

Lecture 7

Functions

What is a function?

A function is a *block of code* which performs a specific sequence of operations and only runs when it is *called*.

You can pass data elements, known as parameters (or arguments), into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

<http://ejudge.kz/reference/en/cpp/language/functions.html>

Function return types

That return a value:

- int
- double
- bool
- string
- ...
- etc, *any type you want*

That return nothing:

- void

Built-in functions

- `max()`

- `min()`

`<algorithm>`

- `sort()`

- `reverse()`

`<cmath>`

- `sqrt()`

`<cctype>`

- `tolower()`

- `toupper()`

- `isalpha()`

- `isdigit()`

- `isalnum()`

- `ispunct()`

Calling a function

Examples:

`max(4, 5);` – takes 2 int arguments and returns the larger one

`sqrt(9);` – takes an int/float/double argument and returns the square root of it

`pow(2, 6);` – returns the first argument to the power of the second argument (2 to the power of 6). The arguments can be of type double

Calling a function

Examples:

```
int addition(int a, int b)
```

.

.

.

```
int c = addition(2, 5);
```

```
// c = 7;
```



Calling a function

Examples:

```
int addition(int a, int b)
```

.

.

.

```
int c = addition(2, 5);
```

```
// c = 7;
```



The number of parameters and their order should be exactly the same as in the function declaration.

Otherwise, you will either get an error or your function will work improperly.

Local and global variables

Two types of variable scopes:

- Local Variables
- Global Variables

```
#include <iostream>
using namespace std;
int a; // global variable - outside all functions and other scopes
// in the global scope
int main() {
    int b; // local variable - inside the function
    return 0;
}
```


Function example

```
void greeting() {  
    cout << "Hello!" << endl;  
}
```

return type

function name

parameter list (currently empty)

```
void greeting() {
```

```
    cout << "Hello!" << endl;
```

```
}
```

function body, within curly
brackets

return type

function name

parameter list

string

greetSomeone

(string name)

{

return "Hello, " + name + "!";

}

return statement
with return value

function body, within curly
brackets

return type

function name

parameter list (currently empty)

```
int main()
```

```
{
```

```
// ...
```

```
// lines of code here ...
```

```
// ...
```

```
return 0;
```

```
}
```

return statement with
return value

function body, within curly
brackets

Void return type

```
void greeting() {  
    cout << "Hello!" << endl;  
    // return "Hello"; - mistake if the return  
type of your function is void  
}
```

Example, calculating the sum of all elements in a 1D array

```
int calculateSum(int n) {  
    int a[n];  
    for(int i = 0; i < n; i++) {  
        cin >> a[i];  
    }  
    int sum = 0;  
    for(int i = 0; i < n; i++) {  
        sum += a[i];  
    }  
    return sum;  
}
```

Calling the function

```
int main() {  
    int n;  
    cin >> n;  
    cout << calculateSum(n) << endl;  
    return 0;  
}
```

Example, accepting a 1D array as a parameter

```
int calculateSum(int a[], int n) {  
    int sum = 0;  
    for(int i = 0; i < n; i++){  
        sum += a[i];  
        a[i] = 0;  
    }  
    return sum;  
}
```


Calling the function

```
int main() {  
    int a[5] = {5, 9, 1, -3, 11};  
    int sum = calculateSum(a, 5);  
    cout << sum << endl;  
    return 0;  
}
```

Example, accepting a 2D array as a parameter

```
int calculateSum(int n, int m, int a[][100]) {  
    int sum = 0;  
    for(int i = 0; i < n; i++){  
        for(int j = 0; j < m; j++) {  
            sum += a[i][j];  
        }  
    }  
    return sum;  
}
```

Calling the function

```
int main() {  
    int n, m;  
    cin >> n >> m;  
    int a[n][100];  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < m; j++) {  
            cin >> a[i][j];  
        }  
    }  
    cout << calculateSum(n, m, a) << endl;  
    return 0;  
}
```

Additional materials

- Paper:
 - C++ How to Program, Seventh Edition, H. M. Deitel, P. J. Deitel:
 - Chapter 6, Sections 6.1 - 6.7, 6.10, 6.12 (available in the KBTU library);
- Digital:
 - informatics.msk.ru:
 - [Теоретический материал \(C++\): Функции - 1](#)
 - [Функции и процедуры. Рекурсия](#)
 - w3schools:
 - [C++ Functions](#)