БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАТИКИ И КОМПЬЮТЕРНЫХ СИСТЕМ

Н. В. Серикова

ПРАКТИЧЕСКОЕ РУКОВОДСТВО
к лабораторному практикуму

«ЛИНЕЙНЫЕ АЛГОРИТМЫ. ВЕТВЛЕНИЯ»

по дисциплине

«ПРОГРАММИРОВАНИЕ НА C++»

2024
МИНСК

Практическое руководство к лабораторному практикуму «ЛИНЕЙНЫЕ АЛГОРИТМЫ. ВЕТВЛЕНИЯ» по дисциплине «ПРОГРАММИРОВАНИЕ НА С++» предназначено для студентов, изучающих базовый курс программирования на языке С++, специальностей «Компьютерная безопасность», «Прикладная информатика», «Радиофизика».

Руководство содержит некоторый справочный материал, примеры решения типовых задач с комментариями.

Автор будет признателен всем, кто поделится своими соображениями по совершенствованию данного пособия.

Возможные предложения и замечания можно присылать по адресу:
E-mail: Serikova@bsu.by

©Серикова Н.В.

ОГЛАВЛЕНИЕ

КЛЮЧЕВЫЕ СЛОВА C/C++

Ключевые слова – формируют синтаксис языка и зарезервированы компилятором:

asm	explicit	operator	true
auto	export	private	try
bool	extern	protected	typedef
break	false	public	typeid
case	float	register	typename
catch	for	reinterpret_cast	union
char	friend	return	unsigned
class	goto	short	using
const	if	signed	uuid
const_cast	inline	sizeof	virtual
continue	int	static	void
default	long	static_cast	volatile
delete	main	struct	wchar_t
do	mutable	switch	while
double	naked	template	
dynamic_cast	namespace	this	
else	new	thread	
enum	noreturn	throw	

ИДЕНТИФИКАТОРЫ

Идентификаторы – имена переменных, функций и других объектов, определенных пользователями.

При записи идентификаторов (имен) объектов можно использовать:

- латинские буквы,
- цифры,
- символ подчеркивания.

Длина – до 1024 символов.

Первый символ не цифра.

Прописные и строчные буквы считаются разными символами.

КОНСТАНТЫ

Константа – информационный объект программы, не изменяющийся в процессе ее исполнения.

Каждой константе в программе соответствует область оперативной памяти ЭВМ, в которой хранится значение этой константы.

Различают неименованные (литералы) и именованные константы.

НЕИМЕНОВАННЫЕ КОНСТАНТЫ

Тип неименованной константы неявно определяется ее значением.

В качестве неименованных констант в C++ могут использоваться:

целые числа:	десятичные	12	-123
	восьмеричные	0738	-0121
	шестнадцатеричные	0X2F56	-0x2A13B
вещественные числа		0.25	-56.12e-12
логические		true	false
символы		'Ж'	'2'
строки		"проба"	"124"
перечислимые		{red, yellow, green}	
неопределенный указатель		NULL	

Символьные константы:

- Значение – символ. Заключаются в одиночные кавычки. Например:‘a’, ‘#’, ‘1’
- Когда компилятор встречает символьную константу, он заменяет ее значением ASCII кода.

Строковые константы:

- Заключаются в двойные кавычки. Например: "Please, enter the value : "
- Компилятор объединяет две следующие одна за другой строковые константы, разделенные любыми символами - разделителями
- Символы и строки могут содержать управляющие последовательности . При их записи они начинаются с символа \ .

Управляющая последовательность	Символ
\a	**Сигнал**
\b	**Пробел**
\f	**Перевод страницы**
\n	**Перевод в начало следующей строки**
\r	**Возврат каретки**
\t	**Табуляция**
****	**Обратная косая черта**
\'	**Одинарные кавычки**
\"	**Двойные кавычки**
\x	**Шестнадцатеричный код символа**

ИМЕНОВАННЫЕ КОНСТАНТЫ

Именованную константу можно описать, присвоив ей идентификатор (имя), который можно будет затем использовать в программе вместо того, чтобы непосредственно записывать значение константы.

Когда следует обязательно использовать именованные константы:

- для задания параметров, управляющих размером структур данных (массивов и др.), числом итераций в циклах, и других, изменение которых может потребоваться при отладке или модернизации программы;
- для обозначения часто встречающихся в программе постоянных величин;
- при использовании констант, имеющих общеупотребительные обозначения.

ПЕРЕМЕННЫЕ

Переменная – информационный объект программы, предназначенный для хранения значений, которые могут изменяться в процессе исполнения программы.

Переменная имеет идентификатор (имя), по которому в программе осуществляется доступ к содержимому и адресу переменной.

Каждой переменной в программе соответствует область оперативной памяти ЭВМ, в которой хранится значение этой переменной.

Каждая переменная характеризуется типом.

Тип переменной определяет:

- размер отведенной для переменной области памяти;
- множество возможных значений, которые может принимать переменная, то есть как интерпретировать информацию (последовательность двоичных чисел), записанных в этой области памяти.

Перед тем, как использовать переменную, ее нужно описать:
Например:

```
float  h;
int    a, b, c;
```

Приветствуется инициализация переменных при их описании.
Например:

```
int    a = 3,  b = 4;
double c (13.0), d (12.4e-4);
```

ТИП ДАННЫХ

Каждый объект программы, предназначенный для хранения данных или вырабатывающий данные (константа, переменная, выражение, функция) обычно относится к определенному типу.

Тип данных определяется набором возможных значений, которые может принимать или вырабатывать объект программы (переменная, выражение, константа, функция и др.), относящийся к этому типу, и совокупностью операций, определенных над этими значениями.

Концепция типа данных основывается на следующих положениях:

1. Любой тип данных определяет множество значений, к которому принадлежит константа, которые может принимать переменная или выражение или вырабатывать операция или функция.

2. Каждая операция или функция требует аргументов фиксированного типа и выдает результат фиксированного типа. Если операция допускает аргументы нескольких типов, то тип результата можно определить по специальным правилам языка.

3. Тип значения, задаваемого константой, переменной или выражением, можно определить по их виду или описанию и остается неизменным для переменных.

В большинстве случаев новые типы данных определяются с помощью ранее определенных типов данных.

Значения, принадлежащие к такому типу, обычно представляют собой совокупности значений компонент, принадлежащих к определенным ранее типам компонент. Такие составные значения называются структурированными.

Если значение имеет всего одну компоненту, принадлежащую определенному ранее типу, то этот тип называется базовым или простым.

ПРОСТЫЕ СТАНДАРТНЫЕ ТИПЫ ДАННЫХ ЯЗЫКА С++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Точность де- сятичн. раз- рядов	Размер в байтах
bool	false	true		1
char	-128	127		1
short short int	-32 768	32 767		2
int long long int	-2 147 483 648	2 147 483 647		4
long long	$-9.2*10^{18}$	$9.2*10^{18}$		8
float	$-3.4*10^{38}$	$3.4*10^{38}$	7	4
double long double	$-1.7*10^{-308}$	$1.7*10^{308}$	15	8
void *				4

Название типа	Нижняя гра- ница диапа- зона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned short	0	65 535	2
unsigned int, unsigned long	0	4 294 967 295	4
unsigned long long	0	$18.4*10^{18}$	8

КОНСТАНТЫ ПРЕДЕЛЬНЫХ ЗНАЧЕНИЙ ДИАПАЗОНОВ ТИПОВ

```
#include <limits.h>            // #include <climits>
```

CHAR_MIN	-128
CHAR_MAX	127
UCHAR_MAX	255
SHRT_MIN	-32768
SHRT_MAX	32767
USHRT_MAX	65535
INT_MIN	-2 147 483 648
INT_MAX	2 147 483 647
UINT_MAX	4 294 967 295

```
#include <float.h>             // #include <cfloat>
```

FLT_MIN	1.17549e-038
FLT_MAX	3.40282e+038
DBL_MIN	2.22507e-308
DBL_MAX	1.79769e+308
LDBL_MIN	2.22507e-308
LDBL_MAX	1.79769e+308
DBL_EPSILON	2.22044e-016

ВЫРАЖЕНИЯ

Любая комбинация переменных, констант, функций и операций, приводящая к вычислению некоторого значения, называется выражением:

Выражения сами могут входить в состав других выражений.

Операторы (операции) – действия над объектами программы (переменными, константами, выражениями, структурами данных, объектами и др.), задаваемые специально определенными символами.

Объекты, над которыми производятся действия называются операндами.

В зависимости от числа требуемых операндов различают унарные, бинарные и тернарные операторы

АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ

Знак операции	Действие (операнды целые и вещественные, результат - в соответствии с типом операндов)
- +	Присвоение противоположного/сохранение знака
+	Сложение
-	Вычитание
*	Умножение
/	Деление (если применяется к целочисленным операндам – целочисленное деление с отбрасыванием остатка: 5/2 = 2)
%	Деление по модулю (остаток целочисленного деления: 14%3 = 2)
--	Декрементация (уменьшение на 1)
++	Инкрементация (увеличение на 1)

Арифметические операторы с присваиванием

Знак операции	Действие
+=	Сложение с замещением: x+=2 эквивалентно x=x+2
-=	Вычитание с замещением: x-=2 эквивалентно x=x-2
*=	Умножение с замещением: x*=2 эквивалентно x=x*2
/=	Деление с замещением: x/=2 эквивалентно x=x/2
%=	Деление по модулю с замещением: n%=2 эквивалентно n=n%2

ОПЕРАТОРЫ СРАВНЕНИЯ И ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Знак операции	**Операции сравнения. Действие (операнды – целые, вещественные, булевские, указательные, результат – булевский)**
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно
Знак операции	**Логические операции. Действие (операнды – булевские и целые, результат – булевский)**
&&	Логическое И
||	Логическое ИЛИ
!	Логическое НЕ

Логические операции

bool a	bool b	a && b
0	0	0
1	0	0
0	1	0
1	1	1

bool a	bool b	a || b
0	0	0
1	0	1
0	1	1
1	1	1

Запись логических выражений

высказывание	запись высказывания с помощью логических операций
0 < X <= 3 и У > 5	`(0 < X) && (X <= 3) && (У > 5)`
X < 0 или X > 5	`(0 < X) || (X > 5)`
Z является min (X, У, Z)	`(Z <= X) && (Z <= Y)`
X, Y, Z равны между собой	`(X == Y) && (X == Z) && (Y == Z)`
каждое из чисел X, Y, Z строго положительно	`(X>0) && (Y >0) && (Z>0)`
хотя бы одно из чисел X, Y, Z положительно	`(X>0) || (Y >0) || (Z>0)`
X и Y имеют разные знаки	`X*Y<0`
X и Y имеют разные знаки	`(X < 0) && (Y > 0) ||` `(Y < 0) && (X > 0)`
точка (X,Y) принадлежит второй четверти	`(X<0) && (Y>0)`
точка (X,Y) находится внутри круга радиуса R, центр которого имеет координаты (0,0)	`X*X + Y*Y <= R*R`
Z является min (X, У, Z)	`(Z <= X) && (Z <= Y)`

только одно из чисел X, Y, Z является положительным	`(X > 0 && Y < 0 && Z < 0) ||`
	`(Y > 0 && X < 0 && Z < 0) ||`
	`(Z > 0 && X < 0 && Y < 0)`

ни одно из чисел X, Y, Z не равно 10	`(X != 10) && (Y != 10) && (Z != 10)`
ни одно из чисел X, Y, Z не равно 10	`int(X == 10) + int(Y == 10)` `+ int(Z == 10) == 0`

ПОБИТОВЫЕ ОПЕРАТОРЫ

Знак операции	Действие (операнды – целые, результат – целый)
&	Побитовое И
|	Побитовое ИЛИ
^	Побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ
>>	Сдвиг вправо
<<	Сдвиг влево
~	Побитовое НЕ

Побитовые операторы с присваиванием

Знак операции	Действие (операнды – целые, результат – целый)
&=	Побитовое И с замещением
|=	Побитовое ИЛИ с замещением
^=	Побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ с замещением
>>=	Сдвиг вправо с замещением
<<=	Сдвиг влево с замещением

Побитовое «исключающее или»

a	b	a ^ b
0	0	0
1	0	1
0	1	1
1	1	0

Примеры использования поразрядных логических операций

Поразрядное логическое умножение (И):

12 & 22 = 4

$12_{10} = 00001100_2$

&

$22_{10} = \underline{00010110_2}$

$00000100_2 = 4_{10}$

Поразрядное логическое сложение (ИЛИ):

12 | 22 = 30

$12_{10} = 00001100_2$

|

$22_{10} = 00010110_2$

$\overline{00011110_2} = 30_{10}$;

Сложение по модулю 2 (исключающее ИЛИ):

12 ^ 22 = 26

$12_{10} = 00001100_2$

^

$22_{10} = 00010110_2$

$\overline{00011010_2} = 26_{10}$;

Логические сдвиги:

2 << 4 = 32

$2_{10} = 0000\ 0010_2$

$32_{10} = 0010\ 0000_2$

32 >> 2 = 8

$32_{10} = 0010\ 0000_2$

$8_{10} = 0000\ 1000_2$

1. Остаток от деления.

С помощью поразрядной операции & можно найти остаток от деления числа x на 2^k:

x & mask;

где mask = 2^k -1.

X & 0x07 (остаток деления на 8)
X & 0x0f (остаток деления на 16)

2. Проверка наличия 1 в любом разряде.

С помощью поразрядной операции & можно проверить наличие единицы в k разряде числа x:

x & mask;

где mask = 2^k.

проверить наличие единицы в бите №4 x & 16 == 16

проверить наличие единиц в битах №4, №2, №1 x & 22 == 22

3. «Выключение бита».

С помощью поразрядной операции & можно «выключить» (обнулить) k бит числа x (при неизменном значении других битов):

x & mask;

где mask = -2^k +255 (для беззнакового типа операндов) или

-2^k - 1 (для знакового типа операндов)

выключить бит №2 x & 251 (11111011_2)

выключить бит №2 x & -5 (11111011_2)

4. «Включение бита».

С помощью поразрядной операции | можно «включить» (занести 1) k бит числа x (при неизменном значении других битов):

x | mask;

где mask = 2^k.

включить бит №2 x | 4;

включить биты №4, №2, №1, №0 x | 23;

5. «Инвертировать бит».

С помощью поразрядной операции ^ можно инвертировать k бит числа x (при неизменном значении других битов):

x ^ mask;

где mask = 2^k.

инвертировать бит №2 x ^ 4;

инвертировать биты №4, №2, №1, №0 x ^ 23;

ОПЕРАЦИИ C++ В ПОРЯДКЕ УБЫВАНИЯ ПРИОРИТЕТОВ

Операция	Назначение	Использование
::	область видимости	::name class::name namespace::name
. -> [ ] ( )	селектор компонента класса доступ к члену класса по указателю индексирование вызов функции	object.member pointer->member variable[exp] function()
++ -- sizeof sizeof ++ -- ~ ! + - * & () new new[] delete delete[] ->* .*	инкремент постфиксный декремент постфиксный размер объекта размер типа префиксный инкремент префиксный декремент побитовое НЕ логическое НЕ унарный плюс унарный минус разыменование адрес приведение типа выделение памяти выделение памяти под массив освобождение памяти освобождение памяти из-под массива разыменование указателя на компонент класса по указателю разыменование указателя на компонент класса	lvalue++ lvalue -- sizeof expr sizeof(type) ++lvalue -- lvalue ~expr !expr -expr +expr *expr &expr (type)expr new type new type(exprlist) new type[] delete pointer delete [] pointer pointer->*pointer_m object.*pointer
* / %	умножение деление деление по модулю	expr1*expr2 expr1/expr2 expr1%expr2
+ -	сложение вычитание	expr1+expr2 expr1-expr2

<< >>	сдвиг влево сдвиг вправо	expr1<<expr2 expr1>>expr2
< <= > >=	меньше меньше или равно больше больше или равно	expr1<expr2 expr1<=expr2 expr1>expr2 expr1>=expr2
== !=	равно не равно	expr1==expr2 expr1 !=expr2
&	побитовое И	expr1&expr2
^	побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ	expr1^expr2
|	побитовое ИЛИ	expr1|expr2
&&	логическое И	expr1&&expr2
||	логическое ИЛИ	expr1||expr2
?:	условная операция	expr1 ?expr2 :expr3
= *= /= %= += -= >>= <<= &= |= ^=	присваивание составное присваивание (относительный порядок приоритетов соответствует порядку приоритетов оераций без прсваивания)	Lznach=expr Lznach*=expr Lznach/=expr Lznach%=expr Lznach+=expr Lznach-=expr Lznach>>=expr Lznach<<=expr Lznach&=expr Lznach|=expr Lznach^=expr
,	запятая	expr,expr

НЕЯВНЫЕ ПРЕОБРАЗОВАНИЯ ТИПОВ: ИЕРАРХИЯ ТИПОВ

Тип данных	старшинство
long double (в MS VS совпадает с double)	Высший
double	
float	
unsigned long (в MS VS совпадает с unsigned int)	
long (в MS VS совпадает с int)	
unsigned int	
int	
short (short int)	
unsigned char	
char	
bool	Низший

ЯВНЫЕ ПРЕОБРАЗОВАНИЯ ТИПОВ

Явные преобразования типов:

- Для явного приведения типов в C++ можно использовать операцию static_ cast:
  Результат = static_cast <*Требуемый_тип*> (*Аргумент*);
- Старый синтаксис:
  Результат = (*Требуемый_тип*) *Аргумент* ;
  или
  Результат = *Требуемый_тип* (*Аргумент*) ;

Например:

```
CharVar = static_cast <char> (IntVar);
CharVar = (char) IntVar;
CharVar = char (IntVar);
```

ЗАГОЛОВКИ В ПРОГРАММАХ НА С++

При использовании библиотечных функций в программах на С необходимо включать соответствующие заголовочные файлы. Это делается с помощью инструкции *#include*.

Например:

```
// iostream.h – имя файла, который используется для ввода-вывода
   #include <iostream.h>

// math.h имя файла, который необходим для использовния математических функций
   #include <math.h>
```

В С++ можно использовать этот же способ. Вместе с тем, в С++ вместо имен заголовочных файлов можно указывать стандартные идентификаторы, по которым компилятор находит соответствующие файлы. Новые заголовки являются абстракциями, гарантирующими объявление соответствующих прототипов и определений библиотеки. Поскольку новые заголовки не являются именами файлов, для них не нужно указывать расширение *.h*:.

```
#include <iostream>
#include <string>
```

С++ содержит всю библиотеку функций С. Однако С++ также определяет заголовки нового стиля вместо этих заголовочных файлов. В С++ к стандартным заголовкам С добавляется префикс *c* и удаляется расширение *.h*.

Например:

Вместо `#include <math.h>` - `#include <cmath>`
Вместо `#include <string.h>` - `#include <cstring>`

В случае включения в программу заголовка нового стиля, содержание этого заголовка включается в пространство имен *std*.

Пространство имен (namespace) - область, необходимая, чтобы избежать конфликтов имен идентификаторов. В С имена библиотечных функций располагаются в глобальном пространстве имен. Содержание заголовков в С++ помещается в пространство имен *std*. Чтобы пространство имен стало видимым необходимо записать

```
using  namespace std;
```

Таким образом, вместо `#include <iostream.h>` будем записывать

```
#include <iostream>
using  namespace std;
```

КОНСОЛЬНЫЙ ВВОД/ВЫВОД В С++

В С++ ввод/вывод выполняется с использованием операторов, а не функций.

Оператор вывода <<, оператор ввода >>.

Вывод на экран строки:

```
cout << " x = \n";
```

Вывод на экран числовой величины

```
cout << 100.45;
```

Вывод на экран значения переменной:

```
cout << x;
```

Ввод с клавиатуры значения для переменной x:

```
cin >> x;
```

Для правильного использования операторов ввода/вывода в С++ необходимо включить в программу

```
#include <iostream>
using  namespace std;
```

НЕКОТОРЫЕ ПОЛЕЗНЫЕ ВСТРОЕННЫЕ ФУНКЦИИ

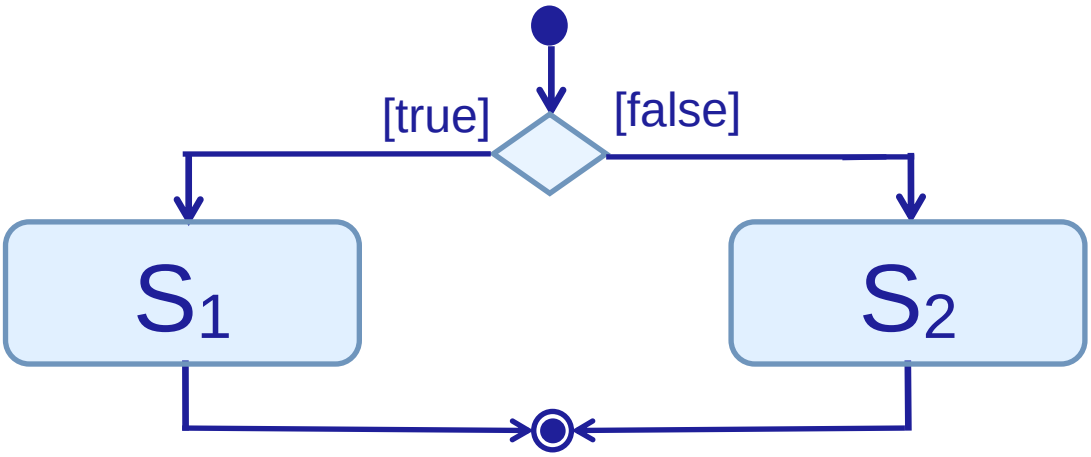
№	Функция	Запись	Описание	Необходимый заголовочный файл
1	abs	int abs(int x)	Абсолютное значение	math.h
2	acos	double acos(double x)	Арккосинус	math.h
3	asin	double asin(double x)	Арксинус	math.h
4	atan	double atan(double x)	Арктангенс	math.h
5	ceil	int ceil(extended x)	Округление вверх	math.h
6	cos	double cos(double x)	Косинус	math.h
7	cosh	double cosh(double x)	Косинус гиперболический	math.h
8	exp	double exp(double x)	Экспонента	math.h
9	fabs	double fabs(double x)	Абсолютное значение	math.h
10	floor	int floor(extended x)	Округление вниз	math.h
11	fmod	double fmod(double x, double y)	Остаток от деления x на y	math.h
12	labs	long labs(long x)	Абсолютное значение	math.h
13	log	double log(double x)	Логарифм натуральный	math.h
14	log10	double log10 (double x)	Логарифм десятичный	math.h
15	modf	double modf(double x, * double y)	Вычисление целой и дробной части числа x	math.h
16	pow	double pow(double x , double y)	Возведение x в степень y	math.h
17	rand	int rand(void)	Генерация случайного числа	stdlib.h
18	sin	double sin(double x)	Синус	math.h
19	sinh	double sinh(double x)	Синус гиперболический	math.h
20	srand	void srand(unsigned n)	Возможность генерировать случайные числа, начиная с заданного	stdlib.h
21	sqrt	double sqrt(double x)	Квадратный корень	math.h
22	tan	double tan(double x)	Тангенс	math.h

НЕКОТОРЫЕ ПОЛЕЗНЫЕ МАТЕМАТИЧЕСКИЕ КОНСТАНТЫ

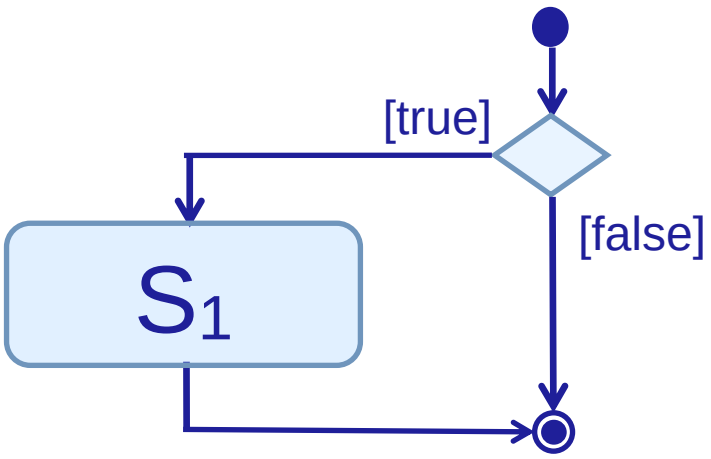
```
#define _USE_MATH_DEFINES
#include <math.h>              // #include <cmath>
```

Symbol	Expression	Value
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830961 6
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

ВЕТВЛЕНИЯ




```
if (Условие)
    инструкция1;
else
    инструкция2;
```




```
if (Условие)
    инструкция1;
```

ВЛОЖЕННЫЕ УСЛОВНЫЕ ИНСТРУКЦИИ

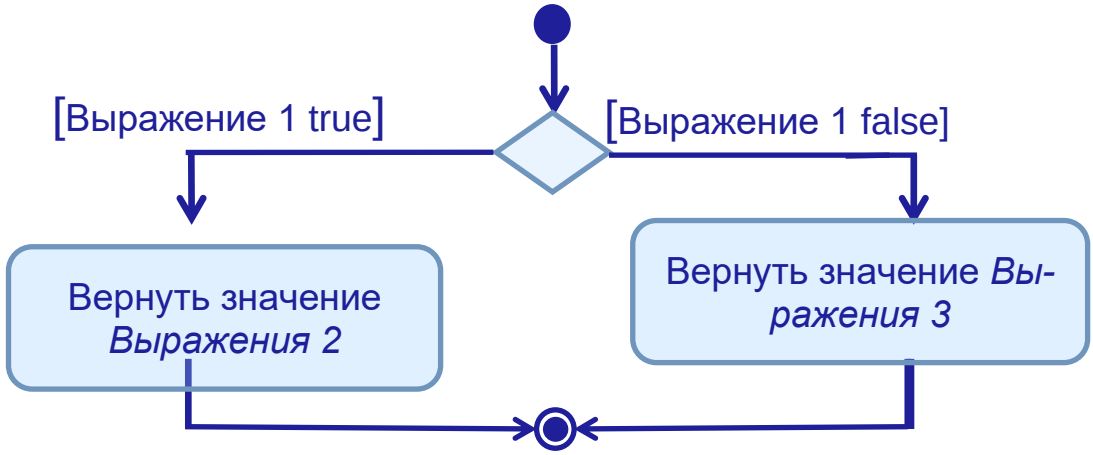
Во вложенных условных инструкциях раздел else всегда связан с ближайшей предшествующей инструкцией if, находящейся с ним в одном блоке и несвязанной с другим разделом else.

```
int main()
{
…

  if (i)
  {
      if (j) инструкция 1;
      if (k) инструкция 2;
      else   инструкция 3;
  }
  else инструкция 4;
…
}
```

```
int main()
{
…

  if (Выражение) инструкция;
  else
      if (Выражение) инструкция;
      else
          if (Выражение) инструкция;
          …
          else инструкция;
…
}
```

ТЕРНАРНЫЙ УСЛОВНЫЙ ОПЕРАТОР ?

Выражение1 ? *Выражение2* : *Выражение3*;



Вместо
```
x = 10;
if (x > 9) y = 100;
else y = 200;
```

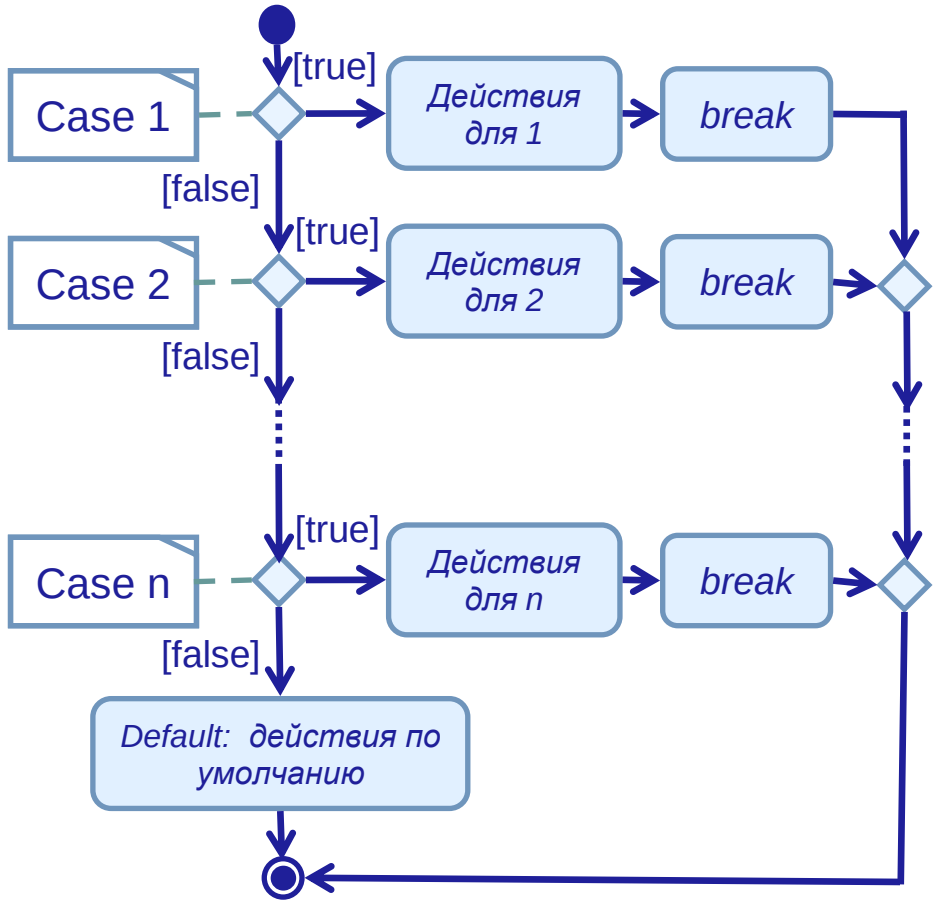
можно
```
x = 10;
y = x > 9 ? 100 : 200;
```

Вместо
```
x = 10;
if (x > 9)
   { y = 100; z = 200;}
else
   {y = 200; z = 100;}
```

можно
```
x = 10;
x > 9 ? (y = 100,z = 200) : (y = 200, z = 100);
```

ИНСТРУКЦИЯ SWITCH

```
switch (Выражение)
{
    case константа1:
        последовательность инструкций;
        break;
   case константа2:
        последовательность инструкций;
        break;
   case константа3:
        последовательность инструкций;
        break;
                .
                .
                .
    default
         последовательность инструкций;
}
```



ПРИМЕР 1. ПЕРВАЯ ПРОГРАММА

Вывод на экран текста “ my first program”.

```
#include <iostream>                    //  cout
using namespace std;

void main()
{
     cout<<" my first program "<<endl;

}


                                         // или такая
/*
#include <iostream>                    //  cout
using namespace std;

int main()
{
     cout<<" my first program "<<endl;
     return 0;

}
*/
```

ПРИМЕР 2. ПРОСТЕЙШИЙ ВВОД-ВЫВОД

```
#include <iostream>                    //  cout
using namespace std;

void main()
{
    int x;          // объявление переменной x целого типа
    cout<<" 1 x = "<<x<<endl; // вывод значения переменной x

    cout<<" x = ";
    cin>>x;                     // ввод целочисленного значения
    cout<<" 2 x = "<<x<<endl; // вывод значения переменной x

    x = 12345;
    cout<<" 3 x = "<<x<<endl; // вывод значения переменной x

    double y;  // объявление переменной y вещественного типа
    cout<<" 4 y = "<<y<<endl; // вывод значения переменной y

    cout<<" y = ";
    cin>>y;                     // ввод вещественного значения
    cout<<" 5 y = "<<y<<endl; // вывод значения переменной y

    y = 12.345;
    cout<<" 6 y = "<<y<<endl; // вывод значения переменной y

    y = 0.12345e2;
    cout<<" 7 y = "<<y<<endl; // вывод значения переменной y

}
```

ПРИМЕР 3. ПРИОРИТЕТ ОПЕРАЦИЙ

Определим приоритет операций в арифметическом выражении без скобок.

```
#include <iostream>                    // cout
using namespace std;

void main()
{
     int x;              // объявление переменной x целого типа

     x = -3 + 4 * 5 - 6;
     cout<<" 1 x = "<<x<<endl;                //  11


     x = -3 + 4 % 5 - 6;
     cout<<" 2 x = "<<x<<endl;               //   -5


     x = -3 * 4 % - 6 / 5;
     cout<<" 3 x = "<<x<<endl;               //   0


     x = (7 + 6) % 5 / 2 ;
     cout<<" 4 x = "<<x<<endl;               //   1
}
```

ПРИМЕР 4. ВЫЧИСЛЕНИЕ МОДУЛЯ ЧИСЛА

```
#include <iostream>          //  cin  cout
#include <math.h>            //  abs  fabs  cabs

using namespace std;

void main()
{
                      //  модуль целого числа
                      // функция  int  abs (int x );
    int x1, y1;
    cout<<" int x= ";
    cin>>x1;
    y1 = abs(x1);
    cout<< " abs("<<x1<<") = "<< y1 << endl;

                   // модуль вещественного числа
                   // функция  double fabs(double x );
    double x2,y2;
    cout<<" double x= ";
    cin>>x2;
    y2 = fabs(x2);
    cout<<" abs("<< x2 <<") = "<< y2 <<endl;

                  //  модуль комплексного числа z
                  // функция double cabs( _complex z );
                  //  d= sqrt( z.x*z.x + z.y*z.y ).
    _complex z;
    double d;
    cout<<" double z.x= "<<endl;
    cin>>z.x;
    cout<<"  double z.y="<<endl;
    cin >> z.y;
    d = _cabs( z );
    cout<<" absolute value "<< z.x
        << "+ i"<< z.y << "  ="<< d <<endl ;

}
```

ПРИМЕР 5. ВЫЧИСЛЕНИЕ АРКТАНГЕНСА ЧИСЛА

```
#include <iostream>       //  cin  cout
#include <math.h>         //  atan  atan2

using namespace std;

void main()
{
     double x, y, x1, x2;

               //  вычисление arctg(x)
               // функция  double atan( double x );
               //  возвращает значения от -Pi/2 до Pi/2 радиан;
               //  если x=0, функция возвращает 0.

     cout << "x=";
     cin >> x;
     y = atan(x);
     cout<<" arctg ("<<x<<")="<<y<<endl;

          //  вычисление arctg(y/x)
          //  функция double atan2( double y, double x );
          //  atan2 возвращает значения от -Pi до Pi радиан;
          //  если x=0 и y=0, функция возвращает 0.

     cout << "x1=";
     cin >> x1;
     cout << "x2=";
     cin >> x2;
     y = atan2(x2,x1);
     cout << " arctg (" << x2 / x1<< ")=" << y << endl;
}
```

ПРИМЕР 6. ВЫЧИСЛЕНИЕ АРКСИНУСА И АРККОСИНУСА ЧИСЛА

```
#include <iostream>     // cin cout
#include <math.h>       // asin

using namespace std;

void main()
{
     double x, y;

           //  вычисление арксинуса   для -1.0 <= x <= 1.0
           // функция  double asin( double x );
           //  возвращает значения от -Pi/2 до Pi/2 радиан;

     cout << "x=";
     cin>>x;
     y = asin(x);
     cout << " asin (" << x << ")=" << y << endl;

          //  вычисление арккосинуса для -1.0 <= x <= 1.0
          //  функция  double acos( double x );
          //  возвращает значения от 0 до Pi радиан;

     cout << "x=";
     cin >> x;
     y = acos(x);
     cout << " acos (" << x << ")=" << y << endl;
}
```

ПРИМЕР 7. ПЕРЕВОД ВЕЛИЧИНЫ УГЛА В ГРАДУСЫ

```
#include <iostream>          // cin cout

using namespace std;

void main()
{
     double x, y, pi = 3.14159265358979;

               //  перевод величины угла из радиан в градусы

     cout << "x=";
     cin >> x;
     y = x * 180 / pi;
     cout<< x << " radian = " << y << "  gradusov " << endl;
}
```

ПРИМЕР 8. ПЕРЕВОД ВЕЛИЧИНЫ УГЛА В РАДИАНЫ

```
#include <iostream>          // cin cout

using namespace std;

void main()
{
     double x, y, pi = 3.14159265358979;

          //  перевод величины угла из градусов в радианы

     cout << "x=";
     cin >> x;
     y = x * pi / 180;
     cout<< x << " gradusov = " << y << "  radian " <<endl;
}
```

ПРИМЕР 9. ВЫЧИСЛЕНИЕ ЭКСПОНЕНТЫ

```
#include <iostream>          // cin cout
#include <math.h>            // exp

using namespace std;

void main()
{
     double x, y;

//   вычисление экспоненты функция  double exp( double x );

     cout << "x=";
     cin >> x;
     y = exp(x);
     cout << " exp (" << x << ")= " << y << endl;

}
```

ПРИМЕР 10. ВОЗВЕДЕНИЕ В СТЕПЕНЬ

```
#include <iostream>        // cin cout
#include <math.h>          // pow

using namespace std;

void main()
{
     double x, y, z;

          //  Возведение x в степень y
          //  функция  double pow( double x,double y );
          //  Если  x<>0, то  y - любое, при y=0 результат=1
          //  Если  x=0,  то y>=0, при y=0 результат=1

     cout << "x=";
     cin >> x;
     cout << "y=";
     cin >> y;

     z = pow(x,y);
     cout << x << " in step " << y << " =" << z << endl;
}
```

ПРИМЕР 11. ВЫЧИСЛЕНИЕ ЛОГАРИФМОВ

```
#include <iostream>       //  cin cout
#include <math.h>         //  log  log10

using namespace std;

void main()
{
     double x, y;

              //  вычисление натурального логарифма
              //  функция  double log( double x );
              //  x>0

     cout << "x=";
     cin >> x;
     y = log(x);
     cout << " ln (" << x << ")= " << y << endl;

              //  вычисление десятичного логарифма
              //  функция  double log10( double x );
              //  x>0

     cout << "x=";
     cin >> x;
     y = log10(x);
     cout << " log10 (" << x << ")= " << y << endl;


             //  вычисление логарифма по основанию n; x>0 n>1
     unsigned n;
     cout << "x=";
     cin >> x;
     cout << "n=";
     cin >> n;
     y = log(x) / log(n);
     cout<< " log " << n << " (" << x << ")= " << y <<endl;

}
```

ПРИМЕР 12. Вычисление логарифмов

Вычислить $$z = \frac{\log_2 x + \log_b y}{\log_{b+2}(x+y)}$$

```
#include <iostream>           // cin cout
#include <math.h>             // log

#define lg(z,a) log(z)/log(a)

using namespace std;

void main()
{
     double x, y, z, b;


     cout << "x = ";
     cin >> x;
     cout << "y = ";
     cin >> y;
     cout << "b = ";
     cin >> b;

     z = (lg(x, 2.0) + lg(y, b))/ lg(x + y, b + 2);

     cout << " z = " << z << endl;


}
```

ПРИМЕР 13. ПОЛУЧЕНИЕ ЗНАЧЕНИЯ ЧИСЛА ПИ

```
#include <iostream>     // cin cout
#include <math.h>       // asin acos

using namespace std;


void main()
{
    double pi;

                          //  вычисление значения числа Pi
    pi = acos(-1.0);
    cout.precision(15);
    cout<<pi<<endl;

                          //  вычисление значения числа Pi
    pi = 2 * asin(1.0);
    cout.precision(15);
    cout<<pi<<endl;
}
```

ПРИМЕР 14. ВЫЧИСЛЕНИЕ СИНУСА И КОСИНУСА УГЛА

```
#include <iostream>     // cin cout
#include <math.h>       // sin sinh cos cosh
using namespace std;

void main()
{
     double x,y;
                    //  вычисление синуса
                    //  функция  double sin( double x );
     cout << "x=";
     cin >> x;
     y = sin(x);
     cout<< " sin ("<<x<<")= " << y << endl;

                    //  вычисление синуса гиперболического
                    // функция  double sinh( double x );
     cout << "x=";
     cin >> x;
     y = sinh(x);
     cout << " sinh (" << x << ")= " << y << endl;

                    //  функция  double cos( double x );
     cout << "x=";
     cin >> x;
     y = cos(x);
     cout << " cos (" << x << ")=" << y << endl;

                    //  вычисление косинуса гиперболического
                    //  функция  double cosh( double x );
     cout << "x=";
     cin >> x;
     y = cosh(x);
     cout<<" cosh ("<<x<<")="<<y<<endl;
}
```

ПРИМЕР 15. ВОЗВЕДЕНИЕ В КВАДРАТ ЧИСЛА

```
#include <iostream>  // cin cout

using namespace std;

void main()
{
     double x, y;

                    //  Возведение в квадрат числа
     cout << "x=";
     cin >> x;
     y = x * x;
     cout<< " sqr (" << x << ")= " << y <<endl;

}
```

ПРИМЕР 16. ИЗВЛЕЧЕНИЕ КВАДРАТНОГО КОРНЯ

```
#include <iostream>       //  cin cout
#include <math.h>         //  sqrt

using namespace std;

void main()
{
     double x, y;

          //  вычисление корня квадратного
          //   функция  double sqrt( double x );  x>0

     cout << "x=";
     cin >> x;
     if (x >= 0)
     {
          y = sqrt(x);
          cout << " sqrt (" << x << ")= " << y << endl;
     }
}
```

ПРИМЕР 17. ИНКРЕМЕНТ АРГУМЕНТА

```
#include <iostream>  // cin cout

using namespace std;

void main()
{
    int x, n;

    cout<<"x=";  cin>>x;

                //    инкремент аргумента x на 1
    x++;                          //     ++x;
    cout<<x<<endl;

    cout<<"n=";  cin>>n;

                      // инкремент аргумента x на n
    x = x + n;
    cout<<x<<endl;

                      // инкремент аргумента x на n
    x += n;
    cout<<x<<endl;
}
```

ПРИМЕР 18. ДЕКРЕМЕНТ АРГУМЕНТА

```
#include <iostream>  // cin cout

using namespace std;

void main()
{
     int x,  n;

     cout << "x=";
     cin >> x;

                                  // декремент аргумента x на 1
     x--;
     cout << x << endl;

     cout<<"n=";
     cin >> n;
                                  // декремент аргумента x на n
     x = x - n;
     cout << x << endl;

                                   // декремент аргумента x на n
     x -= n;
     cout << x << endl;
}
```

ПРИМЕР 19. ФОРМАТИРУЕМЫЙ ВЫВОД В С++

```
#include <iostream>                    //  cout
using namespace std;

void main()
{
     int i = 456;
     double x = 24.6782;
     bool b = true;
     char c = 66;
     cout << -125 / 2 << endl;      // -62
     cout << 5.6e12 * 2 << endl;        // 1.12e+013
     cout << !true << endl;         // 0
     cout << i << endl;             // 456
     cout << x << endl;             // 24.6782
     cout << b << endl;             // 1
     cout << c << endl;             // B


                    // Please,
                    // enter the value:
  cout<<"Please,\nenter the value: \n";
  cout<< "Please,"  "\n"  "enter the value: \n";
  cout << "Please," ;
  cout<< "\n" ;
  cout << "enter the value: ";
  cout<< "\n" ;
  cout<< "Please" <<endl<<"enter the value: " <<endl;
  cout << "Please";
  cout<< endl;
  cout << "enter the value: "<<endl;

                  // Hello, ☼
                  //'☼' соответствует код ASCII 1510=0F16
  cout << "Hello, \x0F \n";
  cout << "Hello, "   " \x0F \n";
  cout  << "Hello, "  << " \x0F "<<endl;


 }
```

ПРИМЕР 20. Явное преобразование типов данных

Рассмотрим результат оного и того же арифметического выражения в зависимости от типов операндов.

```
#include <iostream>                    // cout
using namespace std;

void main()
{
    int    x; // объявление переменной x целого типа
    double y; // объявление переменной y вещественного типа

                //  1   результат заносим в x – целого типа
                //  в y – вещественного типа
    x = 1 + 2 – 3 * 4 / (5 % 6) ;
    y = 1 + 2 – 3 * 4 / (5 % 6) ;
    cout<<" 1 x = "<<x<<"  y = "<<y<<endl;     //  1  1

                  /* 2  явное преобразование типа в C
                       относится только к константе 1 */
    x = (double)1 + 2 – 3 * 4 / (5 % 6);
    y = (double)1 + 2 – 3 * 4 / (5 % 6);
    cout<<" 2 x = "<<x<<"  y = "<<y<<endl;      //  1  1

                 /* 3  явное преобразование типа в C
                       относится только к вычисленному
                       результату */
    x = (double) (1 + 2 – 3 * 4 / (5 % 6));
    y = (double) (1 + 2 – 3 * 4 / (5 % 6));
    cout<<" 3 x = "<<x<<"  y = "<<y<<endl;     //  1   1

                  /* 4  явное преобразование типа в C
                относится  к каждой константе выражения  */
     x = (double)1 + (double)2 – (double)3 *
         (double)4 / double(5 % 6);
     y = (double)1 + (double)2 – (double)3 *
         (double)4 / double(5 % 6);
    cout<<" 4 x = "<<x<<"  y = "<<y<<endl;      //  0  0.6

                  /*  5  явное преобразование типа в C
                    относится  только к  константе 4   */
    x = 1 + 2 – 3 * (double)4 / (5 % 6);
    y = 1 + 2 – 3 * (double)4 / (5 % 6);
    cout<<" 5 x = "<<x<<"  y = "<<y<<endl;    //  0  0.6
```

```
                    /*6  явное преобразование типа в C++
                      относится  только к  константе 4   */
    y = 1 + 2 - 3 * static_cast<double>(4) / (5 % 6);
    y = 1 + 2 - 3 * static_cast<double>(4) / (5 % 6);
    cout<<" 6 x = "<<x<<"  y = "<<y<<endl;      //  0  0.6

}
```

ПРИМЕР 21. НЕЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ (ЦЕЛЫЕ И ВЕЩЕСТВЕННЫЕ)

Рассмотрим результат оного и того же арифметического выражения в зависимости от типов операндов.

```
#include <iostream>                    // cout
using namespace std;

void main()
{
     int    x; // объявление переменной x целого типа
     double y; // объявление переменной y вещественного типа

     x = 1 + 2 - 3 * 4 / (5 % 6) ;
     /*  1   сначала  вычисляем выражение над целыми
     операндами, результат получается целого типа, затем
     неявно преобразуется результат  к вещ. типу  */
     y = 1 + 2 - 3 * 4 / (5 % 6) ;
     cout<<" 1 x = "<<x<<"  y = "<<y<<endl;     //  1  1

            /*   2    вычисляем выражение над  операндами
              разных типов - неявное преобразование типов  */
     x = 1. + 2. - 3. * 4. / (5 % 6);
     y = 1. + 2. - 3. * 4. / (5 % 6);
     cout<<" 2 x = "<<x<<"  y = "<<y<<endl;     //  0  0.6

                                    //   3  только 4.0
     x = 1 + 2 - 3 * 4. /(5 % 6);
     y = 1 + 2 - 3 * 4. /(5 % 6);
     cout<<" 3 x = "<<x<<"  y = "<<y<<endl;     //  0  0.6

                                      // 4 только 3.0
     x = 1 + 2 - 3. * 4 / (5 % 6);
     y = 1 + 2 - 3. * 4 / (5 % 6);
      cout<<" 4 x = "<<x<<"  y = "<<y<<endl;     //  0  0.6


        // 5   1.0 и 2.0  недостаточно для результата 0.6!!!
     x = 1 + 2 - 3. * 4 / (5 % 6);
     y = 1. + 2. - 3 * 4 / (5 % 6);
     cout<<" 5 x = "<<x<<"  y = "<<y<<endl;     //  0  1


}
```

ПРИМЕР 22. НЕЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ (SIGNED CHAR И UNSIGNED CHAR)

```
#include <iostream>                    // cout
using namespace std;

void main()
{
  char c = 0;
  unsigned char cs = 239;
  int i = cs;

  cout << cs <<"   " << i << endl;          // я  239
  i = c = cs;
  cout << c << "   " << i << endl;          // я  -17


  c = -30;
  cs = 0;
  i = c;
  cout << c << "   " << i << endl;         // т  -30
  i = cs = c;
  cout << cs << "   " << i << endl;        // т  226

  }
```

ПРИМЕР 23***. НЕЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ (FLOAT SHORT DOUBLE)

```
#include <iostream>                    // cout
using namespace std;

 void main()
 {
    short i = 0;
    float r = -12.9869e2;
    cout << r << endl;                  // - 1298.69
    i = r;
    cout << i <<"   " <<  endl;         // - 1298

    r = -123e8;
    cout << r << endl;                  // - 1.23e+10

    i = r;
    cout << i <<"   " <<  endl;         // -7168

    i = 123e14;
    cout << i <<"   " <<  endl;         // - 16384

    r = 0;
    double d = -123e14;
    cout << d << endl;                 // -1.23e+016

    r = d;
    cout << r <<"   " <<  endl;         // -1.23e+016

    d = -123e112;
    cout << d << endl;                 // -1.23e+114

    r = d;
    cout << r <<"   " <<  endl;       // -1.#INF ( HUGE_VAL )

    r = 123e-104;
    cout << r <<"   " <<  endl;       // 0

 }
```

ПРИМЕР 24. БИНАРНЫЕ ОПЕРАТОРЫ

```
#include <iostream>                    // cout
using namespace std;

void main()
{
     int X, Y;

                              //      X = X + 2
     X = 4;      X += 2;
     cout<<"X + 2 = "<<X<<endl;

     cout<<"X = ";  cin>>X;
     X += 2;
     cout<<"X + 2 = "<<X<<endl;


                             //      X = X + 1     Y = X
     X = 4;      Y = X ++;
     cout<<"Y = X++ ;       X = "<<X<< " Y =  "<<Y<<endl;

     cout<<"X = ";  cin>>X;
     Y = X++;
     cout<<"Y = X++ ;  X = "<<X<< " Y = "<<Y<<endl;


                             //   Y = X     X = X+1
     X = 4;      Y = ++X ;
     cout<<"Y = ++X ;  X = "<<X<<"  Y = "<<Y<<endl;

     cout<<"X = ";  cin>>X;
     Y=++X ;
     cout<<"Y = ++X ;  X = "<<X<<" Y =  "<<Y<<endl;


                             //    X = X - 2
     X = 4;    X -= 2;
     cout<<" 4 - 2 = "<<X<<endl;

     cout<<"X = ";  cin>>X;
     X -= 2;
     cout<<"X - 2 = "<<X<<endl;



                                 //      X = X * 2
```

```
    X = 4;      X *= 2;
    cout<<"4 * 2 = "<<X<<endl;

    cout<<"X = ";  cin>>X;
    X *= 2;
    cout<<"X * 2 = "<<X<<endl;


                              //   X = X / 2  (частное)
    X = 4;      X /= 2;
    cout<<"4 / 2 = "<<X<<endl;


    cout<<"X = ";  cin>>X;
    X /= 2;
    cout<<"X / 2 = "<<X<<endl;


                            //     X = X mod 2   (остаток)
    X = 4;       X %= 2;
    cout<<"4 % 2 = "<<X<<endl;

    cout<<"X = ";  cin>>X;
    X %= 2;
    cout<<"X % 2 = "<<X<<endl;

}
```

ПРИМЕР 25. ИНКРЕМЕНТАЦИЯ И ДЕКРЕМЕНТАЦИЯ

```
#include <iostream>                    // cout
using namespace std;

void main()
{
        int i = 2, j = 0;
        i++;
        cout << i << endl;                          // 3

        --i;
        cout << i << endl;                          // 2

        j = 5 + i--;
        cout << i <<"  "<< j << endl;               // 1     7

        i = 2;
        j = 5 + --i;
        cout << i <<"  "<< j << endl;                // 1     6

        int x = 2;
        x = ++x * 4;
        cout << x << endl;                           // 12

        x = 2;
        x = x++ * 4;
        cout << x << endl;                           //   9

  }
```

ПРИМЕР 26***. ОГРАНИЧЕННОСТЬ ДИАПАЗОНА ПРЕДСТАВИМЫХ ЦЕЛЫХ ЧИСЕЛ

```
#include <iostream>          //  cout
using namespace std;         // #include <limits.h>
void main()
{    unsigned char UCX, UCY;
     UCX = 127;                                  // CHAR_MAX
     UCY = UCX + 1;
     cout<<"1  "<<UCY<<"  "<<int(UCY)<<endl; // A  128

     UCX = 255;                                  // UCHAR_MAX
     UCY = UCX + 1;
     cout<<"2  "<<UCY<<"  "<<int(UCY)<<endl;  // 0

     char CX, CY;
     CX = 127;                                      // CHAR_MAX
     CY = CX + 1;
     cout<<"3  "<<CY<<"  "<<int(CY)<<endl; //A -128 CHAR_MIN

     unsigned short USX, USY;
     USX = 32767;                        // SHRT_MAX
     USY = USX + 1;
     cout<<"4  "<<USY<<endl;             // 32768

     USX = 65535;                        // USHRT_MAX
     USY = USX + 1;
     cout<<"5  "<<USY<<endl;             // 0

     short SX = 32767;                        // SHRT_MAX
     short SY = SX + 1;
     cout<<"6  "<<SY<<endl;              // -32768  SHRT_MIN

     unsigned int UIX, UIY;
     UIX = 2147483647;                   // INT_MAX
     UIY = UIX + 1;
     cout<<"7  "<<UIY<<endl;             // 2147483648

     UIX = 4294967295;                   // UINT_MAX
     UIY = UIX + 1;
     cout<<"8  "<<UIY<<endl;             // 0

     int IX, IY;
     IX = 2147483647;
     IY = IX + 1;
     cout<<"9  "<<IY<<endl;              // -2147483648  INT_MIN
}
```

ПРИМЕР 27. ПОБИТОВЫЕ (ПОРАЗРЯДНЫЕ) ОПЕРАТОРЫ

Поразрядные операции над переменными типа unsigned

```
#include <iostream>                    // cout
using namespace std;

void main()
{
    unsigned short int X;
                                          //   сдвиг вправо на 2
    X = 4;
    X >>= 2;
    cout<<"4 >> 2 = "<<X<<endl;          //   1

    cout<<"X = ";  cin>>X;
    X >>= 2;
    cout<<"X >> 2 = "<<X<<endl;

                                           //   сдвиг влево на 2
    X = 4;
    X <<= 2;
    cout<<"4 << 2 = "<<X<<endl;          //  16

    cout<<"X = ";  cin>>X;
    X <<= 2;
    cout<<"X << 2 = "<<X<<endl;

                                        // поразрядное отрицание
    X = 4;
    X =~ X;
    cout<<"~4 = "<<X<<endl;            //   65531

    cout<<"X = ";  cin>>X;
    X = ~X;
    cout<<" ~X = "<<X<<endl;

                                         // поразрядное и
    X = 4;
    X &= 2;
    cout<<"4 & 2 = "<<X<<endl;          // 0

    cout<<"X = ";  cin>>X;
    X &= 2;
    cout<<"X & 2 = "<<X<<endl;


                                          // поразрядное или
```

```
    X = 4;
    X |= 2;
    cout<<"4 | 2 = "<<X<<endl;          //  6


    cout<<"X = ";  cin>>X;
    X |= 2;
    cout<<"X | 2 = "<<X<<endl;

                          //   поразрядное исключающее или
    X = 4;
    X ^= 2;
    cout<<"4 ^ 2 = "<<X<<endl;           //    6

    cout<<"X = ";  cin>>X;
    X ^= 2;
    cout<<"X ^ 2 = "<<X<<endl;

    unsigned short y = 0x00FF;
    cout << hex << y << endl;          // ff
    y &= 0xAA80;
    cout << hex << y << endl;          // 80


}
```

ПРИМЕР 28. ПОБИТОВЫЕ (ПОРАЗРЯДНЫЕ) ОПЕРАТОРЫ

Поразрядные операции над переменными типа signed

```
#include <iostream>                    // cout
using namespace std;

void main()
{
     signed short int X;
                                        //   сдвиг вправо на 2
     X = 4;
     X >>= 2;
     cout<<"4 >> 2 = "<<X<<endl;       //   1

     X = -4;
     X >>= 2;
     cout<<"-4 >> 2 = "<<X<<endl;       //  -1

                                        //   сдвиг влево на 2
     X = 4;
     X <<= 2;
     cout<<"4 << 2 = "<<X<<endl;       // 16

     X = -4;
     X <<= 2;
     cout<<"-4 << 2 = "<<X<<endl;        //  -16

                                       // поразрядное отрицание
     X = 4;
     X =~ X;
     cout<<"~4 = "<<X<<endl;           //  -5

     X = -4;
     X = ~X;
     cout<<" ~-4 = "<<X<<endl;          //  3

                                        //    поразрядное и
     X = 3;
     X &= 2;
     cout<<"3 & 2 = "<<X<<endl;       //  2

     X = -3;
     X &= 2;
     cout<<"-3 & 2 = "<<X<<endl;       //  0

                                          //    поразрядное или
```

```
    X = 3;
    X |= 2;
    cout<<"3 | 2 = "<<X<<endl;          //  3

    X = -3 ;
    X |= 2;
    cout<<"-3 | 2 = "<<X<<endl;          //  -1

                                // поразрядное исключающее или
    X = 3;
    X ^= 2;
    cout<<"3 ^ 2 = "<<X<<endl;          //   1

    X = -3;
    X ^= 2;
    cout<<"-3 ^ 2 = "<<X<<endl;         //   -1


}
```

ПРИМЕР 29***. ВЫДЕЛЕНИЕ БИТ В ЧИСЛЕ

От прибора в ЭВМ поступает 16-разрядный код (справа налево номера от 0 до 15). Допустим, в разрядах 0-4 находится информация от первого датчика, 5-11 – от датчика 2, с 12 по 14 – от датчика 3. Рассмотрим алгоритм выделения информации от каждого из датчиков.

```
#include <iostream>
using namespace std;

void main()
{
  short X, XX;          //XX – исходная  информация
  unsigned char x1, x2, x3;
             //x1, x2, x3 – выделенная информация от датчиков
  //cin>>XX;
  XX = 0xf00f;
  cout<<hex<<XX<<endl;

  X = XX;
  x1 = X & 31;
    //выделяются разряды с 0 по 4 :   31(10) = 0001 1111 (2)
  cout<<hex<<int(x1)<<endl;

  X = X >>5;
              //происходит сдвиг X на 5 разрядов вправо
  x2 = X & 127;
                 //выделяются очередные 7 разрядов
                 // (бывшие разряды 5-11)
                //127 (10) = 7F (16) = 0111 1111 (2)
  cout<<hex<<int(x2)<<endl;

  X = X >>7;
    // производится сдвиг вправо очередных 7-и разрядов
  x3 =X & 7;
      // выделяются младшие 3 разряда (бывшие 12 - 14)
 cout<<hex<<int(x3)<<endl;
}
```

ПРИМЕР 30*. ОПЕРАТОР =**

```
#include <iostream>                    // cout
using namespace std;

void main()
{
     int X = 2, Y = 2,  Z = 2;

     X *= 3 + 2;
     cout<<" X = "<<X<<endl;              // X=10

     X *= Y = Z = 4;
     cout<<" X = "<<X<<" Y = "<<Y<<" Z = "<<Z<<endl;
                                          // X=40   Y=4   Z=4

     X = Y == Z;
     cout<<" X = "<<X<<" Y = "<<Y<<" Z = "<<Z<<endl;
                                          // X=1   Y=4   Z=4

     X == (Y = Z);
     cout<<" X = "<<X<<" Y = "<<Y<<" Z = "<<Z<<endl;
                                           // X=1 Y=4   Z=4

     int i = 0;
     double x = 0;
     x =(i = 5.25) + 6.5;
     cout << i << "   " << x << endl;       // 5     11.5

     int ii = 0;
     float xf = 0, yf = 0;
     bool b = false;
     xf = b = ii = yf = 4.567;
     cout<<yf<<"  "<<ii<<"  "<<b<<"  "<<xf<<endl;
                                          // 4.567  4  1  1

     float fx = 0, fy = 4.5, fz = 3.2;
     fx = (fy = fz) = 6.5;
     cout<<" fz = "<<fz<<" fy = "<<fy<<" fx = "<<fx<<endl;
                                    // fz=3.2  fy=6.5  fx=6.5
}
```

ПРИМЕР 31***. ОПЕРАТОР СРАВНЕНИЯ

```
#include <iostream>                    // cout
using namespace std;

void main()
{
    int  i = 2, j = -3, k = 5, l = 1;
    char c1 = 'S', c2 = 'U';
    bool b;

    b = i > 3;
    cout << "1 b= "<< b << endl;                  // 0

    b = i >=3;
    cout << "2 b= "<< b << endl;                  // 0

    b = i < 3;
    cout << "3 b= "<< b << endl;                  // 1

    b = k !=2;
    cout << "4 b= "<< b << endl;                  // 1

    b = j == i - k;
    cout << "5 b= "<< b << endl;                  // 1

    b = 'U' < 'S';
    cout << "6 b= "<< b << endl;                  // 0

    b = l == (k > i);
    cout << "7 b= "<< b << endl;                  // 1
}
```

ПРИМЕР 32***. СРАВНЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ НА ТОЧНОЕ РАВЕНСТВО

```
#include <iostream>                    //  cout
#include <float.h>
#include <math.h>

using namespace std;

void main()
{
    float x = 1.0f;
    x += 0.5f;
    bool b =(x == 1.5f);
    cout << x << "   " << b << endl;      //1.5   1(true)


    float y = 1.0f;
    y += 0.1f;
    b = (y == 1.1f);  // не работает - это число
                      // представляется в формате float
                      // с погрешностью
    cout << y << "   " << b << endl;      // 1.1   0(false)

    x = 1.0f;
    float z = 1.1f;
    x += 0.1f;
    b =(x == z);
    cout << x << "  " << b << endl;          // 1.1  1(true)
    cout<<x - 1.1<<"   "<<x - 1.1f<<endl;   //2.38419e-008 0
                                            // 1.1 != 1.1f

    x = 1.0f;
    x += 0.1f;
    b = abs(x - 1.1) <= FLT_EPSILON;        //  Вот так !!!
    cout << x << "   " << b <<"   "<< FLT_EPSILON << endl;
                            // 1.1   1(true)   1.19209e-07
    x = 10.1f;
    b = abs(x - 10.1) <= FLT_EPSILON;   //  Не работает
                                    // порог выбран неверно
    cout << x << "   " << b << "   " << FLT_EPSILON << endl;
                            // 10.1   0(false)   1.19209e-07
```

```
    x = 10.0f;
    y = 10.3f;
    x += 0.1f;
    x += 0.1f; // не то же самое что x += 0.3f, потому что
    x += 0.1f; // погрешность округления накапливается

    b = abs(x - y) <= FLT_EPSILON;   //  Не работает
                                     // порог выбран неверно
    cout << x-y << "  " << b << "  " << FLT_EPSILON << endl;
                     // 9.53674e-07  0(false)  1.19209e-07
    float limit = FLT_EPSILON * fmax(abs(x), abs(y));
    b = abs(x - y) <= limit;   // Вот так!!!
            // порог выбран с учётом текущего значения x и y
    cout << x-y << "  " << b << "  " << limit << endl;
                     // 9.53674e-07  1(true)  1.22786e-06
}
```

ПРИМЕР 33*. НЕВЫПОЛНЕНИЕ АССОЦИАТИВНОГО ЗАКОНА СЛОЖЕНИЯ ПРИ ВЫЧИСЛЕНИЯХ С ПЛАВАЮЩЕЙ ТОЧКОЙ.**

```
#include <iostream>
#include <conio.h>
using namespace std;

void main()
{
    float x = 1.1e7, y = -1.1e7, z = 1.0e2, r, v;
    r = (x + y)+ z;
    cout <<r<<endl;                                  //100
    r = x +(y + z);
    cout <<r<<endl;                                  //100
    v = x + y;
    r = v + z;
    cout <<r<<endl;                                  //100
    v = y + z;
    r = x + v;
    cout << r<<endl<<endl;                           //100


    x = 1.1e8;      y = -1.1e8;      z = 1.0e2;
    r = (x + y) + z;
    cout <<r<<endl;                                  //100
    r = x + (y + z);
    cout <<r<<endl;                                  //100
    v = x + y;
    r = v + z;
    cout <<r<<endl;                                  //100
    v = y + z;
    r = x + v;
    cout <<r<<endl<<endl;                            //96


    x = 1.1e10;      y = -1.1e10;      z = 1.0e2;
    r = (x + y) + z;
    cout <<r<<endl;                                 //100
    r = x + (y + z);
    cout <<r<<endl;                                 //100
    v = x + y;
    r = v + z;
    cout <<r<<endl;                                 //100
    v = y + z;
    r = x + v;
    cout <<r<<endl<<endl;                           //0

    x = 1.1e22;    y = -1.1e22;    z = 1.0e2;
```

```
    r = (x + y) + z;
    cout <<r<<endl;                                    //100
    r = x+(y + z);
    cout <<r<<endl;                                    //0
    v = x + y;
    r = v + z;
    cout <<r<<endl;                                    //100
    v = y + z;
    r = x + v;
    cout <<r<<endl;                                    //0

}
```

ПРИМЕР 34. ВЫДЕЛЕНИЕ ЦИФР ЦЕЛОГО ЧИСЛА

Дано трехзначное натуральное n. Верно ли, что это число содержит ровно две одинаковые цифры?

```
#include <iostream>  // cin cout

using namespace std;

void main()
{    int   n, m, N,
           a, b, c, d;                          //  цифры

     cout<<" vvod n=";
     cin>>n;

     if ((n < 100) || (n >= 1000))
     {
          cout<<" error";
          return;
     }

                        // вариант 1 определения цифр числа n
     c = n % 10;                          // 3-ая цифра
     b = n / 10 % 10;                     //  2-ая цифра
     a = n / 100 ;                        //   1-ая  цифра
     cout << a << b << c << endl;

     N = int (a == b) + int(a == c) + int(b == c);

     if (N == 1 ) cout << "Yes" << endl;
     else         cout  << "No" << endl;

                         // вариант 2 определения цифр числа n
     m = n;
     c = m % 10;                          // 3-ая цифра
     m = m / 10;
     b = m % 10;                          // 2-ая цифра
     a = m / 10;                          // 1-ая цифра
     cout << a << b << c << endl;

     N=int (a == b) + int(a == c) + int(b == c);

     if (N == 1 ) cout << "Yes" << endl;
     else         cout  << "No"<< endl;
}
```

ПРИМЕР 35. ВЫДЕЛЕНИЕ ЦИФР ВЕЩЕСТВЕННОГО ЧИСЛА

Определить, есть ли среди первых двух цифр дробной части заданного положительного вещественного числа x цифра 7.

```
#include "stdafx.h"
#include <math.h>       // modf
#include <iostream>     // cin cout
using namespace std;

void main()
{    double x;
     int c, d;
                                        // 1 вариант
     cout<<" vvod x=";
     cin>>x;
     c = int(x * 10) % 10;               // 1-ая цифра
     d = int(x * 100) % 10;              // 2-ая цифра
     cout<<c<<d<<endl;
     if ((c ==7) || (d == 7)) cout<<"YES"<<endl;
     else cout<<"NO"<<endl;
                                        // 2 вариант
     cout<<" vvod x=";
     cin>>x;
     x = (x - int(x)) * 10;
     c = int(x);                        // 1-ая цифра
     x = (x - int(x)) * 10;
     d = int(x);                        // 2-ая цифра
     cout<<c<<d<<endl;
     if ((c == 7) || (d == 7)) cout<<"YES"<<endl;
     else cout<<"NO"<<endl;

                                        // 3 вариант
     cout<<" vvod x=";
     cin>>x;
     x += x * DBL_EPSILON;      // для избежания ошибки
     double y;
     x = modf(x,&y);
     x *= 10;
     c = int(x);                        // 1-ая цифра
     x = modf(x,&y);
     x *= 10;
     d = int(x);                        // 2-ая цифра
     cout<< c << d <<endl;
     if ((c == 7) || (d == 7)) cout<<"YES"<<endl;
     else cout<<"NO"<<endl;
}
```

ПРИМЕР 36. Логические выражения

Присвоить логической переменной b значение логического выражения, истинного при выполнении следующего условия и ложного в противном случае: целые *n* и *k* имеют разную четность.

```
#include <iostream>  // cin cout

using namespace std;

void main()
{
     int n, k;
     bool  d;

     cout<<"n= ";
     cin>>n;

     cout<<"k= ";
     cin>>k;

     d = (n  % 2 != k % 2);

     cout<<d<<endl;
}
```

ПРИМЕР 37. Логические переменные и выражения

Вычислить и вывести на экран результат логического выражения для заданных значений логических переменных a, b, c: $a \cdot \overline{b \cdot \overline{c}}$.

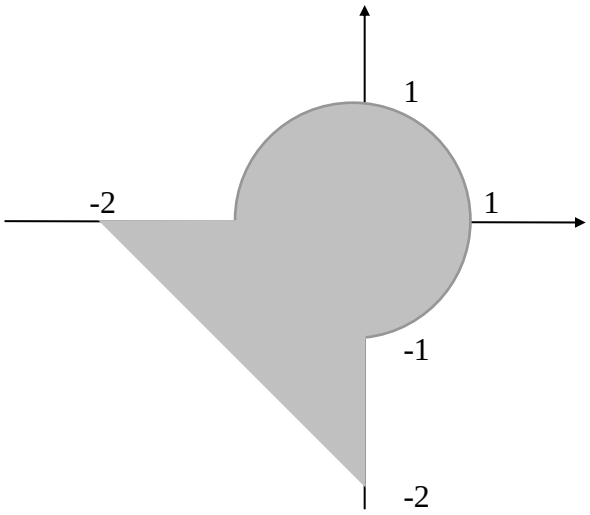
```
#include <iostream>  // cin cout

using namespace std;

void main()
{
    bool  a, b, c, d;

                                          // вариант 1
    a = true;
    b = true;
    c = false;                           // инициализация

    d = a && ( !(b && (!c)));
    cout<<d<<endl;


                                          // вариант 2
    cout<<"a= "; cin>>a;      //  ввод логической переменной
    cout<<"b= "; cin>>b;      //  ввод логической переменной
    cout<<"c= "; cin>>c;      //  ввод логической переменной

    d = a && ( !(b && (!c)));
    cout<<d<<endl;

                                           // вариант 3
    int  a1, b1, c1;
    cout<<"a= ";
    cin>>a1;
    cout<<"b= ";
    cin>>b1;
    cout<<"c= ";
    cin>>c1;

    d = bool(a1) && (!(bool(b1)  && (!bool(c1))));
    cout<<d<<endl;

}
```

ПРИМЕР 38. ОПИСАНИЕ ОБЛАСТИ НА ПЛОСКОСТИ

Написать программу, проверяющую, попадает ли точка в область. Область можно описать как круг, пересекающийся с треугольником.

Точка может попадать либо в круг, либо в треугольник, либо в их общую часть:

$\{x^2 + y^2 <= 1\}$ или $\{x \le 0$ и $y \le 0$ и $y \ge -x-2\}$




```
#include <iostream>  // cin cout

using namespace std;

void main()
{
    double  x, y;
    bool      p;

    cout  << " input x, y" << endl;
    cin >> x >> y;

     p = (x * x + y * y <= 1)
      || (x <= 0 && y <= 0 && y >= -x - 2);

    cout << p << endl;
}
```

ПРИМЕР 39. ОПРЕДЕЛЕНИЕ ОСТАТКА ОТ ДЕЛЕНИЯ

```
#include <math.h>
#include <iostream>  // cin cout

using namespace std;

void main()
{
     double x, y, xy;

     cout << "vvod x=";
     cin >> x;
     cout << "vvod y=";
     cin >> y;

     // определение остатка от деления через функцию fmod
     xy = fmod(x,y);
     cout << " ostatok ot deleniya x na y = " << xy << endl;


     // определение остатка от деления через оператор  %
     // если  это возможно
     xy = int(x) %  int(y);
     cout<<" ostatok ot deleniya x na y = "<< xy << endl;

}
```

ПРИМЕР 40. ПОЛУЧЕНИЕ ПСЕВДОСЛУЧАЙНОГО ЧИСЛА

```
#include <stdlib.h>       // rand  srand
#include <time.h>         // time
#include <iostream>       // cin cout

using namespace std;

void main()
{
        // Установка  начального значения для генерации
        //  псевдослучайной  последовательности чисел
        // функция void srand( unsigned int seed );

    srand( (unsigned)time( NULL ) );

                //  генерация псевдослучайных чисел
                //  функция int rand( void );
                //  числа от 0.. RAND_MAX=32767
    cout<< rand() << endl << endl;

        // генерация псевдослучайных чисел в диапазоне 0..1
    cout << rand() / RAND_MAX << endl << endl;


        // генерация псевдослучайных чисел в диапазоне 0..m
    int m = 100;
    cout << rand() % m << endl << endl;

        // генерация псевдослучайных чисел в диапазоне -m..m
    cout << rand() % m - rand() % m << endl << endl;

    // генерация псевдослучайных чисел в диапазоне m1..m1+m2
    cout << rand() % m2 + m1 << endl << endl;

}
```

ПРИМЕР 41. ПОЛУЧЕНИЕ ДРОБНОЙ ЧАСТИ ЧИСЛА

```
#include <math.h>                    // modf
#include <iostream>                  // cin cout

using namespace std;

void main()
{
    double value,  int_value,  frac_value;

    cout << "vvod value=";
    cin >> value;

// 1.  Получение дробной части числа, используя функцию
//     double modf( double x,  double *intptr );

    frac_value = modf(value,&int_value);
    cout<< " 1 frac_value =" << frac_value << endl;


// 2.  Получение дробной части числа. Приведение типа  в С
    frac_value = value -  int(value);
    cout <<" 2 frac_value = " << frac_value << endl;


// 3.  Получение дробной части числа. Приведение типа в С++
    frac_value = value - static_cast<int>(value);
    cout<<" 3 frac_value = " << frac_value << endl;

}
```

ПРИМЕР 42. ПОЛУЧЕНИЕ ЦЕЛОЙ ЧАСТИ ЧИСЛА

```
#include <math.h>       // modf   floor
#include <iostream>     // cin cout

using namespace std;

void main()
{
     double value, int_value;

     cout<< "vvod value=";
     cin >> value;

     //  1. Получение целой части числа, используя функцию
     //  double modf( double x, double *intptr );
     modf(value, &int_value);
     cout << " 1 int_value=" << int_value << endl;



     // 2.  Отбрасывание дробной части числа через
     //  функцию double floor( double x );
     if (value >= 0)
          int_value = floor(value);
     else
          int_value = floor(value + 1);
     cout << " 2 int_value = " << int_value << endl;


                  // 3.  Получение целой части числа.
                  //  Приведение типа  в С
     int_value = int(value);
     cout <<" 3 int_value = "<< int_value << endl;



                  // 4.   Получение целой части числа.
                  // Приведение типа в С++
     int_value = static_cast<int>(value);
     cout <<" 4 int_value = " << int_value << endl;

}
```

ПРИМЕР 43. ОКРУГЛЕНИЕ ДО БЛИЖАЙШЕГО ЦЕЛОГО (С ИЗБЫТКОМ И НЕДОСТАТКОМ).

```
#include <math.h>
#include <iostream>  // cin cout

using namespace std;

void main()
{
    double x1, x2, y;

    x1 =  2.8;
    x2 = -2.8;
              // floor(2.8)=2     округление с недостатком
    y = floor(x1);
    cout<<"the floor of "<< x1 <<" is "<< y <<endl;

             // floor(-2.8)=-3 округление с недостатком
    y = floor(x2);
    cout<<"the floor of "<< x2 <<" is "<< y <<endl;

             //  ceil(2.8)=3 округление с избытком
    y = ceil(x1);
    cout<<"the ceil of "<< x1 <<" is "<< y <<endl;

              // ceil(-2.8)=-2 округление с избытком
    y = ceil(x2);
    cout<<"the ceil of "<< x2 <<" is "<< y <<endl;


              // floor  ceil
    cout<<"vvod x=";
    cin>>x1;
    cout<<"the floor of "<< x1 <<" is "<< floor(x1) <<endl;
    cout<<"the floor of "<< x1 <<" is "<< floor(-x1) <<endl;
    cout<<"the ceil of "<< x1 <<" is "<< ceil(x1) <<endl;
    cout<<"the ceil of "<< x1 <<" is "<< ceil(-x1) <<endl;

}
```

ПРИМЕР 44. ОКРУГЛЕНИЕ ДО БЛИЖАЙШЕГО ЦЕЛОГО.

```
#include <math.h>    //  ceil
#include <iostream>  //  cin cout

using namespace std;


void main()
{
    double x,y;
              // Округление до ближайшего целого, используя
              //  функцию double ceil( double x );

    x = 2.2;
    y = ceil(x - 0.5);                                  // y = 2
    cout<<" int (2.2) = "<< y <<endl;

    x = 2.8;
    y = ceil(x - 0.5);                                  // y = 3
    cout<<" int (2.8)= "<< y <<endl;

    x = -2.2;
    y = ceil(x - 0.5);                                  // y = -2
    cout<<" int (-2.2) = "<< y <<endl;

    x = -2.8;
    y = ceil(x - 0.5);                                  // y = -3
    cout<<" int (-2.8) = " << y <<endl;

    cout<<"x=";
    cin>>x;
    y = ceil(x - 0.5);
    cout<<" int = " << y <<endl;

}
```

ПРИМЕР 45. ПОЛУЧЕНИЕ НАИБОЛЬШЕГО ЗНАЧЕНИЯ В ДИАПАЗОНЕ ПАРАМЕТРА

```
#include <limits.h>
#include <iostream>  // cin cout

using namespace std;


// содержит константы, в которых максимальные и минимальные
//  значения для стандартных типов

void main()
{
cout<<"Maximum signed char value ="<< SCHAR_MAX <<endl;
                                          // 127

cout<<"Maximum unsigned char value  ="<< UCHAR_MAX <<endl;
                                           // 255 (0xff)

cout<<"Maximum unsigned short value   ="<< USHRT_MAX <<endl;
                                          // 65535 (0xffff)

cout<<"Maximum (signed) short value"<< SHRT_MAX <<endl;
                                          //  32767

cout<<"Maximum unsigned int value "<< UINT_MAX <<endl;
                                     // 4294967295(0xffffffff)

cout<<"Maximum unsigned long value "<< ULONG_MAX <<endl;
                                     // 4294967295(0xffffffff)

cout<<"Maximum (signed) int value  "<< INT_MAX <<endl;
                                     // 2147483647

cout<<"Maximum (signed) long value  "<< LONG_MAX <<endl;
                                      // 2147483647

cout<<"Maximum  char value ="<< CHAR_MAX <<endl;
                                       // 127

}
```

ПРИМЕР 46. ПОЛУЧЕНИЕ НАИМЕНЬШЕГО ЗНАЧЕНИЯ В ДИАПАЗОНЕ ПАРАМЕТРА

```
#include <limits.h>
// содержит константы, в которых максимальные и минимальные
//  значения для стандартных типов
#include <iostream>  // cin cout

using namespace std;

void main()
{

cout<<"Minimum signed char value  ="<< SCHAR_MIN <<endl;
                                          //   -128

cout<<"Minimum (signed) short value "<< SHRT_MIN <<endl;
                                          // -32768

cout<<"Minimum (signed) int value  "<< INT_MIN <<endl;
                                            // -2147483647-1

cout<<" Minimum (signed) long value   "<< LONG_MIN <<endl;
                                           // -2147483647-1

cout<<"Minimum  char value  ="<< CHAR_MIN<<endl;
                                          //   -128
}
```

ПРИМЕР 47. ВЫВОД СИМВОЛА

```
#include <iomanip>          // setw
#include <iostream>         // cin cout

using namespace std;

void main()
{
     char c = 'A';

     cout<<"simvol c="<< c <<endl;     //   вывод символа

                           //  вывод кода введенного символа
     cout<<"kod simvola c="<< int(c) <<endl;

     unsigned kod;
     kod = c;        // или так  вывод кода введенного символа
     cout << " Kod simvola " << c << " = "<<kod<<endl;

                     //  вывод символа по коду
     cout<<"simvol c kodom 55="<< (char)55 <<endl;

     cout << "   Enter a kod simvola ";
     cin >> kod;
     c = kod;                //  или так  вывод символа по коду
     cout << " Simvol s kodom " << kod <<" = "<< c <<endl;


     cout<<"simvol c=";
               // количество позиций для вывода символа =5
     cout.width(5);
     cout<< c <<endl;

          // или так количество позиций для вывода символа =5
     cout<<"simvol c="<< setw(5) << c <<endl;


     cout.width(5);
     cout.setf(ios::left);          // выравнивание влево
     cout<< c << c;
     cout.unsetf(ios::left);      // отмена выравнивания влево
     cout<<endl;
  }
```

ПРИМЕР 48. ВЫВОД ЦЕЛОГО ЧИСЛА

```
#include <iomanip>   // setw
#include <iostream>  // cin cout

using namespace std;

void main()
{    int  x = 1234;

     cout<<"int number"<<endl;
     cout<< x <<endl;

              // задание количества позиций для  вывода числа
     cout.width(20);
     cout<< x <<endl;
     cout<< setw(20) << x <<endl;             // или так

       // задание символа для заполнение свободных позиций
     cout.fill('0');
     cout.width(20);
     cout<< x <<endl;
     cout<< x <<endl;
     cout<< setw(20) << x <<endl;
     cout<< x <<endl;
         // задание символа для заполнение свободных позиций
     cout.fill(' ');

                  // вывод числа в восьмеричной системе
     cout<< oct << x <<endl;

                // вывод числа в шестнадцатиричной системе
     cout<< hex << x <<endl;

                 // вывод числа в шестнадцатиричной системе
                 // с буквами в верхнем ре-гистре
     cout.setf(ios::uppercase);
     cout<< hex << x <<endl;
                             // отмена верхнего регистра
     cout.unsetf(ios::uppercase);

                      // вывод числа в десятичной системе
     cout<<x<<"  "<< dec << x <<endl;
     cout<<endl;

}
```

ПРИМЕР 49. ВЫВОД ВЕЩЕСТВЕННОГО ЧИСЛА

```
#include <iomanip>    // setw
#include <iostream>   // cin cout

using namespace std;

void main()
{
    double  y = 3.14159265;

    cout<<"real number"<<endl;
    cout<< y <<endl;

            // задание количества позиций для  вывода числа
    cout<< setw(5) << y <<endl;

            // задание количества позиций для  вывода числа
    cout<< setw(20) << y <<endl;

        // задание символа для заполнение свободных по-зиций
    cout.fill('0');
    cout<< setw(20) << y <<endl;

            //  вывод значения из 2 значащих цифр
    cout.precision(2);
    cout<< y <<endl;

            // задание количества позиций для  вывода числа
    cout.width(20);
    cout.precision(9);
    cout<< y <<endl;

    cout.precision(4);
    cout<< y <<endl;

                //  научный формат (с экспонентой)
    cout.width(20);
    cout.setf(ios::scientific);
    cout<< y <<endl;
    cout.unsetf(ios::scientific);

    cout<<endl;
}
```

ПРИМЕР 50. ВЫВОД СТРОКИ

```
#include <iomanip>   // setw
#include <iostream>  // cin cout

using namespace std;

void main()
{
     char s[] = "EXAMPLE VYVODA STROKI";

     cout<<"stroka"<<endl;
     cout<<s<<endl;

               // задание количества позиций для  вывода строки
     cout<< setw(5) << s << endl;
     cout<< setw(25) << s <<endl;


               // задание количества позиций для  вывода строки
     cout.width(25);
     cout.setf(ios::left);        // выравнивание влево
     cout<<s<<endl;
     cout.unsetf(ios::left);

}
```

ПРИМЕР 51. Условные операторы

Вычислить значение функции y при заданном значении x:

$$y=\begin{Bmatrix} 0 & x<-2 \\ -x-2 & -2\le x<-1 \\ x & -1\le x<1 \\ -x+2 & 1\le x<2 \\ 0 & x\ge 2 \end{Bmatrix}$$

```
#include <iostream>  // cin cout

using namespace std;

void main()
{
     double x, y;

     cout << " x =";
     cin >>x;

                          // вариант 1
     if (x < -2)                y = 0;
     if (x >= -2 && x < -1)     y = -x - 2;
     if (x >= -1 && x < 1)      y = x;
     if (x >= 1 && x < 2)       y = -x +2;
     if (x >= 2)                y = 0;
     cout << " 1 x =" << x << " y =" << y << endl;

                          // вариант 2
     if (x < -2)    y = 0;
     else     if (x < -1)   y= -x-2;
          else     if (x < 1) y = x;
               else     if (x < 2) y = -x +2;
                    else y = 0;
     cout << " 2 x =" << x << " y =" << y << endl;

                          // вариант 3
     (x < -2) ? y = 0:
     (x < -1) ? y = -x - 2   :
     (x < 1)  ? y = x :
     (x < 2)  ? y = -x +2 : y = 0;

     cout << " 3 x =" << x << " y =" << y << endl;
}
```

ПРИМЕР 52. МНОЖЕСТВЕННЫЙ ВЫБОР

Пример использования множественного выбора: определение нажатой клавиши:

```
#include <conio.h>      // getch()
#include <iostream>     // cin cout

using namespace std;

void main()
{    char key;

     key = getch();  // определение кода нажатой клавиши
     switch (key)
     { case '0':  cout<< "digital" <<endl;  break;
       case 'a':  cout<< "small latin letter" <<endl;  break;
       case 'A':  cout<< "big latin letter" <<endl;  break;
       case '+': case '-':  cout<< "" <<endl;  break;
       default:   cout<< "error" <<endl;
     }
}
```

ПРИМЕР 53. МНОЖЕСТВЕННЫЙ ВЫБОР

Пример использования множественного выбора: определение нажатой клавиши:

```
#include <conio.h>    // getch()
#include <iostream>   // cin cout

using namespace std;

void main()
{
     int key;
                          // вариант 1
     key = getch();       // определение кода нажатой клавиши
     switch (key)
     { case 65:   cout<< "A" <<endl;  break;
       case 97:   cout<< "a" <<endl;  break;
       case 61:   cout<< "=" <<endl;  break;
       case 55:   cout<< "7" <<endl;  break;
       default:    cout<<"error"<<endl;
     }


                          // вариант 2  без default
     key = getch();
     switch (key)
     { case 65:   cout<< "A" <<endl;  break;
       case 97:   cout<< "a" <<endl;  break;
       case 61:   cout<< "=" <<endl;  break;
       case 55:   cout<< "7" <<endl;  break;
     }


                        // вариант 3  код символа - 16-ричный
     key = getch();
     switch (key)
     {  case 0x41:   cout<< "A" <<endl;  break;
        case 0x61:   cout<< "a" <<endl;  break;
        case 0x3D:   cout<< "=" <<endl;  break;
        case 0x37:   cout<< "7" <<endl;  break;
        default:     cout<< "error" <<endl;
     }




                          // вариант 4  без break
     key = getch();
```

```
    switch (key)
    { case 65:   cout<< "A" <<endl;
      case 97:   cout<< "a" <<endl;
      case 61:   cout<< "=" <<endl;
      case 55:   cout<< "7" <<endl;
      default:   cout<< "error" <<endl;
    }
}
```

ПРИМЕР 54. НАХОЖДЕНИЕ МИНИМАЛЬНОГО ЧИСЛА

Даны три вещественных числа a, b, c. Определить наименьшее из этих чисел.

```
#include <iostream>  // cin cout

using namespace std;

void main()
{    double a, b, c,  min;

     cout << " a, b, c =";
     cin>>a>>b>>c;
                                                    // 1
     if (a <= b && a<=c) min = a;
     else
          if (b <= c && b<=a) min = b;
          else
               if (c<=b && c<=a) min = c;
     cout << " 1 min = " << min<<endl;

                                                   // 2
     if (a<=b && a<=c) min = a;
     else
          if (b<=c && b<=a) min = b;
          else min=c;
      cout << " 2 min = " << min<<endl;

                                                    // 3
     if (a<=b && a<=c)  min = a;
     else
          if (b<=c)  min = b;
          else min = c;
     cout << " 3 min = " << min<<endl;

                                                   // 4
     if (a <= b)
          if (a <= c) min = a;
          else min = c;
     else
          if (b <=c ) min = b;
          else min = c;
     cout << " 4 min = " << min<<endl;



                                                   // 5
```

```
    min = a;
    if (b < min) min = b;
    if (c < min) min = c;
    cout << " 5 min = " << min<<endl;


                                                    // 6
    min = a <= b ? (a <= c ? a : c) : (b <= c ? b : c);
    cout << " 6 min = " << min<<endl;
}
```

ПРИМЕР 55. УПОРЯДОЧИВАНИЕ ЗНАЧЕНИЙ

Даны вещественные *x*, *y*, *z*. Поменять значения переменных так, чтобы $x < y < z$.

```
#include <iostream>  // cin cout

using namespace std;

void main()
{   double  x, y, z, k;

cout << " x = ";
cin  >> x;
cout << " y = ";
cin  >> y;
cout << " z = ";
cin  >> z;

if (y < x)
   { k = x; x = y; y = k; }  // меняем значения x и y

if (y > z )
   { k = y; y = z; z = k; }  // меняем значения z и y

if (y < x)
   { k = x; x = y; y = k; }  // меняем значения x и y

cout << x << " < " << y << " < " << z <<endl;

}
```

ПРИМЕР 56. НАХОЖДЕНИЕ КОРНЯ КВАДРАТНОГО УРАВНЕНИЯ

Определить корни квадратного уравнения $ax^2+bx+c=0$

```
#include <math.h>
#include <iostream>  // cin cout

using namespace std;

void main()
{    double a, b, c, x, y, D;

     cout<<"a=";     cin>>a;
     cout<<"b=";     cin>>b;
     cout<<"c=";     cin>>c;

     if (!a && !b)
          if (!c) cout<<"kornei beskonechno mnogo"<<endl;
          else     cout<<"ne imeet kornei"<<endl;
     elseif (!a )
          {    x = -c / b;
               cout<<x<<endl;        // 1 корень
          }
          elseif (!b && !c)
               {  cout<<" 0 "<<endl;
               }
               else
               {              //   вычисляем дискриминант
                    D = b * b - 4 * a * c;
                    if (D >= 0)
                    {         // 2 вещественных корня
                         x = (-b + sqrt(D)) / (2 *  a);
                         y =  (-b -  sqrt(D)) / ( 2  * a);
                         cout<< x <<endl;
                         cout<< y <<endl;
                    }
                    else
                    {//действительная часть комплексного кор-ня
                      x = -b/ (2*a);
                         // мнимая часть комплексного корня
                    y = sqrt(-D) / (2 * a);
                    cout<<x<<" + i "<<y<<endl;
                    cout<<x<<" - i "<<y<<endl;
                            }
               }
}
```

ПРИМЕР 57. ПРЕОБРАЗОВАНИЕ СИМВОЛА В ВЕРХНИЙ И НИЖНИЙ РЕГИСТР

```
#include <ctype.h>   //  islower isupper toupper tolower
#include <iostream>  //  cin cout

using namespace std;

void main()
{                  //  Преобразование символа в верхний регистр
    char c, C;
    cout << "   Enter a char: ";
    cin >> c;
             //  Перевод строчных латинских букв в прописные
              //  используя функции int toupper( int c );
              //  int islower( int c );
    if (islower(c))                  //  c >= 'a' && c <= 'z'
    {   C = toupper(c);
        cout<<"   for  "<< c <<" = "<< C <<endl;
    }

                  // или так перевод строчных латинских букв
                  // в прописные через арифмети-ку
    if (islower(c))             //   c >= 'a' && c <= 'z'
    {   C ='A' + c - 'a';
        cout<<"   for  "<< c  <<" = "<< C <<endl;
    }


             // Преобразование символа в нижний регистр
    cout<<"   Enter a char: ";
    cin>>C;

             // Перевод прописных латинских букв в строчные
             // используя функции int tolower( int c );
              // int isupper( int c );
    if (isupper(C))                  //C >= 'A'  && C <= 'Z'
    {   c = tolower(C);
        cout<<"   for  "<< C  <<" = "<< c <<endl;
    }
                 // Перевод прописных латинских букв
                 // в строчные через арифметику
    if (isupper(C))              //  C >= 'A' && C <= 'Z'
    {   c = 'a' + C - 'A';
        cout<<"   for  "<< C <<" = "<< c <<endl;
    }

}
```

ПРИМЕР 58. ПОЛУЧЕНИЕ ТЕКУЩЕЙ ДАТЫ И ВРЕМЕНИ

```
#include <time.h>              //  _strdate   _strtime
#include <iostream>            // cin cout

using namespace std;

void main( void )
{
     char dbuffer [9];

              // функция   char *_strdate( char *timestr );

     _strdate( dbuffer );
     cout<<"The current date is "<< dbuffer <<endl;

     char tbuffer [9];
              // функция   char *_strtime( char *timestr );

     _strtime( tbuffer );
     cout<< "The current time is "<<  tbuffer <<endl;
}
```

ПРИМЕР 59. Определение времени работы программы

```
#include <time.h>      // clock
#include <iostream>    // cin cout

using namespace std;



void main( void )
{
     double sec;
     clock_t start, finish;

          //  функция clock_t clock( void );
          //  предназначена для определения тактовых импульсов
          //  процессора с начала работы программы
          //  CLOCKS_PER_SEC константа, которая определяет
          //  количество тактовых имульсов в 1 сек.

     start = clock();

                          // ...   программа

     finish = clock();

     sec = double(finish - start) /  CLOCKS_PER_SEC;

     cout<<"time of run programm = "<<sec<<endl;
}
```

ПРИМЕР 60. ПЕРЕКОДИРОВКА СТРОКОВЫХ КОНСТАНТ С РУССКИМИ БУКВАМИ

```
#include <iostream>   // cin cout
using namespace std;
#include <clocale>    //#include <locale.h>

void main()
    {
     system ("cls");              //очистка окна вывода
     cout << "Моя  программа"<< endl;

     setlocale(LC_ALL, "rus");
     cout << "Моя программа"<< endl;

     system ("pause");  //приостановка выполнения программы

    }
```

ЗНАЧЕНИЯ ДВОИЧНЫХ КОДОВ ШЕСТНАДЦАТЕРИЧНЫХ ЦИФР

цифра	код	цифра	Код
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

КОДЫ СИМВОЛОВ ASCII (АЛЬТЕРНАТИВНАЯ)

dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char
0	0		46	2E	.	92	5C	\	138	8A	К
1	1		47	2F	/	93	5D	]	139	8B	Л
2	2		48	30	0	94	5E	^	140	8C	М
3	3		49	31	1	95	5F	_	141	8D	Н
4	4		50	32	2	96	60	'	142	8E	О
5	5		51	33	3	97	61	a	143	8F	П
6	6		52	34	4	98	62	b	144	90	Р
7	7		53	35	5	99	63	c	145	91	С
8	8		54	36	6	100	64	d	146	92	Т
9	9		55	37	7	101	65	e	147	93	У
10	A		56	38	8	102	66	f	148	94	Ф
11	B		57	39	9	103	67	g	149	95	Х
12	C		58	3A	:	104	68	h	150	96	Ц
13	D		59	3B	;	105	69	i	151	97	Ч
14	E		60	3C	<	106	6A	j	152	98	Ш
15	F		61	3D	=	107	6B	k	153	99	Щ
16	10		62	3E	>	108	6C	l	154	9A	Ъ
17	11		63	3F	?	109	6D	m	155	9B	Ы
18	12		64	40	@	110	6E	n	156	9C	Ь
19	13		65	41	A	111	6F	o	157	9D	Э
20	14		66	42	B	112	70	p	158	9E	Ю
21	15		67	43	C	113	71	q	159	9F	Я
22	16		68	44	D	114	72	r	160	A0	а
23	17		69	45	E	115	73	s	161	A1	б
24	18		70	46	F	116	74	t	162	A2	в
25	19		71	47	G	117	75	u	163	A3	г
26	1A		72	48	H	118	76	v	164	A4	д
27	1B		73	49	I	119	77	w	165	A5	е
28	1C		74	4A	J	120	78	x	166	A6	ж
29	1D		75	4B	K	121	79	y	167	A7	з
30	1E		76	4C	L	122	7A	z	168	A8	и
31	1F		77	4D	M	123	7B	{	169	A9	й
32	20	про-бел	78	4E	N	124	7C	¦	170	AA	к
33	21	!	79	4F	O	125	7D	}	171	AB	л
34	22	"	80	50	P	126	7E	~	172	AC	м
35	23	#	81	51	Q	127	7F	Δ	173	AD	н
36	24	$	82	52	R	128	80	А	174	AE	о
37	25	%	83	53	S	129	81	Б	175	AF	п
38	26	&	84	54	T	130	82	В	176	B0	▒
39	27	'	85	55	U	131	83	Г	177	B1	▒
40	28	(	86	56	V	132	84	Д	178	B2	▓
41	29	)	87	57	W	133	85	Е	179	B3	│
42	2A	*	88	58	X	134	86	Ж	180	B4	┤
43	2B	+	89	59	Y	135	87	З	181	B5	╡
44	2C	,	90	5A	Z	136	88	И	182	B6	╢
45	2D	-	91	5B	[	137	89	Й	183	B7	╖

184	B8		202	CA		220	DC		238	EE	ю
185	B9		203	CB		221	DD		239	EF	я
186	BA		204	CC		222	DE		240	F0	Ё
187	BB		205	CD		223	DF		241	F1	ё
188	BC		206	CE		224	E0	р	242	F2	Ґ
189	BD		207	CF		225	E1	с	243	F3	ґ
190	BE		208	D0		226	E2	т	244	F4	
191	BF		209	D1		227	E3	у	245	F5	
192	C0		210	D2		228	E4	ф	246	F6	
193	C1		211	D3		229	E5	х	247	F7	
194	C2		212	D4		230	E6	ц	248	F8	
195	C3		213	D5		231	E7	ч	249	F9	
196	C4		214	D6		232	E8	ш	250	FA	Ў
197	C5		215	D7		233	E9	щ	251	FB	ў
198	C6		216	D8		234	EA	ъ	252	FC	^{n}
199	C7		217	D9		235	EB	ы	253	FD	^{2}
200	C8		218	DA		236	EC	ь	254	FE	■
201	C9		219	DB		237	ED	э	255	FF	

СЛОВАРЬ ПОНЯТИЙ, ИСПОЛЬЗУЕМЫХ В ЗАДАНИЯХ

Натуральное число называется **палиндромом**, если его запись читается одинаково с начала и с конца.