УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретические сведения к лабораторной работе, выполните пример и задание согласно вашему варианту.

Результаты работы, отчет, предоставить преподавателю на e-mail:

e-mail: r.bigangel@gmail.com до 13.02.2023.

Требования к отчету:

Отчет предоставляется преподавателю в электронном варианте и должен содержать:

- **название работы, постановку цели, вывод;**
- **ответы на контрольные вопросы, указанные преподавателем.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

***ВНИМАНИЕ!!!* При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (порасписанию).**

!!!Так же обратите внимание что лабораторная работа рассчитана на три пары (6 часов), в связи с тем, что она включает 3 темы, поэтому, все последующие задания будут выполняться в соответствии с данной методической рекомендацией!!!

ЛАБОРАТОРНАЯ РАБОТА №№ 3-5
Переменные и базовые типы данных языка С++. Создание программы линейного алгоритма

Цель: научится создавать блок-схемы и программы линейного алгоритма, записывать математические выражения на языке С++.

2.1 Теоретическая часть

2.1.1 Понятие алгоритма. Блок-схема

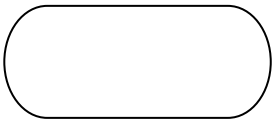
Алгоритм – конечная последовательность предписаний, однозначно определяющая процесс преобразования исходных данных в результат решения задачи.

В процессе разработки алгоритма могут использоваться различные способы его описания. Наиболее распространенные:

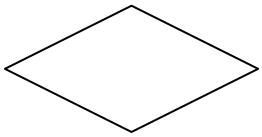
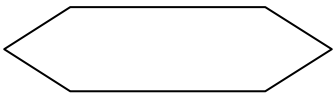
- словесная запись;
- графические схемы алгоритмов (блок-схемы);
- псевдокод (формальные алгоритмические языки);
- структурограммы.

Блок-схема – это графическое представление алгоритма, дополненное элементами словесной записи. На блок-схеме каждый пункт алгоритма изображается соответствующей геометрической фигурой. В таблице 2.1. приведены графические элементы, на которых компонуются блок-схемы, их названия и символы.

Таблица 2.1 – Графические элементы блок-схем

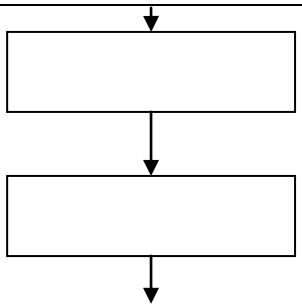
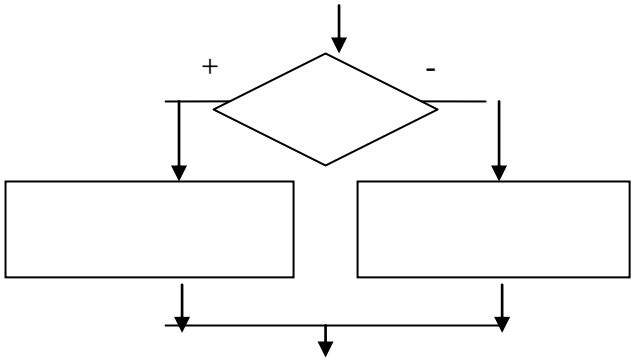
Название блока	**Блок**	**Отображаемая функция**
Начало-конец		Начало, конец, вход-выход в программах
Блок ввода-вывода		Ввод данных либо вывод результатов на экран
Блок вывода		Вывод данных на печать

Продолжение таблицы 2.1

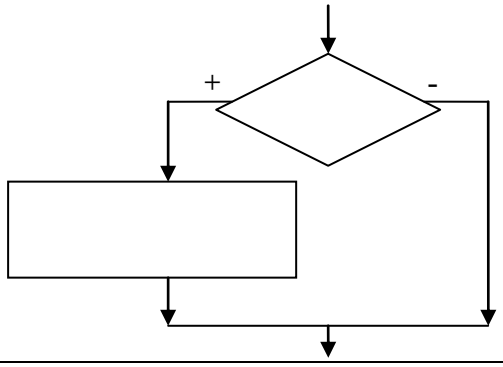
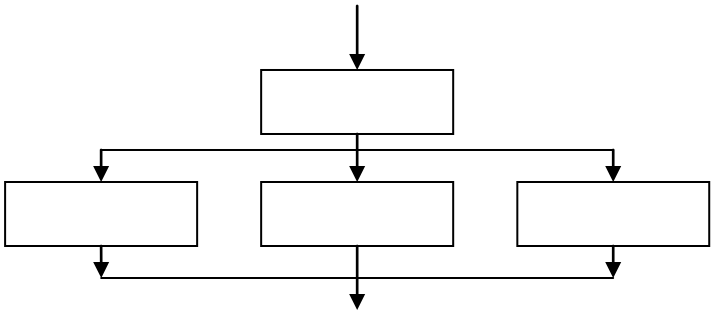
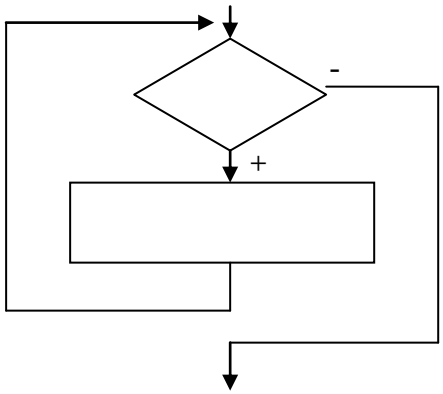
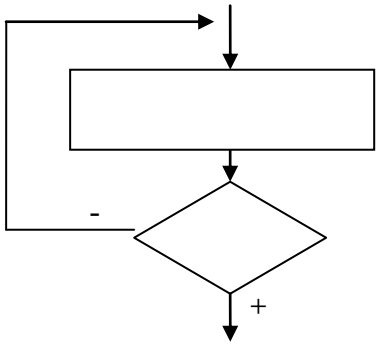
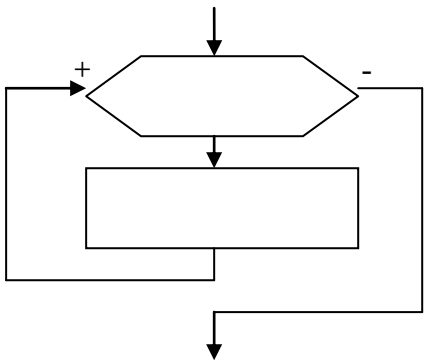
Процесс		Вычисление или последовательность вычислений
Предопределенный процесс		Выполнение подпрограммы
Альтернатива		Проверка условий
Модификация		Начало цикла
Соединитель		Разрыв линий потока информации в пределах одной страницы

В таблице 2.2. приведены основные базовые элементарные структуры для составления блок-схем.

Таблица 2.2 – Базовые структуры блок-схем

Название типа структуры	**Изображение**
Последовательность	
Разветвление (выбор)	

Продолжение таблицы 2.2

Сокращенная запись разветвления	
Выбор варианта	
Цикл с предусловием	
Цикл с постусловием	
Цикл с параметрами	

2.1.2 Алфавит и лексемы языка C++

В алфавит языка С входят:

– прописные и строчные буквы латинского алфавита;

– цифры: 0,1,2,3,4,5,6,7,8,9;

– специальные знаки:

” {} , [] () | + - / \ % ;
‘ : < = > _ ! & # ^ . *
~

Из символов алфавита формируются лексемы языка:

– идентификаторы;

– ключевые (служебные, иначе зарезервированные) слова;

– константы;

– знаки операций;

– разделители (знаки пунктуации).

Рассмотрим эти лексические элементы языка подробнее.

Идентификатор – последовательность из букв латинского алфавита, десятичных цифр и символов подчеркивания, начинающаяся не с цифры:

```
RUN       run       hard_RAM_disk          copy_54
```

Прописные и строчные буквы различаются. Таким образом, в этом примере два первых идентификатора различны. На длину различаемой части идентификатора конкретные реализации накладывают ограничение. Компиляторы различают не более 32-х первых символов любого идентификатора. Некоторые реализации С++ на ЭВМ типа VAX допускают идентификаторы длиной до 8 символов.

Ключевые (служебные) слова – это идентификаторы, зарезервированные в языке для специального использования. Ключевые слова С++:

```
asm             else            private         throw
auto            enum            protected       try
break           extern          public          typedef
case            float           register        typeid
catch           for             return          union
char            friend          short           unsigned
class           goto            signed          virtual
const           if              sizeof          void
continue        inline          static          volatile
default         int             struct          while
delete          long            switch
do              new             template
double          operator        this
```

Ранее в языке C++ был зарезервирован в качестве ключевого слова идентификатор overload. Для компиляторов фирмы Borland (BC++ и TC++) дополнительно введены ключевые слова:

```
cdecl          _export        _loadds        _saveregs
_cs            far            near           _seg
_ds            huge           pascal         _ss
es             interrupt      regparam
```

Там же введены как служебные слова регистровые переменные:

```
_AH            _CH            _SI            _SS
_AL            _CL            _DI            _ES
_AX            _CX            _BP            _FLAGS
_BH            _DH            _SP
_BL            _DL            _CS
_BX            _DX            _DS
```

Отметим, что ранние версии BC++ и TC++ не включали в качестве ключевых слов идентификаторы throw, try, typeid, catch.

Не все из перечисленных служебных слов сразу же необходимы программисту, однако запрещено использовать их в качестве произвольно выбираемых имен, и список служебных слов нужно иметь уже на начальном этапе знакомства с языком C++. Кроме того, идентификаторы, включающие два подряд символа подчеркивания (__), резервируются для реализаций C++ и стандартных библиотек. Идентификаторы, начинающиеся с символа подчеркивания (_), используются в реализациях языка Си. В связи с этим начинать выбираемые пользователем идентификаторы с символа подчеркивания и использовать в них два подряд символа подчеркивания не рекомендуется.

Константа (литерал) – это лексема, представляющая изображение фиксированного числового, строкового или символьного (литерного) значения.

Константы делятся на пять групп: целые, вещественные (с плавающей точкой), перечислимые, символьные (литерные) и строковые (строки или литерные строки). Перечислимые константы проект стандарта языка C++ [2] относит к одному из целочисленных типов.

Компилятор, выделив константу в качестве лексемы, относит её к той или другой группе, а внутри группы – к тому или иному типу данных по ее «внешнему виду» (по форме записи) в исходном тексте и по числовому значению.

Целые константы могут быть десятичными, восьмеричными и шестнадцатеричными.

Фундаментальные объекты данных, с которыми работает программа, – это переменные и константы. Используемые в программе переменные перечисляются в объявлениях или декларациях, в которых указывается их тип, а также иногда их начальные значения.

С именами переменных связывается тип данных, который контролируется компилятором и для которого выделяется определенное количество байтов памяти. Имена переменных должны начинаться с буквы (латинского

алфавита) или символа подчеркивания (например, _aza), за которым могут следовать любые комбинации букв в любом регистре (заглавные или строчные), символы подчеркивания или цифры 0–9. В языке **C** имеется различие между заглавными и строчными буквами. Поэтому переменная World будет отличаться от переменной world и т. п. При этом в определении переменной не разрешается символ пробела (пробелов) и некоторые другие символы, например $... .

Стандарт C89 определяет пять базовых типов данных:

int – целочисленный тип, целое число;

float – вещественное число одинарной точности с плавающей точкой;

double – вещественное число двойной точности с плавающей точкой;

char – символьный тип для определения одного символа;

void – тип без значения.

Кроме того, существуют модификаторы, которые могут применяться к этим базовым типам. Ряд компиляторов может поддерживать еще и логический тип _Bool. Тип void служит для объявления функции, не возвращающей значения, или для создания универсального указателя (pointer).

Объект типа char всегда занимает 1 байт памяти. Размеры объектов других типов, как правило, зависят от среды программирования и операционной системы.

Приведем модификаторы базовых типов данных. К ним относятся следующие спецификаторы, предшествующие им в тексте программы:

```
signed, unsigned, long, short
```

Базовый тип int может быть модифицирован каждым из перечисленных спецификаторов. Тип char модифицируется с помощью unsigned и signed, тип double – с помощью long.

В табл. 2.3 приведены допустимые комбинации типов данных языка **C** с их минимальным диапазоном значений и типичным размером.

Таблица 2.3 – Типы данных языка C

Тип данных	**Типичный размер в битах**	**Минимально допустимый диапазон значений**
char	8 (или 1 байт)	от –127 до 127
unsigned char	8	от 0 до 255
signed char	8	от –127 до 127
int	16 или 32	от –32767 до 32767
unsigned int	16 или 32	от 0 до 65535
signed int	16 или 32	от –32767 до 32767
short int	16	от –32767 до 32767
unsigned short int	16	от 0 до 65535
signed short int	16	от –32767 до 32767
long int	32	от –2147483647 до 2147483647
long long int	64	от $-(2^{63}-1)$ до $(2^{63}-1)$ для C99

Продолжение таблицы 2.3

signed long int	32	от –2147483647 до 2147483647
unsigned long int	32	от 0 до 4294967295
unsigned long long int	64	от 0 до $(2^{64}–1)$ для C99
float	32	от 1E–37 до 1E+37 (с точностью не менее 6 значащих десятичных цифр)
double	64	от 1E–37 до 1E+37 (с точностью не менее 10 значащих десятичных цифр)
long double	80	от 1E–37 до 1E+37 (с точностью не менее 10 значащих десятичных цифр)

Для базового типа int возможны следующие записи с модификатором:

```
signed  или  signed int
unsigned  или  unsigned int
long  или  long int
short  или  short int
```

Для данных вещественного типа максимальные значения абсолютных величин представлены в табл. 2.4.

Таблица 2.4 – Вещественные типы данных языка С

Тип данных	**Типичный размер в битах**	**Диапазон абсолютных величин**
float	32	от 3.4E–38 до 3.4E+37
double	64	от 1.7E–308 до 1.7E+308
long double	80	от 3.4E–4932 до 1.1E+4932

В языке **С** предусматривается преобразование типов в выражениях и приведение типов. Если в выражении смешаны различные типы литералов и переменных, то компилятор преобразует их в один тип. Во-первых, все char и short int значения автоматически преобразуются (с расширением "типоразмера") в тип int. Этот процесс называется целочисленным расширением (integral promotion). Во-вторых, все операнды преобразуются (также с расширением "типоразмера") в тип самого большого операнда. Этот процесс называется расширением типа (type promotion), причем он выполняется пооперационно. Например, если один операнд имеет тип int, а другой – long int, то тип int расширяется в тип long int. Или если хотя бы один из операндов имеет тип double, то любой другой операнд приводится к типу double. Это означает, что такие преобразования, как тип char в тип double, вполне допустимы (если предусматривать, к чему это приведет). После преобразования оба операнда будут иметь один и тот же тип, а результат операции – тип, совпадающий с типом операндов. Приведем последовательность преобразования типов в выражениях по старшинству:

```
ЕСЛИ  операнд имеет тип long double
ТО  второй операнд преобразуется в long double
ИНАЧЕ ЕСЛИ  операнд имеет тип double
ТО  второй операнд преобразуется в double
ИНАЧЕ ЕСЛИ  операнд имеет тип float
ТО  второй операнд преобразуется в float
ИНАЧЕ ЕСЛИ  операнд имеет тип unsigned long
ТО  второй операнд преобразуется в unsigned long
ИНАЧЕ ЕСЛИ  операнд имеет тип long
ТО  второй операнд преобразуется в long
ИНАЧЕ ЕСЛИ  операнд имеет тип unsigned int
ТО  второй операнд преобразуется в unsigned int
```

Кроме того, действует правило: если один из операндов имеет тип `long`, а второй – `unsigned int`, притом значение `unsigned int` не может быть представлено типом `long`, то оба операнда преобразуются в значение типа `unsigned long`.

В языке С предусматривается явное преобразование (приведение) типов. Общая форма оператора явного приведения типа: (тип) выражение.

В приведенной форме тип – это любой поддерживаемый тип данных. Явное преобразование типа – это операция. Оператор приведения типа является унарным и имеет тот же приоритет, что и остальные унарные операторы.

В приводимых ниже программах используются такие средства ввода-вывода, как **`printf(), getchar(), gets(), scanf().`**

Приведем характеристику данных функций.

Прототип функции `printf()` имеет вид:

```
int printf(const char *format, ?);
```

Функция printf() записывает в стандартный поток stdout (стандартный выходной поток данных) значения аргументов из заданного списка аргументов в соответствии со строкой форматирования, адресуемой параметром format. Строка форматирования состоит из элементов двух типов. К элементам первого типа относятся символы, которые выводятся на экран. Элементы второго типа содержат спецификации формата, определяющего способ отображения аргументов. Спецификация формата начинается символом процента, за которым следует код формата. На спецификации формата могут воздействовать модификаторы, задающие ширину поля, точность и признак выравнивания по левому краю. Целое значение, расположенное между знаком % и командой форматирования, играет роль спецификации минимальной ширины поля. Наличие этого спецификатора приводит к тому, что результат будет заполнен пробелами или нулями, чтобы выводимое значение занимало поле, ширина которого не меньше заданной С I минимальной ширины. По умолчанию в качестве заполнителя используется пробел (пробелы).

Для заполнения нулями перед спецификацией ширины поля нужно поместить нуль, т. е. 0. Например, спецификация формата %05d дополнит нулями выводимое целое число, в котором менее пяти цифр, чтобы общая длина равнялась пяти символам. Действие модификатора точности зависит от кода формата, к которому он применяется. Чтобы добавить модификатор точности, следует поставить за спецификатором ширины поля десятичную точку, а после нее – требуемое значение точности (число знаков после десятичной точки). Применительно к целым числам модификатор точности задает минимальное количество выводимых цифр. При необходимости перед целым числом будут добавлены нули. Если модификатор точности применяется к строкам, то число, следующее за точкой, задает максимальную длину поля. Например, спецификация %5.7s выведет строку длиной не менее пяти, но не более семи символов. Если выводимая строка окажется длиннее максимальной длины поля, конечные символы будут отсечены. По умолчанию все выводимые значения выравниваются по правому краю: если ширина поля больше выводимого значения, то оно будет выровнено по правому краю поля. Чтобы установить выравнивание по левому краю, нужно поставить знак "минус" ("–") сразу после знака процента. Например, спецификация формата % – 10.4f обеспечит выравнивание вещественного числа с четырьмя десятичными знаками по левому краю в 10-символьном поле. Существуют два модификатора формата, позволяющие функции printf() отображать короткие и длинные целые числа. Это модификатор l (латинская буква эль) уведомляет функцию printf() о длинном типе значения. Модификатор h сообщает функции printf(), что нужно вывести число короткого целого типа. Кроме того, модификатор l можно поставить перед командами форматирования вещественных чисел. В этом случае он уведомит о выводе значения типа long double.

Спецификаторы формата для функции printf() перечислены в табл. 2.5.

Таблица 2.5 – Спецификаторы формата функции printf()

Код	**Формат**
%c	Символ
%d	Десятичное целое число со знаком
%i	Десятичное целое число со знаком
%e	Экспоненциальное представление числа (в виде мантиссы и порядка, e — на нижнем регистре)
%E	Экспоненциальное представление числа (в виде мантиссы и порядка, E — на верхнем регистре)
%f	Десятичное число с плавающей точкой
%F	Десятичное число с плавающей точкой (только стандарт C99; если применяется к бесконечности или нечисловому значению, то выдает надписи INF, INFINITY(бесконечность) или NAN — Not A Number на верхнем регистре. Спецификатор %f выводит их эквиваленты на нижнем регистре)

Продолжение таблицы 2.5

%g	Использует более короткий из форматов %e или %f
%G	Использует более короткий из форматов %E или %F
%o	Восьмеричное число без знака
%s	Символьная строка
%x	Шестнадцатеричное без знака (строчные буквы)
%X	Шестнадцатеричное без знака (прописные буквы)
%p	Выводит указатель
%n	Соответствующий аргумент должен быть указателем на целое число. (Этот спецификатор указывает, что в целочисленной переменной, на которую указывает ассоциированный с данным спецификатором указатель, будет храниться число символов, выведенных к моменту обработки спецификации %n)
%%	Выводит знак процента

Прототип функции getchar() имеет следующий вид:

```
int getchar(void);
```

Функция getchar() возвращает из стандартного потока stdin (входного потока данных) следующий символ. При чтении символа предполагается, что символ имеет тип unsigned char, который потом преобразуется в целый. При достижении конца файла, как и при обнаружении ошибки, функция getchar() возвращает значение EOF (End Of File – конец файла).

Прототип функции gets имеет следующий вид:

```
char *gets(char *str);
```

Функция gets() читает символы (включая пробелы) из стандартного потока stdin и помещает их в массив символов, адресуемый указателем *str (далее это массив символов). Символы читаются до тех пор, пока не встретится разделитель строк или значение EOF. Для реализации EOF на клавиатуре следует набрать одновременно Ctrl+Z. Вместо разделителя строк в конец строки вставляется нулевой символ, свидетельствующий о ее завершении. Следует учесть, что нет способа ограничить количество символов, которое прочитает функция gets(). Поэтому массив, адресуемый указателем *str, может переполниться, и тогда программа выдаст непредсказуемые результаты.

Прототип функции scanf() имеет следующий вид:

```
int scanf(const char *format, ?);
```

Функция scanf() представляет собой функцию для ввода данных общего назначения, которая читает поток stdin и сохраняет информацию в переменных, перечисленных в списке аргументов. Если в строке форматирования встретится разделитель, то функция scanf() пропустит один

или несколько разделителей во входном потоке. Под разделителем, или пробельным символом, подразумевают пробел, символ табуляции \t или разделитель строк \n. Все переменные должны передаваться посредством своих адресов, например, с помощью символа &. Управляющая строка, задаваемая параметром format, состоит из символов трех категорий: спецификаторов формата, пробельных символов, символов, отличных от пробельных.

Спецификация формата начинается знаком % и сообщает функции scanf() тип данного, которое будет прочитано. Спецификации формата функции scanf() приведены в табл.2.6.

Таблица 2.6 – Спецификаторы формата функции scanf()

Код	**Формат**
%c	Читает один символ
%d	Читает десятичное целое число
%i	Читает целое число в любом формате (десятичное, восьмеричное или шестнадцатеричное)
%u	Читает десятичное целое число типа short int
%e	Читает число с плавающей точкой (и в экспоненциальной форме)
%E	Аналогично коду %e
%f	Читает число с плавающей точкой
%lf	Читает десятичное число с плавающей точкой типа double
%F	Аналогично коду %f (для стандарта C99)
%g	Читает число с плавающей точкой.
%G	Аналогично коду %g
%o	Читает восьмеричное число
%x	Читает шестнадцатеричное число
%X	Аналогично коду %x
%s	Читает строку
%p	Читает указатель
%n	Принимает целое значение, равное количеству прочитанных до сих пор символов
%[ ]	Просматривает набор символов
%%	Читает знак процента

Строка форматирования читается слева направо, и спецификации формата сопоставляются с аргументом в порядке их перечисления в списке аргументов. Символ *, стоящий после знака % и перед кодом формата, прочитает данные заданного типа, но запретит их присваивание. Команды форматирования могут содержать модификатор максимальной длины поля. Он представляет собой целое число, располагаемое между знаком % и кодом формата, которое ограничивает количество читаемых для всех полей символов. Если входной поток содержит больше заданного количества симво-

лов, то при последующем обращении к операции ввода чтение начнется с того места, в котором «остановился» предыдущий вызов функции scanf(). Если разделитель (например, пробел) встретится раньше, чем достигнута максимальная ширина поля, то ввод данных завершится. В этом случае функция scanf() переходит к чтению следующего поля. При чтении одиночных символов символы табуляции и разделители строк читаются подобно любому другому символу.

В программах бывает необходимость определять константы. В языке **C** типы констант можно задавать явно при использовании суффиксов. Например:

```
long int  j = -12345678L;       /* суффикс L */
unsigned int a = 678U;          /* суффикс U */
float x = 123.45F;              /* суффикс F */
long double z = 12345678.99L;   /* суффикс L* /
```

По умолчанию спецификации f, e, g заставляют функцию scanf() присваивать переменным типа float. Если перед одной из этих спецификаций поставить модификатор l, то функция scanf() присвоит прочитанные данные переменной типа double.

Функция scanf() поддерживает спецификатор формата общего назначения, называемый набором сканируемых символов. В этом случае определяется набор символов, которые могут быть прочитаны функцией scanf() и присвоены соответствующему массиву символов. Для определения такого набора символы, подлежащие сканированию, необходимо заключить в квадратные скобки. Открывающая квадратная скобка должна следовать сразу за знаком процента. При использовании набора сканируемых символов функция scanf() продолжает читать символы и помещать их в соответствующий массив символов до тех пор, пока не встретится символ, отсутствующий в данном наборе. Если первый символ в наборе является знаком "^", то получится обратный эффект: входное поле читается до тех пор, пока не встретится символ из заданного набора сканируемых символов, т. е. знак "^" заставляет функцию scanf() читать только те символы, которые отсутствуют в наборе сканируемых символов. Если в строке форматирования встретился символ, отличный от разделителя, то функция scanf() прочитает и отбросит его. Если заданный символ не найден, то функция scanf() завершает работу.

В таких средах разработки как MS Visual Studio 2008 и 2010 рекомендуется для безопасной работы применять функции gets_s() и scanf_s(). Для этих функций при чтении символа или строки следует указать размер в байтах, соответственно для символа или строки. Например, scanf_s("%c", &ch, 1). В Visual Studio 2010 тип данных char занимает 1 байт.

2.1.3 Математические функции в языке программирования С++

Таблица 2.7 – Запись математических функций в С++

Математическая функция	Название функции	Функция на языке программирования С++	Поясненне
$\arccos x$	арккосинус	`#include <math.h> double acos(double x); float acosf(float x);`	аргумент для acos должен находиться в отрезке [-1,1]. Acos и acosf возвращают значения в радианах на промежутке от 0 до pi.
arccosh(x)	обратный гиперболический косинус	`#include <math.h> double acosh(double x); float acoshf(float x);`	x должен быть больше либо равен 1.
$\arcsin x$	арксинус	`#include <math.h> double asin(double x); float asinf(float x);`	аргумент для asin должен находиться в отрезке [-1,1]. Asin и asinf возвращают значения в радианах в промежутке от –pi/2 до pi/2.
Arcsinh(x)	обратный гиперболический синус	`#include <math.h> double asinh(double x); float asinhf(float x);`	ни atanh, ни atanhf не являются ANSI C – функциями.
arctgx	арктангенс	`#include <math.h> double atan(double x); float atanf(float x);`	atan и atanf возвращают значения в радианах на промежутке от –pi/2 до pi/2.
$arctgh(x)$	обратный гиперболический тангенс	`#include <math.h> double atanh(double x); float atanhf(float x);`	ни atanh, ни atanhf не являются ANSI C - функциями.
$\sqrt[3]{x}$	кубический корень	`#include <math.h> double cbrt(double x); float cbrtf(float x);`	Является стандартной функцией ANSI C
$\cosh x$	гиперболический косинус	`#include <math.h> double cosh(double x); float coshf(float x);`	Углы определены в радианах.
e^x	экспонента числа	`#include <math.h> double exp(double x); float expf(float x);`	Является стандартной функцией ANSI C
$\lvert x \rvert$	модуль числа (абсолютная величина)	`#include <math.h> double fabs(double x); float fabsf(float x);`	Является стандартной функцией ANSI C

Продолжение таблицы 2.7

Min и max	наименьшее и наибольшее ближайшие целые	`#include <math.h> double floor(double x); float floorf(float x); double ceil(double x); float ceilf(float x);`	Является стандартной функцией ANSI C
mod	остаток от деления в виде числа с плавающей точкой	`#include <math.h> double fmod(double x, double y); float fmodf(float x, float y);`	Является стандартной функцией ANSI C
$\ln x$	натуральный логарифм	`#include <math.h> double log(double x); float logf(float x);`	Является стандартной функцией ANSI C
$\lg x$	логарифм по основанию 10	`#include <math.h> double log10(double x); float log10f(float x);`	log10 возвращает значение логарифма по основанию 10 от x. Он определяется как ln(x)/ln(10).
x^y	возведение основания ***x*** в степень ***y***	`#include <math.h> double pow(double x, double y); float powf(float x, float y);`	Является стандартной функцией ANSI C
rint, rintf,	округление до ближайшего целого	`#include <math.h> double rint(double x); float rintf(float x);`	Является стандартной функцией ANSI C
	остаток от деления x/y	`double remainder(double x, double y); float remainderf(float x, float y);`	это будет число между –y/2 и y/2.
$\sqrt{x}$	квадратный корень из числа	`#include <math.h> double sqrt(double x); float sqrtf(float x);`	вычисляет арифметический (неотрицательный) квадратный корень из аргумента
$\sin x$	синус	`#include <math.h> double sin(double x); float sinf(float x);`	Углы определены в радианах.
$\cos x$	косинус	`#include <math.h> double cos(double x); float cosf(float x);`	Углы определены в радианах.
$\sinh x$	гиперболический синус	`#include <math.h> double sinh(double x); float sinhf(float x);`	Углы определены в радианах.
$tg(x)$	тангенс	`#include <math.h> double tan(double x); float tanf(float x);`	Углы определены в радианах.
$tgh(x)$	гиперболический тангенс	`#include <math.h> double tanh(double x); float tanhf(float x);`	Углы определены в радианах. tanh(x) определяется как sinh(x)/cos(x)

2.2 Практическая часть

Пример 1. *Напишите программу вычисления площади круга и его длины окружности по заданному радиусу, вводимого пользователем с клавиатуры, а также вывода на консоль максимальных значений чисел типа int, float и double.*

Для решения примера следует воспользоваться математической библиотекой компилятора, т. е. включить в программу заголовочный файл <math.h>., а также заголовочные файлы <limits.h>, <float.h>.

Программный код решения примера:

```
#include <stdio.h>
#include <conio.h>
// Для числа пи (π)
#define _USE_MATH_DEFINES
#include <math.h>
#include <limits.h>
#include <float.h>
int main (void)
{
double  R,  Sr, Lr;
printf("\n Enter a real greater than zero: ");
scanf_s("%lf", &R);
Sr = M_PI*R*R;
Lr = 2*M_PI*R;
printf("\n Area of a circle of radius R = %g is %g", R, Sr);
printf("\n  The  length  of  a  circle  of  radius  R  =  %g  is
%g",R,Lr);
puts("");
printf("\n  Maximum integer: %d\n ", INT_MAX);
printf(" Maximum real number of float: %g\n ", FLT_MAX);
printf("Maximum real number type double: %g\n ", DBL_MAX);

printf("\n Press any key: ");
      _getch();
      return 0;
}
```

В программу включена константа _USE_MATH_DEFINES для работы с числом M_PI (π). Остальные константы можно найти в справочной документации компилятора. Например, через меню Help →Index системы MS Visual Studio 2008.

Функция `scanf_s()` определена в компиляторе языка **C** системы MS Visual Studio 2008. С этой функцией компилятор не выдает предупреждений.

Результат выполнения программы показан на рис. 2.1.

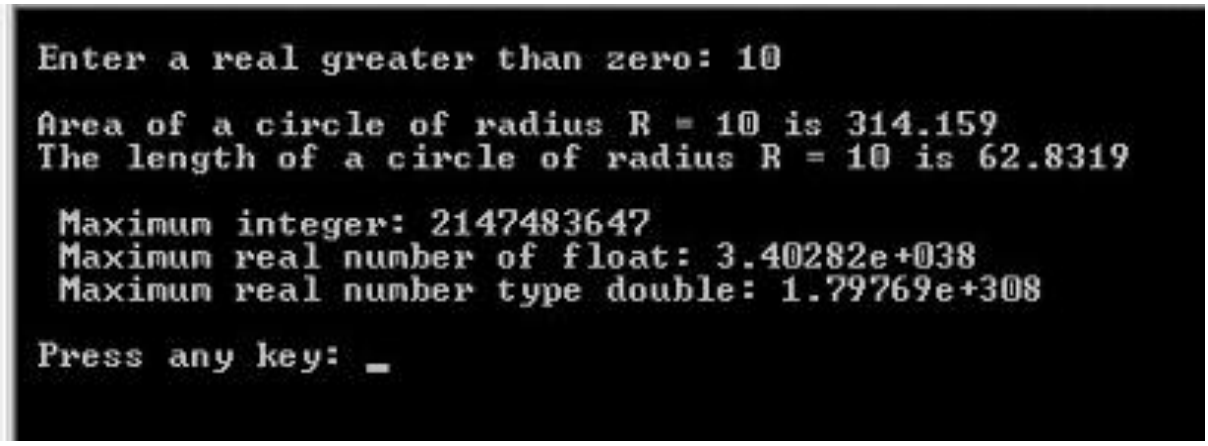


Рисунок 2.1 – Пример использования предопределенных констант

В начале функции `int main(void)` сделано объявление переменных, которые будут использоваться в программе. Каждый тип переменных объявлен через запятую.

Функции `printf()` выводят либо только сообщения, либо еще заданные переменные соответствующих типов.

Функция `gets()` позволяет считывать символы с наличием разделителей, в частности, с пробелами. В Microsoft Visual Studio 2010 рекомендуется использовать `gets_s()`, чтобы не было предупреждений.

Следует обратить внимание на формат записи функций scanf(). Если сканируются числа, или одиночные символы, то присваивание этих символов переменным, которые были объвлены через соответствующие типы данных, осуществляется с помощью взятия адреса этих переменных, т. е. с помощью символа &, например, scanf_s(«%c»,&ch, 1). При сканировании массива символов, т. е. при сканировании строки, символ & не используется. Имя массива само по себе является указателем. Обращение к адресу осуществляется с помощью указателей (будут рассмотрены позднее). Для сканирования чисел типа double в функции scanf_s() используется спецификатор l.

Пример 2. *Создать блок-схему к программе и программу на языке программирования Си++ для вычисления функции* ***B****, которая зависит от трех переменных* ***x, y, z****. Ввод значений переменных сделать с клавиатуры, если* $B = x^2 + 3xy + \sqrt{z}$, где $z = (x + y)^3$.

Окно с программой показано на рис. 2.2, а блок-схема к заданию на рис.2.3.

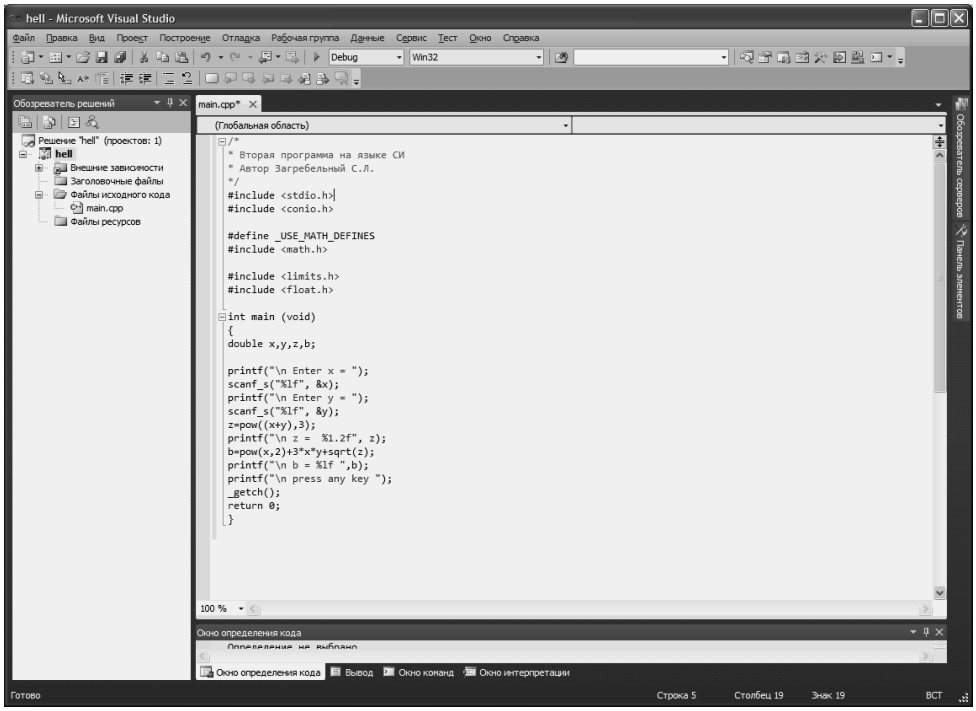


Рисунок 2.2 – Окно программы Примера 2

Программный код решения примера:

```
/*
* Вторая программа на языке СИ
* Автор Загребельный С.Л.
*/
#include <stdio.h>
#include <conio.h>

#define _USE_MATH_DEFINES
#include <math.h>

#include <limits.h>
#include <float.h>

int main (void)
{
double x,y,z,b;
printf("\n Enter x = ");
scanf_s("%lf", &x);
printf("\n Enter y = ");
scanf_s("%lf", &y);
z=pow((x+y),3);
printf("\n z =  %1.2f", z);
b=pow(x,2)+3*x*y+sqrt(z);
printf("\n b = %lf ",b);
printf("\n press any key ");
_getch();
return 0;
}
```

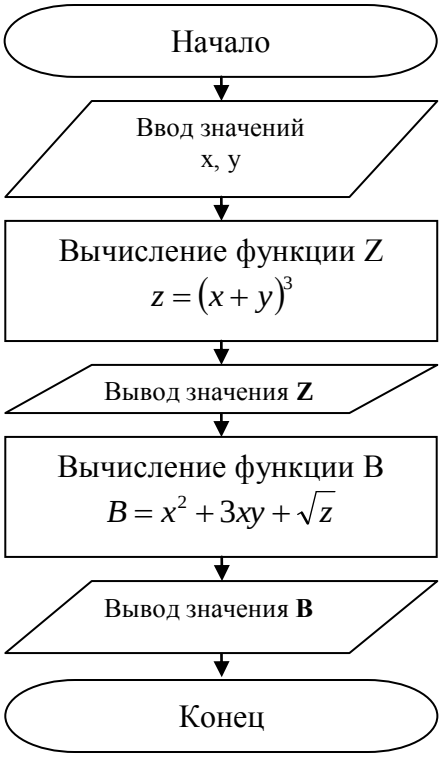


Рисунок 2.3 – Блок-схема примера 2

В программу включена библиотека математических функций math.h, в которой sqrt() – функция извлечения квадратного корня, pow(*a, b*) – функция возводит в степень *b* число по основанию *a*. Все эти функции возвращают результат типа double и вычисляют от числа (выражения) также типа double.

Результат выполнения программы показан на рис. 2.4.

Рисунок 2.4 – Окно вывода результата Примера 2

2.3 Индивидуальные задания

Составить блок-схему к программе и программу на языке программирования C++ для вычисления функций b = f(x,y,z) , где z = w(x,y) при постоянных значениях x и y (см. пример 2). Значения x и y заданы в таблице 2.8.

Таблица 2.8 – Индивидуальные задания

Вариант	*f(x,y,z)*	*w(x,y)*	*x*	*y*
1	$e^{-2x}(tg(z)+2y)$	$\sqrt{\sin^2\|x\|+y}$	-4,52	0,75
2	$\frac{\sqrt{x}\sin(2y)}{z+e^z y}$	$\frac{2xy}{x+\cos(y)}$	2,87	0,84
3	$\frac{y-z/(y-x)}{\cos(x)+(y-x)^2}$	$\frac{\sqrt{15y}}{y+ctg(x)}$	1,82	18,25
4	$y^x+\sqrt[3]{\|x\|+\|y\|}e^z$	$\frac{\sqrt{\|20x\|}}{x^2+y^3}$	-0,85	1,25
5	$\ln(\sqrt{x}-\sqrt{y}+2)z^3$	$\frac{\sin(x/y)}{2x^2}$	25,34	33,85
6	$y+\frac{xarctg(z)}{y+x^2}$	$\sqrt{x}\sin(y)$	0,12	-8,75
7	$\frac{z^3}{x+y^3/(x+z^2)}$	$\frac{15}{x+e^y}$	1,54	3,26
8	$\frac{z^2}{y+x^3}+\sin(y/5)$	$\frac{3x}{\cos^2(y)+3}$	1,58	3,42
9	$\|\cos(x)+\sin(y)\|-2tg^2(z)$	$\frac{\sqrt{x}\sin^2(y)}{x+e^y}$	0,42	-0,87
10	$\ln y^{\sqrt{\|x\|}}(z^2-\frac{y}{\sin(x)})$	$\sqrt{3+2y}$	-15,24	4,67
11	$\frac{z^2}{y+x^3}+\sin(y/5)$	$\frac{\sqrt{x}arctg(2y)}{e^{y+x}}$	6,55	-2,78
12	$\cos^2(z)+\|x+y\|^3$	$\frac{12}{x+e^y}$	-2,75	-1,42
13	$x^{y/x}+\sqrt[3]{\|y^z\|}e^x$	$\ln(\sqrt{e^{x-y}}+x^2)$	1,82	18,23
14	$\frac{e^{z-1}}{2y+x^3}+\sin(y^2)$	$\cos^2(y)+\sin^3(x^2)$	0,84	0,65
15	$\sqrt{\|y\|e^{-(y+x)}}-\cos(z^3)$	$\frac{x+6y}{\sin(x)+\ln(y)}$	1,12	0,87
16	$\frac{4y^2e^{3x}}{8z^3+\ln\|x\|}$	$\frac{x+y\sqrt{x}}{x+10}$	0,27	4,38
17	$\frac{\sqrt{y\ln(x)}-zx^2}{1+tg^2(x^2)}$	$\frac{e^x\sqrt{x^3+y}}{x-1}$	6,35	7,32
18	$\frac{\ln(y+\sqrt{y+x^2})}{(z+x^2)e^{x/2}}$	$\frac{2x\sqrt{y}}{\sin(x^2)}$	0,42	1,23
19	$\frac{x^3+y}{\sin^2(z)+x/5}$	$\frac{\cos^2(3(2+x))}{4-y^2\sqrt{x}}$	43,32	-0,54
20	$\frac{x+y(x^2+\cos(y))}{y(x-z)+\ln\|xz\|}$	$2\sin(3x+y)$	3,25	4,12

Продолжение таблицы 2.8

21	$\frac{1+\cos^2(x+z)}{\left\|x^3-2\ln\sqrt{y}\right\|}$	$\frac{x^2+y^2}{e^{x+y}}$	0,83	2,38
22	$\frac{\ln\|x\|}{\sqrt[3]{\|x\|+\|y\|}+tg(z)}$	$\sqrt{x^2-\sin(y)}$	-0,93	-0,25
23	$\frac{z^3}{x+y^3/(x+z^2)}$	$\frac{\|y+8x\|}{\sin(x)+tg(y)}$	-0,72	-1,42
24	$2^{-x}\sqrt{y+\sqrt[4]{\|z\|}}$	$\frac{xy}{x^2+5}+\cos^2(y)$	3,98	1,63
25	$\sqrt{e^{x-1}\sqrt{\|y\|}}$	$\frac{3y}{3+e^{x-y}}$	3,91	-0,51
26	$\frac{\sqrt[3]{\sqrt{xy}}-x^2}{1+z^2}$	$\frac{x^3+\sqrt{xy}}{\left(x+y^2\right)}$	1,26	3,69
27	$\frac{\ln\left\|z^3+x+y\right\|}{\sqrt{x^2+y^2}-\sqrt{z}}$	$\frac{y\sqrt{(x+y)^4}}{x+10y}$	-4,11	2,99
28	$\frac{\left(x+y+z^3\right)^2}{(x+z)/z^2}$	$\frac{x^2\sqrt{y}-y^2\sqrt{x}}{x^2+y^2}$	1,24	2,55
29	$\frac{\sqrt[3]{tg(x+y+z)}}{\ln\|x+y-z\|}$	$\frac{x-\sqrt[3]{y}}{y+\sqrt[3]{x}}$	-1,25	-3,16
30	$\frac{\sqrt[3]{\cos(x)+\sin(y)}}{tg(z)}$	$\frac{x^2+x^y}{\cos\left(\sqrt[3]{x}\right)}$	1,84	-1,17

2.4 Контрольные вопросы

1 Для каких типов данных используются суффиксы при инициализации переменных?

2 Чем отличаются функции `printf()` и `puts()` при консольном выводе информации?

3 Для чего в программах на **C++** используется заголовочный файл math.h?

4 При использовании функции `gets_s()` с какими разделителями может происходить считывание информации с консоли?

5 Какой тип данных возвращает функция `gets_s()` при считывании информации?

6 Как осуществляется считывание с консоли информация с помощью функции `scanf_s()`?

7 Как с консоли осуществляется считывание последовательности различных типов данных с помощью одной функции `scanf_s()`?

8 Как выводится на консоль последовательность различных типов данных с помощью одной функции `printf()`?