Программирование на языке C++

...

Типы данных

Binary Digit (Bit)

Двоичное число. Единица измерения информации

1 бит информации — символ или сигнал, который может принимать два значения: включено (1) или выключено (0)

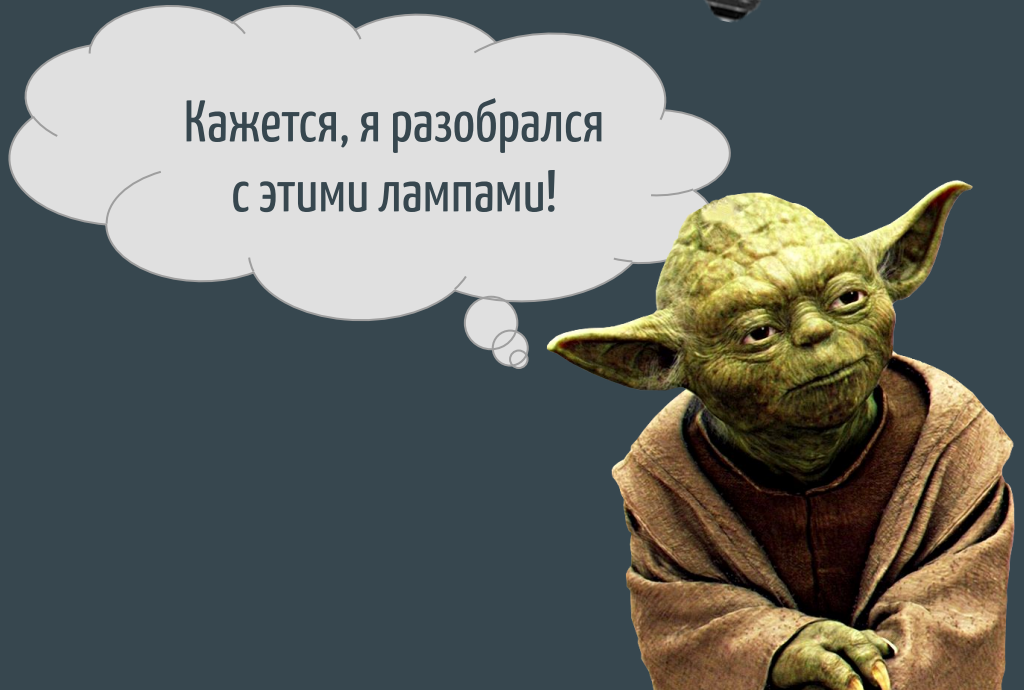
Битовые комбинации

Битовые комбинации

1 лампа
2 значения

2 лампы
2x2 значения

3 лампы
2x2x2
значений



N ламп (бит)
2^N комбинаций

Byte

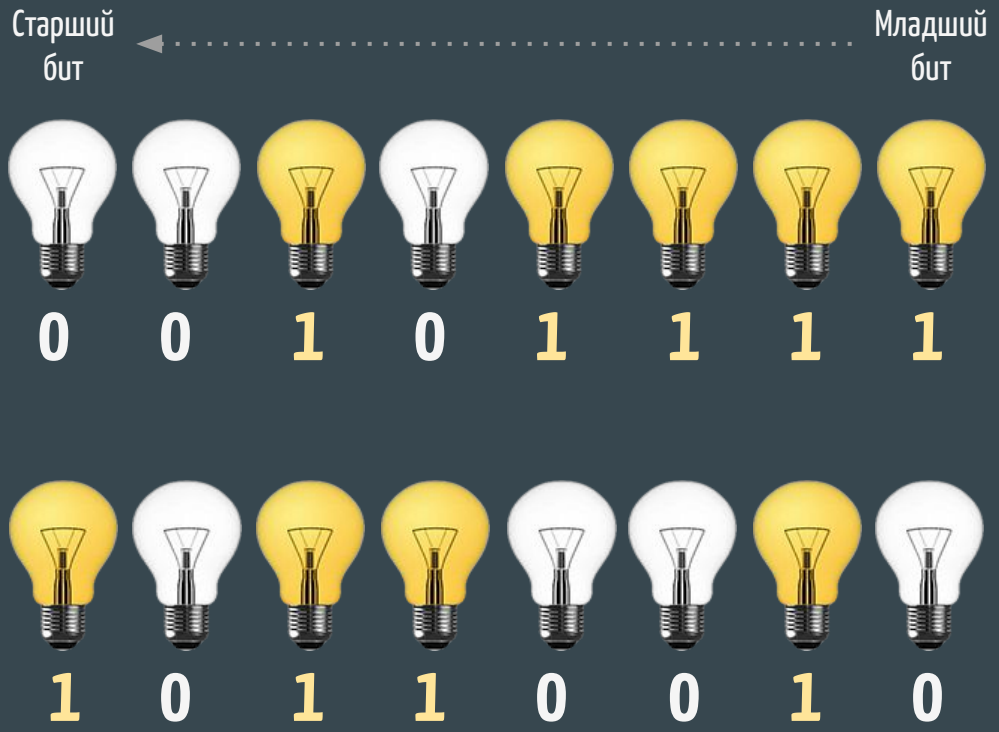
Байт. Совокупность битов, обрабатываемая компьютером одномоментно

В современных вычислительных системах байт состоит из восьми битов и, соответственно, может принимать одно из 256 (2^8) различных значений.

В большинстве вычислительных архитектур байт — это минимальный независимо адресуемый набор данных.

Byte

Байт. Совокупность битов, обрабатываемая компьютером одномоментно



Byte

Внутреннее представление целого числа

0	0	1	0	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	32	0	8	4	2	1

Младший бит интерпретируется как 2^0, каждый последующий как 2^i. Для получения окончательного значения просто сложим значения всех бит:

$$32 + 8 + 4 + 2 + 1 = 47$$

Byte

Диапазон возможных значений
целого числа
от 0 до 255 (2^8-1)

0

0 0 0 0 0 0 0 0

255

128 64 32 16 8 4 2 1

Byte

Внутреннее представление
целого знакового числа
от -127 до 127 (2^7-1)

127

\+ 64 32 16 8 4 2 1

-127

\- 64 32 16 8 4 2 1

Сложение чисел

Типы данных C++

Пустой

void

Логический тип

bool

Символьный

char

Целочисленные типы

short	unsigned short
int	unsigned int
long	unsigned long
long long	unsigned long long

Типы с плавающей точкой

float
double
long double

Символьный тип

```
#include<iostream>

int main()
{
    char ch1 = 'A';
    cout << "ch1 = " << ch1 << ". Symbol code is " << (int)ch1 << endl;

    char ch2 = 77;
    cout << "ch2 = " << ch2 << ". Symbol code is " << (int)ch2 << endl;
}
```

```
ch1 = A. Symbol code is 65
ch2 = M. Symbol code is 77
```

Таблица ASCII

American Standard Code for Information Interchange

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

Символьный тип

```
#include<iostream>

int main()
{
    char ch1 = 'A';
    char ch2 = ch + 10;
    cout << "ch1 = " << ch1 << ". Symbol code is " << (int)ch1 << endl;
    cout << "ch2 = " << ch2 << ". Symbol code is " << (int)ch2 << endl;
}
```

```
ch1 = A. Symbol code is 65
ch2 = K. Symbol code is 75
```

Сложение чисел

```
#include<iostream>

int main()
{
     unsigned char var1 = 39;  // 00100111
     unsigned char var2 = 146; // 10010010

     unsigned char sum = var1 + var2;
     std::cout << "Sum = " << (int)sum << "\n";
}
```

```
Sum = 185
```

Переполнение

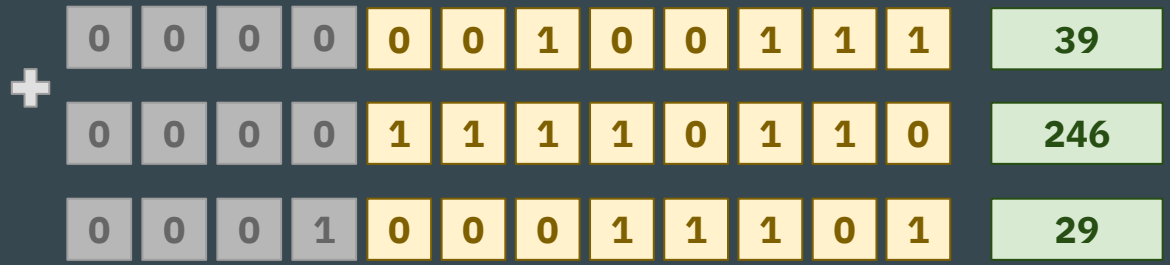
```
#include<iostream>

int main()
{
     unsigned char var1 = 39;  // 00100111
     unsigned char var2 = 246; // 11110110

     unsigned char sum = var1 + var2;
     std::cout << "Sum = " << (int)sum << "\n";
}
```

```
Sum = 29
```

```
cout << "var size = " << sizeof(unsigned char) << " byte" << endl;
```

```
var size = 1 byte
```

```
unsigned char sum = 39 + 246;
```



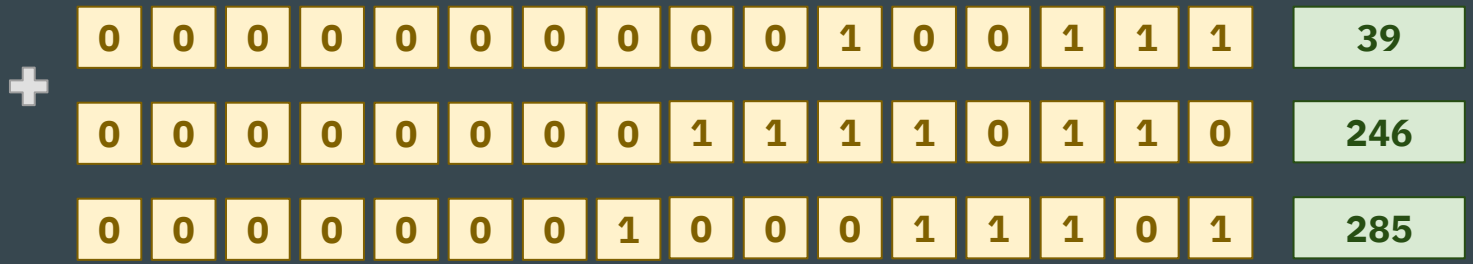

```
std::cout << "Sum = " << (int)sum << "\n";
```

```
Sum = 29
```

```
cout << "var size = " << sizeof(short) << " byte" << endl;
```

```
var size = 2 byte
```

```
short sum = 39 + 246;
```




```
std::cout << "Sum = " << sum << "\n";
```

```
Sum = 285
```

Размеры целых типов

Размер типов не определяется стандартом, а зависит от архитектуры и разрядности процессора и компилятора.

```
#include<iostream>

int main()
{
	std::cout << "Different type sizes in bytes: \n";
	std::cout << "char      = " << sizeof(char)      << "\n";
	std::cout << "short     = " << sizeof(short)     << "\n";
	std::cout << "int       = " << sizeof(int)       << "\n";
	std::cout << "long      = " << sizeof(long)      << "\n";
	std::cout << "long long = " << sizeof(long long) << "\n";
}
```

```
Different type sizes in bytes:
char      = 1
short     = 2
int       = 4
long      = 4
long long = 8
```

Структуры

Простейший пользовательский тип данных.

Позволяет сгруппировать переменные разных типов в единое целое.

```
#include<iostream>

struct Vector {
    float x;
    float y;
    float z;
};

int main()
{
    Vector v1{20.f, -7.f, 0.f };
    v1.x -= 3.5f;
    v1.z = 20;

    std::cout << "v1 = {";
    std::cout << v1.x << "; ";
    std::cout << v1.y << "; ";
    std::cout << v1.z << "}";
    std::cout << std::endl;
}
```

```
v1 = {16.5; -7; 20}
```

Инициализация полей структуры

```
#include<iostream>

struct Vector {
    float x;
    float y;
    float z;
};

int main()
{
    Vector v1;
    std::cout << "v1 = {";
    std::cout << v1.x << "; ";
    std::cout << v1.y << "; ";
    std::cout << v1.z << "}";
    std::cout << std::endl;
}
```

```
v1 = {-5.82538e+28; 4.59149e-41; 9.2347e-35}
```

Инициализация полей структуры
Значения по умолчанию

```cpp
#include<iostream>

struct Vector {
  float x { 0.f };
  float y = { 0.f };
  float z = 0.f;
};

int main()
{
    Vector v1;
    std::cout << "v1 = {";
    std::cout << v1.x << "; ";
    std::cout << v1.y << "; ";
    std::cout << v1.z << "}";
    std::cout << std::endl;
}
```

```
v1 = {0; 0; 0}
```

Выравнивание полей структуры

```cpp
#include<iostream>

struct MyStruct {
    int a;
    char b;
    int c;
};

int main()
{
    std::cout << "Size of MyStruct: ";
    std::cout << sizeof(MyStruct)  << std::endl;

    std::cout << "Size of each field in MyStruct: ";
    std::cout << sizeof(MyStruct::a)  << " + ";
    std::cout << sizeof(MyStruct::b)  << " + ";
    std::cout << sizeof(MyStruct::c)  << std::endl;
}
```

```
Size of MyStruct: 12
Size of each field in MyStruct: 4 + 1 + 4
```