Тема 4: Функции и оператор return.

def: Функция - блок от инструкции, които изпълняват дадена функционалност

```
<сигнатура> <идентификатор>( <параметър1>,<параметър2>,...){
  <тяло>
}
//<параметър> - състои се от <тип> и <идентификатор>
```

- Ако функция, която не е void не връща стойност => грешка при компилация.
- Извикването на функция е операция с много висок приоритет.
- Void е празен тип, който не връща резултат

Пример 1:

```
void hello_world() {
    std::cout << “Hello world!”;
}

int main() {
    hello_world();
}
```

Пример 2:

```
void print_next_number(int number) {
    std::cout << number + 1 << std::endl;
}

int main() {
    print_next_number(2);   // 3
    int a = 5;
    print_next_number(a);   // 5
}
```

Оператор return:

- прекратява изпълнението на функцията
- замества извикването ѝ с резултата

Пример 3:

```
int max(int num1, int num2) {
   return num1 > n um2 ? num1 :
num2;
}

int main() {
   int a = 5;
   int result = max(2,a);
   std::cout << result; //5
```

Плюсове от използването на функции:

- по - четим код
- избягват се повторения
- по - лесно за поддръжка
- подобрява се нивото на абстракция

Оператор **return** може да се използва и при функции, които не връщат нищо(void), тогава **return** единствено прекратява функцията.

Пример 4:

```
void print_even(int num) {
   if(num % 2 != 0){
      return;
   }
   std::cout << num << std::endl;
}

int main() {
   print_even(1);
   print_even(2);
}
```

Пример 5 (подаване на параметри по копие):

```
void multiply_by_two(int num) {
   n*=2;
}

int main() {
   int n = 5;
   multiply_by_two(n);
   std::cout << n << std::endl;//5
```

Плюсове от използването на функции:

- по - четим код
- избягват се повторения
- по - лесно за поддръжка
- подобрява се нивото на абстракция