Семинар – група 1

Седмица 1

В съвремието, за разлика от преди, съществуват много езици за програмиране, които обслужват най различни нужди за софтуер. Тези езици се разделят на няколко типа.

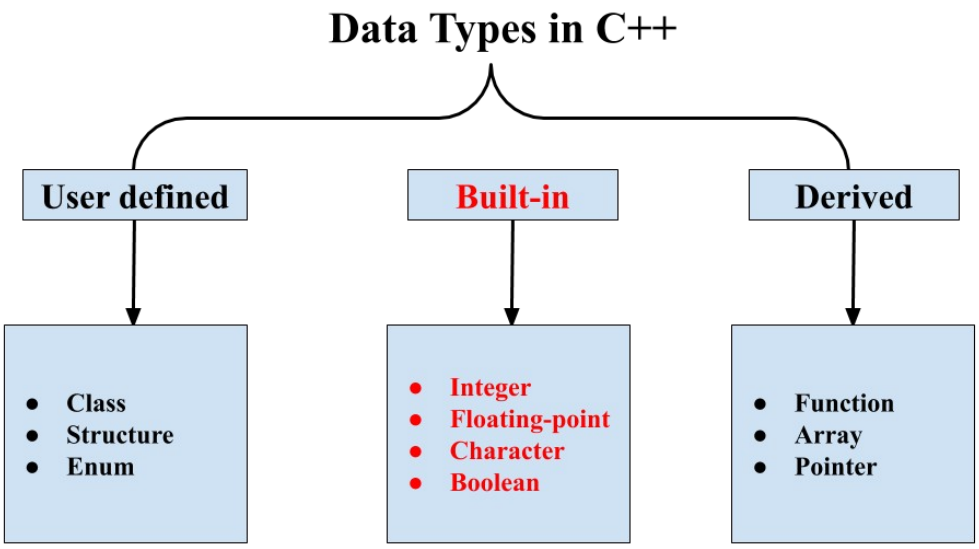
а) по начин на изпълнение – функционален, процедурен, логически и т.н.
б) по начин на структуриране – обектно-ориентиран, структурен и т.н.
в) по ниво на езика – високо ниво, средно ниво, ниско ниво, асемблерен, машинен
г) по типизация – слабо типизиран, силно типизиран.

C++ може да се счете към, така наречените, системни езици или по-точно описано език от средно ниво, който е силно типизиран, поддържащ обектно-ориентиран подход, както и структурен, и процедурен начин на изпълнение.

Защо системен език за програмиране? Защото сам по себе си езикът ни дава достъп до ресурсите на машината, на която изпълняваме софтуера , написан на C++. Тъй като езикът ни дава толкова възможности, това го прави и по-труден за осъвършенстване. В същото време го прави и добра основа за навлизане в света на програмирането, тъй като е подобен на повечето известни езици като Java, C# и т.н.

Както се спомена горе, езикът спада към собствена категория системен, произлизащ от C, затова неговите основни типове данни зависят от системата, на която работим, а именно архитектурата на системата /*64-битова или 32-битова*/. Вече рядко ще се срещнем с 32-битови машини, но те още са използваеми. Основната разлика, която ни интересува е целочисления тип “int”, който при 32-битови архитектури той с размер 2 байта, а при 64-битови е 4.

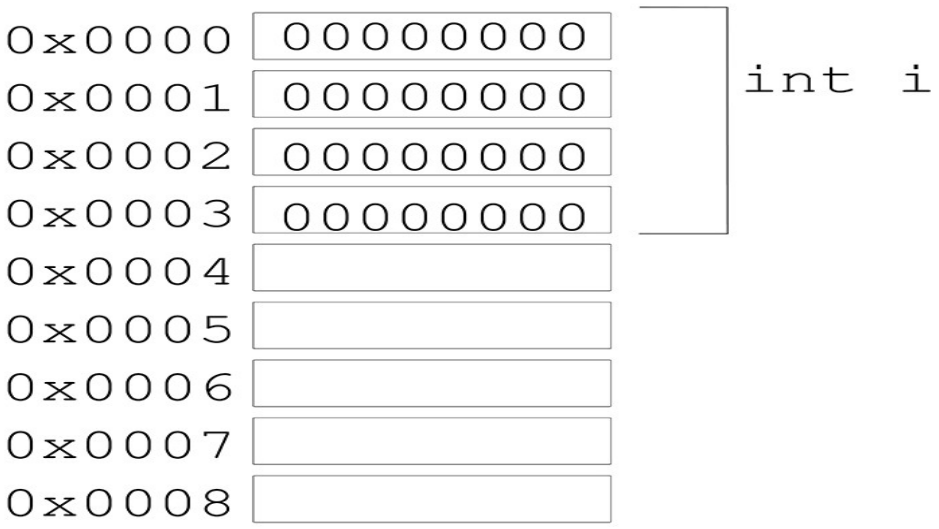
1. Основни типове данни.



Data type	Memory (bytes)	Minimum Value	Maximum Value
Bool	1	Logical Value T/F	Logical Value T/F
char	1	-128	127
unsigned Char	1	0	255
short int	2	-32768	32767
unsigned short int	2	0	65535
int	2	-32768	32767
unsigned int	2	0	65535
long int	4	-2147483648	2147483647
unsigned long int	4	0	4294967295
float	4	10^{-38}	10^{38}
double	8	10^{-308}	10^{308}
long double	10	10^{-4932}	10^{4932}

DATA TYPE	SIZE (IN BYTES)	RANGE
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

Int или целочисленият тип можем да го наречем като основен тип, който да ни е като основа за всички други типове. При деклариране на Int променлива се случва следното в паметта:



2. Първа програма на C++. Hello World. Основни аритметични оператори. Приоритет на операции.

```
1  #include <iostream>
2
3  int main() {
4
5      std::cout << "Hello world!" << std::endl;
6
7      return 0;
8  }
```

→ *вмъкване на библиотека*

→ *основната функция , отваряне на блок от код*
→ *оператори за писане в конзолата и нов ред*
→ *оператор за край на функция, последван от край на блок*

- Основни аритметични оператори - “+”, “-”, “*”, “/”.
- Оператор за търсене на остатък при деление - “%”.
- Особености при аритметични операции.

Допълнително: Побитови оператори - “&”, “|”, “^”, “<<”, “>>”.

Езиците за програмиране спазват същия приоритет на операциите, както в математиката.

лява	[
без асоциативност	++ --
без асоциативност	~ - (int) (float) (string) (array) (object) (bool) @
без асоциативност	instanceof
дясна	!
лява	* / %
лява	+ - .
лява	<< >>
без асоциативност	< <= > >= <>
без асоциативност	== != === !==
лява	&
лява	^
лява	|
лява	&&
лява	||
лява	? :
дясна	= += -= *= /= .= %= &= |= ^= <<= >>=
лява	and
лява	xor
лява	or
лява	,

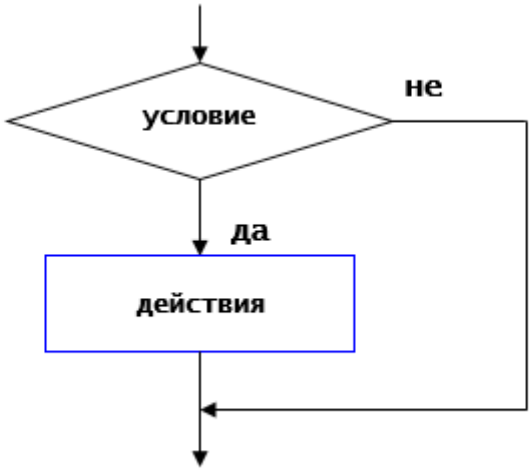
3. Булеви изрази и условен оператор / IF /.

3.1. Какво е булев израз?

- Това е такъв израз, който накрая на всички “сравнения” се оценява с **True** или **False**.
- Резултатът се запаметява в променлива или константа от тип **bool.**
- Оценява се с помощта на булеви оператори - “>”, “=”, “<”, “>=” , “<=”, “==”, ”!=”, “||”, “&&”.

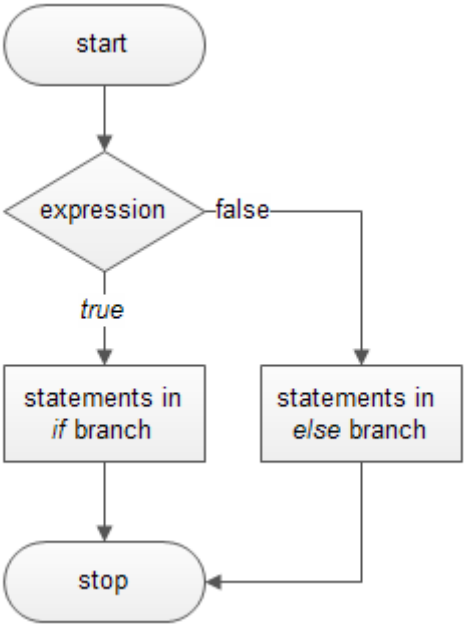
```
bool isEven = (number % 2 == 0);

bool isEven1 = !(number % 2);

bool isEven2 = !(number & 1);
```

3.2. Оператор за условие IF. Структура if-else. Тернарен оператор.




```
if(<условие>) {
    <оператор>
    <оператор>
    <оператор>
    ...
}
```

Действията в блока след if се изпълняват при стойност на условието “истина’. Тези действия може да са много, затворени в блок, или само едно действие. Условието може да е сложен булев израз.




```
if(<условие>) {
    <оператор>
    ...
} else {
    <оператор>
    ...
}
```

Тази конструкция е напълно по подобие на горната, но тук имаме два изхода вместо само един.

```
(<условие>) ? <оператор> : <оператор> ;
```

Тернарният оператор действа по напълно същия начин като конструкцията if-else с една много важна разлика. Докато при if-else можем да имаме цял блок от код при изходите, който да се изпълни, тук имаме само по **ЕДИН** оператор за изпълнение на всеки изход.

4. Задачи

0. Да се провери дали едно число, получено от стандартния вход, е четно и ако е такова, да се изведе в конзолата “Even”. В противен случай да се изведе “Odd”.

1. Да се напише програма, която получава две числа от стандартния вход и извежда по-голямото от тях.

2. Да се напише програма, която получава година и извежда в конзолата дали тя е високосна.

3. Да се напише програма, която получава трицифрено число от стандартния вход и проверява дали всичките му цифри са четни.

4. Да се напише програма, която получава месец и година и изписва в козолата броя дни в този месец.

5. Да се напише програма, която получава три числа от стандартния вход и проверява дали може да бъде съставен триъгълник с такива страни.

6. Да се напише програма, която получава три числа от стандартния вход и проверява дали може да бъде съставен триъгълник с такива ъгли.

7. Да се напише програма, която по пресмята корените на квадратно уравнение. Коефициентите се получават от стандартния вход. Ако уравнението няма корени, да се изведе подходящо съобщение.