Тема 1: Типове данни. Основни аритметични операции. Вход и изход

def: Променлива - блок памет. Има име и стойност.

- Декларация - задаване на име на променливата
- Инициализация - задаване на стойност на променливата

Ако не инициализираме променливата не знаем какво има в нея

1. Примитивни типове данни:

- Целочислен тип (int, short, long ..).
- Числа с плаваща запетая (double, float ..).
- Булев тип (bool).
- Символен тип (char)

1.1 Цяло число - int

Колко байта е int? - В повечето случаи е 4, но може да е и 2. Зависи от операционната система.

1.2 Число с плаваща запетая - float(4 байта)/double(8 байта)

1.3 Bool - има стойност true(1)/false(0)

Това е тип, който съществува за наше улеснение. Вместо него може да се ползва int.

1.4 Символ - char

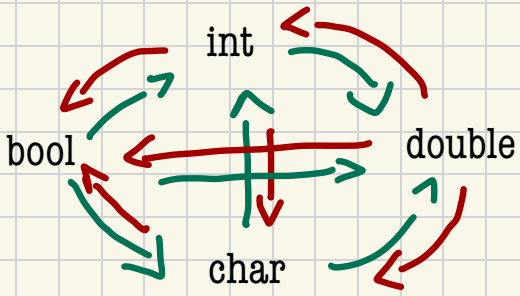
Въпрос: Защо ограждаме символите с ‘’?

Отговор: Защото може да имаме променлива именувана по същия начин като този символ.

char vs. int

Разликата между char и int е в това какво се отпечатва.

2. Преобразуване между типовете



- със загуба на информация
- без загуба на информация

3. Операторите

3.1 Аритметични оператори

• Събиране	+	3+2 = 5
• Изваждане	-	3-2 = 1
• Умножение	*	3*2 = 6
• Деление	/	5/2 = 2
• Деление с остатък	%	5%2=1

% и / са дефинирани само за int

3.2 Оператори за сравнение - връщат bool

• По-малко	<	2 < 5	True
• По-голямо	>	2 > 5	False
• По-малко или равно	<=	2 <= 5	True
• По-голямо или равно	>=	2 >= 2	True
• Равно на	==	1 == 1	True
• Не е равно на	!=	1 != 1	False

Не сравняваме 2 float-а/ double-и с ==.

Правим го по следния начин:

```
double a = 3.5;
double b = 3.4;
return abs(a-b) <= 0,0001;
```

3.3 Логически оператори

- Негация - !
- Логическо “или” - ||
- Логически “и” - &&

a	b	a || b	a && b	!a	!b
F	F	F	F	T	T
F	T	T	F	T	F
T	F	T	F	F	T
T	T	T	T	F	F

Характеристики на операторите:

- Приоритет
- Асоциативност
- Позиция на оператора спрямо аргумента - префиксен, инфиксен и суфиксен.

3.4 Оператори за присвояване - променят левия аргумент

+=, -=, *=, =, %=, /=

Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right
2	`a++` `a--`	Suffix/postfix increment and decrement	
	`type``()` *`type`*`{}`	Functional cast	
	`a()`	Function call	
	`a[]`	Subscript	
	`.` `->`	Member access	
3	`++a` `--a`	Prefix increment and decrement	Right-to-left
	`+a` `-a`	Unary plus and minus	
	`!` `~`	Logical NOT and bitwise NOT	
	(*`type`*)	C-style cast	
	`*a`	Indirection (dereference)	
	`&a`	Address-of	
	`sizeof`	Size-of[note 1]	
	`new` `new[]`	Dynamic memory allocation	
	`delete` `delete[]`	Dynamic memory deallocation	
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 ≤ respectively	
	`>` `>=`	For relational operators > and ≥ respectively	
10	`==` `!=`	For relational operators = and ≠ respectively	
11	`&`	Bitwise AND	
12	`^`	Bitwise XOR (exclusive or)	
13	`|`	Bitwise OR (inclusive or)	
14	`&&`	Logical AND	
15	`||`	Logical OR	
16	`a?b:c`	Ternary conditional[note 2]	Right-to-left
	`throw`	throw operator	
	`=`	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	
17	`,`	Comma	Left-to-right

https://stackoverflow.com/questions/55085938/post-increment-vs-assignment-in-c-operation-precedence-table