Московский государственный областной технологический университет

Колледж космического машиностроения и технологии

Утверждаю

Директор ККМТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шанаурова И.Ш.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2015г.

**Методические рекомендации**

**по лабораторным работам**

Дисциплина: **Основы программирования**

Специальность: **09.02.03**

2015

**Автор**: Гусятинер Л.Б. (Королев), 2015, 171 стр.

Одобрены

Цикловой комиссией

Протокол №\_\_\_\_от \_\_\_\_\_\_\_\_\_\_\_

Председатель

Начальник научно-методического центра

Заместитель директора по УМР работе

Оглавление

[Раздел 1. Программирование на языке Pascal 6](#_Toc435355794)

[Тема 1.1. Простейшие программы на языке Pascal 6](#_Toc435355795)

[Практическая работа 1. «Ручная» отладка программ с ветвлением 6](#_Toc435355796)

[Практическая работа 2. «Ручная» отладка программ с циклами 8](#_Toc435355797)

[Практическая работа 3. Построение блок-схем 12](#_Toc435355798)

[-----Лабораторная работа 1. Систематизация файлов с решенными задачами 15](#_Toc435355799)

[----Лабораторная работа 2. Построение блок-схем в программе Dia 16](#_Toc435355800)

[-----Лабораторная работа 3. Подготовка файлов с тестовыми данными 17](#_Toc435355801)

[Лабораторная работа 4. Работа с циклами REPEAT-UNTIL. Выход из цикла 18](#_Toc435355802)

[Лабораторная работа 5. Работа с вложенными циклами FOR и WHILE 20](#_Toc435355803)

[Тема 1.2. Структурное программирование 23](#_Toc435355804)

[Практическая работа 4. Техника работы с массивами 23](#_Toc435355805)

[-----Практическая работа 5. Построение блок-схем при работе с массивами 24](#_Toc435355806)

[Практическая работа 6. Техника сортировки массива 25](#_Toc435355807)

[Практическая работа 7. Техника поиска в массиве 30](#_Toc435355808)

[Лабораторная работа 6. Работа с двумерными массивами 32](#_Toc435355809)

[Практическая работа 8. Лабораторная работа 7. Работа со строками 34](#_Toc435355810)

[Лабораторная работа 8. Работа с массивами строк 36](#_Toc435355811)

[Тема 1.3. Процедурное программирование 38](#_Toc435355812)

[Практическая работа 9. Лабораторная работа 9. Работа с процедурами. 38](#_Toc435355813)

[--Практическая работа 10. Техника работы с функциями 40](#_Toc435355814)

[Лабораторная работа 10. Создание функций 41](#_Toc435355815)

[Лабораторная работа 11. Создание типов данных . Работа с множествами, перечислениями и интервалами 43](#_Toc435355816)

[Лабораторная работа 12. Работа с записями 45](#_Toc435355817)

[Практическая работа 11. Написание процедур и функций для работы с массивами 47](#_Toc435355818)

[Практическая работа 12. Написание процедур и функций для работы со строками 50](#_Toc435355819)

[Лабораторная работа 13. Работа с массивами записей 52](#_Toc435355820)

[Практическая работа 13. Поиск ошибок в программах 54](#_Toc435355821)

[Лабораторная работа 14. Создание модулей 56](#_Toc435355822)

[--Тема 1.4. Работа с файлами 58](#_Toc435355823)

[Лабораторная работа 15. Работа с текстовыми файлами 59](#_Toc435355824)

[--Практическая работа 14. Сортировка типизированных файлов 61](#_Toc435355825)

[--Практическая работа 15. Проектирование системы, основанной на данных 62](#_Toc435355826)

[--Лабораторная работа 16. Работа с типизированными файлами 63](#_Toc435355827)

[--Практическая работа 16. Техника работы с бестиповыми файлами 64](#_Toc435355828)

[--Лабораторная работа 17. Работа с бестиповыми файлами 65](#_Toc435355829)

[Раздел 2. Программирование на языке C++ 66](#_Toc435355830)

[Тема 2.1. Краткое введение 66](#_Toc435355831)

[Л1.Знакомство со средой Code::Blocks/GCC 66](#_Toc435355832)

[--Тема 2.2. Управляющие конструкции 75](#_Toc435355833)

[--П1.Анализ программ с ветвлением 75](#_Toc435355834)

[--Лабораторная работа 2. Разработка программ с ветвлением и выбором 75](#_Toc435355835)

[--П2.Анализ программ с циклами for 77](#_Toc435355836)

[--П3.Анализ программ с циклами while 77](#_Toc435355837)

[--П4.Анализ программ с циклами do..while 77](#_Toc435355838)

[Лабораторная работа 3.Разработка программ с циклами 78](#_Toc435355839)

[4. Все строки вектора должны быть записаны нижним регистром. 83](#_Toc435355840)

[--Тема 2.3 Встроенные и библиотечные типы данных 84](#_Toc435355841)

[--Л4.Работа с целыми и вещественными типами 84](#_Toc435355842)

[Лабораторная работа 5.Разработка программ со строками 85](#_Toc435355843)

[Лабораторная работа 6.Разработка программ со строками в стиле C 87](#_Toc435355844)

[--Л7.Разработка программ с файлами 89](#_Toc435355845)

[--П5.Построение трассировочных таблиц с одномерными массивами (векторами) 89](#_Toc435355846)

[Лабораторная работа 8.Работа с векторами 90](#_Toc435355847)

[--Тема 2.4 Массивы и указатели 93](#_Toc435355848)

[--П6.Построение трассировочных таблиц с многомерными массивами 93](#_Toc435355849)

[Лабораторная работа 9.Поиск в массиве 94](#_Toc435355850)

[Лабораторная работа 10.Сортировка массива 97](#_Toc435355851)

[--Л11.Обработка таблиц 99](#_Toc435355852)

[--Л12.Контрольная работа 99](#_Toc435355853)

[--Тема 2.5 Функции 100](#_Toc435355854)

[--П7."Ручное" тестирование функций 101](#_Toc435355855)

[Лабораторная работа 13.Работа с функциями 102](#_Toc435355856)

[Лабораторная работа 14.Работа с указателями 104](#_Toc435355857)

[--Л15.Создание и удаление динамических объектов 106](#_Toc435355858)

[--Л16.Передача массивов в функции 106](#_Toc435355859)

[--Тема 2.6 Структуры 106](#_Toc435355860)

[--П8.Проектирование структур данных 106](#_Toc435355861)

[--Л17.Работа со структурами 106](#_Toc435355862)

[--Л18.Работа с массивами структур 106](#_Toc435355863)

[--Л19.Передача структур в функции 106](#_Toc435355864)

[--Тема 2.7. Проекты 106](#_Toc435355865)

[--Л20.Разработка каркаса проекта 106](#_Toc435355866)

[--Л21.Разработка функций и сборка проекта 106](#_Toc435355867)

[--Л22.Тестирование и отладка проекта 106](#_Toc435355868)

[--Л23.Разработка пользовательской документации 106](#_Toc435355869)

[--Л24.Зачетная работа 106](#_Toc435355870)

[--Лабораторная работа 19. Разработка простейших программ 107](#_Toc435355871)

[--Лабораторная работа 20. Использование классов 110](#_Toc435355872)

[-----Тема 2.2. Переменные и базовые типы 111](#_Toc435355873)

[-----Лабораторная работа 21. Работа с переменными 111](#_Toc435355874)

[--Лабораторная работа 22. Работа со ссылками 112](#_Toc435355875)

[-----Лабораторная работа 23. Создание типов классов 114](#_Toc435355876)

[-----Лабораторная работа 24. Создание многофайловых проектов 115](#_Toc435355877)

[--Лабораторная работа 31. Работа с многомерными массивами 115](#_Toc435355878)

[-----Лабораторная работа 41. Работа с препроцессором 118](#_Toc435355879)

[--Лабораторная работа 49. Программирование ввода-вывода в файл 119](#_Toc435355880)

# Раздел 1. Программирование на языке Pascal

## Тема 1.1. Простейшие программы на языке Pascal

### Практическая работа 1. «Ручная» отладка программ с ветвлением

**Цель работы:**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.
2. Переменные. Имена. Объявление var. Тип integer
3. Ввод read, readlnивывод write, writeln.
4. Ветвление. if..else
5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

program p01;

{ Сумма двух чисел }

var

a, b, c: integer;

begin

write( 'Введитеа' );

readln( a );

write( 'Введите b' );

readln( b );

с := a + b;

{ Складываем значения переменных a и b и результат заносим

в переменную (ячейку) с }

writeln( c );

end.

program p02;

{ Абсолютная величина числа }

var

x, y: integer;

begin;

write( 'Введитечисло' );

readln( x );

if x >= 0 then

y := x

else

y := -x;

writeln( y );

end.

program p03;

{ Наибольшее из трех чисел. Вариант 1 }

var

a, b, c, max: integer;

begin

write( 'Первоечисло' );

readln( a );

write( 'Второе число' );

readln( b );

write( 'Третье число' );

readln( c );

if (a > b) and (a > c) then

max := a

else if (b > a) And (b > c) then

max := b

else

max := c;

writeln( max );

end.

program p04;

{ Наибольшее из трех чисел. Вариант 2 }

var

a, b, c, max: integer;

begin

write( 'Первоечисло' );

readln( a );

write( 'Второе число' );

readln( b );

write( 'Третье число' );

readln( c );

max := a;

if b > max then max := b;

if c > max then max := c;

writeln( max );

readln;

end.

**Задания для самостоятельной работы**

1. Наибольшее из четырех чисел

2. Сколько корней имеет квадратное уравнение a\*x^2+b\*x+c=0

3. Название дня недели по его номеру

4. Количество уроков по номеру дня недели

5. Название месяца по его номеру

6. Количество дней в месяце по номеру (год - невисокосный)

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Практическая работа 2. «Ручная» отладка программ с циклами

**Цель работы: Научиться отлаживать программы без помощи отладчика**

**Необходимые предварительные знания:**

1. Циклы For, While, Repeat-Until

2. Трассировка

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы.
2. Выполнить задания.

**Трассировка**

Для понимания чужой программы и для проверки правильности написания своей используют метод пошагового выполнения программы с отслеживанием значений всех переменных.

Пример 2.1.   
Вычисление суммы чисел от 6 до 10

Var

N: integer; { Это будет счетчик цикла for }  
S: integer; { В этой переменной будем накапливать сумму }

Begin

S:=0; { Вначале обнулим сумматор }

for N:=6 to 10 do

S:=S + N;

{Эта строка - тело цикла. При его выполнении каждый раз к S прибавляется очередное N. Переменную S можно сравнить с аккумулятором, в котором накапливается сумма }

writeln('Сумма чисел=', S:6);

readln

end.

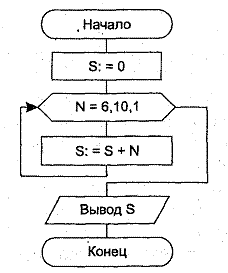


Рис. 2.1. Блок-схема алгоритма вычисления суммы чисел от 6 до 10

Для проверки правильности работы программы рекомендуется пошагово отслеживать изменение всех переменных после выполнения каждого оператора программы.

Такой процесс называется трассировкой. Продемонстрируем этот прием (табл. 2.1). В результате работы программы на экране получим число 40.

Таблица 2.1. Трассировка программы из примера 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оператор | Условие | N | S | Примечание |
| S:=0 |  |  | 0 |  |
| for N:= 6 to 10 do | Да | 6 |  |  |
| S:=S + N |  |  | 6 | 0+6-6 |
| For N:= 6 to 10 do | Да | 7 |  |  |
| S:= S + N |  |  | 13 | 6 + 7 = 13 |
| For N:= 6 to 10 do | Да | 8 |  |  |
| S:= S + N |  |  | 21 | 13 + 8 = 21 |
| For N:= 6 to 10 do | Да | 9 |  |  |
| S:= S + N |  |  | 30 | 21 + 9 = 30 |
| For N:= 6 to 10 do | Да | 10 |  |  |
| S:= S + N |  |  | 40 | 30 + 10 = 40 |
| For N:=6 to 10 do | Нет | 11 |  |  |
| writeln ('Сумма чисел',S:3) |  | ??? |  | На экране: Сумма чисел=40 |

Для операторов, выполняющих проверку условий (if, for и т. п.) в столбце «Условие» принято указывать результат проверки. В данном случае в цикле for проверяется условие продолжения цикла.

Символы «???» подчеркивают, что значение счетчика цикла по выходе из цикла считается неопределенным.

Метод трассировки очень помогает при отладке программы, когда программа выдает не тот результат, который должна выдать. Осуществляя пошаговую трассировку, мы вникаем в логику работы программы и на каждом шаге проверяем, правильны ли были наши рассуждения при ее написании.

**Вычисление суммы ряда**

Рассмотрим задачу вычисления суммы ряда:

1/(1\*1) + 1/(2\*2) + 1/(3\*3) + 1/(4\*4) + 1/(5\*5)

Здесь мы имеем ряд дробей, у которых в знаменателях записаны квадраты чисел от 1 до 5.

Рассмотрим каждую дробь как произведение двух дробей, например:

1/(3\*3) = 1/3 \* 1/3

В общем виде это можно записать так:

1/(N \* N) = 1/N \* 1/N

Блок-схема алгоритма решения задачи представлена на рис. 2.2.

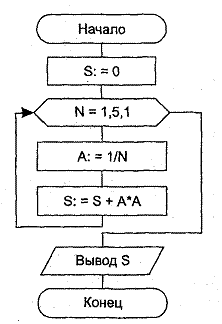


Рис. 2.2. Блок-схема алгоритма вычисления суммы ряда

Пример 2.2.

Вычисление суммы ряда

Var

N: integer; { Это будет счетчик цикла for }

S: real;

{ В этой переменной будем накапливать сумму. Выбрали для нее тип real, так как искомая сумма будет нецелым числом }

A: real; { Для записи очередной дроби 1/N (тоже число не целое) }

Begin

S:=0; { Первоначально сумматор обнулим }

for N:=1 to 5 do

begin

A:=l / N;

{ Эти строки - тело цикла. В цикле необходимо выполнить два оператора, поэтому объединяем их в блок begin ... end }

S:=S + A\*A

end;

writeln('Сумма чисел=', S:6:4);

readln

end.

Задание 1.

Написать программу вычисления n! (факториал числа n), где n положительно. Определение факториала:

0! = 1  
1! = 1   
2! = 1\*2   
3! = 1\*2\*3  
n! = 1 \* 2 \* 3 \*.... \* (n - 1) \* n

Другими словами, n! — это произведение первых n натуральных чисел.

Каждый следующий результат (обозначим его Р) получается путем умножения предыдущего результата (предыдущего Р) на счетчик, который пробегает значения от 1 до n.   
Обозначим значение счетчика буквой k.

Получаем общий вид выражения: Р = Р \* k (то есть воспользуемся рекуррентной формулой вычисления факториала: n! = (n - 1)! \* n).

Программа должна быть организована так: с клавиатуры вводится число n (n— положительно), а затем на экран выдается таблица факториалов чисел до n включительно.

**Задания для самостоятельной работы**

Задание 2.

Написать программу вычисления суммы ряда S=1 + 2 + 3 + 4 + 5 + 6. Нарисовать блок-схему и заполнить таблицу трассировки. Убедиться при трассировке, что сумма равна 21.

Таблица 2.2. Заготовка для таблицы трассировки алгоритма из задания 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оператор | Условие | S | K (счетчик) | Примечание |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Практическая работа 3. Построение блок-схем

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура линейной программы.
2. Структура программы с ветвлением
3. Блок-схемы алгоритмов

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Блок-схемы алгоритмов линейной структуры**

|  |
| --- |
| C:\ККК\Студ\13П-1\Агафонов Никита\Pascal\Begin33.png |

Рис. 3.1 Блок-схема линейного алгоритма

**{ Begin33. Программа к блок-схеме 3.1 }**

**var**

px, py, x, y, a: real;

**begin**

write('X = ');

readln(x);

write('A = ');

readln(a);

write('Y = ');

readln(y);

px := a / x;

py := px \* y;

write(px, ' ', py);

**end**.

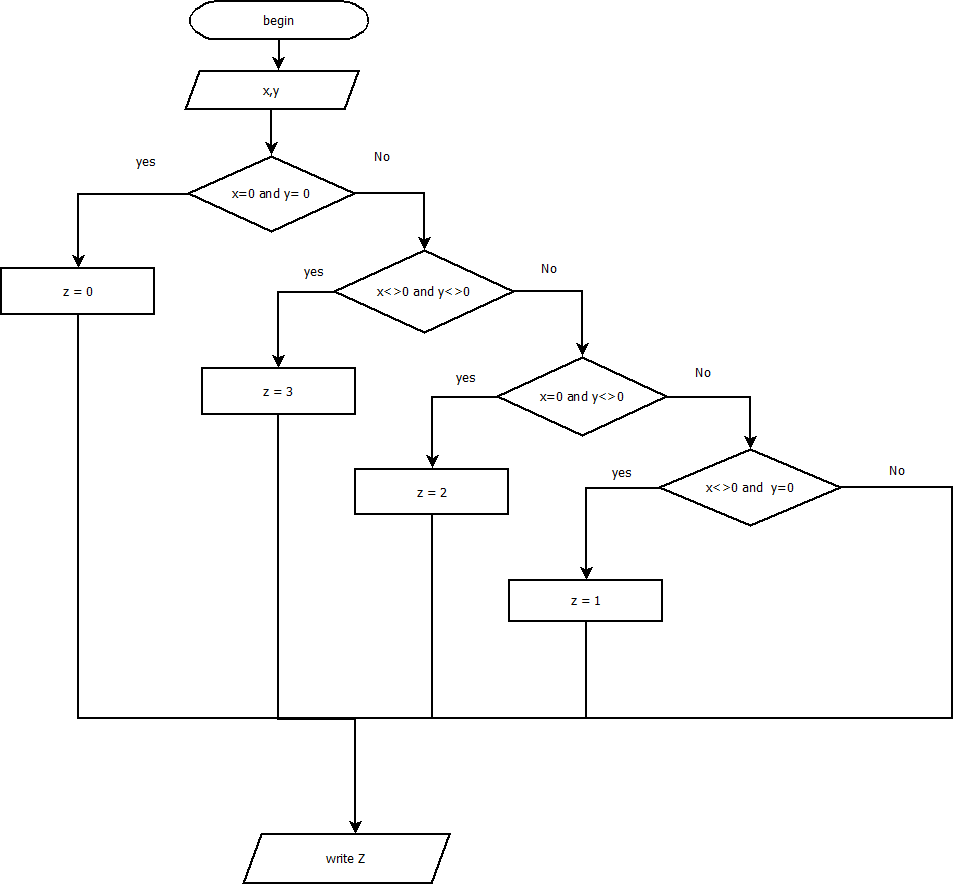
****

Рис. 3.2 Блок-схема алгоритма с ветвлением

**Программа к рис. 3.2.**

var

x,y,z:integer;

begin

read(x,y);

if (x=0) and (y=0) then z:=0;

if (x<>0) and (y<>0) then z:=3;

if (x=0) and (y<>0) then z:=2;

if (x<>0) and (y=0) then z:=1;

write(z);

end.

**Задания для самостоятельной работы**

1. Даны длины сторон треугольника A, B, C. Найти площадь треугольника S. Составьте блок-схему алгоритма решения поставленной задачи.
2. Даны координаты вершин треугольника АВС. Найти его площадь. Составьте блок-схему алгоритма решения поставленной задачи.
3. В квадратной комнате шириной A и высотой B есть окно и дверь с размерами C на D и M на N соответственно. Вычислите площадь стен для оклеивания их обоями. Составьте блок-схему алгоритма решения поставленной задачи
4. Дана величина A, выражающая объем информации в байтах. Перевести А в более крупные единицы измерения информации. Составьте блок-схему алгоритма решения поставленной задачи.
5. Вычислить путь, пройденный лодкой, если ее скорость в стоячей воде v км/ч, скорость течения реки v1 км/ч, время движения по озеру t1 ч, а против течения реки – t2 ч. Составьте блок-схему алгоритма решения поставленной задачи.
6. Вычислите значение функции Y при X=2, используя блок-схему алгоритма. Y = 2 РЕШЕНИЕ: 1. X = 2 2. Z = 8 \* 2 = 16 3. Z = = 4 4. Z = 4 – 1 = 3 5. Y = 3 \* 2 = 6 6. Y = 6 / 3 = 2

**Содержание отчета:**

1. Тексты заданий
2. Блок-схемы

### -----Лабораторная работа 1. Систематизация файлов с решенными задачами

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### ----Лабораторная работа 2. Построение блок-схем в программе Dia

**Цель работы: Научиться создавать блок-схемы в программе Dia.**

**Необходимые предварительные знания:**

1. Структура линейной программы.
2. Ввод и вывод
3. Ветвление

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### -----Лабораторная работа 3. Подготовка файлов с тестовыми данными

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 4. Работа с циклами REPEAT-UNTIL. Выход из цикла

**1. Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**2. Необходимые предварительные знания:**

2.1. Структура простейшей программы. program, begin, end.

2.2. Переменные. Имена. Объявление var. Тип integer

2.3. Ввод read, readln, и вывод write, writeln.

2.4. Цикл. REPEAT-UNTIL

2.5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**program p04\_01;**

{ Выводит n-ое количество раз нуль }

var

n, i: integer;

begin

write('Введите n: ');

read(n);

i := 0;

repeat

write(0, ' ');

inc(i); // i := i + 1

until(i = n); // повторять до того как i станет равным n

end.

**program p04\_02;**

{Даны два целых числа A и B (A < B).

Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.}

{задачник Абрамяна задача FOR 2}

var

a, b, i, j: integer;

begin

write('Введите а: ');

read(a);

write('Введите b: ');

read(b);

i := 0; // счетчик

j := 0;

repeat

j:= a + i; // записывает в перменную j полученный результат

inc(i); // i := i + 1

write(j, ' '); // вывод j

until(j = b); // делть до того как j стане равным b

writeln;

writeln('N = ', i);

end**.**

**program p04\_03;**

{Даны два целых числа A и B (A < B).

Найти сумму всех целых чисел от A до B включительно.}

{ Задачник Абрамяна задача FOR 7}

var

a, b, i, j, sum: integer;

begin

write('Введите а: ');

read(a);

write('Введите b: ');

read(b);

i := 0;

j := 0;

sum := 0;

repeat

j := a + i;

inc(i); // i := i + 1

sum := sum + j; // считает сумму от a до b

until(j = b);

write(sum);

end.

**Задания для самостоятельной работы**

1. Даны два целых числа *A* и *B* (*A* < *B*). Найти произведение всех целых чисел от *A* до *B* включительно.
2. Даны два целых числа *A* и *B* (*A* < *B*). Найти сумму квадратов всех целых чисел от *A* до *B* включительно.
3. 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.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 5. Работа с вложенными циклами FOR и WHILE

**Цель работы: Изучить технику работы с вложенными циклами**

**Необходимые предварительные знания:**

1.Тип integer, real.

2. Ввод read, readlnивывод write, writeln.

3. Ветвление. if..else

**Ход работы:**

1. Запустить PascalABC.NET

2. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

3. Выполнить задания.

program p01;

{Даны целые положительные числа N и K. Найти сумму 1^K + 2^K + … + N^K.}

varn,k,i,j,p: integer;

s:real;

begin

write('N=');

readln(n);

write('K=');

readln(k);

s:=0;

fori:=1 to ndo

begin

p:=i;

forj:=2 to k do

p:=p\*i;

s:=s+p;

end;

writeln(s);

end.

program p02;

{Даноцелоечисло N (> 0). Найтисумму 1^1 + 2^2 + … + N^N.}

varn,i,j,p: integer;

s:real;

begin

write('N=');

readln(n);

s:=0;

fori:=1 to ndo

begin

p:=i;

forj:=2 to i do

p:=p\*i;

s:=s+p;

end;

writeln(s);

end.

programp03;

{Даны целые положительные числа N и K. Используя цикл While, найти сумму 1^K + 2^K + … + N^K.}

varn,k,i,j,p: integer;

s:real;

begin

write('N=');

readln(n);

write('K=');

readln(k);

s:=0;

i:=1;

while(i<=n) do

begin

p:=i;

j:=2;

while(j<=k) do

begin

p:=p\*i;

j:=j+1;

end;

s:=s+p;

i:=i+1;

end;

writeln(s);

end.

program p04;

{Дано целое число N (> 0). Используя цикл While, найти сумму 1^1 + 2^2 + … + N^N.}

varn,i,j,p: integer;

s:real;

begin

write('N=');

readln(n);

s:=0;

i:=1;

while(i<=n) do

begin

p:=i;

j:=2;

while(j<=i) do

begin

p:=p\*i;

j:=j+1;

end;

s:=s+p;

i:=i+1;

end;

writeln(s);

end.

**Задания для самостоятельной работы**

1. Даны целые числа *A* и *B* (*A* < *B*). Вывести все целые числа от *A* до *B* включительно; при этом число *A* должно выводиться 1 раз, число *A* + 1 должно выводиться 2 раза и т. д.

2. Даны целые положительные числа *A* и *B* (*A* < *B*). Вывести все целые числа от *A* до *B* включительно; при этом каждое число должно выводиться столько раз, каково его значение (например, число 3 выводится 3 раза).

3. Вывести на экран матрицу размерности NxM, заполеную ненулевыми значениями.  
4. Дано целое число *N* (> 1). Последовательность *чисел Фибоначчи* *FK* определяется следующим образом:

*F*1 = 1,        *F*2 = 1,        *FK* = *FK*−2 + *FK*−1,    *K* = 3, 4, … .

Проверить, является ли число *N* числом Фибоначчи. Если является, то вывести True, если нет — вывести False.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

## Тема 1.2. Структурное программирование

### Практическая работа 4. Техника работы с массивами

**Цель работы: Научиться создавать программы с одномерными массивами**

**Необходимые предварительные знания:**

1. Константа

2. Массив

3. Циклы For, While

Рассмотрим следующие задачи при обработке массива:

- чтение (заполнение);

- печать;

- нахождение максимума.

// 4.1. Дан массив из N целых элементов. Отпечатать массив. Найти и отпечатать максимум.

Const MaxN=100;

// Гусятинер Л.Б., МГОТУ ККМТ, 10.11.2015

Var

I, n, max: integer;

V: array[1..MaxN] of integer;

Begin

Readln(n);

For i:= 1 to n do

Readln(v[i]);

Max := v[1];

For i:= 2 to n do

If v[i]>max then

Max := v[i];

Writeln(‘Array’);

For i:= 1 to n do

Write(v[i]:4);

Writeln;

Writeln(‘max=’,max);

End.

**Ход работы:**

1. Рассмотреть пример программ из настоящей работы.
2. Выполнить задания.

**Задания для самостоятельной работы**

1. Найти максимальный отрицательный элемент массива. Гарантируется, что такой элемент есть.
2. Отпечатать каждый k-й элемент массива.
3. Найти минимальный среди элементов массива с четными индексами.
4. Отпечатать массив с конца

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### -----Практическая работа 5. Построение блок-схем при работе с массивами

### Практическая работа 6. Техника сортировки массива

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Массив

2. Сортировка «обменом»

3. Сортировка «вставками»

4. Сортировка «выбором»

//

// 1. Проценко В.С., Чаленко П.И., Сорока Р.А. Техника программирования:

// Учеб.пособие. Киев: Выща шк., 1990

//

// Gusyatiner Leonid. Kovrov, 2005 - 2014

// Simple sorting in VBA. All these programs are based on

// the algorithmes, described in [1]

const

n=7;

type

TVector = array[1..n] of integer;

**procedure VecInit(var v: TVector; n: integer);**

// Заполнение массива

begin

v[1] := -20;

v[2] := -5;

v[3] := 10;

v[4] := 8;

v[5] := 7;

v[6] := 30;

v[7] := 40;

end;

**procedure swap(var a, b: integer);**

// Обмен содержимого ячеек

var

tmp: integer;

begin

tmp := a;

a := b;

b := tmp;

end;

**function BoolToStr(b: boolean): string;**

begin

if b then

BoolToStr := 'true'

else

BoolToStr := 'false';

end;

**procedure VecPrint(v: TVector; first, last: integer);**

// Печать одномерного массива (вектора)

var

i: integer;

begin

write('< ');

for i:= first to last do

begin

write(v[i]);

if i<n then write(',');

end;

write(' >');

end;

**procedure bubble1(var v: TVector; const n: integer);**

// Сортировка обменом (пример -"пузырьковая")

// Упорядоченный список b' получается из b систематическим обменом

// пары рядом стоящих элементов, не отвечающих требемому порядку,

// пока такие пары существуют

var

i, j: integer;

Swapped: boolean;

begin

i := 0;

writeln;

writeln('bubble1');

write(i, ' --> ');

VecPrint(v,1,n);

writeln;

Swapped := true;

while Swapped do

begin

i := i + 1;

Swapped := false;

for j := n downto i + 1 do

if v[j] < v[j - 1] then

begin

swap(v[j], v[j - 1]);

Swapped := true;

end;

write(i, ' --> ');

VecPrint(v,1,n);

writeln(' ', boolToStr(Swapped));

end;

end;

**procedure bubble2(var v: TVector; const n: integer);**

// Сортировка обменом (пример -"пузырьковая")

var

i, j: integer;

begin

i := 0;

writeln;

writeln('bubble2');

write(i, ' --> ');

VecPrint(v,1,n);

writeln;

for i := 1 to n - 1 do

begin

for j := n downto i + 1 do

if v[j] < v[j - 1] then

swap(v[j], v[j - 1]);

write(i, ' --> ');

VecPrint(v,1,n);

writeln;

end;

end;

**procedure byInsert(var v: TVector; const n: integer);**

// Сортировка вставками

var

i, j: integer;

tmp, k : Integer;

printed :Boolean;

begin

printed := False;

writeln;

writeln('by insert');

write('before v=');

VecPrint(v,1,n);

writeln('; v''=< >');

For i := 2 To n Do

Begin

If Not printed Then

Begin

write('v=');

VecPrint(v,i,n);

write('; v''=');

VecPrint(v,1,i-1);

writeln;

printed := True;

End;

tmp := v[i];

For k := 1 To i do

// Поиск места вставки

If (v[k] >= tmp) Then Break;

For j := i - 1 DownTo k do

// Освобождение места вставки

v[j + 1] := v[j];

v[k] := tmp;

write('v=');

VecPrint(v,i+1,n);

write('; v''=');

VecPrint(v,1,i);

writeln;

End;

write('after v=');

VecPrint(v,i+1,n);

write('; v''=');

VecPrint(v,1,i);

writeln;

End;

**procedure bySelect(var v: TVector; const n: integer);**

// Сортировка выбором

// Упорядоченный список b' получается из b многократным применением

// выборки из b минимального элемента и добавлением его в конец

// списка b', который первоначально должен быть пустым

var

i, j: integer;

begin

// Состояние до сортировки

writeln;

writeln('by select');

i := 0;

write('before v=');

VecPrint(v,i+1,n);

write('; v''=');

VecPrint(v,1,i);

writeln;

i := 1;

while i<= n-1 do

begin

for j := i + 1 to n do

If v[j] < v[i] then

swap(v[j], v[i]);

// Состояние после каждого прохода сортировки

write('v=');

VecPrint(v,i+1,n);

write('; v''=');

VecPrint(v,1,i);

writeln;

i:= i+1;

end;

// Состояние после сортировки

write('after v=');

VecPrint(v,i+1,n);

write('; v''=');

VecPrint(v,1,i);

writeln;

end;

var

v: TVector;

begin

VecInit(v,n);

bubble1(v,n);

VecInit(v,n);

bubble2(v,n);

VecInit(v,n);

ByInsert(v,n);

VecInit(v,n);

BySelect(v,n);

end.

**Ход работы:**

1. Рассмотреть пример программы из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Задания для самостоятельной работы**

1. Дан массив целых чисел. Отсортировать в порядке убывания
2. Дан массив строк. Отсортировать в порядке возрастания длины строки.
3. Дан массив целых чисел. Размер массива не превышает 1000. Сколько разных чисел встречается в массиве? Предварительно отсортировать.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Практическая работа 7. Техника поиска в массиве

**Цель работы: Приобрести умения поиска в массиве**

**Необходимые предварительные знания:**

1. Массив
2. Цикл

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**МЕТОДЫ ПОИСКА**

Поиск информации в несортированном массиве требует проведения последовательного просмотра массива. Просмотр начинается с первого элемента и завершается либо найденным элементом, либо достижением конца массива. Этот метод должен использоваться для несортированных данных, но он также может использоваться для отсортированных данных. Если данные отсортированы, то может использоваться двоичный поиск, который выполняется значительно быстрее.

**Последовательный поиск**

Алгоритм последовательного поиска имеет очень простой вид. Ниже представлена функция, которая выполняет поиск в символьном массиве заданной длины элемента с заданным значением ключа:

function SeqSearch(item: DataArray; count:integer;

key:DataItem):integer;

var t:integer;

begin

t:=1;

while (key<>item[t]) and (t<=count) do

t:=t+1;

if t>count then

SeqSearch:=0

else

SeqSearch:=t;

end; { конец последовательного поиска }

Эта функция выдает либо значение индекса для найденного элемента массива, либо нулевое значение, когда требуемый элемент не найден.

При прямом последовательном поиске в среднем проверяются n/2 элементов. В лучшем случае будет проверяться только один элемент, а в худшем случае будут проверятся n элементов. Если информация размещается на диске, то поиск может быть очень долгим. Однако, если данные не отсортированы, то последовательный поиск является единственным возможным в данном случае методом поиска.

**Двоичный поиск**

Если данные отсортированы, то может использоваться очень хороший метод поиска, названный двоичным поиском. При таком поиске используется метод "разделяй и властвуй". Сначала производится проверка среднего элемента. Если его ключ больше ключа требуемого элемента, то делается проверка для среднего элемента из первой половины. В противном случае делается проверка среднего элемента из второй половины. Этот процесс повторяется до тех пор, пока не будет найден требуемый элемент или не будет больше элементов для проверки.

Например, для поиска числа 4 в массиве 1 2 3 4 5 6 7 8 9 указанным методом сначала делается проверка среднего элемента, которым является число 5. Поскольку этот элемент больше 4, поиск будет продолжен в первой половине массива, т.е. среди чисел 1 2 3 4 5. Здесь средним элементом является 3. Это значение меньше 4 и поэтому первая половина не будет больше рассматриваться и поиск продолжается среди чисел 4 5. На следующем шаге нужный элемент будет найден. При двоичном поиске число сравнений в худшем случае равно log n. Для среднего случая это значение будет несколько лучше, а в лучшем случае оно равно единице.

Приводимую ниже функцию, которая реализует двоичный поиск для символьных массивов, можно использовать для поиска любой произвольной структуры данных, изменив блок сравнения и определение типа данного "DataItem".

function BSearch (item: DataArray; count:integer;

key:DataItem):integer;

var

low, high, mid: integer;

found:boolean;

begin

low:=1; high:=count;

found:=false; { не найден }

while (low<=high) and (not found) do

begin

mid:=(low+high) div 2;

if key<item[mid] then

high:=mid-1

else if key>item[mid] then

low:=mid+1

else

found:=true; { найден }

end;

if found then

BSearch:=mid

else

BSearch:=0; { не найден }

end;

**Задания для самостоятельной работы**

1. Даны два массива целых чисел. Для каждого элемента первого массива выяснить, входит ли он во второй массив.
2. Даны три массива целых чисел. Какие из элементов первого массива входят во второй, но не входят в третий?

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 6. Работа с двумерными массивами

1. Цель работы: знакомство с двумерными массивами и выполнение простейших программ.

Ход работы:

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

2. Выполнить задания.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004. }

{ Matrix1. Даны целые положительные числа M и N. Сформировать целочисленную

матрицу размера M \* N, у которой все элементы i-й строки имеют значение

10\*i (i = 1, …, m). }

{ Решение Гусятинер Л.Б., 10.2012 }

const

m = 10;

n = 10;

var

i, j: integer;

a: array[1..m, 1..n] of integer;

k: integer;

begin

for i := 1 to m do

begin

k := 10 \* i;

for j := 1 to n do

a[i, j] := k;

end;

for i := 1 to m do

begin

for j := 1 to n do

write(a[i, j]:4);

writeln;

end;

readln;

end.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004. }

{ Даны целые положительные числа M и N. Сформировать целочисленную матрицу размера M × N,

у которой все элементы J-го столбца имеют значение 5·J (J = 1, …, N)}

{ Решение Гусятинер Л.Б., 10.2012 }

const

m = 10;

n = 10;

var

i, j: integer;

a: array[1..m, 1..n] of integer;

k: integer;

begin

for j := 1 to n do

begin

k := 5 \* j;

for i := 1 to m do

a[i, j] := k;

end;

for i := 1 to m do

begin

for j := 1 to n do

write(a[i, j]:4);

writeln;

end;

readln;

end.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004. }

{ Даны целые положительные числа M, N и набор из M чисел. Сформировать матрицу

размера M × N, у которой в каждом столбце содержатся все числа из исходного

набора (в том же порядке) }

{ Решение Гусятинер Л.Б., 10.2012 }

const

m = 5;

n = 5;

var

i, j: integer;

a: array[1..m, 1..n] of integer;

v: array [1..n] of integer;

begin

for i := 1 to m do

Readln(v[i]);

for i := 1 to m do

for j := 1 to n do

a[i, j] := v[i];

for i := 1 to m do

begin

for j := 1 to n do

write(a[i, j]:4);

writeln;

end;

readln;

end.

Задания для самостоятельной работы [1]

1. Matrix4. Даны целые положительные числа *M*, *N* и набор из *N* чисел. Сформировать матрицу размера *M* × *N*, у которой в каждой строке содержатся все числа из исходного набора (в том же порядке).
2. Matrix7°. Дана матрица размера *M* × *N* и целое число *K* (1 ≤ *K* ≤ *M*). Вывести элементы *K*-й строки данной матрицы.
3. Matrix17. Дана матрица размера *M* × *N* и целое число *K* (1 ≤ *K* ≤ *M*). Найти сумму и произведение элементов *K*-й строки данной матрицы.

Содержание отчета:

1. Тексты заданий

2. Тексты программ

### Практическая работа 8. Лабораторная работа 7. Работа со строками

**Цель работы**: Приобретение умений чтения и разработки программ со строками.

**Необходимые предварительные знания**:

1. Строка. Тип string
2. Функции работы с символами и строками chr(), ord(), length().

**Ход работы**:

1. Запустить PascalABC.NET
2. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
3. Выполнить задания.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004.}

{ String13. Дана строка. Подсчитать количество содержащихся в ней цифр}

{ Решение: Л.Б.Гусятинер 10.2012 }

var

s: string;

i, k: integer;

begin

readln(s);

k := 0;

for i := 1 to length(s) do

if s[i] in ['0'..'9'] then

k := k + 1;

writeln(k);

end.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004.}

{ String16. Дана строка. Преобразовать в ней все прописные латинские буквы в строчные. }

{ Решение: Л.Б.Гусятинер 09.2012 }

var

s: string;

i: integer;

begin

readln(s);

for i:= 1 to length(s) do

if s[i] in ['A'..'Z'] then

s[i]:= chr(ord(s[i])-ord('A')+ord('a'));

writeln(s);

end.

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004.}

{ String22. Дана строка, изображающая целое положительное число. Вывести сумму цифр этого числа.}

{ Решение: Л.Б.Гусятинер 10.2012 }

var

sum, c, i: integer;

s: string;

begin

readln(s);

sum := 0;

for i := 1 to length(s) do

begin

c := ord(s[i]) - ord('0');

sum := sum + c;

end;

writeln(sum);

end.

Задания для самостоятельной работы [1]

1. Дан символ *C*. Вывести его *код* (т. е. номер в кодовой таблице).

2. Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.

3. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.

4. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.

Содержание отчета:

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 8. Работа с массивами строк

**Цель работы**: изучение техники работы с массивами строк

**Необходимые предварительные знания**:

1. Структура простейшей программы. program, begin, array, string, end.

2. Переменные. Имена. Объявление var. Тип array of string

3. Ввод read, readlnивывод write, writeln.

4. Ветвление. if..else

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{ Ввод-вывод массива строк }

{ Гусятинер Л.Б., 17.09.2015 }

**const**

n = 10;

**var**

i: integer;

a: **array** [1..n] **of** string;

**begin**

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

readln(a[i]);

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

write(a[i], ' ');

writeln;

**end**.

{ Сравнить массивы поэлементно и выдать результат }

{ Гусятинер Л.Б., 17.09.2015 }

const

n = 10;

var

a, b: array [1..n] of string;

i: integer;

equal: boolean;

begin

for i := 1 to n do

readln(a[i]);

for i := 1 to n do

readln(b[i]);

equal := true;

for i := 1 to n do

if a[i] <> b[i] then

begin

equal := false;

break;

end;

if equal then

writeln('равны')

else

writeln('не равны');

end.

**Задания для самостоятельной работы**

1. Сложить массивы из строк

2. Отсортировать массивы из строк по возрастанию

3. Отсортировать массивы из строк по убыванию

4. Создать массив из строк и вывести четные элементы массива

5. Создать массив из строк и вывести нечетные элементы массива

**Содержание отчета**:

1. Тексты заданий

2. Тексты программ

## Тема 1.3. Процедурное программирование

### Практическая работа 9. Лабораторная работа 9. Работа с процедурами.

**Цель работы**:

1. Приобретение умений чтения и анализа процедур.
2. Приобретение умений использования процедур при разработке программ в среде PascalABC.Net

**Необходимые предварительные знания:**

1. Процедуры.
2. Локальные переменные
3. Параметры

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{ сложение двух чисел}

uses

Crt;

procedure Add(a, b: Integer);

var

C: Integer;

begin

C := A + B;

Write(C);

end;

var

N1, N2: Integer;

C: Char;

begin

repeat

{ Очищаем экран }

ClrScr;

{ Читаем переменные }

Write('Введите число N1: ');

Readln(N1);

Write('Введитечисло N2: ');

Readln(N2);

Writeln;

{ Вызываем нашу процедуру }

Add(N1, N2);

{ Спрашиваем - Выйти? }

Writeln;

Write('Выйти? (Y/N): ');

Readln(C);

until UpCase(C) = 'Y';

end.

{ Описать процедуру PowerA3(A, B), вычисляющую третью степень числа A и

возвращающую ее в переменной B (A - входной, B - выходной параметр;

оба параметра являются вещественными).

С помощью этой процедуры найти третьи степени пяти данных чисел. }

var

i: integer;

A, B: real;

procedure PowerA3(A: real; varB: real);

begin

B := A \* A \* A;

end;

/////// основная программа /////////////

begin

Writeln('Введите пять вещественных чисел');{заголовок}

i := 0;

while(i < 5) do

begin

Read(a);

inc(i);

PowerA3(A, B);

Write(' ', B);

end;

end.

**Задания для самостоятельной работы**

При помощи процедур выполнить следующее:

1. Наибольшее из трёх чисел
2. Вычислить среднее арифметическое при помощи процедур
3. Даны целые числа *K* и *N* (*N* > 0). Вывести *N* раз число *K*.
4. Количество уроков по номеру дня недели
5. Даны два ненулевых числа. Найти сумму, разность, произведение и частное их модулей.
6. Дано целое число *N* (> 0). Найти произведение 1.1 · 1.2 · 1.3 · … (*N* сомножителей).

Содержание отчета:

1. Тексты заданий

2. Тексты программ

### --Практическая работа 10. Техника работы с функциями

### Лабораторная работа 10. Создание функций

1. Цель работы: Научиться создавать пользовательские функции

2. Необходимые знания:

2.1 Условные операторы

2.2 Циклы

Ход работы:

1. Запустить PascalABC.NET

2. Рассмотреть примеры

3. Выполнить задания

10-1

//Функция biggest возвращает большее из 2-х чисел.

Function biggest(a,b: integer):integer;

Begin

If a>b then biggest:=a

Else biggest:=b

End;

Var

A,b:integer;

Begin

Readln(a,b);

Writeln(biggest(a,b));

End.

10-2  
//Функция вфчисляет число А в 5-й степени.

function powerA5(a:integer):integer;

begin

powerA5:=a\*a\*a\*a\*a;

end;

Var

a:integer;

Begin

Readln(a);

Writeln(powerA5(a));

End.

10-3

//Функция определяет количество корней уравнения A•x2 + B•x + C = 0

functionRootCount(a,b,c:integer) :integer;

var

d:integer;

begin

d:=b\*b-4\*a\*c;

if d<0 then RootCount:=0

else

if d=0 then RootCount:=1

elseRootCount:=2;

end;

Var

a,b,c:integer;

Begin

Readln(a,b,c);

Writeln(RootCount(a,b,c));

End.

**Задания для самостоятельной работы**

1. Описать функцию CircleS(R) вещественного типа, находящую площадь круга радиуса R (R — вещественное). С помощью этой функции найти площади трех кругов с данными радиусами. Площадь круга радиуса R вычисляется по формуле S = π·R2. В качестве значения π использовать 3.14.
2. Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых чисел от A до B включительно (A и B — целые). Если A > B, то функция возвращает 0. С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.
3. Описать функцию Even(K) логического типа, возвращающую True, если целый параметр K является четным, и False в противном случае. С ее помощью найти количество четных чисел в наборе из 10 целых чисел.
4. Описать функцию Calc(A, B, Op) вещественного типа, выполняющую над ненулевыми вещественными числами A и B одну из арифметических операций и возвращающую ее результат. Вид операции определяется целым параметром Op: 1 — вычитание, 2 — умножение, 3 — деление, остальные значения — сложение. С помощью Calc выполнить для данных A и B операции, определяемые данными целыми N1, N2, N3.

Содержание отчета:

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 11. Создание типов данных . Работа с множествами, перечислениями и интервалами

**Цель работы: Приобрести умения работы с пользовательскими типами данных.**

**Необходимые предварительные знания:**

1. Множества. Элементы. Операции
2. Интервалы
3. Перечисления

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Пример Л11.1**

// Дана строка. Преобразовать строку удалив из неё все символы,

// не являющиеся латинскими буквами. Если преобразование строки

// невозможно,выдать об этом сообщение с указанием причины.

// Гусятинер Л.Б., РФПРО, 21.03.2011 г.

**program p182575;**

**var**

s: string;

ErrLevel:integer;

**procedure DeleteNonLatins(var s: string; var ErrLevel: integer);**

**const**

ABC = ['A'..'Z','a'..'z'];

**var**

i,j,len, nDeleted: integer;

**begin**

len:= length(s);

nDeleted:= 0;

j:= 1;

for i:= 1 to len do

begin

if (s[i] in ABC) then

begin

if i<>j then

s[j] := s[i];

inc(j);

end

else

inc(nDeleted);

end;

if nDeleted=len then

ErrLevel := 1

else if nDeleted=0 then

ErrLevel := 2

else

begin

s[0]:= chr(len-nDeleted);

Errlevel := 0;

end;

**end;**

**begin**

write('s=');

readln(s);

DeleteNonLatins(s, Errlevel);

case ErrLevel of

0: writeln('после преобразования s=', s);

1: writeln('Ошибка: после преобразования пустая строка');

2: writeln('Ошибка: строка содержит только латинские буквы');

else

writeln('Неизвестная ошибка');

end;

end.

Пример работы

s=Учим Pascal Учим Pascal Учим Pascal

после преобразования s=PascalPascalPascal

s=

Ошибка: после преобразования пустая строка

s=abcdefghijklmn

Ошибка: строка содержит только латинские буквы

**Задания для самостоятельной работы**

1. Даны 2 строки символов. Сколько различных символов 1й строки являются одновременно символами 2й?
2. Даны три строки символов. Какие из символов 1й строки входят во 2ю, но не входят в 3ю?

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 12. Работа с записями

**Цель работы**: приобретение умений работы с записями.

**Необходимые предварительные знания**:

1. Записи.
2. Поля.
3. Оператор присоединения.

**Ход работы**:

1. Запустить PascalABC.NET

2. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

3. Выполнить задания.

**Пример Л12.1 Записи**

// Записи

// Гусятинер Л.Б.

// Ковров, КГТА, 2005 / Королёв, ФТА, ККМТ, 2012

// 1. Тип комплексное число

type TComplex = record // Комплексное число

re: real; // действительная часть

im: real; //мнимая часть

end;

**function CAbs(c: TComplex): real;**

begin

CAbs := Sqrt(sqr(c.re)+sqr(c.im));

end;

var

c1: TComplex;

c2: Tcomplex;

**begin**

readln(c1.re);

readln(c1.im);

with c2 do

begin

readln(re);

readln(im);

end;

writeln('c1->', CAbs(c1));

writeln('c2->', CAbs(c2));

**end.**

**Пример Л12.2 Записи**

// Записи

// Гусятинер Л.Б.

// Ковров, КГТА, 2005 / Королёв, ФТА, ККМТ, 2012

// 2. Типы Папа, Сын, Внук

// Папа – фамилия

// Сын – фамилия + отчество

// Внук – фамилия + отчество + имя

**type**

Papa = record

LastName: string;

end;

Son = record

p: Papa;

MidName: String;

end;

GrandSon = record

s: Son;

FirstName: String;

end;

**var**

gs: GrandSon;

st: String;

**begin**

st := '';

with gs do

begin

FirstName := 'Иван';

with s do

begin

MidName := 'Иванович';

with p do

LastName := 'Иванов';

end;

end;

with gs do

st := st + FirstName + ' ' + s.MidName + ' ' + s.p.LastName;

writeln(st);

**end.**

**Задания для самостоятельной работы**

1. Создать запись «Студент» с полями: группа, имя, фамилия, оценка по математике , оценка по программированию.

Напечатать среднюю оценку по математике и по программированию.

1. Создать запись «Треугольник» с полями сторона1, сторона2, сторона3.Напечатать площадь этих треугольников.
2. Создать тип данных «Автор»
3. Создать тип данных «Книга»
4. Создать тип данных «Читатель»
5. Создать тип данных «Библиотека» на основе пп.3-5

**Содержание отчета**:

1. Тексты заданий

2. Тексты программ

### Практическая работа 11. Написание процедур и функций для работы с массивами

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Массив
2. Процедура
3. Функция

**Пример 11.1**

{ М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004. }

{ Param11. Описать процедуру Sort(A, N), выполняющую сортировку по возрастанию

вещественного массива A размера N. Массив A является входным и выходным параметром.

С помощью этой процедуры отсортировать массивы A, B, C размера NA, NB, NC соответственно. }

{ Решение Гусятинер Л.Б., 10.2012 }

const

MaxN = 10;

type

TVector = array[1..MaxN] of integer;

var

i: integer;

a:TVector;

na: integer;

**procedure VPrint(v: TVector; n: integer; s: string);**

var

i: integer;

begin

write(s, ' <');

for i:= 1 to na do

begin

if i>1 then write(',');

write(a[i]);

end;

writeln('>');

end;

**procedure VSort(var v: TVector; n: integer);**

var

x: integer;

i, j: integer;

begin

for i:= 1 to n-1 do

for j:= n downto i+1 do

if a[j-1]>a[j] then

begin

x:= a[j];

a[j] := a[j-1];

a[j-1] := x;

end;

end;

begin

randomize;

na:= 2 + random(maxN-1);

for i:= 1 to na do

a[i] := random(na+1);

VPrint(a,na,'Массив A до сортировки');

VSort(a,na);

VPrint(a,na,'Массив A после сортировки');

readln;

end.

**Пример 11.2**

Дан массив. Найти максимальное значение.

// Максимальный элемент массива

// Гусятинер Л.Б., МГОТУ ККМТ, 10.11.2015

**const**

MaxN = 1000;

**type**

TVector = **array**[1..MaxN] **of** integer;

**procedure** VInput(**var** v: TVector; **var** n: integer);

**var**

i: integer;

**begin**

readln(n);

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

readln(v[i]);

**end**;{ VInput }

**function** VMax(**const** v: TVector; **const** n: integer): integer;

**var**

i: integer;

max: integer;

**begin**

max := v[1];

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

**if** v[i] > max **then**

max := v[i];

Vmax := max;

**end**;{ VMax }

**var**

a: TVector;

n: integer;

**begin**

VInput(a, n);

writeln('max=', VMax(a, n));

**end**.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Задания для самостоятельной работы**

1. Написать функцию для нахождения 2го по значению элемента массива
2. Написать процедуру сортировки «вставками» массива строк
3. В процедуру ввода из примера добавить приглашение на ввод.
4. В процедуру ввода из примера добавить проверку вхождения n в интервал 1..MaxN.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Практическая работа 12. Написание процедур и функций для работы со строками

**Цель работы:** Научиться создавать несложные процедуры и функции для работы со строками.

**Необходимые предварительные знания:**

1. Строки.

2. Процедуры

3. Функции

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

Задание 1. Написать функцию, TrimLeft, возвращающую строку с удаленными начальными пробелами.

{ Гусятинер Л.Б., 17.09.2015 }

function TrimLeft(s: string): string;

// Возвращает строку с удаленными начальными пробелами

var

s1: string;

i: integer;

begin

s1:= '';

i:= 1;

while (i<=length(s)) and (s[i]=' ') do

i:= i+1;

while (i<=length(s)) do

begin

s1 := s1+s[i];

i:= i+1;

end;

TrimLeft := s1;

end; // TrimLeft

var

s: string;

begin

readln(s);

writeln(trimleft(s));

end.

Задание 2. Написать процедуруInsert, вставляющую подстроку source в строку s с позиции index.

{ Гусятинер Л.Б., 17.09.2015 }

**procedure** Insert(source: string; **var** s: string; index: integer);

// Вставляет подстроку source в строку s с позиции index

**var**

res: string;

i, j: integer;

**begin**

res := '';

**for** i:= 1 **to** index-1 **do**

res := res+s[i];

**for** j:= 1 **to** length(source) **do**

res := res+source[j];

**for** i:= index **to** length(s) **do**

res := res+s[i];

s:= res;

**end**; // Insert

**var**

source, s: string;

index: integer;

**begin**

readln(source);

readln(s);

readln(index);

Insert(source, s, index);

writeln(s);

**end**.

**Задания для самостоятельной работы**

1. **Написать реализацию процедур**
   1. procedure Delete(var s: string; index,count: integer); // Удаляет из строки s count символов с позиции index
   2. procedure Val(s: string; var value: integer; var err: integer);

// Преобразует строковое представление s целого числа к числовому значению и записывает его в переменную value.

// Если преобразование успешно, то err=0, иначе err>0

* 1. procedure Str(i: integer; var s: string); // Преобразует целое значение i к строковому представлению и записывает результат в s

1. **Написать реализацию функций**
   1. function Pos(subs,s: string): integer; // Возвращает позицию подстроки subs в строке s. Если не найдена, возвращает 0
   2. function PosEx(subs,s: string; from: integer := 1): integer; // Возвращает позицию подстроки subs в строке s начиная с позиции from. Если не найдена, возвращает 0
   3. function Copy(s: string; index,count: integer): string; // Возвращает подстроку строки s длины count с позиции index
   4. function Trim(s: string): string; // Возвращает строку с удаленными начальными и конечными пробелами
   5. function IntToStr(a: integer): string; // Преобразует целое число к строковому представлению

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Лабораторная работа 13. Работа с массивами записей

**Цель работы: приобрести умения работы с массивом записей**

**Необходимые предварительные знания:**

1. Массив
2. Запись
3. Цикл.
4. Процедура

**Ход работы:**

1. Рассмотреть пример программы из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Пример 13.1.**

{ Гусятинер Л.Б., Ковров, КГТА, 09.05.2006 }

const

size = 10;

type

TStudent = record

fm, im,ot, group: string;

end;

TStudents = array[1..size] of TStudent;

var

arr\_st: TStudents;

**procedure InputArr( var arr\_st: TStudents );**

var

i: integer;

begin

clrscr;

for i:= low( arr\_st ) to high( arr\_st ) do begin

Writeln( 'student[', i, ']');

Write('fm=' );

Readln( arr\_st[i].fm );

Write( 'im=' );

Readln( arr\_st[i].im );

Write( 'ot=' );

Readln( arr\_st[i].ot );

Write( 'group=' );

Readln( arr\_st[i].group );

end

end; { InputArr }

**procedure PrintArr( arr\_st: TStudents );**

var

i: integer;

begin

clrscr;

for i:= low( arr\_st ) to high( arr\_st ) do begin

With arr\_st[i] do

Write( fm, im, ot, group );

Writeln

end

end; { PrintArr }

begin

InputArr( arr\_st );

PrintArr( arr\_st );

Readln;

end.

**Задания для самостоятельной работы**

1. Построить тип студент: фамилия, имя, отчество, группа.
2. Написать процедуры ввода и печати массива студентов.
3. Написать программу для теста процедур

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### Практическая работа 13. Поиск ошибок в программах

**Цель работы: Научиться обнаруживать ошибки в программе**

**Необходимые предварительные знания:**

1. Трассировка
2. «Ручной» прогон.
3. Отладочная печать

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

Пример П13.1. Найти ошибки в вычислении факториала числа n

//

k:= 1;

f:= 0;

while (k<n) do

begin

f:= f\*k;

k:= k+1;

end;

Для обнаружения ошибки достаточно провести «ручной» прогон при n=3

|  |  |  |
| --- | --- | --- |
| F | K | K<n |
| 0 | 1 | True |
| 0\*1=0 | 2 | True |
| 0\*2=0 | 3 | False |

Обнаруживается неверная инициализация F (нужно было f=1)

**Задания для самостоятельной работы.**

**Найти ошибки в программах**

1. **Возвести число 2 в степень n.**

P:= 1;

For k:= 1 to n do

P:= k\*2;

Writeln(p);

1. **Вводятся 3 числа. Сколько среди них положительных и отрицательных?**

var

a,b,c,q,w:integer;

begin

readln(a,b,c);

if a < 0 then

q := q + 1

else

w := w + 1;

if b < 0 then

q := q + 1

else

w := w + 1;

if c < 0 then

q := q + 1

else

w := w + 1;

writeln('Количество отрицательных чисел',' ', w);

writeln('Количество положительных чисел',' ', q);

end.

**Содержание отчета:**

1. Тексты заданий
2. Трассировочные таблицы
3. Тексты программ

### Лабораторная работа 14. Создание модулей

**Цель работы: Приобрести умения работать с модулями**

**Необходимые предварительные знания:**

1. Структура программы.
2. Структура модуля

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Структура модуля.**

Модули предназначены для разбиения текста программы на несколько файлов. В модулях описываются переменные, константы, типы, классы, процедуры и функции. Для того чтобы эти объекты можно было использовать в вызывающем модуле (которым может быть и основная программа), следует указать имя файла модуля (без расширения .pas) в разделе uses вызывающего модуля. Файл модуля (.pas) или откомпилированный файл модуля (.pcu) должен находиться либо в том же каталоге, что и основная программа, либо в подкаталоге Lib системного каталога программы PascalABC.NET.

Модуль имеет следующую структуру:

unit имя модуля;

interface

// раздел интерфейса

implementation

// раздел реализации

initialization

// раздел инициализации

finalization

// раздел финализации

end.

Имеется также упрощенный синтаксис модулей без разделов интерфейса и реализации. Первая строка обязательна и называется заголовком модуля. Имя модуля должно совпадать с именем файла. Раздел интерфейса и раздел реализации модуля могут начинаться с раздела uses подключения внешних модулей и пространств имен .NET. Имена в двух разделах uses не должны пересекаться. Раздел интерфейса включает объявление всех имен, которые экспортируются данным модулем в другие модули (при подключении его в разделе uses). Это могут быть константы, переменные, процедуры, функции, классы, интерфейсы. Реализация методов классов может быть дана прямо в разделе интерфейса, но это не рекомендуется. Раздел реализации содержит реализацию всех процедур, функций и методов, объявленных в разделе интерфейса. Кроме этого, в разделе реализации могут быть описания внутренних имен, которые не видны вне модуля и используются лишь как вспомогательные. Раздел инициализации и раздел финализации представляют собой последовательность операторов, разделяемых символом ;. Операторы из раздела инициализации модуля выполняются до начала основной программы, операторы из раздела финализации модуля - после окончания основной программы. Порядок выполнения разделов инициализации и разделов финализации подключенных модулей непредсказуем. Как раздел инициализации, так и раздел финализации могут отсутствовать. Вместо разделов инициализации и финализации может присутствовать только раздел инициализации в виде

begin

последовательность операторов

end.

**Пример Л14.1**

unit Lib;

interface

uses GraphABC;

const Dim = 5;

var Colors: array [1..Dim] of integer;

function RandomColor: integer;

procedure FillByRandomColor;

implementation

function RandomColor: integer;

begin

Result := RGB(Random(255),Random(255),Random(255));

end;

procedure FillByRandomColor;

begin

for i: integer := 1 to Dim do

Colors[i] := RandomColor;

end;

initialization

FillByRandomColor;

end.

**Задания для самостоятельной работы**

1. Спроектировать модуль для операций с векторами. Подключить его к программе, демонстрирующей его использование.
2. Спроектировать модуль для операций с комплексными числами. Подключить его к программе, демонстрирующей его использование.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

## --Тема 1.4. Работа с файлами

### Лабораторная работа 15. Работа с текстовыми файлами

**Цель работы**: выполнение простейших программ с текстовыми файлами.

**Необходимые предварительные знания**:

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Типы string, Text

3. Ввод read, readlnивывод write, writeln.

4. Ветвление. if..else

5. Комментарии.

**Ход работы**:

1. Запустить PascalABC.NET

2. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

3. Выполнить задания.

program p1;

{ Вывод текстового документа}

var

f: text;

fname: string;

s: string;

begin

writeln('Введите путь к файлу');

readln(fname); //прочли имя файла

assign(f, fname); //связали файловую переменную с именем файла

reset(f); //открыли для чтения

while not eof(f) do

begin

readln(f, s); //чтение файла в переменную типа string

writeln(s); //вывод переменной типа string

end;

close(f); //закрытиефайла

readln;

writeln('Конец операции');

end.

program p2;

{ Удаление файла/текстового документа }

var

f: text;

fname: string;

begin

writeln('Введите путь нуждающегося в удалении файла');

readln(fname); //ввод имени файла

assign(f, fname); //связка файловой переменной с именем файла

erase(f); //выполнение удаления файла

end.

program p3;

{ Запись в файл }

var

f: text;

fname: string;

s: string;

begin

writeln('Введите путь нуждающегося в переименовывании файла');

readln(fname); //ввод имени файла

assign(f, fname); //связка файловой переменной с именем файла

rewrite(f);

write(f, s);

close(f);

readln;

end.

**Задания для самостоятельной работы**

1. Дано имя файла и целые положительные числа *N* и *K*. Создать текстовый файл с указанным именем и записать в него *N* строк, каждая из которых состоит из *K* символов «\*» (звездочка).

2. Дано имя файла и целое число *N* (0 <*N*< 27). Создать текстовый файл с указанным именем и записать в него *N* строк: первая строка должна содержать *строчную* (т. е. маленькую) латинскую букву «a», вторая — буквы «ab», третья — буквы «abc» и т. д.; последняя строка должна содержать *N* начальных строчных латинских букв в алфавитном порядке.

3. Дано имя файла и целое число *N* (0 <*N*< 27). Создать текстовый файл с указанным именем и записать в него *N* строк длины *N*; строка с номером *K* (*K* = 1, …, *N*) должна содержать *K* начальных *прописных* (т. е. заглавных) латинских букв, дополненных справа символами «\*» (звездочка). Например, для *N* = 4 файл должен содержать строки «A\*\*\*», «AB\*\*», «ABC\*», «ABCD».

4. Дан текстовый файл. Вывести количество содержащихся в нем символов и строк (маркеры концов строк EOLN и конца файла EOF при подсчете количества символов не учитывать).

5. Дана строка *S* и текстовый файл. Добавить строку *S* в конец файла.

**Содержание отчета**:

1. Тексты заданий

2. Тексты программ

### --Практическая работа 14. Сортировка типизированных файлов

### --Практическая работа 15. Проектирование системы, основанной на данных

### --Лабораторная работа 16. Работа с типизированными файлами

### --Практическая работа 16. Техника работы с бестиповыми файлами

### --Лабораторная работа 17. Работа с бестиповыми файлами

# Раздел 2. Программирование на языке C++

## Тема 2.1. Краткое введение

### Л1.Знакомство со средой Code::Blocks/GCC

**1. Цель работы:** Научиться работать в среде Code::Blocks.

**2. Необходимые предварительные знания:**

2.1. Работа в редакторе NotePad++, PascalABC.Net.

**Ход работы:**

1. Запустить программу Code::Blocks, рассмотреть примеры.
2. Выполнить задания.

**Описание работы**

1. **Открываем Code::Blocks** (Рис. 18.1)

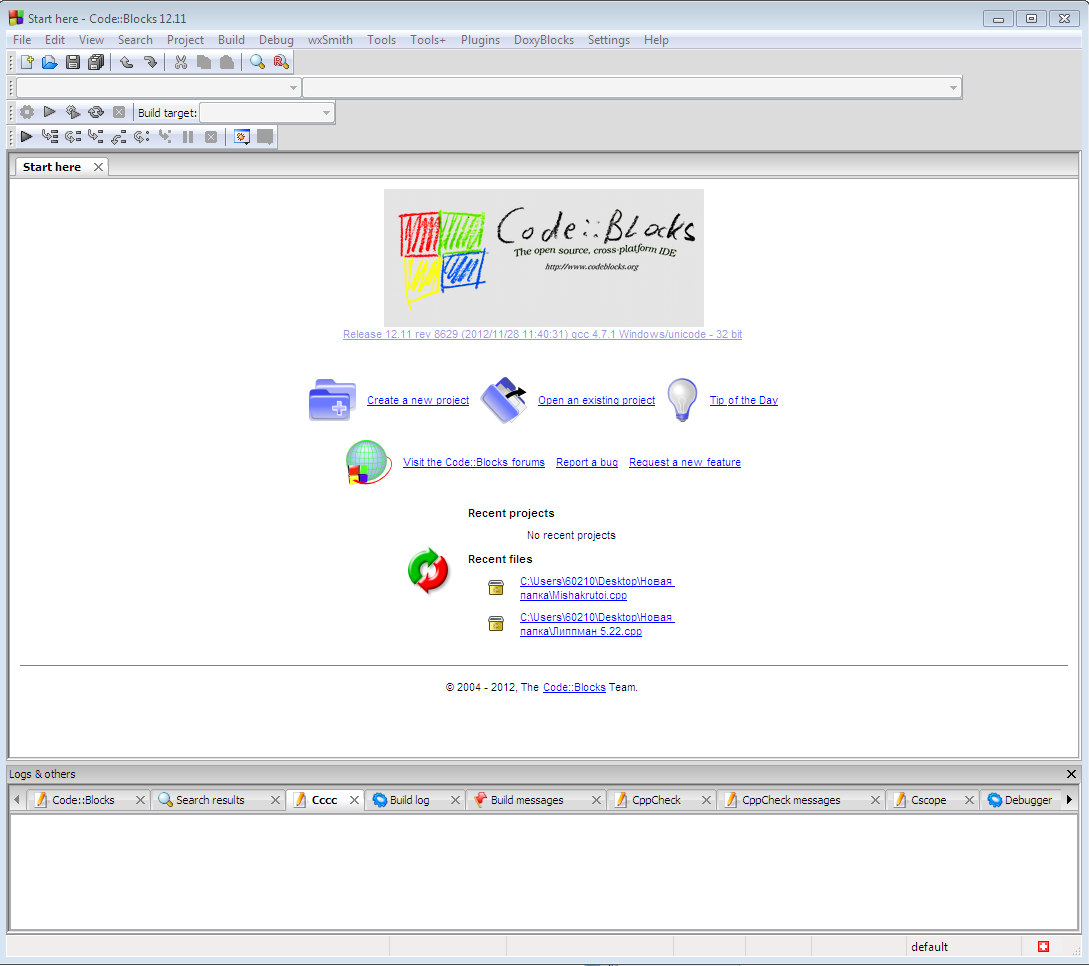


Рис. 18.1 Открытие Code:: Blocks

1. **Создание Empty File. Сохранение работы.**

Выбрать в главном меню команду File → New → Empty File

(Файл → Создать → Проект). Для компиляции необходимо сохранить работу. (Рис. 18.2, 18.3)

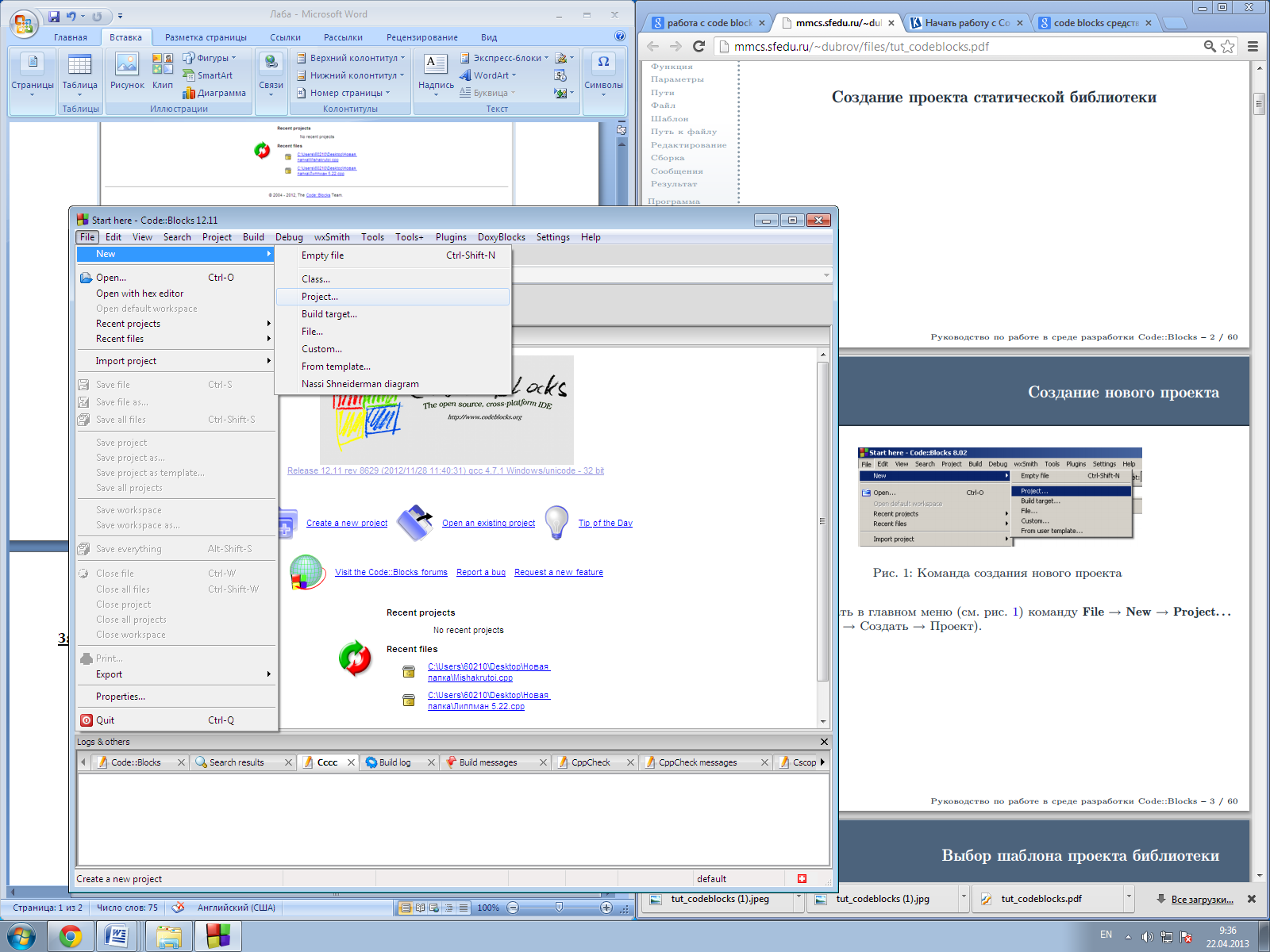


Рис. 18.2

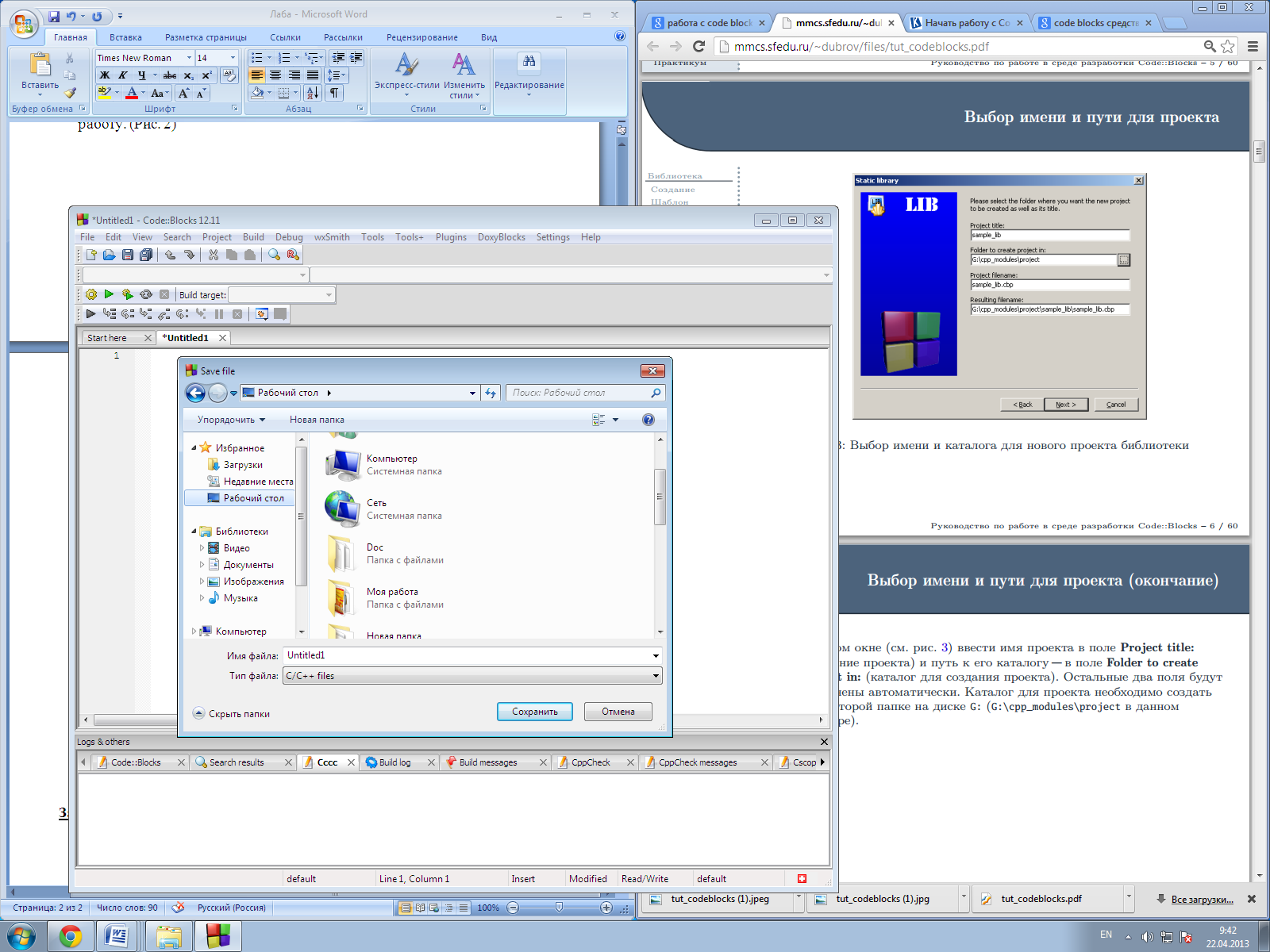


Рис. 18.3 Создание нового проекта. Сохранение работы

1. **Выбор шаблона проекта библиотеки**.

В окне выбора шаблона нового проекта выбрать шаблон Console application -> Go. Запустится мастер создания проекта. (Рис. 18.4)



Рис. 18.4 Выбор шаблона проекта библиотеки

1. **Выбор языка, имени и пути для проекта.**

На странице выбора имени и каталога проекта выбрать имя sample\_program и тот же самый каталог, в котором был создан под каталог с проектом библиотеки. На странице выбора языка проекта выбрать язык C ++ (Рис. 18.5)

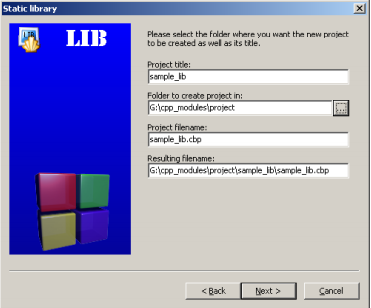


Рис. 18.5 Выбор имени и пути для проекта

1. **Выбор каталогов для генерируемых файлов.**

На странице выбора каталогов исполняемых и объектных файлов для каталогов Output dir.: (Выходной каталог) указать те же самые каталоги, что и в проекте статической библиотеки. Для каталогов Objects output dir.: (выходной каталог для объектных модулей) ввести значение ..\..\bin\Debug\sample\_program. Это позволит предотвратить возможное перезаписывание объектных файлов с одинаковыми именами двух проектов. (Рис. 18.6)

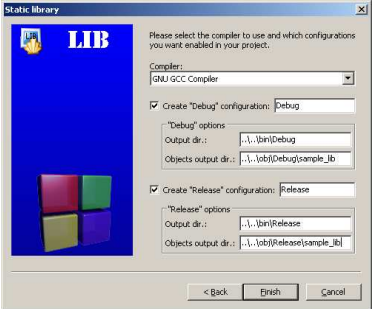


Рис. 18.6 Выбор каталогов для генерируемых файлов

1. **Редактирование исходного модуля.**

Открыть в текстовом редакторе созданный файл main.cpp (Рис. 18.7, 18.8)

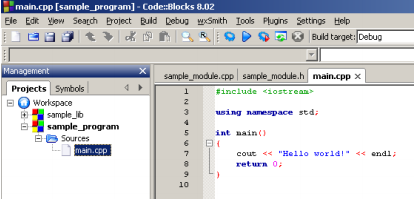


Рис. 18.7

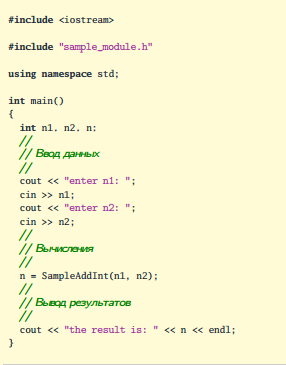


Рис. 18.8 Редактирование исходного модуля

1. **Переименование рабочего пространства.**

В окне управления проектами найти запись Workspace -> команда Rename workspace. (Рис. 18.9)

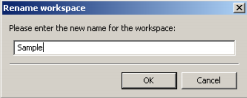
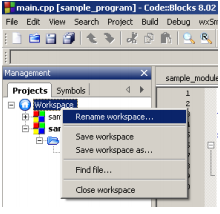
****

Рис. 18.9 Переименование рабочего пространства

1. **Сохранение рабочего пространства.**

Выбрать в главном меню команду File → Save workspace as (Рис. 18.10)

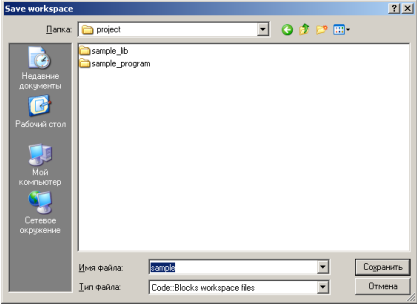
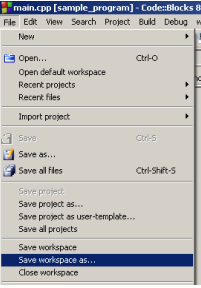


Рис. 18.10 Сохранение рабочего пространства.

1. **Файл рабочего пространства**

Файл рабочего пространства будет создан в указанном каталоге (G:\cpp\_modules\project). Таким образом, чтобы сохранить весь набор проектов в другом месте, достаточно в дальнейшем скопировать указанный каталог (Рис. 18.11)

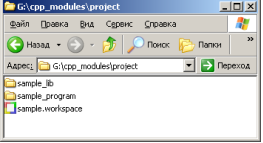
****

Рис. 18.11 Файл рабочего пространства.

1. **Вызов параметров сборки проекта.**

В окне управления проектами в контекстном меню проекта

Sample\_program выбрать команду Build options (Рис. 18.12)

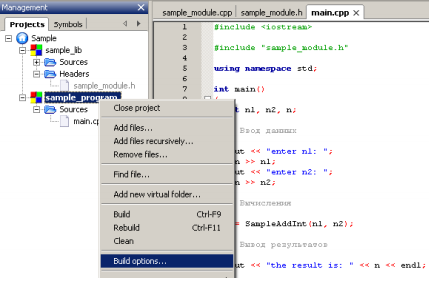


Рис 18.12 Команда вызова окна параметров сборки.

1. **Переход к общим параметрам сборки** (Рис. 18.13)

В окне параметров сборки проекта в левой части располагается дерево конфигураций проекта. Выбор конфигурации в нём приводит к отображению в правой части настроек для данной конфигурации. Для отображения настроек, общих для всех конфигураций, перейти в корневой элемент дерева, имеющий название проекта.

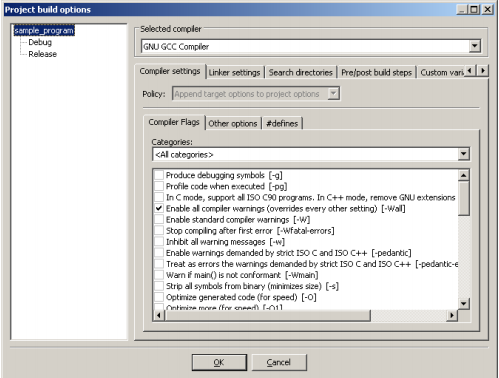
****

Рис. 18.13 Переход к параметрам сборки, общим для всех конфигураций

1. **Настройка путей для компилятора.**

В правой части окна параметров сборки проекта перейти на вкладку Search directories. Здесь на вкладке Compiler указываются дополнительные каталоги для поиска заголовочных файлов, подключаемых из программы при помощи директив препроцессора #include. Для добавления пути необходимо нажать кнопку Add. (Рис. 18.14)

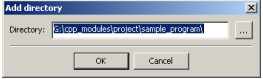
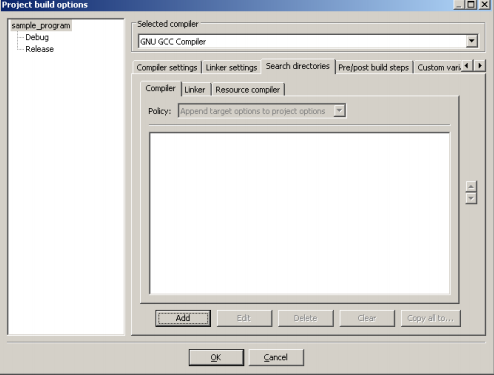
****

Рис. 18.14 Переход к настройке дополнительных путей проекта для поиска заголовочных файлов.

1. **Сборка проекта.**

Выбрать в главном меню команду Build→ Build для сборки проекта. Та же самая команда доступна на панели инструментов. (Рис. 18.15)

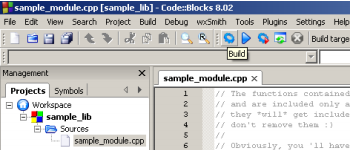


Рис. 18.15 Сборка проекта

1. **Окно сообщений о ходе сборки проекта библиотеки.** При отсутствии ошибок в окне Logs & others во вкладке Build log выведется информация о ходе сборки проекта, в конце которой будет строка:0 errors, 0 warnings. (Рис. 18.16)

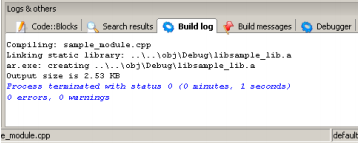
****

Рис. 18.16Окно сборки

1. **Запуск приложения.**

Выбрать в главном меню команду Build → Run. Та же самая команда доступна на панели инструментов. (Рис. 15)

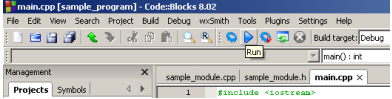
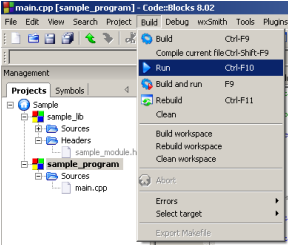


Рис. 18.17 Запуск приложения

1. **Работа запущенного приложения.**

Сообщения собранного приложения в ходе выполнения (Рис. 18.18)

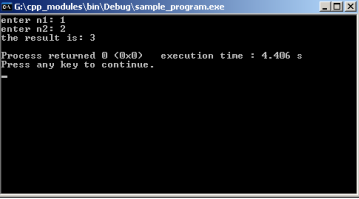


Рис. 18.18 Запуск приложения

**Контрольные вопросы**

1) Как сохранить проект?

2) Что будет, если указать неправильные настройки компилятора?

3) Что будет с программой, если не указать в настройках путь к для подключаемых библиотек?

4) Важно ли сохранять разрешение файла?

5) Где производятся настройки компилятора Code::Blocks?

## --Тема 2.2. Управляющие конструкции

### --П1.Анализ программ с ветвлением

### --Лабораторная работа 2. Разработка программ с ветвлением и выбором

/\*\* Добавить выбор \*/

**Цель работы**: Получить первоначальные навыки работы с операторами ветвления

Необходимые предварительные знания: ([2], стр.221-225)

1. Структура простейшей программы.
2. Базовые типы, переменные.

**Ход работы**:

1. Запустить CodeBlocks;

2. Создать новый .cpp-файл (File – New – File);

3. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

4. Выполнить задания.

**Пример Л2.1**

// 2.1 Дано число. Вывести его модуль.

#include <iostream>

int main()

{

int a;

std::cout << "Enter a" << std::endl;

std::cin >> a;

if (a<0)

a\*= -1;

std::cout << "module is: " << a << std::endl;

return 0;

}

**Пример Л2.2**

// 2.2 Даны два числа a и b. Вывести результат их сравнения

#include <iostream>

int main()

{

int a, b;

std::cout << "Enter a" << std::endl;

std::cin >> a;

std::cout << "Enter b" << std::endl;

std::cin >> b;

if (a>b)

std::cout << "a > b" << std::endl;

else

if (b>a)

std::cout << "b > a" << std::endl;

else

std::cout << "b = a" << std:: endl;

return 0;

}

**Пример Л2.3**

// 2.3 Дано 3 числа a, b и c. Вывести количество положительных чисел // n.

#include <iostream>

int main()

{

int a,b,c;

std::cout << "Enter a" << std::endl;

std::cin >> a;

std::cout << "Enter b" << std::endl;

std::cin >> b;

std::cout << "Enter c" << std::endl;

std::cin >> c;

int n = 0;

if (a>0)

++n;

if (b>0)

++n;

if (c>0)

++n;

std::cout << "n= " << n << std::endl;

return 0;

}

**Задания для самостоятельной работы**

1. Дано целое число. Если оно является положительным, то прибавить к нему 1; если отрицательным, то вычесть из него 2; если нулевым, то заменить его на 10. Вывести полученное число
2. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т. д.
3. Даны три числа. Найти сумму двух наибольших из них.

**Содержание отчета**:

1. Тексты заданий
2. Тексты программ

### --П2.Анализ программ с циклами for

### --П3.Анализ программ с циклами while

### --П4.Анализ программ с циклами do..while

### Лабораторная работа 3.Разработка программ с циклами

**Цель работы: Научиться создавать программы с циклами**

**Необходимые предварительные знания:**

1. Циклы
2. Ввод-вывод

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Часть 1. Введение в циклы**

**Пример 3.1**

/// М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону 2004.

/// For1. Даны целые числа K и N (N > 0). Вывести N раз число K.

/// Решение: Л.Б.Гусятинер 03.2014

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

int main()

{

int k,n;

cin >> k >> n;

for (int i=0; i<n; i++)

cout << k << ' ';

cout << endl;

return 0;

}

/// М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону

/// 2004.

/// For2. Даны два целых числа A и B (A < B).

/// Вывести в порядке возрастания все целые числа, расположенные между /// A и B (включая сами числа A и B),

/// а также количество N этих чисел.

/// Решение: Л.Б.Гусятинер 02.2014

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

int main()

{

int a, b;

cin>>a>>b;

int n=0;

for (int i=a; i<=b; ++i, ++n)

cout<<i<<' ';

cout << endl << n << endl;

return 0;

}

**Пример 3.2**

/// М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону

/// 2004.

/// For4. Дано вещественное число — цена 1 кг конфет. Вывести

/// стоимость 1,

/// 2, …, 10 кг конфет. }

/// Решение: Л.Б.Гусятинер 02.2014

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

int main()

{

double a;

std::cin>>a;

for (int i=1; i<=10; ++i)

cout<<a\*i <<' ';

cout<<endl;

return 0;

}

**Пример 3.3**

/// While1°. Даны положительные числа A и B (A > B). На отрезке длины A размещено

///максимально возможное количество отрезков длины B (без наложений).

///Не используя операции умножения и деления, найти длину незанятой части отрезка A.

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

int x=a;

while (x>=b)

x-=b;

cout << x;

return 0;

}

**Пример 3.4**

/// М. Э. Абрамян. 1000 ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ, Ростов-на-Дону

/// 2004.

/// While2. Даны положительные числа A и B (A > B). На отрезке длины A

/// размещено

/// максимально возможное количество отрезков длины B (без наложений).

/// Не используя операции умножения и деления, найти количество

/// отрезков B, размещенных на отрезке A. }

/// Решение: Л. Б. Гусятинер, 02.2014

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

int main()

{

double a, b;

cin >> a >> b;

int k=0;

while(a>=b)

{

a=a-b;

++k;

}

cout<<k<<endl;

return 0;

}

**Пример 3.5**

/// Вводятся N чисел, найти сумму. Цикл do..while

/// Решение: Л. Б. Гусятинер, 09.2015

#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

int s=0;

int i=0;

while (i<n)

{

int x;

cin >> x;

s=s+x;

i=i+1;

}

cout<<"s=" << s <<endl;

return 0;

}

**Задания для самостоятельной работы** [1]

Решить при помощи 3-х управляющих конструкций каждую из 3х задач:

For5, while4, while6

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

**Часть 2. Вложенные циклы**

**Пример 3.6**

#include <iostream>

#include <vector>

#include <string>

using std::cout;

using std::cin;

using std::vector;

using std::string;

int main()

{

vector<string> v;

string s;

bool b=0;

cin>>s;

v.push\_back(s);

while(cin>>s && !s.empty())

{

int i=0;

while(i!=v.size())

{

if(v[i]==s)

{

b=1;

break;

}

++i;

}

if(b==1)

break;

else

v.push\_back(s);

}

if(b==1)

cout<<s;

else

cout<<"Povtor9I-0IIIuxs9 slov net";

return 0;

}

**Пример 3.7**

#include <iostream>

#include <vector>

#include <string>

using std::cout;

using std::cin;

using std::vector;

using std::string;

int main()

{

vector<string> v;

string s;

bool b=0;

cin>>s;

v.push\_back(s);

beg:

{

cin>>s;

int i=0;

while(i!=v.size())

{

if(v[i]==s)

{

b=1;

break;

}

++i;

}

if(b==0)

{

v.push\_back(s);

goto beg;

}

}

if(b==1)

cout<<s;

else

cout<<"Povtor9I-0IIIuxs9 slov net";

return 0;

}

**Пример 3.8**

#include <iostream>

#include <string>

using std::cout;

using std::cin;

using std::string;

int main()

{

string s1;

string s;

bool b=0;

while(cin>>s && !s.empty())

{

if(s[0]!='!')

continue;

s[0]=' ';

s1+=s;

}

for(size\_t i=0;i!=s1.size();++i)

s1[i]=s1[i+1];

cout<<s1;

return 0;

}

**Задания для самостоятельной работы:**

1. Записать текст без повторяющихся слов.

2. Все символы строки записать в нижней регистр.

3. Из вектора строк найти строку, которая будет начинаться с нижнего регистра.

## 4. Все строки вектора должны быть записаны нижним регистром.

## --Тема 2.3 Встроенные и библиотечные типы данных

### --Л4.Работа с целыми и вещественными типами

### Лабораторная работа 5.Разработка программ со строками

**Цель работы**: приобрести практические навыки работы со строками в C++.

**Необходимые первоначальные знания**: структуры простейших программ, переменные, ввод, вывод, тип string ([2], стр.104-114)

**Ход работы**:

1. Запустить Code::Blocks;
2. Создать новый .cpp-файл (File – New – File);
3. Рассмотреть примеры программ из приложенных ниже работ, скопировать, запустить;
4. Выполнить задания.

**Пример Л5.1**

//5.1 Дана строка. Подсчитать количество содержащихся в ней цифр.

#include <iostream>

#include <string>

int main()

{

int digitCount=0;

std::string s="";

getline(std::cin,s);

for(int i=0; i!=s.size(); ++i)

if(isdigit(s[i]))

++digitCount;

std::cout << "Number of digits is " << digitCount << std::endl;

return 0;

}

**Пример Л5.2**

//5.2. Дана строка. Преобразовать в ней все латинские строчные буквы в // прописные.

#include <iostream>

#include <ctype.h>

#include <string>

int main() {

std::string s;

getline(std::cin,s);

for (int i=0; i!=s.size(); ++i)

s[i]=toupper(s[i]);

std::cout << s;

return 0;

}

**Пример Л5.3**

//5.3 Подсчитать сумму цифр в строке.

#include <iostream>

#include <string>

int main()

{

std::string s;

int a,sum=0;

getline(std::cin, s);

for (int i=0; i!=s.size(); ++i)

{

a=s[i]-'0';

sum+=a;

}

std::cout << "Sum of digits is " << sum << std::endl;

return 0;

}

**Задания для самостоятельной работы**:

1. Преобразовать в ней все латинские прописные буквы в строчные.
2. Подсчитать количество пробелов в строке.
3. Дана строка, содержащая полное имя файла, т. е. имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки имя файла (без расширения).

### Лабораторная работа 6.Разработка программ со строками в стиле C

**Цель работы**: приобрести практические навыки работы со строками в с стиле С.

**Необходимые первоначальные знания**: структуры простейших программ, переменные, ввод, вывод, строки в стиле С ([2], стр.154-165).

**Функции для символьных строк в стиле С:**

strlen(s) – возвращает длину строки s без учёта нулевого символа;

strcmp(s1,s2) – проверяет равенство строк s1 и s2. Возвращает 0, если s1==s2, положительное значение, если s1>s2, и отрицательное значение, если s1<s2.

strcat(s1,s2) – добавляет строку s2 к строке s1. Результат возвращает в строку s1.

strcpy(s1,s2) – копирует строку s2 в строку s1. Результат возвращает в строку s1.

strncat(s1,s2,n) – добавляет n символов из строки s2 в строку s1. Результат возвращает в строку s1.

strncpy(s1,s2,n) – копирует n символов из строки s2 в строку s1. Результат возвращает в строку s1.

**Ход работы:**

1. Запустить Code::Blocks;
2. Создать новый .cpp-файл (File – New – File);
3. Рассмотреть примеры программ из приложенных ниже работ, скопировать, запустить;
4. Выполнить задания.

**Пример Л6.1**

//6.1 Вывод строки в стиле С.

#include <iostream>

int main()

{

char c[] = "Hello, world!";

char \*cp=c;

while(\*cp) //цикл осуществляется до тех пор, пока не встретится нулевой символ

{

std::cout << \*cp;

++cp;

}

return 0;

}

**Пример Л6.2**

//6.2 Сложение двух строк и вывод результата.

#include <iostream>

#include <cstring>

int main()

{

char c1[]="Hello,";

char c2[]=" world!";

strcat(c1,c2);

while(c1)

{

std::cout << c1;

break;

}

return 0;

}

**Пример Л6.3**

//6.3. Заменяет в строке в стиле С все прописные буквы на строчные.

#include <iostream>

#include <cstring>

int main()

{

char c[]="hello, world!";

char \*cp=c;

while(\*cp)

{

\*cp=toupper(\*cp); // функция toupper(\*элемент\*) заменяет букву на

// строчную.

std::cout << \*cp;

++cp;

}

return 0;

}

**Задания для самостоятельной работы:**

1. Подсчитать колчисество цифр в строке;
2. Заменить все вхождения в строку символа A символом B;
3. Вычислить сумму всех цифр в строке;
4. Вычислить длину строки в стиле Си.

### --Л7.Разработка программ с файлами

### --П5.Построение трассировочных таблиц с одномерными массивами (векторами)

### Лабораторная работа 8.Работа с векторами

**Цель работы:** Научиться работать с векторами в языке C++.

**Необходимые предварительные знания:** структуры простейших программ, переменные, ввод, вывод, тип string, int, double ([2], стр.114-120)

**Ход работы:**

1. Запустить Code::blocks;
2. Рассмотреть примеры программ из настоящей работы, скопировать, запустить
3. Выполнить задания для самостоятельной работы

**Часть 1. Введение в работу с векорами.**

**Пример Л8.1**

//Ввод и вывод данных в вектор

#include <iostream>

#include <vector>

using std::cin;

using std::cout;

using std::endl;

using std::vector;

int main() {

vector<int> v;

int c;

for(size\_t i=0;i<5;++i) {

cin >>c;

v.push\_back(c);

}

for(size\_t i=0;i<v.size();++i)

cout<<endl<<v[i];

return 0;

}

**Пример Л8.2**

//сортировка вектора

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using std::cin;

using std::cout;

using std::endl;

using std::vector;

using std::string;int main()

{ vector<string> v;

string s;

setlocale( LC\_ALL,"Russian" );

for(size\_t i=0;i<5;++i)

{

cin>>s;

v.push\_back(s);

}

cout<<"Cортировка: \n";

for(size\_t i=0;i<v.size();++i)

{

sort(v.begin(),v.end());

cout<<v[i]<<endl;

}

return 0;

}

**Пример Л8.3**

//сумма четных чисел в векторе

#include <iostream>

#include <vector>

using std::cin;

using std::cout;

using std::endl;

using std::vector;

int main()

{ vector<int> v;

int c;

for(size\_t i=0;i<5;++i) {

cin >>c;

v.push\_back(c);

}

int sum=0;

for(size\_t i=0;i<v.size();++i){

if(v[i]%2==0)

sum+=v[i];

}

cout<<sum;

return 0;

}

**Задания для самостоятельной работы:**

1. Укажите три способа создания вектора и добавления в него 10 элементов, каждый из которых содержит значение 42.
2. Прочитайте некоторый текст, сохраняя каждое введенное слово как отдельный элемент вектора. Преобразуйте символы каждого слова в прописные. Отобразите преобразованный текст, выводя по 8 слов в строке
3. Прочитайте в вектор набор целых чисел. Вычислите и отобразите сумму каждой пары смежных элементов в векторе.

**Часть 2. Работа с итераторами**

**Пример Л8.4**

/\* Дан вектор. С помощью итератора обнулите все элементы вектора.\*/

#include <iostream>

#include <vector>

int main()

{

std::vector<int> ivec(10,333);

for (std::vector<int>::iterator iter=ivec.begin(); iter!=ivec.end();++iter)

\*iter=0;

for (std::vector<int>::iterator iter=ivec.begin(); iter!=ivec.end();++iter)

std::cout<<"Element vektora obnulen "<<\*iter<<std::endl;

return 0;

}

**Пример Л8.5**

/\*

Напишите программу, где создаётся вектор из 10 элементов.

При помощи итератора присвойте каждому элементу значение,

которое вдвое больше его текущего значения.

\*/

#include <iostream>

#include <vector>

int main ()

{

std::vector<int> ivec(10,3);

for (std::vector<int>::iterator iter=ivec.begin(); iter!=ivec.end(); ++iter)

\*iter\*=2;

for (std::vector<int>::iterator iter=ivec.begin(); iter!=ivec.end(); ++iter)

std::cout<< \*iter <<std::endl;

return 0;

}

**Задания для самостоятельной работы:**

1.-2. При помощи итераторов выполните задания первой части лабораторной работы

## --Тема 2.4 Массивы и указатели

### --П6.Построение трассировочных таблиц с многомерными массивами

### Лабораторная работа 9.Поиск в массиве

**Цель работы:** научиться осуществлять поиск в массиве в С++

**Необходимые предварительные знания:**

* Структуры простейших программ
* Переменные, типы данных, умение работы с циклами
* Ввод cin, вывод cout
* Массивы в С++
* Использованная литература: [2] стр.134-138

**Начало работы:**

- Запустить Code::Blocks

- Создать новый файл и сохранить его как .cpp файл

- Рассмотреть пример

**Ход работы:**

Рассмотреть примеры

Выполнить задания

**Пример Л9.1**

// Есть ли в массиве заданное число x?

// Гусятинер Л.Б., 15.11.2015

#include <iostream>

using namespace std;

int main()

{

const int MAXN=1000;

int n;

cin >> n;

int v[MAXN];

for (int i=0; i<n; i++)

cin >> v[i];

int x;

cin >> x;

int i=0;

while (i<n && v[i] !=x)

++i;

if (i<n)

cout << "YES";

else

cout << "NO";

cout << endl;

return 0;

}

11-2.cpp

**Пример Л9.2** [**http://purecodecpp.com/archives/1977**](http://purecodecpp.com/archives/1977)

Существует ли число key в массиве arr

#include <iostream>

using namespace std;

// функция с алгоритмом двоичного поиска

int Search\_Binary (int arr[], int left, int right, int key)

{

int midd = 0;

while (1)

{

midd = (left + right) / 2;

if (key < arr[midd]) // если искомое меньше значения в ячейке

right = midd - 1; // смещаем правую границу поиска

else if (key > arr[midd]) // если искомое больше значения в ячейке

left = midd + 1; // смещаем левую границу поиска

else // иначе (значения равны)

return midd; // функция возвращает индекс ячейки

if (left > right) // если границы сомкнулись

return -1;

}

}

int main()

{

setlocale (LC\_ALL, "rus");

const int SIZE = 12;

int array[SIZE] = {};

int key = 0;

int index = 0; // индекс ячейки с искомым значением

for (int i = 0; i < SIZE; i++) // заполняем и показываем массив

{

array[i] = i + 1;

cout << array[i] << " | ";

}

cout << "\n\nВведите любое число: ";

cin >> key;

index = Search\_Binary (array, 0, SIZE, key);

if (index >= 0)

cout << "Указанное число находится в ячейке с индексом: " << index << "\n\n";

else

cout << "В массиве нет такого числа!\n\n";

return 0;

}

**Задания:**

Дан массив целых чисел A и значение X.

1. Напечатать номер первого вхождения числа X или -1, если вхождения нет.
2. Напечатать номер последнего вхождения числа X или -1, если вхождения нет.
3. Напечатать количество вхождений числа X.

Дан упорядоченный по возрастанию массив целых чисел A и значение X.

1. Сколько раз входит X в A?

Дан упорядоченный по возрастанию массив целых чисел A и неупорядоченный массив B.

1. Сколько элементов B не входят в массив A?

### Лабораторная работа 10.Сортировка массива

**Цель работы:** научиться выполнять сортировку массива в С++

**Необходимые предварительные знания:**

1. Структуры простейших программ
2. Переменные, типы данных, циклы
3. Ввод cin, вывод cout
4. Массивы в С++

**Ход работы:**

1. Запустить Code::Blocks
2. Создать новый файл и сохранить его как .cpp файл
3. Рассмотреть пример
4. Выполнить задания

**Пример Л10.1**

// Сортировка "пузырьком"

// Гусятинер Л.Б., 15.11.2015 г.

#include <iostream>

using namespace std;

int main()

{

const int MAXN=1000;

int n;

cin >> n;

int v[MAXN];

for (int i=0; i<n; i++)

cin >> v[i];

bool swapped=true;

int i=-1;

while (swapped)

{

i++;

swapped = false;

for (int j=n-1; j>i; j--)

if (v[j] < v[j - 1])

{

int x=v[j];

v[j]=v[j-1];

v[j-1]=x;

swapped = true;

}

}

for (int i=0; i<n; i++)

cout << v[i] << ' ';

cout << endl;

return 0;

}

**Задания для самостоятельной работы**

Рассмотреть программы сортировки на языке Pascal данной методички. Переписать на C++ без использования функций

1. Сортировка обменом
2. Сортировка вставками

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### --Л11.Обработка таблиц

### --Л12.Контрольная работа

## --Тема 2.5 Функции

### --П7."Ручное" тестирование функций

### Лабораторная работа 13.Работа с функциями

**Цель работы:** приобрести основные практические навыки работы с функциями.

**Необходимые первоначальные знания:** определение функций, передача аргумента, оператор return, объявление функций.

**Ход работы:**

1. Запустить Code::Blocks;
2. Создать новый .cpp-файл (File – New – File);
3. Рассмотреть примеры программ из приложенных ниже работ, скопировать, запустить;
4. Выполнить задания.

**Пример Л13.1**

//13.cpp Функция вычисляющая сумму двух целых чисел.

#include<iostream>

int Summ(int n1, int n2);

int main(int argv, char\*\* argc)

{

std::cout << Summ(3, 4) << '\n';

return 0;

}

int Summ(int n1, int n2)

{

return n1 + n2;

}

**Пример Л13.2**

// 13.2.cpp Функция вычисляющая значение y из линейной функции y = x + 5 с помощью

// передачи аргумента y;

#include<iostream>

int Func(int x);

int main(int argv, char\*\* argc)

{

std::cout << Func(5) << '\n';

return 0;

}

int Func(int x)

{

int y = x + 5;

return y;

}

**Пример Л13.3**

//13.2.cpp Встраиваемая функция вычисляющая сумму двух целых чисел.

#include<iostream>

inline int Summ(int n1, int n2);

int main(int argv, char\*\* argc)

{

std::cout << Summ(3, 4) << '\n';

return 0;

}

int Summ(int n1, int n2)

{

return n1 + n2;

}

// 26\_2.cpp Функция вычисляющая сумму двух целых чисел и сохраняет сумму в

// переменную по ссылке out.

#include<iostream>

void Summ(int n1, int n2, int &out);

int main(int argv, char\*\* argc)

{

int out;

Summ(3, 4, out);

std::cout << out << '\n';

return 0;

}

void Summ(int n1, int n2, int &out)

{

out = n1 + n2;

}

**Задания для самостоятельной работы:**

1. Написать функцию возвращающую решение выражения y = x / 2 – 7.

Тип возвращаемого значения: int,

Аргументы: int x.

1. Написать функцию возвращающую кол-во символов в строке Си.

Тип возвращаемого значения: int,

Аргументы: int x.

### Лабораторная работа 14.Работа с указателями

1. **Цель работы:** получить навыки по работе с указателями
2. **Необходимые предварительные знания:** устройство памяти, указатели
3. **Ход работы:** запустить среду, рассмотреть примеры, выполнить задания

**Пример Л14.1**

//Изменение значения переменной при помощи указателя

#include <iostream>

int main()

{

int a1=5, a2=7; //создаем две переменные

int \*pa=&a1; //присваиваем указателю pa адрес переменной

\*pa=a2; //изменяем значение переменной, на которую

//указывает указатель

std::cout << \*pa; //вывод: 7

return 0;

}

**Пример Л14.2**

//Изменение элементов массива при помощи указателя

#include <iostream>

int main()

{

const int a\_size = 5;

int m[a\_size] = {1,2,3,4,5};

//в цикле создаем указатель на тип int

//и присваиваем адрес первого элемента массива m

//также создаем указатель на конец массива

for (int \*pbegin = m, \*pend = m+a\_size; pbegin < pend; ++pbegin)

{

\*pbegin = 0;

std::cout << \*pbegin;

}

//вывод: 00000

return 0;

}

**Пример Л14.3**

//выражения с указателями

#include <iostream>

int main()

{

//создаем 2 переменных и 2 указателя

int a=3, b=0, \*c=&b, \*d;

//указателю d присваиваем адрес указателя c

d=c;

int m[]={1,2,3,4,5,6,7,8,9,10};

//в цикле:

//указателю d присваиваем адрес переменной a

//условие \*c<\*d (эквивалент b<a)

// (\*d)++ эквивалент a=a+1

// \*с+=2 эквивалент b=b+2

for (d=&a; \*c<\*d; (\*d)++, \*c+=2)

std::cout << "a=" << a << " b=" << b << endl;

//вывод:

//a=3 b=0

//a=4 b=2

//a=5 b=4

return 0;

}

**Задания для самостоятельной работы**

1. Что такое указатель, для чего он используется
2. Есть ли среди приведенных определений недопустимые?
3. int \*ip;
4. string \*s=0;
5. int i; double\* dp = I;
6. int \*ip,\* ip2;
7. string \*ps=Null;
8. Что выполняет следующий код:

int i = 42, j = 1024;

int \*p1 = &i, \*p2 = &j

\*p2 = \*p1 \* \*p2;

\*p1 \*= \*p1;

1. Напишите код, суммирующий элементы массива, при помощи указателя.

### --Л15.Создание и удаление динамических объектов

### --Л16.Передача массивов в функции

## --Тема 2.6 Структуры

### --П8.Проектирование структур данных

### --Л17.Работа со структурами

### --Л18.Работа с массивами структур

### --Л19.Передача структур в функции

## --Тема 2.7. Проекты

### --Л20.Разработка каркаса проекта

### --Л21.Разработка функций и сборка проекта

### --Л22.Тестирование и отладка проекта

### --Л23.Разработка пользовательской документации

### --Л24.Зачетная работа

### --Лабораторная работа 19. Разработка простейших программ

1. **Цель работы**: Научиться разрабатывать простейшие программы в среде Code:;Blocks.
2. **Необходимые первоначальные знания**: Ориентировочные познания в С++
3. **Начало работы:** Запустить программу Code::Blocks, разработать программу HelloWorld и другие.

**Задание 1 Разработка простейших программ.**

#include <iostream>

using std::cout;

using std::endl;

int main()

{

cout << "Hello, World!" << endl;

return 0;

}

1. **Разбор кода С++ на примере программы Hello World!**

**#include <iostream> -** поддержка потоков **(streams)** ввода/вывода

**using std::cout; -**использовать имена из стандартной библиотеки **C++**

**using std::endl; -** использовать имена из стандартной библиотеки **C++**

**int main() -** любая программа на **C++** должна включать

функцию с этим именем; ключевое слово **int** означает, что

программа возвратит операционной системе целое число

**{ -** тело любой функции заключается в фигурные скобки

**cout << "Hello, World!" << endl; -** то, что нужно вывести, разделяется символами **«<<»** Вывод **endl** приводит к переводу курсора на новую строку

**return 0; -** точка выхода из программы; операционной системе передается значение 0 (признак того, что программа завершилась успешно)

**} -** тело любой функции заключается в фигурные скобки

!!! std можно записать в нескольких вариантах – std::… и using std::

**Задания на самостоятельную работу**

* 1. Измените программу так, чтобы она напечатала еще одну строчку: «I love C++»
  2. Сколько раз пришлось воспользоваться объектом cout?
  3. Можно ли обойтись лишь одним обращением к объекту cout?

1. **Необходимо вывести на экран решение уравнения a + x = b, значения a и b пользователь должен ввести с клавиатуры**

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

int main()

{

double a, b; - объявление переменных a и b, представляющих из себя

вещественные числа двойной точности

cout << "Input a: ";

cin >> a; - запись числа, введенного пользователем, в переменную a

cout << "Input b: ";

cin >> b; - запись числа, введенного пользователем, в переменную b

double x = b - a; - объявление переменной x с одновременным присвоением ей значения

cout << "Root of equation a+x=b is " << x << endl; - Вывод

return 0;

}

**Задания на самостоятельную работу**

* 1. Вывести на экран сумму a + b чисел a и b; значения a и b

пользователь должен ввести с клавиатуры

* 1. Необходимо ли здесь вводить дополнительную переменную, в

которой будет сохраняться значение суммы?

1. **Даны две целые переменные a и b. Составить программу, после работы которой значения переменных поменялись бы местами (т.е. новое значение переменной a равно старому значению переменной b и наоборот)**

#include <iostream>

using std::cout;

using std::cin;

using std::endl;

int main()

{

int a,b,c;

cin >> a >> b;

c=a;

a=b;

b=c;

cout << a << b << endl;

return 0;

}

**Задания на самостоятельную работу**

* 1. Можно ли записать область видимости другим способом?
  2. Как перевести вывод на следующую строку?

### --Лабораторная работа 20. Использование классов

1. **Цель работы**: Научиться разрабатывать простейшие программы в среде Code:;Blocks.
2. **Необходимые первоначальные знания**: Введение в классы
3. **Начало работы:** Запустить программу Code::Blocks, научиться использовать классы в программах.

**Определение класса**

class soldier

{

public:

int x,y;

int ammo;

};

## -----Тема 2.2. Переменные и базовые типы

### -----Лабораторная работа 21. Работа с переменными

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### --Лабораторная работа 22. Работа со ссылками

**Цель работы: Приобрести умения работы со ссылками**

**Необходимые предварительные знания:**

1. Переменные.
2. Ввод, вывод.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

Ссылочный тип, иногда называемый псевдонимом, служит для задания объекту дополнительного имени. Ссылка позволяет косвенно манипулировать объектом, точно так же, как это делается с помощью указателя. Однако эта косвенная манипуляция не требует специального синтаксиса, необходимого для указателей. Обычно ссылки употребляются как формальные параметры функций. В этом разделе мы рассмотрим самостоятельное использование объектов ссылочного типа.

Ссылочный тип обозначается указанием оператора взятия адреса (&) перед именем переменной. Ссылка должна быть инициализирована значением объекта. Например:

**Пример Л22.1**

int ival = 1024;

// правильно: refVal - ссылка на ival

int &refVal = ival;

// ошибка: ссылка должна быть инициализирована

int &refVal2;

int ival = 1024;

Определив ссылку, уже нельзя изменить ее так, чтобы работать с другим объектом.

**Пример Л22.2**

// оператор присваивания не меняет значения refVal, новое значение присваивается

// переменной ival – ту, которую адресует refVal.

int min\_val = 0;

// ival получает значение min\_val,

// а не refVal меняет значение на min\_val

refVal = min\_val;

Все операции со ссылками реально воздействуют на адресуемые ими объекты.

**Пример Л22.3**

refVal += 2;

прибавляет 2 к ival – переменной, на которую ссылается refVal. Аналогично

int ii = refVal;

присваивает ii текущее значение ival,

Если мы определяем ссылки в одной инструкции через запятую, перед каждым объектом типа ссылки должен стоять амперсанд (&) – оператор взятия адреса. Например:

**Пример Л22.4**

// определено два объекта типа int

int ival = 1024, ival2 = 2048;

// определена одна ссылка и один объект

int &rval = ival, rval2 = ival2;

// определены две ссылки

int &rval3 = ival3, &rval4 = ival2;

Константная ссылка может быть инициализирована объектом другого типа (если, конечно, существует возможность преобразования одного типа в другой), а также безадресной величиной – такой, как литеральная константа. Например:

**Пример Л22.5**

// верно только для константных ссылок

const int &ir = 1024;

**Задания для самостоятельной работы**

1. Разработать по одной программе из каждого раздела задачника М.Э.Абрамяна с применением ссылок.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### -----Лабораторная работа 23. Создание типов классов

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### -----Лабораторная работа 24. Создание многофайловых проектов

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### --Лабораторная работа 31. Работа с многомерными массивами

**Цель работы**: Научиться работать с многомерными массивами в С++.

**Необходимые предварительные знания:**

1. Ветвление. if..else.
2. Одномерные массивы
3. Вложенные циклы

**Ход работы:**

1. Запустить CodeBlocks.

2. Перейти в режим: New ->File ->Name.

3. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.

4. Выполнить задания.

#include<iostream>

using namespace std;

int main()

{

int k,n=5,m=5;

int mas [m][n];

for (size\_t i=0;i!=m;i++)//формирование массива

for (size\_t j=0;j!=n;j++)

{

k = 5 \* j;//ltqcndbt

mas[i][j] = k;

}

for (size\_t i=0;i!=m;i++)//вывод массива

{

for (size\_t j=0;j!=n;j++)

cout<<mas[i][j];

cout<<endl;

}

return 0;

}

#include<iostream>

using namespace std;

int main()

{

int n=5,m=5;

int mas1 [m][n];

int mas2 [m];

for (size\_t i=0;i!=m;i++)//ввод данных в однамерый массив

cin>>(mas2[i]);

for (size\_t i=0;i!=m;i++)//формирование массива

for (size\_t j=0;j!=m;j++)

mas1[i][j] = mas2[i];

for (size\_t i=0;i!=m;i++)//вывод массива

{

for (size\_t j=0;j!=m;j++)

cout<<mas1[i][j]<<" ";

cout<<endl;

}

return 0;

}

#include<iostream>

using namespace std;

int main()

{

int k,n=5,m=5;

int mas [m][n];

for (size\_t i=0;i!=m;i++)//формирование массива

{

k = 10\* i;

for (size\_t j=0;j!=n;j++)

mas[i][j] = k;

}

for (size\_t i=0;i!=m;i++)//вывод массива

{

for (size\_t j=0;j!=n;j++)

cout<<mas[i][j];

cout<<endl;

}

return 0;

}

**Задания для самостоятельной работы:**

1. Найти число с помощью координат.

2. Найти сумму переменных во всех столбцах.

3. Найти сумму переменных во всех строках.

4. Найти сумму всех строк и столбцов.

### -----Лабораторная работа 41. Работа с препроцессором

**Цель работы: Научиться создавать несложные программы с циклами Repeat-Until**

**Необходимые предварительные знания:**

1. Структура простейшей программы. program, begin, end.

2. Переменные. Имена. Объявление var. Тип integer

3. Ввод read, readln, и вывод write, writeln.

4. Цикл. REPEAT-UNTIL

5. Комментарии.

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

{задачник Абрамяна задача FOR 2}

**Задания для самостоятельной работы**

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ

### --Лабораторная работа 49. Программирование ввода-вывода в файл

**Цель работы: Приобрести умения работы с текстовыми файлами в c++**

**Необходимые предварительные знания:**

1. Потоки
2. Строки
3. Циклы

**Ход работы:**

1. Рассмотреть примеры программ из настоящей работы, скопировать, запустить.
2. Выполнить задания.

**Потоковый ввод/вывод дисковых файлов**

Для работы с дисковыми файлами необходимо подключение заголовочного файла <fstream>, содержащего наборы специальных классов:

* ifstream — для ввода,
* ofstream — для вывода,
* fstream — для чтения и записи данных в один и тот же файл.

Чтобы получить возможность работать с дисковым файлом, нужно открыть его с указанием режима доступа, который определяется значением константы open-mode класса ios.

|  |  |
| --- | --- |
| Режим доступа | Действие |
| app | открывает файл для дозаписи |
| ate(atend) | при открытии файла устанавливает файловый указатель на конец файла |
| binary(bin) | открыть файл в двоичном представлении |
| in | открыть файл для чтения (ввода) |

**Текстовые файлы**

Для создания текстового файла определяют объект класса ofstream и передают конструктору класса имя дискового файла в качестве первого параметра и режим доступа в качестве второго параметра:

ofstream out\_file(“Out.txt”, ios::out);

Можно объявить константу, определяющую режим открытия файла, например:

const ios::open\_mode=ios::out | ios::app;

После того, как предпринималась попытка открыть файл, следует убедиться в том, что файл открыт и готов для записи (или перезаписи):

if (!out\_file)

{

cerr<<”Error: unable to write to out.txt”<<endl;

exit(1);

}

Все сказанное верно и для файлов, открываемых для чтения (или входных файлов):

ifstream in\_file(“Input.txt”, ios::in);

if (! in\_file)

{

cerr<<”Error: unable to open Input.txt”<<endl;

exit(1);

}

**Пример 17.1.** Посимвольное чтение файла и вывод его на экран.

# include <fstream>

# include <iostream>

using namespace std;

int main()

{ char sym;

ifstream in\_file (“Input\_file.txt”, ios::in);

if (! in\_file) { cerr<<”Error input file”<<endl;

exit(1);

}

while (in\_file)

{ in\_file.get(sym); // функция get() –

//для посимвольного чтения

//текстового файла.

cout<<sym;

}

cout<<endl;

return 0;

}

**Пример 17.2.** Посимвольная запись текста.

# include <fstream>

# include <iostream>

# include <string>

using namespace std;

int main(){

string guote = “В синей вышине сомкнулись кроны в желтое кольцо.”;

ofstream out\_file (“Out\_file.txt”, ios::out);

if (! out\_file)

{ cerr<<”Error output file”<<endl;

exit(1);

}

for (int i=0; i<guote.size(); i++)

out\_file.put(guote[i]); ); // функция put() –

//для посимвольной записи

//текстового файла.

cout<<”Конец записи”<<endl;

}

**Пример 17.3.**Построчное чтение файла.

# include <fstream>

# include <iostream>

using namespace std;

int main()

{ const int LEN=80;

char BUF[LEN];

ifstream in\_file (“Input\_file.txt”, ios::in);

if (! in\_file) { cerr<<”Error input file”<<endl;

exit(1);

}

while (in\_file)

{ in\_file.getline(BUF,LEN); // функция getline() - чтения строки из файла

cout<<BUF<<endl;

}

}

**Пример 17.4.**Построчная запись текста.

# include <fstream>

# include <iostream>

# include <string>

using namespace std;

int main()

{ ofstream out\_file (“Out\_file.txt”, ios::out);

if (! out\_file) { cerr<<”Error output file”<<endl;

exit(1);

}

out\_file<<”На далекой Амазонке\n”;

out\_file<<”Не бывл я никогда.\n”;

out\_file<<”Никогда туда не ходят\n”;

out\_file<<”Быстроходные суда\n”;

}

**Контрольные вопросы**

1. Что такое поток?
2. Что представляет собой файловый указатель?
3. Перечислить режимы доступа к файлу.
4. Как открыть и как закрыть файл?

**Задания для самостоятельной работы**

1. Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g*, компоненты файла *f*, исключив повторные вхождения чисел.
2. Дан символьный файл *f*. Найти и записать в файл *g* самое длинное слово файла *f*, снабдив его комментарием.

**Содержание отчета:**

1. Тексты заданий

2. Тексты программ