
2) правил построения алгоритмов;
3) описания и использования типов и структур данных языка программирования;
4) тестирования и отладки программ.

1.ТЕМАТИЧЕСКИЙ ПЛАН ДИСЦИПЛИНЫ

Наименование тем и разделов			
	лекции	**Практические занятия**	**лабораторные занятия**
Раздел 1. Язык программирования Паскаль	**4**	**0**	**0**
Тема 1.1 Основные понятия языка Pascal.	8	0	0
Тема 1.2.Типы данных и выражения. Операторы языка	4	0	0
Тема 1.3 Массивы. Строки.	4	0	0
Тема 1.4 Подпрограммы	4	0	4
Тема 1.5 Файлы. Множества. Записи.	6	0	6
Тема 1.6 Модули. Объекты.	6	0	6
Раздел 2 Программирование в среде Lazarus		**0**	
Тема 2.1 Среда Lazarus	12	2	6
Тема 2.2 Компоненты	12	4	6
Тема 2.3 Линейные алгоритмы. Разветвляющиеся алгоритмы Циклические алгоритмы	14	4	14
Тема 2.4 Массивы. Строки. Файлы	10	2	10

2.СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ

Раздел 1 Язык программирования Паскаль

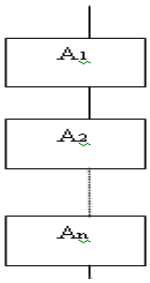
Практическая работа № 1

Составление блок-схем алгоритмов.

Цель: Изучение способов задания алгоритмов, приобретение практических навыков составления блок - схем решения задач на ЭВМ.

Краткая теория

Структура следования - представляет собой последовательность размещенных блоков или групп блоков друг за другом.



 - функциональный блок

$i=\overline{1.n}$

Функциональный блок – это любая базовая структура или их комбинация

Задание 1. Описать исходные, выходные и промежуточные данные следующих задач:

Задача 1. Даны стороны прямоугольника a и b. Найти его периметр p,и площадь s этого прямоугольника.

Задача 2. Скорость звездолета 100 км/час. Звездолет летит до некоторой звезды, свет от которой до Земли идет 14 минут. Определить, сколько времени потребуется звездолету, чтобы долететь до данной звезды.

Задача 3. В треугольнике ABC известны длины сторон a, b, c. Вычислить длину высоты, проведенной из вершины A.

$$h=\frac{2\sqrt{p(p-a)(p-b)(p-c)}}{a}$$

Задача 4. Дано два числа. Найти наименьшее.

Задача 5.Студент сдал четыре экзамена a, b, c, d и получил оценку по каждому из экзаменов по пятибалльной шкале. Определить средний балл студента.

Практическая работа 2,3

Тема: составление программ разветвляющейся структуры

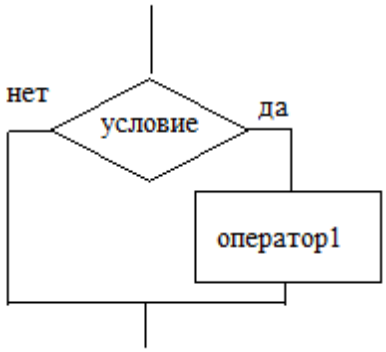
Краткая теория

Ветвление — алгоритм, в котором предусмотрены разветвления, указанные в последовательности действий на два направления в зависимости от итогов проверки заданного условия**.** То есть такой алгоритм, обязательно содержит условие и в зависимости от результата выполнения условия происходит выбор действия.

Таких примеров можем привести много из обычной жизни и наук. К примеру, физика: **Если** удар упругий, **то** масса тела сохраняется, **иначе** масса изменяется.

Алгоритмическая конструкция ветвление программируется с помощью **условного оператора If**, который может быть представлен двумя вариантами, представленными в таблице 1.

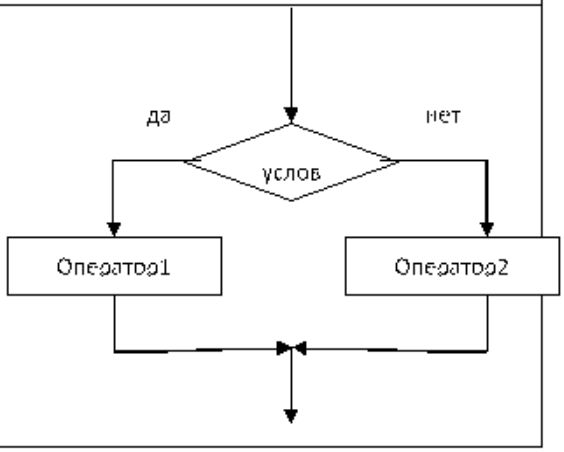
Таблица 1

Конструкция	**Графическое представление блок - схема)**
1 Вариант — неполное ветвление	
If <условие>**Then**<оператор> Неполное ветвление — в зависимости от результата проверки условия либо выполняются действия одной ветви «да» (оператор), либо эти действия не выполняются.	

2 вариант - полное ветвление

if < условие > **Then** <оператор 1 > **else** <оператор 2 >

Полное ветвление - в зависимости от результата проверки условия выполняются только оператор 1 ветви "да" или только оператор 2 ветви "нет"



Условие — это логическое выражение, которое может принимать одно из двух значений: true (истина — условие выполняется) и false (ложь — условие не выполняется). В условии используются операции отношения (=, <> , >,<, >=, <=) и логические операции (and (И), or (ИЛИ), xor (исключающее ИЛИ), not (отрицание)). Если требуется проверить несколько условий, их объединяют с помощью логических операций. Примеры логических выражений: A<2 или (x<>0) and (y<>0)

Если между служебными словами стоят несколько операторов, то они заключаются в операторные скобки Begin…End

Правила пунктуации при записи операторов

Точка с запятой не ставится в разделах описаний после зарезервированных слов uses, label, type, const, var и ставится после завершения каждого описания

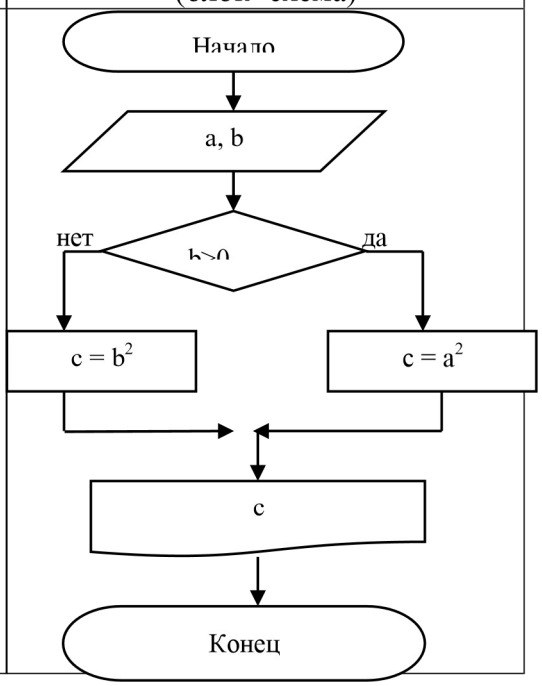
Точка с запятой не ставится после begin и перед end, так как эти слова являются операторными скобками, а не операторами;

Точка с запятой является разграничителем операторов, ее отсутствие между операторами вызывает ошибку компиляции;

В операторах цикла точка с запятой не ставится после служебных слов while, repeat, do, until;

В условных оператора точка с запятой не ставится после then и перед else

Рассмотрим пример: даны 2 вещественных числа. Если числа положительные, то возвести в квадрат первое число, иначе возвести в квадрат второе число.

Листинг программы	Графическое представление (блок -схема)
Program Primer_1; Uses crt; {Обозначим 1- ое число через переменную a, 2-ое через переменную b, результат — c} Var a, b, c: real; Begin Clrscr; Writeln('Введите первое число '); Readln(a); Writeln('Введите второе число '); Readln(b); **If** (a>0) and (b>0) **Then** c:=sqr(a) **Else** c:=sqr(b); Writeln('Результат = ', c:5:2); End.	

Задания для выполнения практической работы

Вариант 1

1.Даны три действительные числа. Возвести в квадрат те из них, значения которых положительны, и в четвертую степень — отрицательные.

2. Написать программу, которая вычисляет частное от деления двух чисел. Программа должна проверять правильность введенных пользователем данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление частного.

Введите в одной строке делимое и делитель, затем нажмите <Enter>

->12 0

Вы ошиблись. Делитель не должен быть равен нулю.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 2

1.Даны два действительные числа. Если числа положительны найти их сумму, если отрицательны — произведение

2. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.

Введите сумму покупки и нажмите <Enter>

->1200

Вам предоставляется скидка 10%

Сумма покупки с учетом скидки: 1080.00 руб.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 3

1.Даны действительные числа x и y, не равные друг другу. Меньшее из этих чисел заменить половиной их суммы, а большее — их удвоенным произведением

2. Написать программу проверки знания даты основания Санкт-Петербурга. В случае неверного ответа пользователя программа должна выводить правильный ответ. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

В каком году был основан город Пермь?

Введите число и нажмите <Enter>

->1705

Вы ошиблись, город Пермь был основан в 1723 году.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 4

1.Данные два вещественных числа. Если первое число больше второго, то возвести его в третью степень, если равно второму — прибавить к нему второе число

2. Написать программу определения стоимости разговора по телефону с учетом скидки 20%, предоставляемой по выходным дням. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости разговора по телефону.

Введите исходные данные:

*Длительность разговора (целое количество минут) —>**3***

*День недели (1 - понедельник, ... 7 — воскресенье) —>**6***

Предоставляется скидка 20%. Стоимость разговора: 5.52 руб.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 5

1.Даны три действительные числа. Если первое число больше второго, умножить данное число на 5, если первое число больше третьего — разделить на два

2. Написать программу — модель анализа пожарного датчика в помещении, которая выводит сообщение «Пожароопасная ситуация», если температура в комнате превысила 60^0C

3.Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 6

1.Даны действительные числа a, b, c. Удвоить эти числа, если $a > b \leq c$, иначе оставить без изменения.

2. Написать программу, которая анализирует данные о возрасте и относит человека к одной из четырех групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 7

1.Даны три числа a, b, c. Определить какое из них равно d. Если ни одно не равно d, то найти сумму чисел a, b, c.

2. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% — если сумма больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.

Введите сумму покупки и нажмите <Enter>

->640

Вам предоставляется скидка 3%

Сумма покупки с учетом скидки: 620.80 руб.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 8

1.Даны три действительные числа. Найти минимальное и максимальное число.

2. Написать программу проверки знания истории развития вычислительных средств. Программа должна вывести вопрос и три варианта ответа. Пользователь должен выбрать правильный ответ и ввести его номер. Ниже

представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Идея программируемой вычислительной машины принадлежит
1 Ада Лавлейс
2 Джон фон Нейман
3 Чарльз Беббидж
*Введите номер правильного ответа и нажмите <Enter> ->**2***
Вы ошиблись.
Идея программируемой вычислительной машины принадлежит — Дж. фон Нейману

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 9

1.Даны три действительные числа. Если все числа положительны, найти среднее арифметическое, иначе произведение.

2.Написать программу проверки знания истории развития языков программирования. Программа должна вывести вопрос и три варианта ответа, а пользователь — выбрать правильный ответ и ввести его номер. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

К машинно-зависимым языкам программирования относятся:
1.Ассемблер
2. Паскаль
3. С+
*Введите номер правильного ответа и нажмите <Enter> ->**3***
Вы ошиблись.
Правильный ответ: 1.

3. Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 10

1.Даны два вещественных числа, если числа не равны нулю, возвести из в третью степень, иначе во вторую степень.

2. Написать программу, которая сравнивает два числа, введенных с клавиатуры. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Ниже представлен рекомендуемый вид экрана во время работы программы.

*Введите в одной строке два целых числа ->**34 67***
34 меньше 67.

3.Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Практическая работа 4

Тема: составление программ циклической структуры

Краткая теория

Операторы цикла используются для многократного повторения аналогичных вычислений.
Для организации цикла в Паскале имеются три различных оператора.

1. Цикл с параметром For
For <параметр цикла>:=<начальное значение> **to** <конечное значение> **do** S;
S- простой или составной оператор.
инструкция for используется для организации циклов с фиксированным, определяемым во время разработки программы, числом повторений;
количество повторений цикла определяется начальным и конечным значениями переменной-счетчика; переменная-счетчик должна быть целого типа (integer).
При каждом прохождении цикла < параметр цикла >, начиная с <начального значения>, увеличивается на единицу. Цикл выполняется, пока <параметр цикла> не станет больше <конечного значения>.
Другой вариант записи оператора For:
For <параметр цикла >:=< начальное значение>**downto** <конечное значение> **do** S**;**
В этом случае при каждом прохождении цикла <параметр цикла> уменьшается на единицу от <начального значения> до <конечного значения>.

2. Цикл с предусловием (цикл ПОКА) While:
While <условие> **do** S; {Пока выполняется условие, делать}
Цикл выполняется, пока условие истинно (true).
3. Цикл с постусловием (цикл ДО) **Repeat**:
Repeat S **until** <условие>; {Выполнять до тех пор, пока не будет выполнено условие}
Цикл выполняется, пока условие ложно (false).

Пример
Найти сумму 5 целых чисел от 1 до 5. Написать программы для определения суммы с помощью трех рассмотренных операторов цикла.

Структограмма и программа приведены в таблице 1

Таблица 1. Операторы цикла

1. Цикл For	2. While	3. Repeat	

Описание S, i
S:=0;
For i:=1 to 5 do
S:=S+i;
 	
Вывод S
	Описание S, i
	S:=0; i:=1;
	While i<=5 do
	S:=S+i;
	i:=i+1;
	Вывод S
		Описание S, i
		S:=0; i:=1;
		S:=S+i;
		i:=i+1;
		Until i>=6
		Вывод S
			Структограммы

Program P1;
Var i,S:integer;
Begin
S:=0;
For i:=1 to 5 do
S:=S+i;
Writeln (‘S=’,S:5);
End.
	Program P2;
	Var i,S:integer;
	Begin
	S:=0; i:=1;
	While i<=5 do
	begin
	S:=S+i; i:=i+1;
	End;
	Writeln (‘S=’,S:5);
	End.
		Program P3;
		Var i,S:integer;
		Begin
		S:=0; i:=1;
		Repeat
		S:=S+i; i:=i+1;
		Until i>=6;
		Writeln (‘S=’,S:5);
		End.
			Текст программы

Задания для выполнения практической работы

Задание 1.

Составьте программу, используя операторы цикла For или Repeat по варианту, предложенному преподавателем.

Вариант 1

1.Написать программу, которая выводит таблицу квадратов первых десяти чисел. Ниже представлен рекомендуемый вид экрана во время работы программы.

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы.

Вариант 2

1. Написать программу, которая выводит таблицу квадратов целых положительных чисел, введенных пользователем с клавиатуры. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов нечетных чисел.

Число	*Квадрат*
1	*1*
3	*9*
..	..

9 81

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы.

Вариант 3

1.Написать программу, которая вводит с клавиатуры 5 дробных чисел и вычисляет их среднее арифметическое. Рекомендуемый вид экрана во время работы программы приведен ниже:

Вычисление среднего арифметического последовательности дробных чисел. После ввода каждого числа нажимайте <Enter>

-> 5.4

-> 7.8

-> 3.0

-> 1.5

-> 2.3

Среднее арифметическое введенной последовательности: 4.00

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 4

1.Написать программу, которая выводит на экран таблицу стоимости, например, яблок в диапазоне от 100 г до 1 кг с шагом 100. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

*Введите цену одного килограмма и нажмите <Enter> (копейки от рублей отделяйте точкой) ->**16.50***

Вес (гр)	*Стоимость (руб.)*
100	*1.65*
200	*3.30*
...	*...*
1000	*16.50*

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 5

1.Написать программу, которая выводит на экран таблицу перевода из градусов Цельсия (C) в градусы по Фаренгейту (F) для значений от 15 до 30 с шагом 1 градус. Перевод осуществляется по формуле F =C*1.8+32

2.Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 6

1.1. Написать программу, которая выводит таблицу значений функции $y=2,4x^2+5x-3$ в диапазоне от – 2 до 2 с шагом 0,5. Ниже представлен рекомендуемый вид экрана во время работы программы:

X y
-2 -22,60
-1,5 -15,90
… …
1,5 -0,90

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 7

1. Написать программу, которая вводит с клавиатуры 7 дробных чисел и вычисляет сумму положительных чисел и произведение отрицательных чисел. Рекомендуемый вид экрана во время работы программы приведен ниже:

Вычисление среднего арифметического последовательности дробных чисел.
После ввода каждого числа нажимайте <Enter>
-> 1.4
-> 7.8
-> 3.0
-> -7,6
-> -9,2
-> 1.5
-> 2.3
Сумма положительных чисел равна =
Произведение отрицательных чисел равно =

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 8

1.Написать программу, которая вычисляет среднее арифметическое последовательности дробных чисел, вводимых с клавиатуры. После того, как будет введено последнее число, программа должна вывести минимальное и максимальное число последовательности.

Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Обработка последовательности дробных чисел. Введите количество чисел последовательности -> 5
*Вводите последовательность. После ввода каждого числа нажимайте <Enter> ->**5.4** ->*

7.8 ->3.0 ->1.5 ->2.3
Количество чисел: ***5***
Среднее арифметическое: 4.00 Минимальное число:
Максимальное число:

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 9

1. Написать программу, которая выводит таблицу значений функции $y=-9x^2+2x$ в диапазоне от – 3 до 3 с шагом 1. Ниже представлен рекомендуемый вид экрана во время работы программы:

X	y
-3	-87
-2	-40
-1	-11
0	0
..	..

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 10

1.Написать программу, которая вычисляет произведение последовательности целых чисел, вводимых с клавиатуры. После того, как будет введено последнее число, программа должна вывести минимальное и максимальное число последовательности. Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Обработка последовательности дробных чисел. Введите количество чисел последовательности -> 5
*Вводите последовательность. После ввода каждого числа нажимайте <Enter> ->**5** ->7>**3** ->**1** ->**2***
Количество чисел: ***5*** *Произведение: 210 Минимальное число:*
Максимальное число:

2.Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Задание 2. Составьте программы, с использованием оператора While по варианту, предложенному преподавателем. Программа, выводит таблицу значений функции для аргумента, изменяющегося в заданных пределах с заданным шагом.

Вариант	Функция
1.	$y = \frac{\sin(3x)}{x^2}$, 0<x<10
2.	$y = 1 - x + \frac{x^2}{2} + 5x^4$, -5<x<6
3.	$y = \cos^2 x + \sin 5x$ -3<x<3
4.	$y = -4\sin(x+5) * \cos x$ -6<x<6
5.	$y = \frac{\sin(\frac{\pi}{2} - 5x)}{x^3}$ -5<x<5
6.	$y = \frac{\sin 2x + \sin 6x}{x^4}$ -3<x<6
7.	$y = \cos^3 x + \sin 10x$ -2<x<5
8.	$y = 2\sqrt{\cos x} - tgx$ -5<x<2
9.	$y = \frac{\sin 10x + x^2 - \cos(6x-2)}{x^4}$ -4<x<2
10.	$y = 2\sqrt{\cos 5x} - \sin x$ -6<x<6

Контрольные вопросы

1 В каких случаях в программе необходимо использовать итерационный цикл, а в каких регулярный цикл?

2 Назовите отличия итерационных циклов и цикла с параметром.

3 Какова структура оператора цикла с параметром? Как выполняется цикл с параметром?

4 Какого типа должны быть параметр цикла, его начальное и конечное значения в цикле с параметром в языке pascal?

5 Могут ли параметр цикла, его начальное и конечное значения в цикле с параметром в языке pascal быть разных типов? Обоснуйте ответ.

6 Чем отличается цикл «до» от цикла «пока»?

7 Сколько раз повторится итерационный цикл?

8 Какова структура цикла с постусловием?

9 Какова структура цикла с предусловием?

Раздел 2 Программирование в среде Lazarus

Практическая работа 5

Тема: интегрированная среда разработки приложений

Краткая теория

Процесс создания программы в Lazarus состоит из двух этапов: формирование внешнего вида программы, ее интерфейса и написание программного кода на языке программирования Free Pascal, заставляющего работать элементы интерфейса.

Начнем знакомство с визуальным программированием с создания простой программы про кнопку, которая хочет, чтобы по ней щелкнули. После чего появляется сообщение о работе программы.

Начнем с создания нового проекта. Для этого выполним команду главного меню Проект- Создать проект.... В появившемся диалоговом окне рисунок 1 выберем из списка слово Приложение и нажмем кнопку Создать. Результатом этих действий будет появление окна формы и окна редактора программного кода.

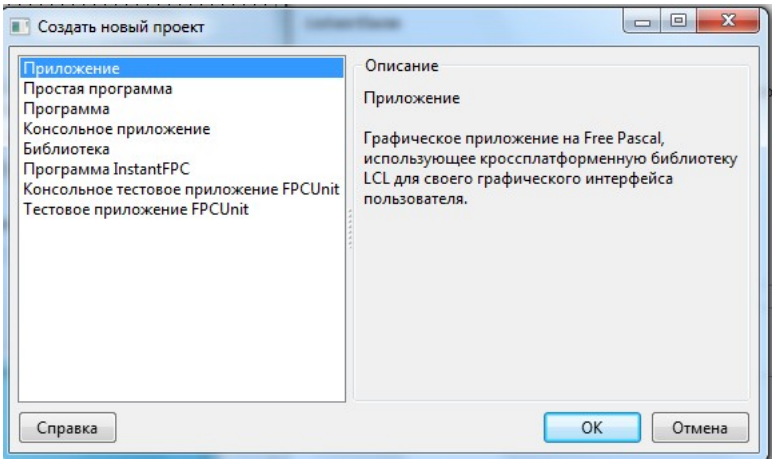


Рисунок 1. Создание нового проекта в среде Windows.

Сохраним созданный проект, воспользовавшись командой Проект-Сохранить проект как…. Откроется окно сохранения программного кода Сохранить Unit1. Создадим в нем новую папку Пример 1 (можно воспользоваться кнопкой Создание новой папки (Создание каталога)), откроем ее и щелкнем по кнопке Сохранить. Тем самым мы сохраним файл Unit1.pas, содержащий текст программы. Сразу же откроется окно Сохранить проект, в котором также необходимо щелкнуть по кнопке Сохранить. Теперь мы сохранили файл Project1, содержащий общие сведения о проекте. На самом деле все наши манипуляции привели к сохранению более чем двух файлов. Теперь в каталоге Пример 1 хранится файл с текстом программы

Unit1.pas, файл Unit1.lfm19, со сведениями о форме Form1, а также файлы Project1.lpn и Project1.lpi, содержащие настройки системы программирования и параметры конфигурации проекта рисунок 2.

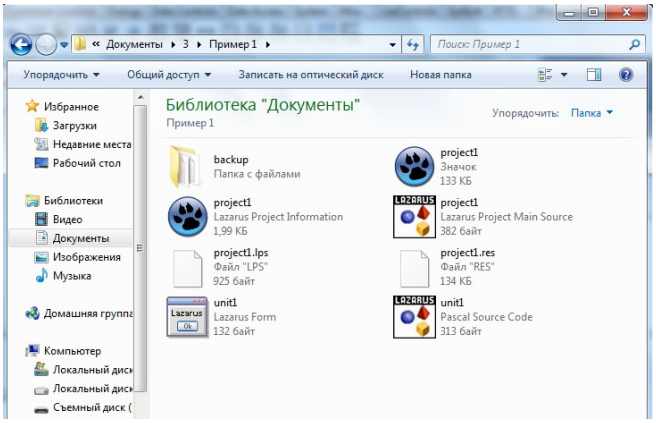


Рисунок 2 Файлы проекта в среде Windows

Теперь можно приступить к визуальному программированию. У нас есть один объект - форма Form1. Для изменения её свойств надо перейти в окно инспектора объектов и Найти нужное свойство и изменить его значение.

Для помещения на форму объекты, следует выбрать требуемый объект на Панели компонентов и поместить как графический объект на форму. Помещенные на форму объекты присоединяются к Главной форме и отображаются в окне Компоненты рисунок 3.

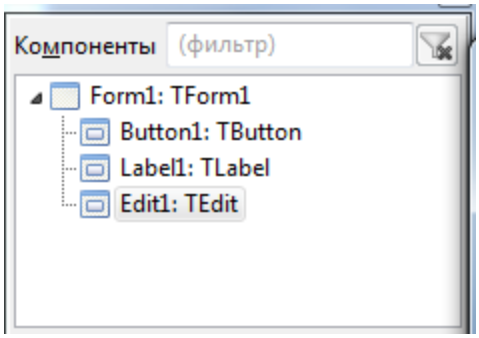


Рисунок 3. Окно компоненты проекта

Для выполнения компиляции проекта, следует его предварительно собрать Запуск- Собрать, затем выполнить компиляцию проекта Запуск-Компилировать.

Для того чтобы посмотреть, как работает наша программа, ее необходимо запустить на выполнение. Сделать это можно командой Запуск — Запустить, функциональной клавишей F9 или кнопкой Запуск в панели инструментов.

Практическая работа 6

Тема: объекты и подключение процедуры событий, создание интерфейса программы

Создать форму, установить 2 кнопки Button как представлено на рисунке 1.

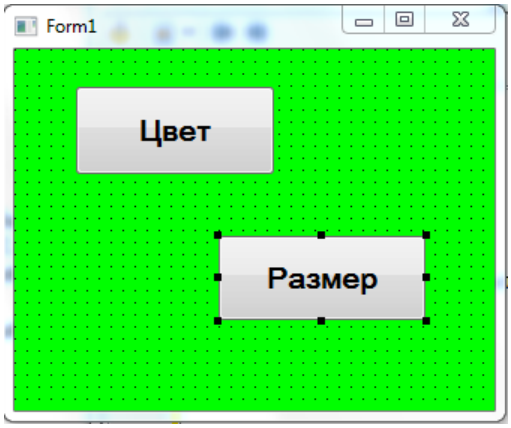


Рисунок 1. Интерфейс программы

Измените свойства формы и свойства кнопок:

Свойство Form1	**Значение**
Color	clLime
Width	140
Height	58
Свойство Button1	**Значение**
Caption	Цвет
Font	Arial- 14- полужирный курсив
Height	60
Width	134
Свойство Button2	**Значение**
Caption	Размер
Font	Arial- 14- полужирный курсив
Height	58
Width	140

Сохраните проект в папке Первый проект.

Выделите кнопку **Button1**, в окне **Инспектора объектов** Выберите вкладку **События,** выберите событие OnClik и двойным щелчком по пустому

полю подключите событие по кнопке- щелчок мышью, у вас в окне редактора должен появиться шаблон процедуры события на кнопке **Button1,** рисунок 2:

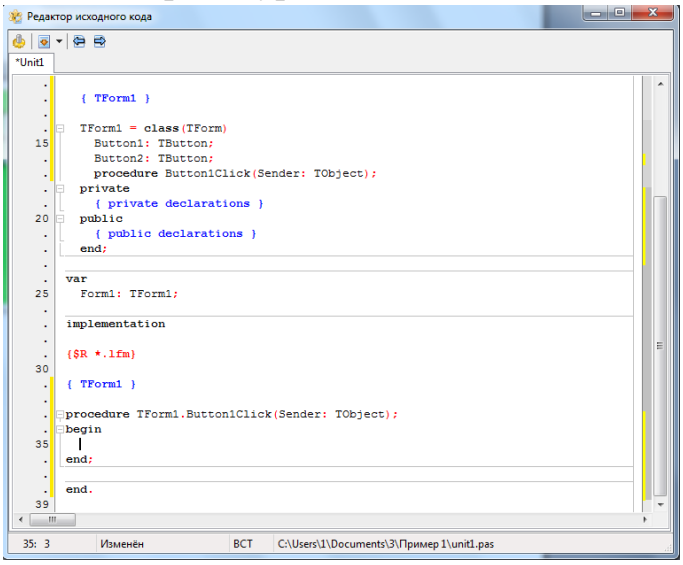


Рисунок 2. Окно редактора кода программы

В окне редактора кода в процедуре напишите код программы

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Color:=clRed;
end;

Подключите процедуру щелчок к кнопке **Button2** и в процедуре напишите код программы:

procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Width:=189;
end;

Выполните сборку проекта, компиляцию и запуск. Нажмите на кнопку цвет, затем на кнопку размер. Что изменилось в программе ?
Запишите в тетради алгоритм создания проекта.

Практическая работа 7

Тема: Основные понятия объектно-ориентированного языка

Краткая теория

Объектно-ориентированное программирование позволяет программировать в терминах классов:

– определять классы;

– конструировать новые и производные (дочерние) классы на основе существующих классов;

– создавать объекты, принадлежащие классу (экземпляры класса).

Класс описывает свойства (атрибуты) объекта и его методы (включая обработчики событий). При создании объекта он наследует структуру (переменные) и поведение (методы) своего класса. В свою очередь, класс, называемый потомком, производным или дочерним классом (подклассом), также может быть создан на основе другого родительского класса (предка) и при этом наследует его структуру и поведение. Любой компонент (элемент управления) или объект в всегда является экземпляром класса. Программно объект представляет собой переменную объектного типа. Для каждого компонента Lazarus существует свой класс, наследуемый от TComponent. Предком всех объектов, включая компоненты, является класс TObject. Наследование позволяет определять новые классы в терминах существующих классов.

Инкапсуляция - это создание защищенных объектов, доступ к свойствам и методам которых разрешен только через определенные разработчиком «точки входа». Иначе говоря, инкапсуляция - это предоставление разработчику конкретного набора свойств и методов для управления поведением и свойствами объекта, определяемыми внутри класса.

Полиморфизм - это возможность различных объектов реагировать по-разному на одни и те же события. Синтаксис языка поддерживает общепринятую для объектно-ориентированного программирования нотацию: **имя_объекта.свойство** для ссылки на свойство объекта или **имя_объекта.метод** для вызова метода объекта.

Каждый объект обладает набором свойств. Свойства могут быть как наследуемые от родительского класса, так и добавленные индивидуально для создаваемого объекта. Список всех свойств объекта и их значений отображается в диалоговом окне Инспектор объектов. Ссылка на свойство в программном модуле записывается как Имя_объекта.Свойство. Метод - это процедура или функция, ассоциируемая с некоторым объектом. Ссылка на метод в программном модуле записывается как Имя_0бъекта. Метод. Lazarus-приложение выполняется в среде Windows, и как любое Windows-приложение, получает сообщения о возникающих для него событиях. Управление приложением фактически сводится к обработке получаемых сообщений. Методы, в которых содержится код обработки события, называются обработчиками событий (События). Lazarus автоматически генерирует процедуры обработки событий - обработчики событий для

любого компонента. При этом имя обработчика событий формируется из имени компонента и названия события (например, EditlClick). Имя обработчика события автоматически квалифицируется именем класса формы. Например: TForml.ButtonlClick(Sender: TObject). Для каждого компонента предусмотрено одно стандартное событие. Например, для командной кнопки, флажка, списка, поля ввода - это событие Click, а для формы - событие FormCreate. Для того чтобы автоматически добавить в модуль объявление и описание разработчика стандартного события, достаточно выполнить на компоненте формы или самой форме двойной щелчок мышью.

Структура программы.

Любой проект в Lazarus – это совокупность файлов, из которых создается единый выполняемый файл. В простейшем случае список файлов проекта имеет вид:

- файл описания проекта (.lpi);
- файл проекта (.lpr);
- файл ресурсов (.lrs);
- модуль формы (.lfm);
- программный модуль (.pas);

После компиляции программы из всех файлов проекта создается единый выполняемый файл, имя этого файла совпадает с именем проекта. Программный модуль, или просто модуль, – это отдельно компилируемая программная единица, которая представляет собой набор типов данных, констант, переменных, процедур и функций. Любой модуль имеет следующую структуру:

unit имя_модуля; //Заголовок модуля.
interface
//Раздел описаний.
implementation
//Раздел реализаций.
end. //Конец модуля.

Заголовок модуля – это зарезервированное слово unit, за которым следует имя модуля и точка с запятой. В разделе описаний, который открывается служебным словом interface, описывают программные элементы – типы, классы, процедуры и функции:

interface
uses список_модулей;
type список_типов;
const список_констант;

var список_переменных;
procedure имя_процедуры;
...
function имя_функции;
...

Раздел implementation содержит программный код, реализующий механизм работы описанных программных элементов (тексты процедур обработки событий, процедуры и функции, созданные программистом). Процедуры и функции в Lazarus также построены по модульному принципу.

Тело программы начинается со слова begin, затем следуют операторы языка Pascal, реализующие алгоритм решаемой задачи. Операторы в языке Pascal отделяются друг от друга точкой с запятой и могут располагаться в одну строчку или начинаться с новой строки (в этом случае их также необходимо разделять точкой с запятой). Назначение символа « ; » - отделение операторов друг от друга. Тело программы заканчивается служебным словом end. Несмотря на то что операторы могут располагаться в строке как угодно, рекомендуется размещать их по одному в строке, а в случае сложных операторов отводить для каждого несколько строк. Рассмотрим более подробно структуру программы:

program имя_программы;
uses modul1, modul2, …, moduln;
const описания_констант;
type описания_типов;
var описания_переменных;
begin
оператор_1;
оператор_2;
end.

Элементы языка

Программа на языке Free Pascal может содержать следующие символы:

•латинские буквы A, B, C…, x, y, z;
•цифры 0, 1, 2…, 9;
•специальные символы + ,–, /, =, <, >, [,], ., (,), ;, :, {, }, $, #, _, @, ‘, ^.

Из символов алфавита формируют ключевые слова и идентификаторы. Ключевые слова – это зарезервированные слова, которые имеют специальное

значение для компилятора. Примером ключевых слов являются операторы языка, типы данных и т.п. Ключевые слова используются только так, как они определены при описании языка. Идентификатор – это имя программного объекта, представляющее собой совокупность букв, цифр и символа подчеркивания. Первый символ идентификатора – буква или знак подчеркивания, но не цифра. Идентификатор не может содержать пробел. Прописные и строчные буквы в именах не различаются, например ABC, abc, Abc – одно и то же имя. Каждое имя (идентификатор) должно быть уникальным и не совпадать с ключевыми словами.

В тексте программы можно использовать комментарии. Комментарий это текст, который компилятор игнорирует. Комментарии используют для пояснений программного кода или для временного отключения команд программного кода при отладке. Комментарии бывают однострочные и многострочные. Однострочный комментарий начинается с двух символов «косая черта» // и заканчивается символом перехода на новую строку. Многострочный комментарий заключен в фигурные скобки {} или располагается между парами символов (* и *). Понятно, что фигурные скобки {} или символы (* и *). К программным объектам относятся константы, переменные, метки, процедуры, функции, модули и программы.

Например:

```
{Комментарий может
выглядеть так!}
(*Или так.*)
//А если вы используете такой способ,
//то каждая строка должна начинаться
//с двух символов ≪косая черта≫.
```

Преобразование строк в другие типы

Таблица 1.

Обозначение	**Тип аргументов**	**Тип результата**	**Действие**
StrToDateTame(S)	строка	Дата и время	преобразует символы из строки s в дату и время
StrToFloat(S)	строка	вещественное	преобразует символы из строки s в вещественное число
StrToInt(S)	строка	целое	преобразует символы из строки s в вещественное число

Val(S,X,Kod)	строка		преобразует строку символов S во внутреннее представление числовой переменной X, если преобразование прошло успешно Kod=0.

Обратное преобразование

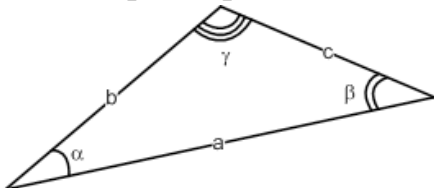
Таблица 2.

Обозначение	Тип аргументов	Тип результата	Действие
DataTimeToStr(V)	Дата и время	строка	преобразует дату и время в строку.
FloatToStr(V)	вещественное	строка	преобразует вещественное число в строку
IntToStr(V)	целое	строка	преобразует целочисленное число в строку
FloatToStrF(V, F,P,D)	вещественное	строка	преобразует вещественное число V в строку символов с учетом формата F и параметров P, D.

Практическая работа 8, 9

Тема: использование объектов для решения задач

ЗАДАЧА. Известны длины сторон треугольника a, b и c. Вычислить площадь S, периметр P и величины углов α, β и γ треугольника.



Прежде чем приступить к написанию программы, вспомним математические формулы, необходимые для решения задачи.

Для вычисления площади треугольника применим теорему Герона:

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

где полупериметр: $p=\frac{a=b=c}{2}$

один из углов найдем по теореме косинусов:

$$\cos(\alpha)=\frac{b^2+c^2-a^2}{2bc}$$;

второй — по теореме синусов: $\sin(\beta)=\frac{b}{a}\sin(\alpha)$

третий — по формуле: $\gamma=\pi-(\alpha+\beta)$

Решение задачи можно разбить на следующие этапы:

1. Определение значений a, b и c (ввод величин a, b и c в память компьютера).

2. Расчет значений S, P, α, β и γ по приведенным формулам.

3. Вывод значений S, P, α, β и γ .

Попытайтесь самостоятельно разработать внешний вид данной программы. Разместите на форме десять меток, три поля ввода и одну кнопку, рисунок 1.

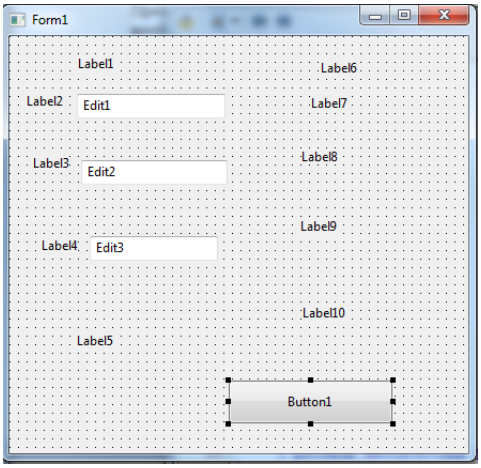


Рисунок 1. Вид формы с компонентами

Измените их заголовки (свойство Caption) в соответствии с таблицей 1.

Таблица 1. Свойства компонента

Компонент	**Свойство Caption**
Form1	Параметры треугольника
Label1	Введите длины сторон
Label2	a=
Label3	b=
Label4	c=
Label5	Величины углов
Label6	alfa=
Label7	betta=
Label8	gamma=
Label9	Периметр P=
Label10	Площадь S=
Button1	ВЫЧИСЛИТЬ

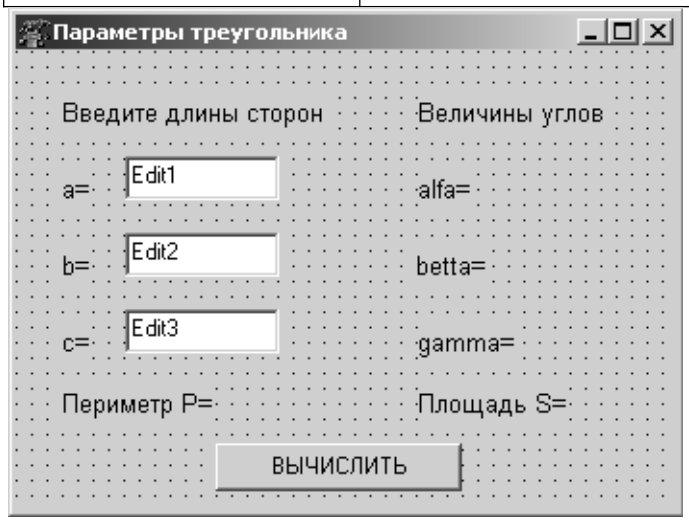


Рисунок 2. Интерфейс программы

Двойной щелчок по кнопке **Вычислить** приведет к созданию процедуры TForm1.Button1Click.

Задача программиста заполнить шаблон описаниями и операторами. Все команды, указанные в процедуре между словами **begin** и **end**, будут выполнены при щелчке по кнопке Выполнить. В нашем случае процедура TForm1.Button1Click будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
//Описание переменных:
// a, b, c – стороны треугольника;
// alfa, betta, gamma – углы треугольника;
// S - площадь треугольника;
// r - полупериметр треугольника
//Все переменные вещественного типа.
var a, b, c, alfa, betta, gamma, S,p: real;
begin
//Из полей ввода Edit1, Edit2, Edit3
//считываются введенные строки,
//с помощью функции StrToFloat(x)
//преобразовываются в вещественные числа
//и записываются в переменные a, b, c.
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
c:=StrToFloat(Edit3.Text);
//Вычисление значения полупериметра.
p:=(a+b+c)/2;
//Вычисление значения площади,
```

```
//для вычисления применяется функция:
// sqrt(x) – корень квадратный из x.
S:=sqrt(p*(p-a)*(p-b)*(p-c));
//Вычисление значения угла alfa в радианах.
//Для вычисления применяем функции:
// arccos(x) - арккосинус x;
// sqr(x) – возведение x в квадрат.
alfa:=arccos((sqr(b)+sqr(c)-sqr(a))/2/b/c);
//Вычисление значения угла betta в радианах.
//Для вычисления применяем функции:
// arcsin(x) - арксинус x;
betta:=arcsin(b/a*sin(alfa));
//Вычисление значения угла gamma в радианах.
//Математическая постоянная определена
//функцией без аргумента pi.
gamma:=pi-(alfa+betta);
//Перевод радиан в градусы.
alfa:=alfa*180/pi;
betta:=betta*180/pi;
gamma:=gamma*180/pi;
//Для вывода результатов вычислений используем
//операцию слияния строк «+»
//и функцию FloatToStrF(x), которая
//преобразовывает вещественную переменную x
//в строку и выводит ее в указанном формате,
//в нашем случае под переменную отводится
//три позиции, включая точку
//и ноль позиций после точки.
//Величины углов в градусах выводятся на форму
//в соответствующие объекты типа надпись.
Label6.Caption:='alfa='+ FloatToStrF(alfa,ffFixed,3,0);
Label7.Caption:='betta='+ FloatToStrF(betta,ffFixed,3,0);
Label8.Caption:='gamma='+FloatToStrF(gamma,ffFixed,3,0);
//Используем функцию FloatToStrF(x)
//для форматированного вывода, в нашем случае
//под все число отводится пять позиций,
//включая точку, и две позиций после точки.
//Значения площади и периметра
//выводятся на форму.
```

*Label9.Caption:='Периметр P='+FloatToStrF(2*p,ffFixed,5,2);*
Label10.Caption:='Площадь S='+FloatToStrF(S,ffFixed,5,2);
end;

Обратите внимание, что было написано всего десять команд, предназначенных для решения поставленной задачи, все остальное комментарий, который писать необязательно.

Задачи для самостоятельного решения

Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

1. Заданы два катета прямоугольного треугольника. Найти гипотенузу и углы треугольника.

3. Известна диагональ квадрата d. Вычислить площадь S и периметр P квадрата.

5. Треугольник задан величинами своих сторон – a, b, c. Найти углы треугольника – α, β, γ.

6. Тело имеет форму параллелепипеда с высотой h. Прямоугольник в основании имеет диагональ d. Известно, что диагонали основания пересекаются под углом a. Найти объем тела V и площадь поверхности S.

7. Задан первый член геометрической прогрессии и ее знаменатель. Вычислить сумму n членов геометрической прогрессии и значение n-го члена.

8. Тело падает с высоты h. Какова его скорость в момент соприкосновения с землей и когда это произойдет.

Практическая работа 10

Тема: составление программ циклической структуры

Задание Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию, при решении задач использовать разные виды циклов..

1. Дана последовательность действительных чисел. Найти максимальный элемент в последовательности.

2. Даны натуральные числа n и k. Найти сумму 1k + 2k + 3k +…+ nk. Заблокировать ввод ненатуральных чисел.

3. Составить программу вычисления значения функции $y=\frac{(x-2)(x-4)(x-6)..(x-64)}{(x-1)(x-3)(x-5)...(x-63)}$

4. Составить программу, которая проверяет, является ли заданное число совершенным. Совершенным называется натуральное число, равное сумме всех своих делителей (исключая само число). Например, 28 = 1 + 2 + 4 + 7 + 14.

5.Спортсмен-лыжник начал тренировки, пробежав в первый день 10 км. Каждый следующий день он увеличивал длину пробега на P процентов от пробега предыдущего дня (P — вещественное, 0 < P < 50). По данному P определить, после какого дня суммарный пробег лыжника за все дни превысит 200 км, и вывести найденное количество дней K (целое) и суммарный пробег S (вещественное число).(WHILE)

Итог работы: отчет, защита работы.

Практическая работа 11, 12

Тема: массивы

Краткая теория

Массив — это структура данных, представляющая собой набор переменных одинакового типа, имеющих общее имя. Массивы удобно использовать для хранения однородной по своей природе информации, например, таблиц и списков.

Массив, как и любая переменная программы, перед использованием должен быть объявлен в разделе объявления переменных. В общем виде инструкция объявления массива выглядит следующим образом:

Имя: **array** [нижний_индекс. .верхний_индекс] of тип

где:

имя — имя массива;

array — зарезервированное слово языка Delphi, обозначающее, что объявляемое имя является именем массива;

нижний_индекс и верхний_и«декс — целые константы, определяющие диапазон изменения индекса элементов массива и, неявно, количество элементов (размер) массива;

тип — тип элементов массива.

Примеры объявления массивов:

temper:array[1..31] of real;

koef:array[0. .2] of integer;

name:array[1..30] of string[25];

При объявлении массива удобно использовать именованные константы. Именованная константа объявляется в разделе объявления констант, который обычно располагают перед разделом объявления переменных. Начинается раздел объявления констант словом const. В инструкции объявления именованной константы указывают имя константы и ее значение, которое отделяется от имени символом "равно". Например, чтобы объявить именованную константу нв, значение которой равно 10, в раздел const надо записать инструкцию: нв=10. После объявления именованной константы ее можно использовать в программе как обычную числовую или символьную константу. Ниже в качестве примера приведено объявление массива названий команд-участниц чемпионата по футболу, в котором используются именованные константы.

const

NT = 18; // число команд

SN = 25; // предельная длина названия команды

var

team: array[1..NT] of string[SN];

Использование компонента StringGrid

Для ввода массива удобно использовать компонент StringGrid. Значок компонента StringGrid находится на вкладке Additional рисунок 1.

Рисунок 1. Компонент StringGrid

Компонент StringGrid представляет собой таблицу, ячейки которой содержат строки символов. В таблице перечислены некоторые свойства компонента таблица 4. StringGrid.

Таблица 1. Свойство компонента

Свойство	Определяет
ColCount	Количество колонок таблицы
RowCount	Количество строк таблицы
Cells	Соответствующий таблице двумерный массив. Ячейка таблицы, находящаяся на пересечении столбца номер col и строки номер row определяется элементом cells [col, row]
FixedCols	Количество зафиксированных слева колонок таблицы. Зафиксированные колонки выделяются цветом и при горизонтальной прокрутке таблицы остаются на месте
FixedRows	Количество зафиксированных сверху строк таблицы.

	Зафиксированные строки выделяются цветом и при вертикальной прокрутке таблицы остаются на месте
Options . goEditing	Признак допустимости редактирования содержимого ячеек таблицы. True — редактирование разрешено, False — запрещено
Options . goTab	Разрешает (True) или запрещает (False) использование клавиши <TaЬ> для перемещения курсора в следующую ячейку таблицы

Использование компонента Memo.

Компонент Memo позволяет вводить текст, состоящий из достаточно большого количества строк, поэтому его удобно использовать для ввода символьного массива. Компонент Memo добавляется в форму обычным образом. Значок компонента находится на вкладке Standard.

В таблице 2 перечислены некоторые свойства компонента Memo.

Таблица 2. Свойства компонента

Свойство	Определяет
Name	Имя компонента. Используется в программе для доступа к свойствам компонента
Text	Текст, находящийся в поле Memo. Рассматривается как единое целое
Lines	Текст, находящийся в поле Memo. Рассматривается как совокупность строк. Доступ к строке осуществляется по номеру
Lines .Count	Количество строк текста в поле Memo

Практическое задание

Цель- научиться использовать компоненты StringGrid и поле Memo, для работы с массивом.

Задача 1.

Рассмотрим программу, которая вычисляет среднее арифметическое значение элементов массива. Диалоговое окно программы приведено на рисунке 2 Компонент StringGrid используется для ввода массива, компоненты Label1 и Label2- для вывода пояснительного текста и результата расчета, Button1- для запуска процесса расчета.

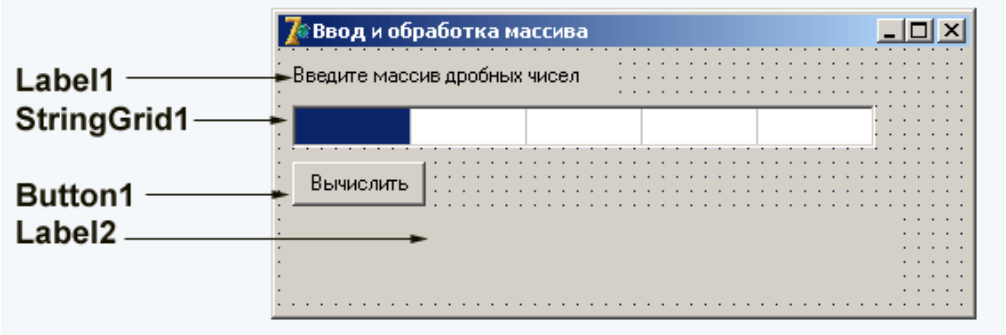


Рисунок 2. Окно интерфейса программы

Добавляется компонент stringGrid в форму точно так же, как и другие компоненты. После добавления компонента к форме нужно выполнить его настройку в соответствии с таблицей 3. Значения свойств Height и Width следует при помощи мыши установить такими, чтобы размер компонента был равен размеру строки.

Таблица 3. Значение свойств

Свойство	**Значение**
ColCount	5
FixedCols	0
RowCount	1
DefaultRowHeight	24
Height	24
DefaultColWidth	64
Width	328
Options . goEditing	True
Options . AlwaysShowEditing	True
Options .goTabs	True

На кнопку посадите процедуру:

```
procedure TForml.ButtonlClick(Sender: Tobject);
 var
a : array[1..5] of integer; // массив
summ: integer; // сумма элементов
sr: real; // среднее арифметическое
i: integer; // индекс
begin
// ввод массива
// считаем, что если ячейка пустая, то соответствующий
// ей элемент массива равен нулю
for i:= 1 to 5 do
if Length(StringGridl.Cells[i-1, 0]) <>0
then a[i] := StrToInt(StringGridl.Cells[i-1,0])
else a[i] := 0;
// обработка массива
summ := 0;
for i :=1 to 5 do
summ := summ + a[i]; sr := summ / 5;
Label2.Caption :='Сумма элементов: ' + IntToStr(summ)+ #13+
'Среднее арифметическое: ' + FloatToStr(sr);
```

end;

Для того, чтобы курсор автоматически переходил в следующую ячейку таблицы, например, в результате нажатия клавиши <Enter>. Добавим процедуру обработки события onKeyPress. На эту же процедуру можно возложить задачу фильтрации вводимых в ячейку таблицы данных. В нашем случае надо разрешить ввод в ячейку только цифру.

Текст процедуры обработки события OnKeyPress приведен ниже. Следует обратить внимание на свойство Col, которое во время работы программы содержит номер колонки таблицы, в которой находится курсор. Это свойство можно также использовать для перемещения курсора в нужную ячейку таблицы. Однако нужно учитывать, что колонки таблицы, впрочем, как и строки, нумеруются с нуля.

Procedure Tform1.StringGridlKeyPress(Sender: Tobject; var Key: Char);
begin
case Key of
#8,'0'..'9' : ; // цифры и клавиша <Backspace>
#13: // клавиша <Enter>
if StringGridl.Col < StringGridl.ColCount-1
then StringGridl.Col := StringGridl.Col+1;
else key := Chr(0); // остальные символы запрещены
end;
end;

Запустите программу, посмотрите как изменилась работа программы.

Задача 2.

Рассмотрим задачу, в которой компонент Memo используется для вывода символьного массива. Создайте интерфейс окна программы, как показано на рисунке 3.

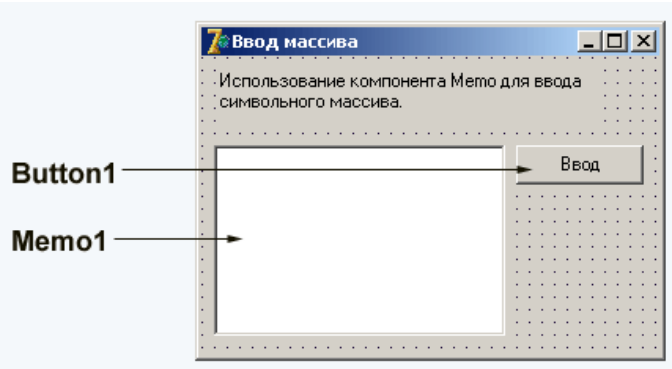


Рисунок 3. Диалоговое окно приложения Ввод массива

На кнопку посадите процедуру:

```
procedure Tforml .ButtonlClick(Sender: Tobject);
const
SIZE=5; // размер массива
var
a:array[l..SIZE]of string[30]; //массив
n: integer; // количество строк, введенных в поле Memo
i:integer; // индекс элемента массива
st:string;
begin
n:=Memo1.Lines.Count;
if n = 0 then begin
ShowMessage('Исходные данные не введены!');
Exit; // выход из процедуры обработки события
end;
// в поле Memo есть текст
if n > SIZE then begin
ShowMessage('Количество строк превышает размер массива.');
n:=SIZE; // будем вводить только первые SIZE строк
end;
for i:=1 to n do
a[i]:=Form1.Memol.Lines[i-1]; // строки Memo пронумерованы с нуля
// вывод массива в окно сообщения
if n > 0 then begin
st:='Введенный массив:'+#13;
for i: =1 to n do
st:=st+IntToStr(i)+' '+ a[i]+f13; ShowMessage(st);
end;
end;
```

Задачи для самостоятельного решения.

Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

1. Записать положительные элементы массива X подряд в массив Y. Вычислить сумму элементов массива X и произведение элементов массива Y.
2. Из массива Y удалить элементы, расположенные между максимальным и минимальным элементами.

3. Сформировать массив B, записав в него элементы массива A с нечетными индексами. Вычислить среднее арифметическое элементов массива B и удалить из него максимальный, минимальный и пятый элементы.
4. Дан массив целых чисел X. Переписать пять первых положительных элементов массива и последних два простых элемента в массив Y. Найти максимальный отрицательный элемент массива X.

Практическая работа №13

Тема: Составление программ с применением строковых процедур и функций.

Порядок выполнения практической работы.

Задание.

1. Преобразовать исходные данные в новую строку (строки) с помощью операций копирования, удаления, вставки, конкатенации. При необходимости использовать функцию подсчета длины строки.

   Морозы
   Вороны
   Тропы
2. Введите два слова с клавиатуры. Соедините их, определите длину полученного слова. Напечатайте введенные слова, полученное слово и его длину.

   Вес и на
   Конь и як
   Само и лет
3. Из заданного слова Черепаха получите:

   слово «чех»
   слово «папаха»
   число 8
   число 3
4. Составьте программу, которая вырезает пять символов из данного текста, соединяет эту вырезку со вторым текстом и определяет длину полученного текста.

   1 текст Радиоэлектроника
   2 текст Вещание
5. Введите с клавиатуры три слова и соедините их в предложение. Напечатайте полученное предложение и длину каждого слова.

Слова произвольно ввести с клавиатуры

6. Составьте программу, которая получает следующие слова из заданного слова Квартира и определяет их длину:

   тир

   рита

   тара

7. Заменить три последние буквы заданного слова. Вывести длину слова, исходное слово и результат.

   Слово самостоятельно ввести с клавиатуры

8. Составьте программу, которая получает из заданного слова слово «смешная». Напечатайте длину слова.

   Смешной

9. Получите новые слова из заданных, заменяя в них вторую букву на букву «о».

   Сам

   Бар

   Кирка

10. Напишите программу для получения из следующих слов '17', '4', ‘21’:

    1721

    38

    424

11 Из заданного слова САЛАКА получите слова:

- *собака*
- *сало*
- *ласка*

Напечатайте их длину.

Контрольные вопросы:

1. Понятие строкового типа данных.
2. Функция определения длины строки.
3. Процедуры вставки, удаления, копирования, конкатенации.

Практическая работа №14

Тема: Разработка алгоритма и составление программы поиска в строке.

Порядок выполнения практической работы.

Задание.

Во всех вариантах ввести фамилию, имя и отчество как ОДНУ СТРОКУ. Определить длину получившейся строки. Выполнить задачу своего варианта.

1. Вывести имя и количество букв в третьем слове.
2. Определить, сколько букв «а» есть в фамилии.
3. Вывести инициалы с точками.
4. Вывести длины фамилии и имени.
5. Вывести фамилию и инициалы.
6. Вывести имя и количество букв в фамилии.
7. Определить, сколько букв «о» есть в имени.
8. Вывести самое длинное слово.
9. Удалить все буквы «а» и «о» из фамилии.
10. Вывести имя по буквам в столбик.
11. Проверить, начинается хотя бы одно слово с буквы «М».
12. Все буквы «и» в имени.
13. Вывести фамилию и количество букв в имени.
14. Вывести имя в обратном порядке.
15. Вывести фамилию в столбик.
16. Вывести имя, отчество и количество букв в фамилии.
17. Вывести самое короткое слово.
18. Вывести строку без пробелов. Сколько букв в имени?
19. Вывести длины трех слов.
20. Вывести имя и количество букв в фамилии.
21. Вывести имя и фамилию на отдельных строках.
22. Каждую букву имени продублировать.
23. Вывести фамилию в обратном порядке.
24. Определить, сколько букв «а» и «б» в фамилии.
25. Вывести третье слово и количество букв в фамилии.

Контрольные вопросы:

1. Понятие строкового типа данных.
2. Функция определения длины строки.
3. Обращение к отдельному символу строки.
4. Алгоритм поиска по строке.

3.СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ

Раздел 2 Программирование в среде Lazarus

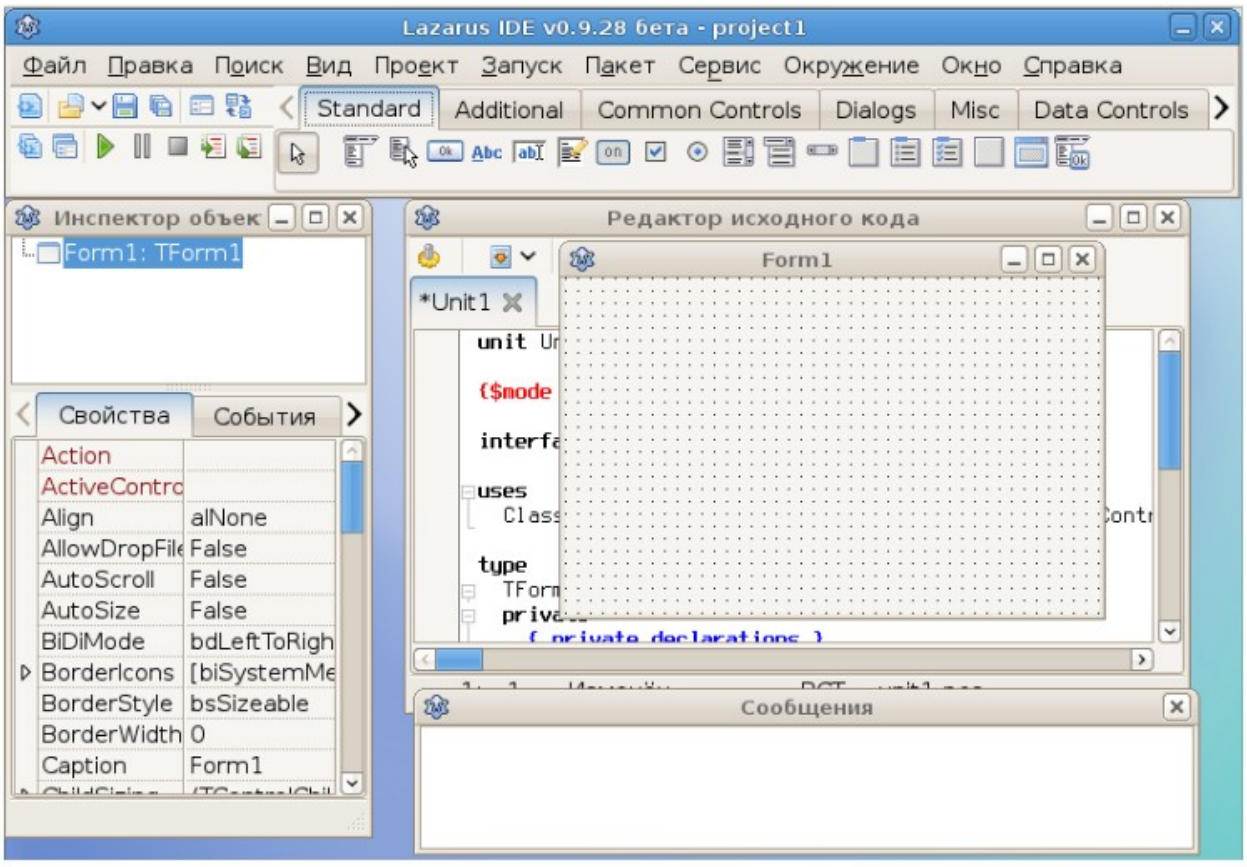
Лабораторная работа №1

Знакомство с интегрированной средой Lazarus

Цель: Знакомство с интегрированной средой разработки программного обеспечения Lazarus. Создание нового проекта.

Справочный материал

.1. Пользовательский интерфейс Lazarus Интегрированная среда разработки Lazarus представляет собой многооконную систему, вид (пользовательский интерфейс) которой может различаться в зависимости от настроек. После загрузки интерфейс Lazarus выглядит следующим образом:



и первоначально включает:

1. Главное окно;
2. окно Инспектора объектов;
3. окно Формы, или Конструктора формы;

4. окно Редактора кода;
5. окно Сообщений.

Главное окно Lazarus содержит:

1 главное меню;

2 панели инструментов;

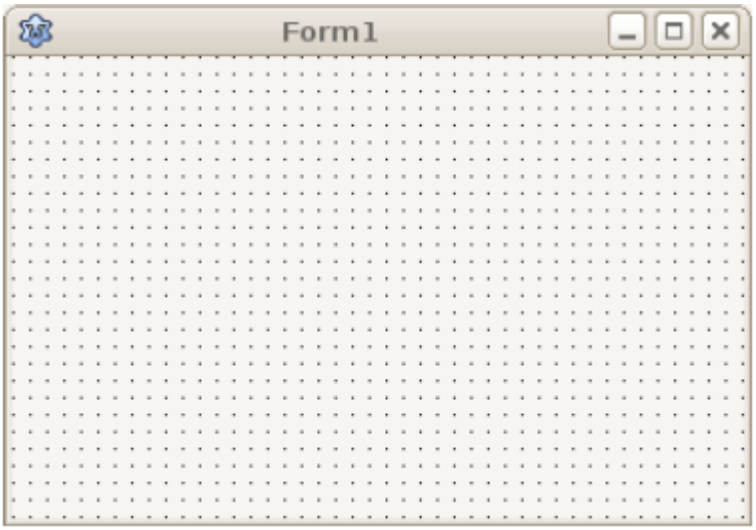
3 палитру компонентов.

Главное меню содержит обширный набор команд для доступа к функциям Lazarus.

Панели инструментов находятся под главным меню в левой части главного окна и содержат кнопки быстрого доступа к наиболее частым командам главного меню.

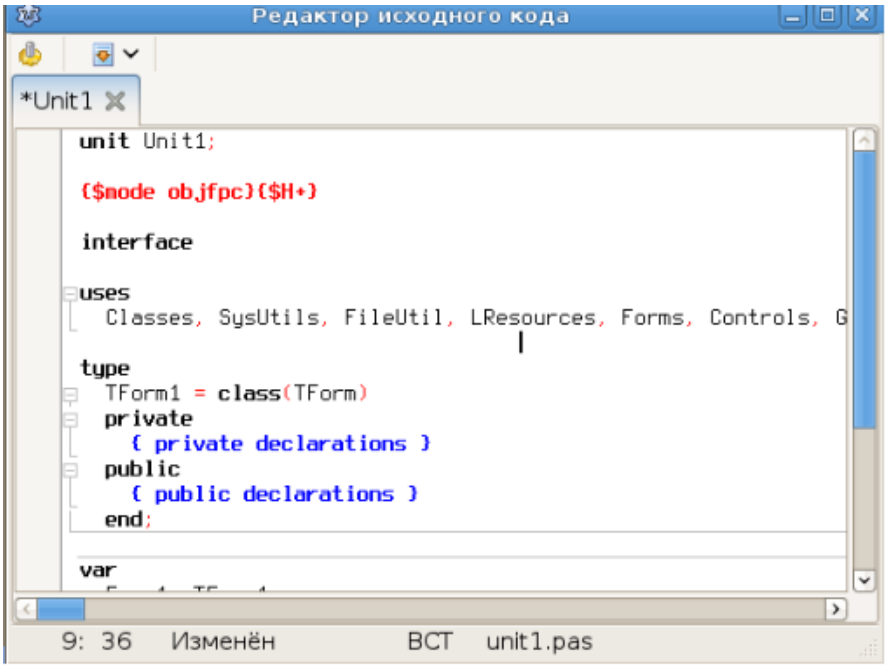
Палитра компонентов находится под главным меню в правой части главного окна и содержит множество компонентов, размещаемых в создаваемых формах.

Компоненты являются своего рода строительными блоками, из которых конструируются формы приложения.

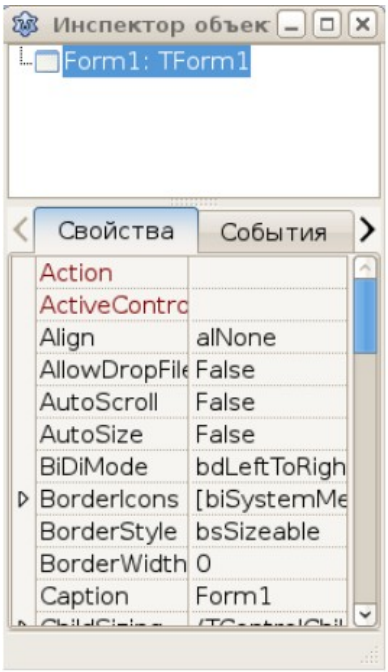


Окно формы (или *Конструктора формы*) первоначально находится в центре экрана и имеет заголовок Form1.

В нем выполняется проектирование формы, путем размещения на форме различных компонентов.



Окно редактора кода после запуска системы программирования находится под (или поверх) окна Формы. Редактор кода представляет собой обычный текстовый редактор, с помощью которого можно редактировать текст программы (листинг)



Окно Инспектора объектов расположено под Главным окном в левой части экрана и состоит из двух частей: в верхней части окна Инспектора объектов располагается список всех созданных объектов (форма и все компоненты,

которые расположены на форме) ; в нижней части отображены свойства и события объектов для текущей формы или компонента. Вкладка Свойства отражает свойства выбранного компонента или формы, а вкладка

События – процедуры, которые должны быть выполнены при возникновении указанного события.

2. Создание нового проекта

Для создания нового проекта необходимо воспользоваться пунктом «Создать» меню «Файл» главного меню и выбрать пункт «Project Application».

3. Объекты формы и их свойства

На палитре компонентов располагаются различные объекты пользовательского интерфейса. Наиболее востребованные объекты:

•Кнопка – TButton 

•Текстовое поле – Tedit 

•Метка - TLabel 

Данные объекты расположены на вкладке *Standart* Палитры компонентов.

Все объекты характеризуются:

•свойствами (цвет, положение на экране и пр.),

•методами (действия или задачи, которые выполняет объект) и

•событиями (на какое событие должен реагировать объект)

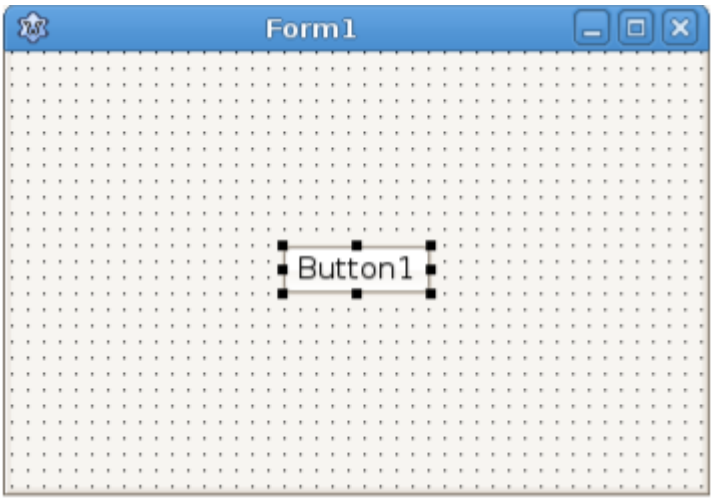
4. События и процедуры обработки событий

Большинство компонентов могут реагировать на определенные действия пользователя - *события*. Наиболее распространенными событиями являются: щелчок кнопкой мыши на кнопке, ввод данных в поле ввода, наведение курсора мыши на компонент и т.д. При этом именно разработчик программы решает, как компонент будет

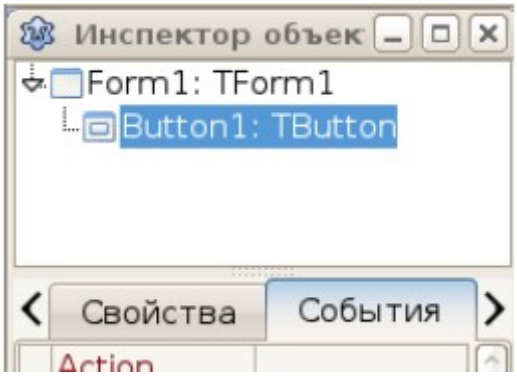
реагировать на возникшее событие. Реакция программы на определенное событие реализуется с помощью *процедур обработки событий*.

Рассмотрим пример процедуры обработки события нажатия кнопки TButton левой кнопкой мыши – OnClick.

Для этого поместим компонет TButton на форму. Это можно сделать следующим образом: щелкаем на компонент TButton на Палитре компонентов и затем щелкаем в том месте формы где необходимо разместить кнопку. Кнопка появится на форме:



а в списке объектов окна Инспектора объектов появится запись Button1: TButton



Выполнив двойной щелчок на кнопке Button1, расположенной на форме мы вызовем обработчик события, который создаст «скелет» процедуры обработки события OnClick нажатия кнопки в окне Редактора кода

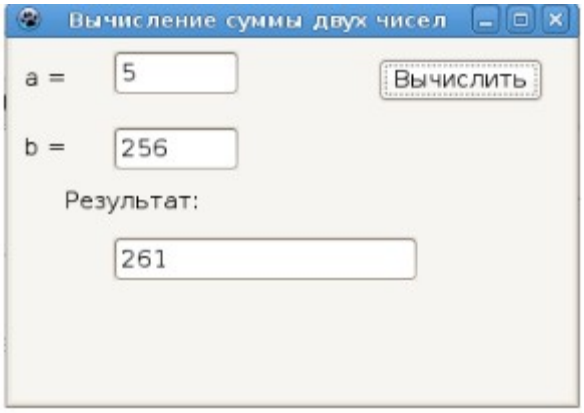
```
{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin

end;

initialization
  {$I unit1.lrs}

end.
```

Рассмотрим структуру процедуры обработки события. Первая строка включает в себя служебное слово procedure, которое означает начало процедуры, затем идет имя процедуры, которое состоит из двух частей: Тип объекта - родителя (в нашем случае это тип TForm1, так как объектом-родителем является форма Form1) и, через точку, имя самого события. В скобках указан объект- инициатор события. Затем идут операторные скобки begin... end, внутри этих скобок и будет содержаться код, описывающий действия, которые должны происходить при возникновении события нажатия кнопки.

Практические задания.

Задание №1.

Создать простейший пользовательский интерфейс, для программы сложения двух чисел (a + b = c), содержащий объекты Button, TextBox -для ввода значений переменных a и b и вывода значения c, Label -для поясняющих надписей.



Выполнение задания

Выполнение задания начнем с размещения на форме необходимых компонентов и задания их основных свойств:

1.Поместите на форму три метки TLabel и измените значение свойства Caption на «a=», «b=» и «Результат:» соответственно.
2.Поместите на форму три компонента TEditи измените начение свойства Textна пустое значение (т.е. просто удалите текст «Edit1», «Edit2» и «Edit3»).
3.Поместите на форму кнопку TButtonи измените значение ее свойства Caption на «Вычислить».
4. Измените значение свойства Caption формы Form1 на «Вычисление суммы двух чисел» (если вводимый текст не помещается в заголовке формы, то необходимо увеличить ширину формы, это можно сделать путем изменения значения свойства формы Width).

Теперь создадим процедуру обработки события (или обработчик события) OnClick нажатия кнопки «Вычислить». Для этого необходимо сделать двойной щелчок левой кнопкой мыши по кнопке «Вычислить» - при этом появится «скелет» процедуры Buuton1Click.

Далее создадим само «тело» процедуры. Прежде всего необходимо задать переменные, которые будут использоваться в процедуре. Дляэтогонеобходимопослестроки

procedure TForm1.Button1Click(Sender: TObject);

написать следующую строчку:

Vara, b, c: integer;

таким образом мы определяем три переменные, в которых будут хранится, соответственно, первое слагаемое, второе слагаемое и результат.

Далее, в операторных скобках необходимо написать следующие строки кода:

```
a := StrToInt(Edit1.Text);
b := StrToInt(Edit2.Text);
c:= a+b;
Edit3.Text:= IntToStr(c);
```

Рассмотри подробнее введенные строки кода.

Первые две строчки кода предназначены для присвоения введенных в поля Edit1 и Edit2 первого и второго слагаемых соответствующим им переменным a и b.

При этом используется функция *StrToInt(s: String):Integer*, которая выполняет перевод введенного значения из строкового типа (свойство Text

компонента TEdit имеет строковый тип данных) в числовой (т.к. переменные a и b числового типа).

Далее происходит операция сложения двух чисел (третья строка кода), а затем вывод результата в поле Edit3, причем здесь происходит обратное действие – мы переводим числовой тип данных в строковый.

Таким образом, полный код процедуры Button1Click имеет вид:

procedure TForm1.Button1Click(Sender: TObject)

```
procedure TForm1.Button1Click(Sender: TObject);
var a, b, c: integer;
begin
a := StrToInt(Edit1.Text);
b := StrToInt(Edit2.Text);
c:= a+b;
Edit3.Text := IntToStr(c);
end;
```

Теперь можно запустить проект на компиляцию и сборку (пункт Главного меню Запуск, подпункт Запуск или нажмите кнопку на панели инструментов или нажмите кнопку F9, в результате чего мы получим работающую программу. Попробуйте ввести в поля a и b два числа и нажать кнопку «Вычислить», в поле «Результат» должна появиться сумма введенных чисел.

Задание №2.

Изменить название кнопки «Вычислить» на «Сложение».

Задание №3.

Добавить к существующей программе еще три кнопки TButton, которые выполняют соответственно операции вычитания, умножения и деления (в данном случае необходимо учесть, что вместо типа integer необходимо использовать тип real, так как результатом операции деления не всегда является целое число).

Лабораторная работа №2

Составление программ разветвляющейся структуры

Задание. Решить задачу своего варианта и оформить согласно этапам решения задач на ЭВМ.

Вариант	Задача
1	Дано трехзначное число. В нем зачеркнули первую справа цифру и

	приписали ее слева. Вывести полученное число.
2	Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее справа. Вывести полученное число.
3	Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду сотен в записи этого числа..
4	Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду тысяч в записи этого числа.
5	Дано трехзначное число. Вывести вначале его последнюю цифру (единицы), а затем — его среднюю цифру (десятки).
6	Дано трехзначное неотрицательное число. Используя одну операцию деления нацело, вывести первую цифру данного числа (сотни).
7	Дано трехзначное число. Найти сумму и произведение его цифр.
8	Дано трехзначное число. Вывести число, полученное при перестановке цифр десятков и единиц исходного числа.
9	Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа.
10	Дано трехзначное число. Вывести число, полученное при прочтении исходного числа справа налево.

Итог работы: отчет, защита работы.

Лабораторная работа №3

Составление программ циклической структуры.

Цель: изучить операторы, реализующие циклические алгоритмы.

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

1. Дана последовательность действительных чисел. Найти максимальный элемент в последовательности.
2. Даны натуральные числа n и k. Найти сумму 1k + 2k + 3k +…+ nk. Заблокировать ввод ненатуральных чисел.
3. Составить программу вычисления значения функции $y=\frac{(x-2)(x-4)(x-6)..(x-64)}{(x-1)(x-3)(x-5)...(x-63)}$
4. Составить программу, которая проверяет, является ли заданное число совершенным. Совершенным называется натуральное число, равное сумме всех своих делителей (исключая само число). Например, 28 = 1 + 2 + 4 + 7 + 14.

5.Спортсмен-лыжник начал тренировки, пробежав в первый день 10 км. Каждый следующий день он увеличивал длину пробега на P процентов от пробега предыдущего дня (P — вещественное, 0 < P < 50). По данному P определить, после какого дня суммарный пробег лыжника за все дни превысит 200 км, и вывести найденное количество дней K (целое) и суммарный пробег S (вещественное число).(WHILE)

Итог работы: отчет, защита работы.

Лабораторная работа №4

Составление программ циклической структуры.

Цель: изучить операторы, реализующие циклические алгоритмы.

Задание. Создать простейший пользовательский интерфейс, для выполнения следующих задач, использовать компоненты Label-l для пояснений, TextBox – для ввода данных, Button - для задания процедур. Перечень вариантов и заданий представлены в таблице

Вариант	Задача
1	Даны два целых числа A и B ($A < B$). Вывести в порядке убывания все целые числа, расположенные между A и B (не включая числа A и B), а также количество N этих чисел.
2	Дано целое число N (> 0). Найти квадрат данного числа, используя для его вычисления следующую формулу: $N^2 = 1 + 3 + 5 + \ldots + (2 \cdot N - 1)$.
3	Дано вещественное число A и целое число N (> 0). Найти A в степени N: $A^N = A \cdot A \cdot \ldots \cdot A$
4	Дано вещественное число A и целое число N (> 0). Используя один цикл, вывести все целые степени числа A от 1 до N.
5	Дано целое число N (> 0). Найти произведение $N! = 1 \cdot 2 \cdot \ldots \cdot N$
6	Дано целое число N (> 0). Найти произведение $1.1 \cdot 1.2 \cdot 1.3 \cdot \ldots$ (N сомножителей).
7	Дано целое число N (> 0). Найти значение выражения $1.1 - 1.2 + 1.3 - \ldots$ (N слагаемых, знаки чередуются). Условный оператор не использовать.
8	Даны два целых числа A и B ($A < B$). Найти сумму всех целых чисел от A до B включительно.
9	Даны два целых числа A и B ($A < B$). Найти произведение всех целых чисел от A до B включительно
10	Даны два целых числа A и B ($A < B$). Найти сумму квадратов всех целых чисел от A до B включительно.

Итог работы: отчет, защита работы.

Лабораторная работа №5,6

Обработка одномерных и двумерных массивов.

Цель: научиться описывать, заполнять, выводить и обрабатывать одномерные и двумерные массивы.

Теоретические сведения:

Массив – группа элементов одного типа, объединенных под общим именем.

Описание массивов

Массивы описываются в разделе описания переменных Var.

Общий вид описания одномерного массива:

<имя массива>: array [<начальный индекс>..<конечный индекс>] of <тип элемента>;

где имя - имя переменной-массива; array - ключевое слово, обозначающее, что переменная является массивом; нижний_индекс и верхний_индекс - целые числа, определяющие диапазон изменения индексов (номеров) элементов массива и, неявно, количество элементов (размер) массива; тип - тип элементов массива.

Общий вид описания двумерного массива:

<имя массива>:array[<m_1>..<m_2>,<n_1>..<n_2>] of <тип>;

где *Имя* - имя массива; *array* - слово языка Pascal, показывающее, что описываемый элемент данных - массив; m_1, m_2, n_1, n_2- константы или выражения типа INTEGER, определяющие диапазон изменения индексов и, следовательно, число элементов массива; *Тип* - тип элементов массива.

Заполнение массива

Под вводом массива понимается ввод значений элементов массива. Ввод удобно реализовать при помощи инструкции FOR. Чтобы пользователь программы знал, ввода какого элемента массива ожидает программа, следует организовать вывод подсказок перед вводом очередного элемента массива. В подсказке обычно указывают индекс элемента массива. Заполнение массива можно производить:

1) с клавиатуры: For i:=1 to n do readln(a[i]);
2) через датчик случайных чисел: Randomize; For i:=1 to n do begin a[i]:=random(i);

Если требуется, чтобы значения элементов массива выбирались из определенного интервала [a,b], то a+Random(b-a+1);

3) через оператор присваивания (по формуле): For i:=1 to n do a[i]:=i*3;

Вывод массива

Если в программе необходимо вывести значения всех элементов массива, то для этого удобно использовать инструкцию FOR, переменная-счётчик которой может быть реализована как индекс элемента массива. Например, For i:=1 to n do writeln(a[i]);

Удаление элементов из одномерного массива.

Для того, чтобы удалить из массива k-ый элемент нужно: найти номер элемента k; сдвинуть все элементы, начиная с k-го, на один элемент влево; последнему элементу присвоить значение, равное 0; уменьшить количество элементов массива на единицу.

Вставка элемента в одномерный массив.

Вставлять элемент можно до или после данного элемента, номер этого элемента можно вводить с клавиатуры или искать при определенных условиях. Пусть k - это номер элемента, после которого мы должны вставить элемент x. Тогда вставка осуществляется следующим образом: первые k элементов массива остаются без изменения, все элементы, начиная с (k+1)-го, необходимо сдвинуть на один назад, на место (k+1)-го элемента записываем значение x; увеличить количество элементов в массиве на единицу.

При решении задач с использованием двумерных массивов организуются вложенные циклы:

```
For i:=1 to m do begin
изменяется номер строки
For j:=1 to n do begin
изменяется номер столбца
a[i , j]:=x;
```

Запись элемента массива: a[i , j]

Главная диагональ i = j	**Побочная диагональ i + j = n+1**
a_{11} a_{12} a_{13} a_{14} a_{21} a_{22} a_{23} a_{24} a_{31} a_{32} a_{33} a_{34} a_{41} a_{42} a_{43} a_{44}	a_{11} a_{12} a_{13} a_{14} a_{21} a_{22} a_{23} a_{24} a_{31} a_{32} a_{33} a_{34} a_{41} a_{42} a_{43} a_{44}

Практические задания.

Создать простейший пользовательский интерфейс, для выполнения следующих задач, для ввода и вывода элементов массива использовать

компоненты Memo и StringGrid, Label -для поясняющих надписей и Button, для задания процедур.

1. Ввести одномерный массив, состоящий из 10 элементов. Заменить отрицательные элементы на противоположные по знаку. Вывести полученный массив на экран.
2. Ввести одномерный массив, состоящий из n элементов. Найти и вывести на экран номера четных элементов.
3. Ввести одномерный массив, состоящий из т элементов. Найти количество положительных и отрицательных элементов в данном массиве.
4. Ввести одномерный массив, состоящий из п элементов. Удалите из массива третий элемент.
5. Ввести одномерный массив, состоящий из п элементов. Вставьте в массив число 100 после пятого элемента.
6. Ввести двумерный массив, состоящую из n*m элементов. Найдите сумму всех элементов.
7. Заполните двумерный массив размером N*N следующим образом:

0 1 1 1 1 0
2 0 1 1 0 4
2 2 0 0 4 4
2 2 0 0 4 4
2 0 3 3 0 4
0 3 3 3 3 0.

Запишите полученные результаты в тетрадь. Оформите отчет о проделанной работе, который должен содержать тему, цель работы, формулировки задач с решениями.

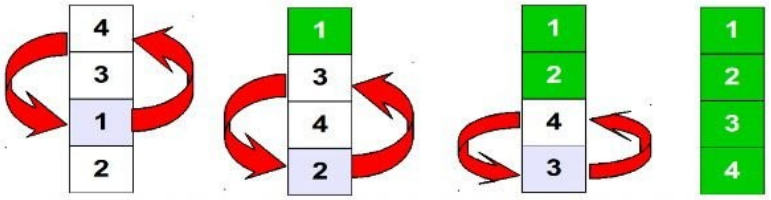
Лабораторная работа №7

Сортировка одномерных массивов.

Цель: Знакомство с разными способами сортировки данных.

Идея сортировки методом выбора следующая:

1. Найти минимальный элемент и поставить на первое место (поменять местами с A[1]).
2. Из оставшихся найти минимальный элемент и поставить на второе место (поменять местами с A[2]), и т.д.



Порядок выполнения работы

1. Создайте форму с компонентами, как указано ниже на рисунке

В инспекторе объектов измените их начальные значения:

Объект	Свойство	Значение свойства
форма **Form1**	**caption**	Сортировка массива методом выбора
метка **Label1**	**caption**	Количество элементов массива
метка **Label2**	**caption**	Границы диапазона чисел
метка **Label3**	**caption**	от
метка **Label4**	**caption**	до
Кнопка **Button1**	**caption**	Генерация и вывод массива
Кнопка **Button2**	**caption**	Сортировка и вывод массива
Текстовое поле **Edit1**	text	10
Текстовое поле **Edit2**	text	-10
Текстовое поле **Edit3**	text	10

2. Перейдем в редактор кода и в разделе описания глобальных переменных опишем переменные

N — количество элементов массива;

S — понадобится для вывода строки с результатом;

A — массив целых чисел;

k1, k2 — границы диапазона значений элементов массива.

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
StdCtrls, ComCtrls;
```

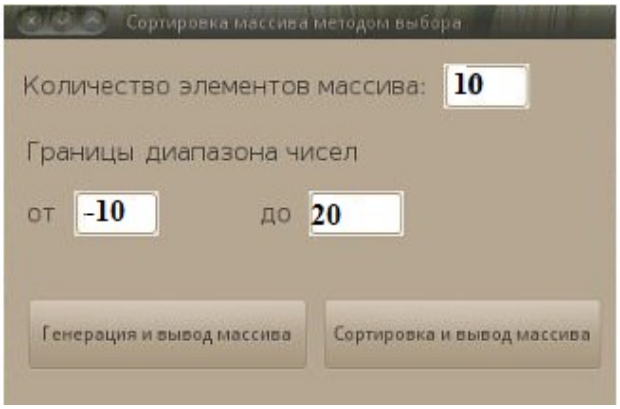
```
type
{ TForm1 }
TForm1 = class(TForm)
Button1: TButton;
Button2: TButton;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;

private
{ private declarations }
public
{ public declarations }
end;

var
Form1: TForm1;
S:string;
N,k1,k2:integer;
A:array[1..100] of integer; {пустой массив из 100 элементов, с запасом}

implementation
{ TForm1 }
initialization
{$I unit1.lrs}
end.
```

3**. Процедура генерации и вывода элементов массива:**

Для этого в инспекторе объектов выберите кнопку Button1 и на закладке события выберите OnClick. Можно поступить проще — выполнить двойной щелчок по кнопке Button1 и сразу попадаем в заготовку процедуры-обработчика (событие OnClick). Текст процедуры следующий:

```
procedure TForm1.Button1Click(Sender: TObject);
var i:integer; {локальная переменная цикла}
begin
```

```
  S:='';{начальная строка пустая}
  N:=StrToInt(Edit1.Text); {количество элементов массива}
  k1:=StrToInt(Edit2.Text); {начало диапазона}
  k2:=StrToInt(Edit3.Text); {конец диапазона}
  for i:=1 to N do begin
        A[i]:=k1+random(k2-k1); {создаем элемент массива со случайным значением в диапазоне от k1 до k2}
      S:=S+IntToStr(A[i])+' '; {преобразуем численные значения элементов массива в символы и добавляем к строке S}
  end;

  ShowMessage(S); {выводим строку в окне вывода}
end;
```



4. Сортировка массива по возрастанию по методу выбора

Обратите внимание, мы переменную массива специально объявляли как глобальную (вне процедуры) для того, чтобы после генерации элементов массива они были доступны для обработки процедурой сортировки. Ниже мы видим текст процедуры:

```
procedure TForm1.Button2Click(Sender: TObject);
var nMin,i,j,c:integer;{локальные переменные}
begin
  S:='';{начальная строка пустая}
  for i:= 1 to N-1 do begin {нужно N-1 проходов}
    nMin:= i ;
    for j:= i+1 to N do {поиск минимального от A[i] до A[N]}
      if A[j] < A[nMin] then nMin:=j; {запоминаем номер текущего минимума}
    if nMin <> i then begin {если нужно, переставляем элементы}
      c:=A[i];
      A[i]:=A[nMin];
      A[nMin]:=c;
```

```
    end;
  end;

  for i:=1 to N do S:=S+IntToStr(A[i])+' ';{преобразуем численные значения элементов массива в символы и добавляем к строке S}
  ShowMessage(S); {выводим строку в окне вывода}
end;
```

Самостоятельная работа.

1. Добавьте на форму кнопки для нахождения максимального и минимального элементов массива, напишите для них программный код.
2. Вычислите сумму всех положительных (отрицательных) элементов массива.
3. Найдите количество отрицательных (положительных) элементов массива.

Лабораторная работа №8

Обработка массивов.

Цель: Знакомство с интегрированной средой разработки программного обеспечения Lazarus. Создание нового проекта.

Лабораторная работа № 9, 10

Работа со строковым типами данных.

Цель: научиться составлять программы решения задач с использованием символьных и строковых типов данных.

Теоретические сведения:

Строковый тип данных - один из самых часто используемых в программах тип. Действительно, без него не обходится практически ни один алгоритм. Даже программы, выполняющие исключительно математические операции, порой, написаны с использованием строковых типов данных.

Строка - это последовательность символов.

В Lazarus существует несколько строковых типов. Вот основные из них:

Тип данных	Максимальная	Используемая	Используется для...

	длина	память	
ShortString	255 символов	от 2 до 256 байт	Минимальная совместимость, хранение небольших строк
AnsiString	около 2^31 символов	от 4 байт до 2 Гб	8-битные символы (ANSI), DBCS ANSI, MBCS ANSI и т.д.
WideString	около 2^30 символов	от 4 байт до 2 Гб	Юникод-символы - многопользовательские сервера, мультиязыковые приложения

Для большинства целей подходит тип AnsiString (иногда называется Long String).

Стандартные функции обработки строк:

1) Функция Length (Str: String) - возвращает длину строки (количество символов). Пример:

```
Var
  Str: String; L: Integer;
{ ... }
Str:='Hello!';
L:=Length(Str);  { L = 6 }
```

2) Функция SetLength(Str: String; NewLength: Integer) позволяет изменить длину строки. Если строка содержала большее количество символов, чем задано в функции, то "лишние" символы обрезаются. Пример:

```
var Str: String;
{ ... }
Str:='Hello, world!';
SetLength(Str, 5); { Str = "Hello" }
```

3) Функция Pos(SubStr, Str: String) - возвращает позицию подстроки в строке. Нумерация символов начинается с единицы (1). В случае отсутствия подстроки в строке возвращается 0. Пример:

```
var Str1, Str2: String; P: Integer;
{ ... }
Str1:='Hi! How do you do?';
Str2:='do';
P:=Pos(Str2, Str1);  { P = 9 }
```

4) Функция Copy(Str: String; Start, Length: Integer) - возвращает часть строки Str, начиная с символа Start длиной Length. Ограничений на Length нет - если оно превышает количество символов от Start до конца строки, то строка будет скопирована до конца. Пример:

```
var Str1, Str2: String;
{ ... }
```

```
Str1:='This is a test for Copy() function.';
Str2:=Copy(Str1, 11, 4); { Str2 = "test" }
```

5) Процедура Delete(Str: String; Start, Length: Integer) - удаляет из строки Str символы, начиная с позиции Start длиной Length. Пример:

```
var Str1: String;
{ ... }
Str1:='Hello, world!';
Delete(Str1, 6, 7); { Str1 = "Hello!" }
```

6) Процедура Insert(SubStr: String; Str: String; Pos: Integer) - вставляет в строку Str подстроку SubStr в позицию Pos. Пример:

```
var Str: String;
{ ... }
Str:='Hello, world!';
Insert('my ',Str, 8); { Str1 = "Hello, my world!" }
```

7) Функции UpperCase(Str: String) и LowerCase(Str: String) преобразуют строку соответственно в верхний и нижний регистры:

```
var Str1, Str2, Str3: String;
{ ... }
Str1:='hELLo';
Str2:=UpperCase(Str1); { Str2 = "HELLO" }
Str3:=LowerCase(Str1); { Str3 = "hello" }
```

Строки можно сравнивать друг с другом стандартным способом:

```
var Str1, Str2, Str3: String; B1, B2: Boolean;
{ ... }
Str1:='123';
Str2:='456';
Str3:='123';
B1:=(Str1 = Str2); { B1 = False }
B2:=(Str1 = Str3); { B2 = True }
```

Если строки полностью идентичны, логическое выражение станет равным True.

Дополнительные функции обработки строк:

В модуле StrUtils.pas содержатся полезные функции для обработки строковых переменных. Чтобы подключить этот модуль к программе, нужно добавить его имя (StrUtils) в раздел Uses.

1) PosEx(SubStr, Str: String; Offset: Integer) - функция аналогична функции Pos(), но позволяет задать отступ от начала строки для поиска. Если

значение Offset задано (оно не является обязательным), то поиск начинается с символа Offset в строке. Если Offset больше длины строки Str, то функция возратит 0. Также 0 возвращается, если подстрока не найдена в строке. Пример:

```
uses StrUtils;
{ ... }
var Str1, Str2: String; P1, P2: Integer;
{ ... }
Str1:='Hello! How do you do?';
Str2:='do';
P1:=PosEx(Str2, Str1, 1); { P1 = 12 }
P2:=PosEx(Str2, Str1, 15); { P2 = 19 }
```

2) Функция AnsiReplaceStr(Str, FromText, ToText: String) - производит замену выражения FromText на выражение ToText в строке Str. Поиск осуществляется с учётом регистра символов. Следует учитывать, что функция НЕ изменяет самой строки Str, а только возвращает строку с произведёнными заменами. Пример:

```
uses StrUtils;
{ ... }
var Str1, Str2, Str3, Str4: String;
{ ... }
Str1:='ABCabcAaBbCc';
Str2:='abc';
Str3:='123';
Str4:=AnsiReplaceStr(Str1, Str2, Str3); { Str4 = "ABC123AaBbCc" }
```

3) Функция AnsiReplaceText(Str, FromText, ToText: String) - выполняет то же самое действие, что и AnsiReplaceStr(), но с одним исключением - замена производится без учёта регистра. Пример:

```
uses StrUtils;
{ ... }
var Str1, Str2, Str3, Str4: String;
{ ... }
Str1:='ABCabcAaBbCc';
Str2:='abc';
Str3:='123';
Str4:=AnsiReplaceText(Str1, Str2, Str3); { Str4 = "123123AaBbCc" }
```

4) Функция DupeString(Str: String; Count: Integer) - возвращает строку, образовавшуюся из строки Str её копированием Count раз. Пример:

```
uses StrUtils;
```

```
{ ... }
var Str1, Str2: String;
{ ... }
Str1:='123';
Str2:=DupeString(Str1, 5); { Str2 = "123123123123123" }
```

5) Функции ReverseString(Str: String) и AnsiReverseString(Str: AnsiString) - инвертируют строку, т.е. располагают её символы в обратном порядке. Пример:

```
uses StrUtils;
{ ... }
var Str1: String;
{ ... }
Str1:='0123456789';
Str1:=ReverseString(Str1); { Str1 = "9876543210" }
```

6) Функция IfThen(Value: Boolean; ATrue, AFalse: String) - возвращает строку ATrue, если Value = True и строку AFalse если Value = False. Параметр AFalse является необязательным - в случае его отсутствия возвращается пустая строка.

```
uses StrUtils;
{ ... }
var Str1, Str2: String;
{ ... }
Str1:=IfThen(True, 'Yes'); { Str1 = "Yes" }
Str2:=IfThen(False, 'Yes', 'No'); { Str2 = "No" }
```

Мы рассмотрели функции, позволяющие выполнять со строками практически любые манипуляции. Как правило, вместо строки с указанным типом данных, можно использовать и другой тип - всё воспринимается одинаково. Но иногда требуются преобразования. Например, многие методы компонент требуют параметр типа PChar, получить который можно из обычного типа String функцией PChar(Str: String):

```
uses ShellAPI;
{ ... }
var FileName: String;
{ ... }
FileName:='C:\WINDOWS\notepad.exe';
ShellExecute(0, 'open', PChar(FileName), '', '', SW_SHOWNORMAL);
```

Тип Char представляет собой один-единственный символ. Работать с ним можно как и со строковым типом. Для работы с символами также существует несколько функций:

Chr(Code: Byte) - возвращает символ с указанным кодом (по стандарту ASCII):

```
var A: Char;
{ ... }
A:=Chr(69); { A = "E" }
```

Ord(X: Ordinal) - возвращает код указанного символа, т.е. выполняет противоположное действие функции Chr():

```
var X: Integer;
{ ... }
X:=Ord('F'); { X = 70 }
```

Из строки можно получить любой её символ - следует рассматривать строку как массив. Например:

```
var Str, S: String; P: Char;
{ ... }
Str:='Hello!';
S:=Str[2]; { S = "e" }
P:=Str[5]; { P = "o" }
```

Задание. Создать проекты, решающие следующие задачи:

1. Составьте программу решения задачи: *Ввести строку символов, заменить все буквы «а» на буквы «о».* Откомпилируйте программу. Проверьте правильность решения задачи на примере.

2. Составьте программу решения задачи: *Даны две фамилии определить, какая из них длиннее.* Откомпилируйте программу. Проверьте правильность решения задачи на примере.

3. Составьте программу решения задачи: *Определить и вывести на экран количество слов во введенной строке.* Откомпилируйте программу. Проверьте правильность решения задачи на примере.

4. Составьте программу решения задачи: *Заданы фамилия, имя и отчество учащегося, разделенные пробелами. Напишите программу, печатающую фамилию ученика и его инициалы.*

Откомпилируйте программу. Проверьте правильность решения на конкретном примере.

5. Составьте программу решения задачи: *Введённую с клавиатуры строку А записать в обратном порядке в строку В. Строку В вывести на экран.* Откомпилируйте программу. Проверьте правильность решения задачи на конкретном примере.

6. Составьте программу решения задачи: *Составить программу, определяющую, является ли введенное слово перевертышем.*

Откомпилируйте программу. Проверьте правильность решения задачи на конкретном примере.

7. Запишите полученные результаты в тетрадь. Оформите отчет о проделанной работе, который должен содержать тему, цель работы, формулировки задач с решениями.

Лабораторная работа № 10,11

Работа со строковым типом данных.

Задание. Выполнить задания своего варианта Разработать программы в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию, создать блок-схемы.

вариант 1

1. Как можно обратиться к отдельным символам строки?
2. Дан некоторый фрагмент программы:

```
Var    a:shortString;
Begin  …
a:=’Hello, Dolly’;
```

Что является результатом функции length(a)?

3. Какое слово называется пустым?
4. Поставьте знаки отношений для строк:

   Ord(‘B’) ? Ord(‘b’)

5. Записать фрагмент программы перевода строки строчных русских букв в прописные.
6. Составить программу, вычеркивающую каждую третью букву слова X.

вариант 2

1. Какие операции допустимы над строковыми данными?
2. Переменная b описана в разделе переменных как

```
Var    b:shortString;
```

   Может ли значение функции length(b) быть равным 300?

3. Поставьте знаки отношений для строк:
   ‘Balkon’ ? ‘balkon’
4. Какое слово при соединении слов играет роль нуля?
5. Введите два целых числа. Объедините их в одну строку и выведите результат на экран.

6. Напишите программу, которая вводит строку и выводит ее, сокращая каждый раз на 1 символ до тех пор пока в строке не останется 1 символ.

вариант 3

1. Как сравниваются два слова между собой?
2. Переменная str1 описана в разделе переменных как

   Var str1:AnsiString;

   Может ли значение функции length(str1) быть равным 300?
3. Верно ли, что символы «A» и «a» имеют одинаковые коды?
4. Что является результатом функции Concat(s1,s2)?
5. Дана строка, состоящая из цифр и текста. Вывести на экран только цифры.
6. Записать фрагмент программы перевода строки прописных русских букв в строчные.

вариант 4

1. Как определить текущую длину строки?
2. Какое слово при соединении слов играет роль нуля?
3. Поставьте знаки отношений для строк:

   ‘balkon’ ? ‘balken’
4. Что является результатом функции POS(‘as’,a)?
5. Убрать из текста уже встречающиеся символы.
6. Введено слово, переставить буквы слова в обратном порядке.

вариант 5

1. Как сравниваются два слова между собой?
2. Переменная s описана в разделе переменных как

   Var s:Char;

   Может ли значение функции length(s) быть равным 300?
3. Верно ли, что символы «B» и «b» имеют одинаковые коды?
4. Дан фрагмент программы:

   St1:=’ABCDEFG’;

   Delete(st1,3,4);

   Что будет результатом функции Delete?
5. Написать фрагмент программы, в которой два заданных с клавиатуры слова объединяются в одно. Результат вывести на экран
6. Составьте программу, удаляющую все пробелы из введенной строки.

вариант 6

1. Дан фрагмент программы:

   St1:=’ABCDEFG’;

   St2:=’blokoton’

   Что будет результатом функции St2:=Copy(st1,4,3)?
2. Какие модули надо подключить в раздел uses, чтобы можно было работать с русским текстом?
3. Поставьте знаки отношений для строки:

   Ord(‘o’) ? Ord(‘e’)
4. Что является результатом функции length(st2) (значение st2 брать из п.1)?
5. Напишите программу, которая определяет, какая из букв первая или последняя встречается в заданном слове чаще.
6. Составьте программу, определяющую на каком месте в слове стоит буква «t».

вариант 7

1. Поставьте знаки отношений для строки:

   ‘Slon’ ? ‘SLON’
2. Переменная st описана в разделе переменных как

   Var st:PChar;

   Может ли значение функции length(st) быть равным 300?
3. Верно ли, что символы «B» и «b» имеют одинаковые коды?
4. Дан фрагмент программы:

   St1:=’Arabika’;

   Delete(st1,2,3);

   Что является результатом функции Delete?
5. Замените в заданном слове все буквы «o» пробелами.
6. В тексте, состоящем из латинских букв и заканчивающимся точкой, подсчитайте количество гласных букв.

вариант 8

1. **Поставьте знаки отношений для строки:**

   ‘Moloko’ ? ‘moloko’
2. **Какие операции допустимы над строковыми данными с кириллицей?**
3. **Какая функция определяет позицию в строке равную заданному значению (подстрока)?**
4. **Переменная st описана в разделе переменных как**

   Var st:Char;

Какие значения может принимать функция length(st)?

5. **Замените каждую букву заданного с клавиатуры слова её кодом.**
6. **Подсчитать сколько раз в заданном тексте встречается прописная буква «А».**

вариант 9

1. **В чем отличие строковых типов ShortString и AnsiString&**
2. **Дан фрагмент программы:**

   St1:=’ABCDEFG’;

   St2:=’1234567’
3. **Что будет результатом функции St1:=Copy(st2,3,4)?**
4. **Какие модули надо подключить в раздел uses, чтобы можно было работать с русским текстом?**
5. **Поставьте знаки отношений для строки:**

   Ord(‘D’) ? Ord(‘d’)
6. **Что является результатом функции length(st1)** (значение st1 брать из п.2)**?**
7. Вычеркните i –ю букву слова.
8. Введите два числа. Объедините их в одну строку, убрав 0, и выведите результат на экран.

вариант 10

1. **Как определить текущую длину строки?**
2. **Переменная str1 описана в разделе переменных как**

   Var str1:AnsiString;

   Может ли значение функции length(str1) быть равным 300?
3. **Поставьте знаки отношений для строк:**

   ‘Balkon’ ? ‘balkon’
4. **Что является результатом функции Concat('система',’345’)?**
5. **Составить программу удаления лишних пробелов в тексте (лишним считать пробел, следующий непосредственно за пробелом).**
6. **Введите 3 числа. Объедините их в одну строку и выведите результат на экран.**

4.РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основные источники:

1 Казанский А. А. Программирование на visual c# 2013:Учебное пособие/Казанский А.А..-М.:Издательство Юрайт,2018, ISBN 978-5-534-02721-1.-191.

2 Огнева М. В. Программирование на языке c++: практический курс:Учебное пособие/Огнева М.В., Кудрина Е.В..-М.:Издательство Юрайт,2018, ISBN 978-5-534-05780-5.-335.

Дополнительные источники:

1 Андреева Т. А. Программирование на языке Pascal:учебное пособие/Андреева Т. А..-Москва:Интернет-Университет Информационных Технологий (ИНТУИТ),2016, ISBN 5-9556-0025-6.-277.

2 Ачкасов В. Ю. Введение в программирование на Delphi:учебное пособие/ Ачкасов В. Ю..-Москва:Интернет-Университет Информационных Технологий (ИНТУИТ),2016.-295.

Интернет-ресурсы:

1. http://www.gramota.ru – Грамота.Ру: справочно-информационный портал «Русский язык».
2. http://refoteka.ru/r-194991.html – обработка массивов в среде программирования Lazarus.
3. http://likbez.spb.ru – Тесты по русскому языку.