Типы данных C++

Название типа (type)	Ключевое слово (keyword)	Размер (size)	Диапазон значений (limits)
логический	`bool`	1 byte	true/false
символьный	`char`	1 byte	-128 ÷ 127
беззнаковый символьный	`unsigned char`	1 byte	0 ÷ 255
широкий символьный	`wchar_t`	2 byte	-32768 ÷ 32767
короткий целый	`short`	2 bytes	-32768 ÷ 32767
беззнаковый короткий целый	`unsigned short`	2 bytes	0 ÷ 65535
целый	`int`	4 bytes	-2147483648 ÷ 2147483647
беззнаковый целый	`unsigned int`	4 bytes	0 ÷ 4294967295
длинный целый	`long`	4 bytes	-2147483648 ÷ 2147483647
беззнаковый длинный целый	`unsigned long`	4 bytes	0 ÷ 4294967295
вещественный	`float`	4 bytes	(±)1.175494351e–38 ÷ (±)3.402823466e+38
вещественный с двойной точностью	`double`	8 bytes	(±)2.2250738585072014e-308 ÷ (±)1.7976931348623158e+308
длинный вещественный	`long double`	8 bytes	(±)2.2250738585072014e-308 ÷ (±)1.7976931348623158e+308
пустой	`void`	–	–

Циклические конструкции

Цикл с предусловием

```
while(условие)
{
    блок операторов;
}
```

Цикл с постусловием

```
do
{
    блок операторов;
}while(условие);
```

Пошаговый цикл

```
for(инициализация; условие; модификация)
{
    блок операторов;
}
```

`break` – оператор досрочного выхода из цикла

`continue` – оператор досрочного перехода к следующему шагу цикла

Условные конструкции

```
if(условие)
{
    блок операторов;
}
```

```
if(условие)
{
    блок операторов;
}
else
{
    блок операторов;
}
```

Селективные конструкции

```
switch(переменная)
{
    case целая константа:
    {
        блок операторов;
        break;
    }
    case целая константа:
    {
        блок операторов;
        break;
    }
    default:
    {
        блок операторов;
    }
}
```

```
if(условие)
{
    блок операторов;
}
else if(условие)
{
    блок операторов;
}
else
{
    блок операторов;
}
```

Остальные ключевые слова C++

`asm`	`mutable`	`this`
`auto`	`namespace`	`try`
`catch`	`operator`	`typedef`
`class`	`private`	`typeid`
`const`	`protected`	`typename`
`enum`	`public`	`union`
`explicit`	`register`	`unsigned`
`export`	`return`	`using`
`extern`	`signed`	`virtual`
`friend`	`static`	`volatile`
`goto`	`struct`	
`inline`	`template`	

Операторы C++

Оператор (operator)	Назначение (meaning)	Тип оператора (type)	Порядок выполнения (associativity)
`::`	область видимости (scope resolution)	unary	–
`[ ]`	доступ к элементам массива (array subscript)	unary	слева направо
`( )`	вызов функции (function call)	multiple	слева направо
`type( )`	оператор преобразования типов, конструирования значения (conversion, constructor)	unary	–
`->`	операторы доступа к членам гетерогенных типов данных (member selection)	unary	слева направо
`++` `--`	постфиксный инкремент/декремент (postfix increment/decrement)	unary	–
`new`	Динамическое размещение в памяти переменных и массивов (allocate)	binary	–
`delete` `delete []`	удаление из памяти переменной/массива, размещенного динамически (deallocate)	binary	–
`++` `--`	префиксный инкремент/декремент (prefix increment/decrement)	unary	–
`*`	получение значения по указателю (dereference)	unary	–
`&`	получения указателя на переменную (address-of)	unary	–
`+`	унарный плюс (unary plus)	unary	–
`-`	арифметическое отрицание (arithmetic negation)	unary	–
`!`	отрицание/логическое НЕ (logical NOT)	unary	–
`~`	побитовое дополнение (bitwise complement)	unary	–
`sizeof`	определение размера переменной (size of object)	unary	–
`typedef`	определение имени типа (type name)	unary	–
`(type)`	приведение типов (type cast)	unary	справа налево
`const_cast` `dynamic_cast` `reinterpret_cast` `static_cast`	приведение типов (type cast)	unary	–
`.*` `->*`	получение указателя на член класса (pointer to class member)	unary	слева направо
`*` `/`	умножение/деление (multiplication/division)	binary	слева направо
`%`	остаток от деления (remainder)	binary	слева направо
`+` `-`	сложение/вычитание (addition/subtraction)	binary	слева направо
`<<` `>>`	побитовый левый/правый сдвиг (left/right shift)	unary	слева направо
`<` `>` `<=` `>=`	сравнение (comparison)	binary	слева направо
`==` `!=`	равенство/неравенство (equality/inequality)	binary	слева направо
`&`	побитовое И (bitwise AND)	binary	слева направо
`^`	побитовое исключающее ИЛИ (bitwise exclusive OR)	binary	слева направо
`|`	побитовое ИЛИ (bitwise OR)	binary	слева направо
`&&` `||`	логическое И/ ИЛИ (logical AND/OR)	binary	слева направо
`? :`	арифметическая условная конструкция (ariftnetic if)	ternary	справа налево
`=`	присваивание (assignment)	binary	справа налево
`*=` `/=`	умножение/деление с присваиванием (multiplication/division assignment)	binary	справа налево
`%=`	остаток от деления с присваиванием (modulus assignment)	binary	справа налево
`+=` `-=`	сложение/вычитание с присваиванием (addition/subtraction assignment)	binary	справа налево
`<<=` `>>=`	побитовый левый/правый сдвиг с присваиванием (left/right shift assignment)	binary	справа налево
`&=`	побитовое И с присваиванием (bitwise AND assignment)	binary	справа налево
`|=`	побитовое ИЛИ с присваиванием (bitwise inclusive OR assignment)	binary	справа налево
`^=`	побитовое исключающее ИЛИ с присваиванием (bitwise exclusive OR assignment)	binary	справа налево
`throw`	генерация исключения	unary	–
`,`	разделитель (comma)	unary	слева направо

```
// однострочный комментарий
/* многострочный комментарий */
```

Основные математические функции

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

Функция	Возвращаемое значения
double atof(const char* string)	вещественное число, заданное массивом символов `string`
int atoi(const char* string)	целое число, заданное массивом символов `string`
int abs(int n)	абсолютное значение (модуль) целочисленного аргумента `n`
double acos(double x)	арккосинус `x` в пределах `-1.0` до `1.0`
double asin(double x)	арксинус `x` в пределах `-1.0` до `1.0`
double atan(double x)	арктангенс `x` в пределах $-\frac{\pi}{2}$ до $\frac{\pi}{2}$
double atan2(double x, double y)	арктангенс `x/y` в пределах $-\pi$ до π
double ceil(double x)	наименьшее целое большее или равное `x`
double cos(double x)	косинус `x`
double exp(double x)	e^x
double fabs(double x)	абсолютное значение (модуль) вещественного числа `x`
double floor(double x)	наибольшее целое большее или равное `x`
double fmod(double x, double y)	остаток от деления `x` на `y`
double hypot(double x, double y)	длина гипотенузы при заданных катетах `x` и `y`
double log(double x)	натуральный логарифм `x`
double log10(double x)	десятичный логарифм `x`
double pow(double x, double y)	`x` в степени `y`
double sin(double x)	синус `x`
double sqrt(double x)	$\sqrt{x}$
double tan(double x)	тангенс `x`

Функции для генерации случайных чисел

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

Функция	Возвращаемое значения
int rand()	псевдослучайное число в диапазоне от 0 до RAND_MAX
void srand(unsigned int seed)	устанавливает стартовую точку для генерации случайных чисел

Функции для работы со временем

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

Функция	Возвращаемое значения
clock_t clock()	процессорное время
double difftime(time_t timerl, time_t timer2)	разность между двумя временными значениями
time_t time(time_t* timer)	системное время

Манипуляторы потоков ввода/вывода

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

Манипулятор	Действие на поток
showbase **noshowbase**	инициирует отображение основания системы счисления
showpos **noshowpos**	инициирует явное отображение знака (+) для положительных значений
uppercase **nouppercase**	инициирует преобразование выводимых символов в верхний регистр
showpoint **noshowpoint**	инициирует отображение десятичной точки при выводе вещественных чисел
skipws **noskipws**	инициирует пропуск пробельных символов при вводе
left	инициирует левое выравнивание, заполнение справа
right	инициирует правое выравнивание, заполнение слева
internal	инициирует внутреннее заполнение (между значением и знаком или основанием системы счисления)
scientific	инициирует вывод вещественных значений в научном формате (d.ddddddddE+dd)
fixed	инициирует вывод вещественных значений в фиксированном формате (d.dddddd)
setbase (int base)	изменяет систему счисления (10, 8, 16)
dec	инициирует вывод целочисленных значений в десятичной системе счисления
oct	инициирует вывод целочисленных значений в восьмеричной системе счисления
hex	инициирует вывод целочисленных значений в шестнадцатеричной системе счисления
setfill (char n)	задает заполняющий символ
setprecision (int n)	изменяет точность выводимых значений
setw (int n)	задает ширину поля вывода

Строковая библиотека

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

Функция	Описание
конструкторы	
`string()`	конструктор по умолчанию, создает пустую строку
`string(const char* p)`	преобразующий конструктор
`string(const string& str, size t pos=0, size t n=npos)`	копирующий конструктор (npos обычно равен -1 и указывает, что память не была выделена)
`string(const char* p, size_t n)`	копирует n символов, p является базовым адресом
`string(char c, size t n=l)`	создает строку из n символов c
перегруженные операторы	
`string& operator= (const string& s)`	оператор присваивания
`string& operator+= (const string& s)`	добавляет строку
`char operator[] (size t pos) const`	возвращает символ из позиции pos
`char& operator[] (size t pos)`	возвращает ссылку на символ из позиции pos
функции-члены	
`string& append(const string& s, size_t pos=0, size t n=npos);`	Добавляет n символов начиная от позиции pos
`string& assign(const string& s, size_t pos=0, size_t n=npos);`	строковому объекту присваивается n символов, начиная от позиции pos
`string& insert(size_t posl, const string& str, size_t pos2=0 , size_t n=npos);`	вставляет n символов, полученных из str, начиная с позиции pos2, в строку с позиции posl
`string& remove(size_t pos=0 , size_t n=npos);`	Удаляются n символов из строки начиная с позиции pos
`string& replace(posl, nl, str, pos2=0, n2=npos);`	в неявной строке начиная с позиции posl заменяет nl символов n2 символами из подстроки str с позиции pos2
`string& replace(pos, n, p, n2);`	заменяет n символов в позиции pos используя char* p из n2 символов или char* p до завершающего нуля, или повторяя символ c rep раз
`char get_at (pos) const;`	возвращает символ из позиции pos
`void put_at (pos, c);`	помещает символ c в позицию pos
`size_t length() const;`	возвращает длину строки
`const char* c_str() const;`	преобразует строку в традиционное char* представление
`const char* data() const;`	возвращает базовый адрес строкового представления
`void resize(n, c);` `void resize(n);`	изменяет строку, делая ее длину равной n; в первой функции в качестве заполняющего символа выступает c, а во второй - символ eos () (end-of-string, конец строки)
`void reserve(size_t res_arg);` `size_t reserve() const;`	выделяет память под строку; первая функция переустанавливает this; вторая возвращает закрытый член res - размер выделенного фрагмента
`size_t copy(p, n, pos=0) const;`	n симолов строки, начиная с позиции pos, копируются в char* p
`string substr(pos=0, n=pos) const;`	возвращается подстрока из n символов строки
`int compare(const string& str, size_t pos=0, size_t n=npos) const;`	сравнивает n символов строки, начиная с позиции pos, со строкой str. Возвращается ноль, если строки равны; в противном случае возвращается положительное или отрицательное целое значение, показывающее, что неявная строка лексикографически больше или меньше чем строка str.
`size_t find (const string& str, size_t pos=0) const;`	в строке начиная с позиции pos производится поиск строки str. Если она найдена, возвращается позиция, в которой она начинается; в противном случае возвращается позиция npos
`size_t rfind(str, pos=npos) const;`	похоже на find (), но при поиске первого совпадения строка сканируется в обратном направлении
`size_t find_first_of(str, pos=0) const;`	производится поиск первого вхождения str начиная с позиции pos
`size_t find_last_of(str, pos=npos) const;`	аналогично, но в обратном направлении
`size_t find_first_not_of(str, pos=0) const;`	производится поиск первого символа, который не соответствует ни одному из символов str начиная с позиции pos
`size_t find_last_not_of(str, pos=npos) const;`	аналогично, но в обратном направлении