

Типи даних

```
55+15 // legal C++ expression
//Both operands of the + operator are integers

55 + "John" // illegal
// The + operator is not defined for integer and string
```

Числові типи даних

Цілі числа (цілі числа), такі як -7, 42.

Числа з плаваючою комою , такі як 3.14, -42.67

Рядки та символи

Рядок складається з цифр, букв або символів. Рядкові літерали розміщуються у **подвійних лапках** ; наприклад "Hello", "My name is David".

Символи - це окремі літери або символи, і вони повинні бути закриті між **окремими лапками**, наприклад 'a', 'b'. У C ++ одиничні лапки позначають **символ** ; подвійні лапки створюють **рядковий** літерал. У той час як 'a' - це **символьний** літерал, "a" – це рядковий літерал.

Булевий тип

Boolean тип даних повертає тільки два можливих значення: **true** (1) і **false** (0)

```
int a = 42;
```

```
unsigned long int a;
```

Цілі числа

signet : Знакове ціле число може містити як від'ємні, так і додатні числа;

unsigned : ціле число без знаку може містити лише додатні значення;

shot : половина розміру за замовчуванням;

long : подвоєний розмір за замовчуванням.

Числа з плаваючою точкою

float, double, long double.

float - 4 байти, **double** - 8, а **long double** - 16 байт.

```
double temp = 4.21;
```

Типи даних з плаваючою комою завжди є знаковими, тобто вони мають можливість зберігати як позитивні, так і негативні значення.

String

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

int main() {
    string a = "I am learning C++";
    return 0;
}
```

Бібліотека `<string>` включена в бібліотеку `<iostream>`, тому не потрібно включати `<string>` окремо, якщо ви вже використовуєте `<iostream>`.

Char

```
char test = 'S';
```

Змінна **char** містить 1-байтове ціле число. Однак замість інтерпретації значення **char** як цілого числа, значення змінної `char` зазвичай інтерпретується як символ ASCII.

Булеві змінні

Якщо булеве значення присвоюється цілому числу, то `true` стає 1, а `false` стає 0. Якщо ціле число присвоюється булевому значенню, 0 стає хибним і будь-яке значення, яке має ненульове значення, стає істинним.

Правила при іменуванні змінних:

- Усі імена змінних повинні починатися з літери алфавіту або підкреслення (`_`).
- Після початкової літери назви змінних можуть містити додаткові літери, а також цифри. Порожні пробіли або спеціальні символи заборонені в назвах змінних.

Існує два відомих принципи іменування:

- **Pascal case:** Перша літера в ідентифікаторі та перша літера кожного наступного сполученого слова з великої літери. Наприклад: `BackColor`
- **Camel case:** Перша літера ідентифікатора - це малі літери, а перша буква кожного наступного сполученого слова пишеться з великої літери. Наприклад: `backColor`.

Масиви

```
int a [5];
```

```
int b [5] = {11, 45, 62, 70, 88};
```

```
int b [] = {11, 45, 62, 70, 88};
```

11	45	62	70	88
[0]	[1]	[2]	[3]	[4]

```
int b[] = {11, 45, 62, 70, 88};
```

```
cout << b[0] << endl;  
// Outputs 11
```

```
cout << b[3] << endl;  
// Outputs 70
```

```
int b [] = {11, 45, 62, 70, 88};  
b [2] = 42;
```

```
int myArr[5];
```

```
for(int x=0; x<5; x++) {  
    myArr[x] = 42;
```

```
    cout << x << ": " << myArr[x] << endl;  
}
```

/* Outputs

0: 42

1: 42

2: 42

3: 42

4: 42

*/

```
int arr[] = {11, 35, 62, 555, 989};  
int sum = 0;
```

```
for (int x = 0; x < 5; x++) {  
    sum += arr[x];  
}
```

```
cout << sum << endl;  
//Outputs 1652
```

```
type name[size1][size2]...[sizeN];
```

```
int x[3][4];
```

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

```
int x[2][3] = {  
    {2, 3, 4}, // 1st row  
    {8, 9, 10} // 2nd row  
};
```

```
int x[2][3] = {{2, 3, 4}, {8, 9, 10}};
```

```
int x[2][3] = {{2, 3, 4}, {8, 9, 10}};  
cout << x[0][2] << endl;
```

//Outputs 4

```
string threeD[42][8][3];
```

Вказівники

```
int score = 5;
cout << &score << endl;

//Outputs "0x29fee8"
```

```
int *ip; // pointer to an integer
double *dp; // pointer to a double
float *fp; // pointer to a float
char *ch; // pointer to a character
```

```
int score = 5;
int *scorePtr;
scorePtr = &score;

cout << scorePtr << endl;

//Outputs "0x29fee8"
```

```
int var = 50;
int *p;
p = &var;

cout << var << endl;
// Outputs 50 (the value of var)

cout << p << endl;
// Outputs 0x29fee8 (var's memory location)

cout << *p << endl;
/* Outputs 50 (the value of the variable
stored in the pointer p) */
```

```
int x = 5;
int *p = &x;

x = x + 4;
x = *p + 4;
*p = *p + 4;
```

```
int *p = new int; // request memory
*p = 5; // store value

cout << *p << endl; // use value

delete p; // free up the memory
```

Є два оператори для вказівників:

- **Взяття адреси(&):** повертає адресу пам'яті свого операнда;
- **Непряме звернення до об'єкту (або разадресація)**
Dereferencing (*): повертає значення змінної, розташованої за адресою

Динамічна пам'ять

Стек (The stack): Усі локальні змінні займають пам'ять зі стека.

Куча (The heap): Невикористана програма пам'ять, яка може бути використана, коли програма **динамічно** виділяє пам'ять.

```
new int;
```

```
int *p = new int;
*p = 5;
```

```
delete pointer;
```

```
int *p = new int; // request memory
*p = 5; // store value
```

```
delete p; // free up the memory
// now p is a dangling pointer
```

```
p = new int; // reuse for a new address
```

```
int *p = NULL; // Pointer initialized with null
p = new int[20]; // Request memory
delete [] p; // Delete array pointed to by p
```

Category	Type	Minimum Size
boolean	bool	1 byte
character	char	1 byte
integer	short	2 bytes
	int	2 bytes
	long	4 bytes
	long long	8 bytes
floating point	float	4 bytes
	double	8 bytes
	long double	8 bytes

sizeof (data type)

```

cout << "char: " << sizeof(char) << endl;
cout << "int: " << sizeof(int) << endl;
cout << "float: " << sizeof(float) << endl;
cout << "double: " << sizeof(double) << endl;
int var = 50;
cout << "var: " << sizeof(var) << endl;

/* Outputs
char: 1
int: 4
float: 4
double: 8
var: 4
*/

```

```

double myArr[10];
cout << sizeof(myArr) << endl;

//Outputs 80

```

```

int numbers[100];
cout << sizeof(numbers) / sizeof(numbers[0]);

// Outputs 100

```