

Приоритеты операторов

Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right →
2	a++ a-- type() type{} a() a[] . ->	Suffix/postfix increment and decrement Functional cast Function call Subscript Member access	
3	++a --a +a -a ! ~ (type) *a &a sizeof co_await new new[] delete delete[]	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT C-style cast Indirection (dereference) Address-of Size-of ^[note 1] await-expression (C++20) Dynamic memory allocation Dynamic memory deallocation	Right-to-left ←
4	.* ->*	Pointer-to-member	Left-to-right →
5	a*b a/b a%b	Multiplication, division, and remainder	
6	a+b a-b	Addition and subtraction	
7	<< >>	Bitwise left shift and right shift	
8	<=>	Three-way comparison operator (since C++20)	
9	< <= > >=	For relational operators < and ≤ and > and ≥ respectively	
10	== !=	For equality operators = and ≠ respectively	
11	a&b	Bitwise AND	
12	^	Bitwise XOR (exclusive or)	
13		Bitwise OR (inclusive or)	
14	&&	Logical AND	
15		Logical OR	
16	a?b:c throw co_yield = += -= *= /= %= <<= >>= &= ^= =	Ternary conditional ^[note 2] throw operator yield-expression (C++20) Direct assignment (provided by default for C++ classes) Compound assignment by sum and difference Compound assignment by product, quotient, and remainder Compound assignment by bitwise left shift and right shift Compound assignment by bitwise AND, XOR, and OR	Right-to-left ←
17	,	Comma	Left-to-right →

Указатели, ссылки, время жизни объектов

- `ptr.cpp`

Текстовая информация

- Представление строки - это последовательность байт.
- `char[]`, где в конце ставится `\0` байт, который является нулевым байтом и обозначает конец строки.
- `cstring` - работа со строками, которая пришла с C.
- в C++ пользуются `std::string` классом

int main

- `intmain.cpp`

Выравнивание

- Выравнивание - гарантирует размещение переменной так, чтобы адрес размещения был кратен размеру выравнивания

Невыровненные данные

- На некоторых платформах вызывает `Bus Error`
- Обращение к невыровненным данным требует два цикла обращения к памяти вместо одного
- `Undefined behavior`

Sizeof & alignof

type	X86 Linux		X64 Linux	
	size	alignment	size	alignment
char	1	1	1	1
short	2	2	2	2
int	4	4	4	4
long	4	4	8	8
long long	8	4	8	8
void *	4	4	8	8
float	4	4	4	4
double	8	4	8	8
long double	12	4	16	16

Struct

```
struct Structure {  
    char f1;  
    long long f2;  
    char f3;  
};
```

- `sizeof(Structure) == 24` на X86-64

```
struct Structure {  
    long long f2;  
    char f1;  
    char f3;  
};
```

- `sizeof(Structure) == 16` на X86-64

Struct

```
struct Structure {  
    char f1; // offset: 0  
    // padding + 7  
    long long f2; // offset: 8  
    char f3; // offset 9  
    // padding +7  
};
```

Максимальное выравнивание - 8 (поле f2), поэтому:

- `Structure` требует выравнивания 8
- `sizeof(Structure)` должен быть кратен 8

struct.cpp

Enum

```
enum class Color {  
    RED,  
    GREEN,  
    YELLOW,  
};
```

```
auto color = Color::GREEN;  
auto color2 = Color::RED;
```

Enum

```
switch (color) {  
case Color::RED:  
    std::cout << "RED" << std::endl;  
    break;  
case Color::GREEN:  
    std::cout << "GREEN" << std::endl;  
    break;  
case Color::YELLOW:  
    std::cout << "YELLOW" << std::endl;  
    break;  
default:  
    // UNREACHABLE  
    std::abort();  
}
```

Константы

```
namespace task_constants {  
    constexpr uint64_t UINT64CONST = 64;  
    constexpr auto STRINGCONSTANT = "da"; // char*  
    constexpr std::string_view STRINGVIEWCONST = "da";  
}
```