I1101

Functions In C

Headlines

1. Introduction
2. Syntax to define a function in C
3. Instruction return
4. Examples
5. Files locations of the functions
6. Call a function
7. Exercises

Headlines

1. **Introduction**
2. Syntax to define a function in C
3. Instruction return
4. Examples
5. Files locations of the functions
6. Call a function
7. Exercises

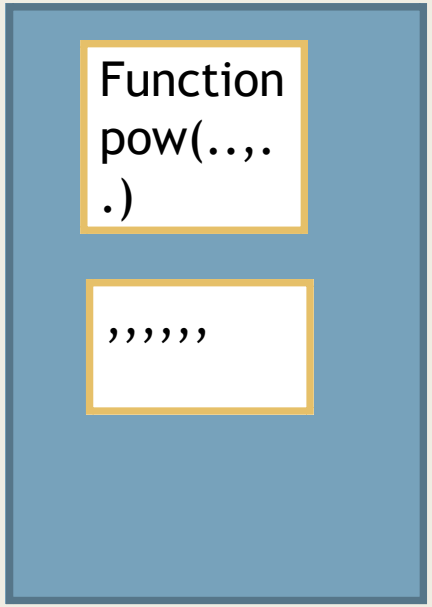
1. Introduction

- We have used some predefined function in our previous program,

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x,y;
cin>>x>>y;
cout<<pow(x,y);
return 0;
}
```

1. Introduction

math.h




```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x,y;
cin>>x>>y;
cout<<pow(x,y);
return 0;
}
```

(Call and Send inputs values)

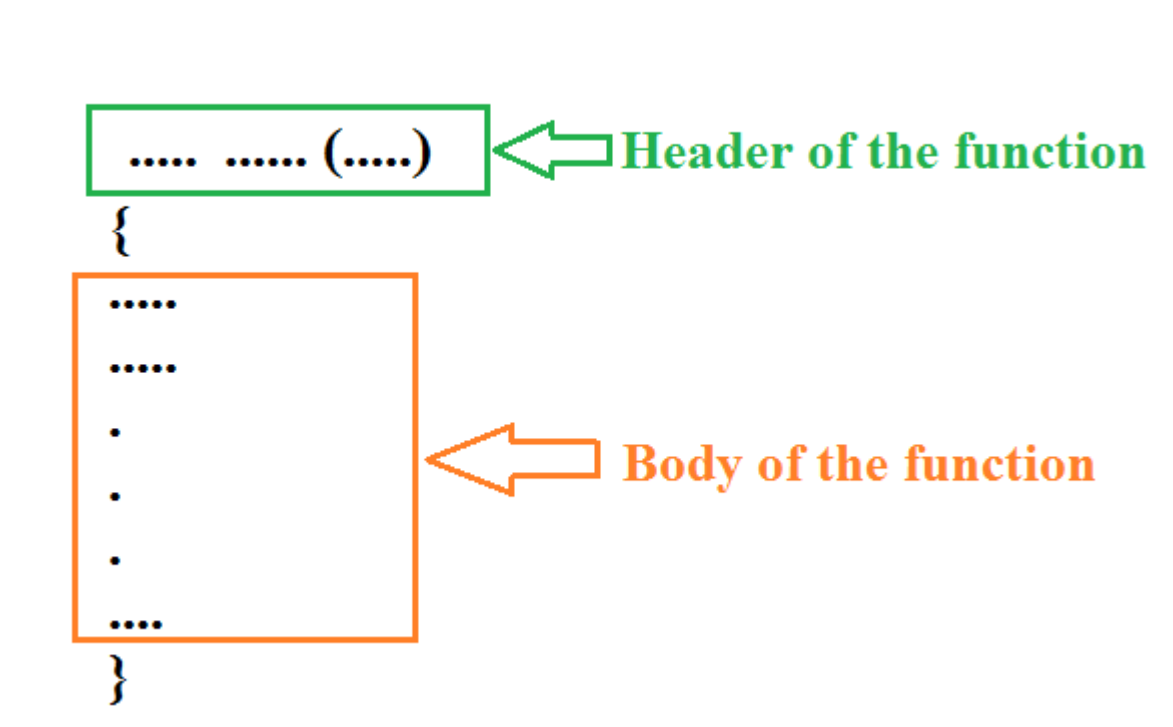
(return value at the line of calling)

Headlines

1. Introduction
2. **Syntax to define a function in C**
3. Instruction return
4. Examples
5. Files locations of the functions
6. Call a function
7. Exercises

2. Syntax to define a function in C

- A function has a header and a body.
- The header of the function contains THREE information
- The body of the function, enclosed between parentheses, contains the set of instructions to be executed when the function is called.



2. Syntax to define a function in C

```
<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)
  {
      ……
  }
```

- <ReturnedType>: Defines the type of the result/value provided by the function. It could be any type: **int, float, double, short, char, bool, long**, …etc.
- When the function is not returning a value, we use in this case **void** as the <returnedType> of the function.

2. Syntax to define a function in C

```
<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)
{
    ……
}
```

- <FunctionName>: Defines the name of the function to be used when called by others functions. The naming of the functions respect the same rules of those of variables and arrays.

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

- (…) : Between braces we define the list of parameters/inputs that the function needs to be received/taken from the caller/exterior to accomplish the required functionality/operation.
- <TypeParameter1> and <TypeParameter2> : Defines the types of the first and the second parameters of the function.
- <NameParameter1>, <NameParameter2>: Defines the names of the first and second parameters of the function

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ……
}
```

Header of the function

```
ReturnedType    FunctionName  ( TypeParameter1 NameParameter1,....)
{
  <local_declarations>
  <instructions>
    <instruction_returned_value>
}
```

Body of the function

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

Example 1:

Header of a Function that returns the factorial of an integer number

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

Example 1:

Header of a Function that returns the factorial of an integer

```
int factorial(int n)
{
.....

}
```

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

{

……

}

Example 1:

Header of a Function that returns the factorial of an integer

```
int factorial(int n)
{
…..

}
```

Example 2:

Header of a Function that returns the maximum of two float numbers.

2. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

Example 1:

Header of a Function that returns the factorial of an integer

```
int factorial(int n)
{
.....

}
```

Example 2:

Header of a Function that returns the maximum of two float numbers.

```
float max(float a, float b)
{
.....

}
```

3. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

Example 1:

Header of a Function that returns the factorial of an integer n

```
int factorial(int n)
{
.....

}
```

Example 2:

Header of a Function that returns the maximum of two float numbers.

```
float max(float a, float b)
{
.....

}
```

Example 3:

Header of a Function that returns a positive integer value read from the keyboard.

3. Syntax to define a function in C

<ReturnedType> <FunctionName> (<TypeParameter1> <NameParameter1>, <TypeParameter2> <NameParameter2>,…)

```
{
    ......
}
```

Example 1:

Header of a Function that returns the factorial of an integer n

```
int factorial(int n)
{
.....

}
```

Example 2:

Header of a Function that returns the maximum of two float numbers.

```
float max(float a, float b)
{
.....

}
```

Example 3:

Header of a Function that returns a positive integer value read from the keyboard.

```
int get_positive()
{
.....
}
```

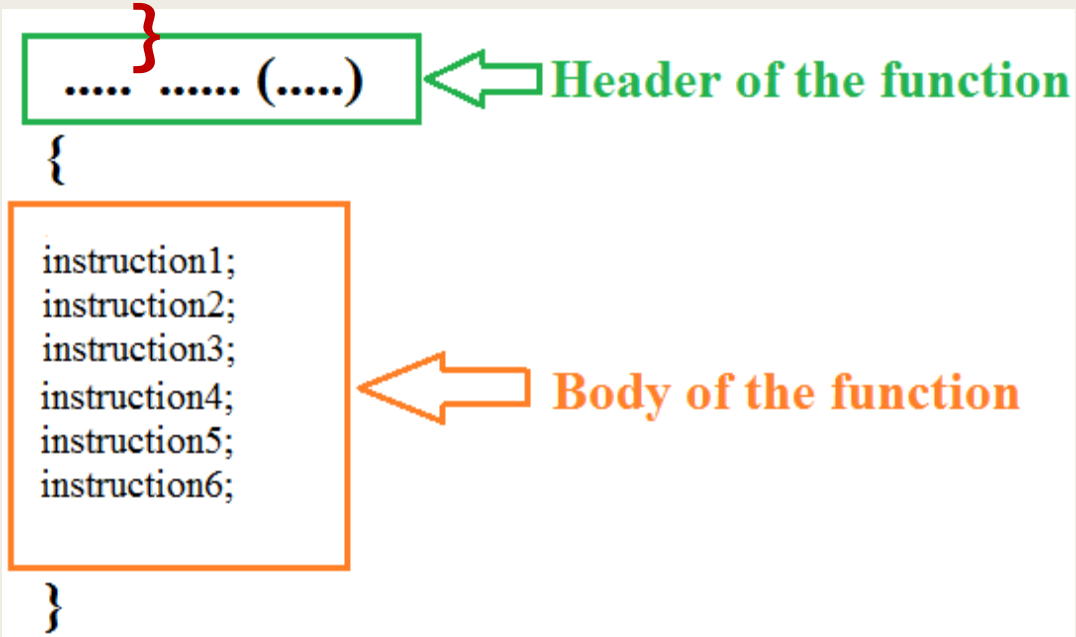
Headlines

1. Introduction
2. Syntax to define a function in C
3. **Instruction return**
4. Examples
5. Files locations of the functions
6. Call a function
7. Exercises

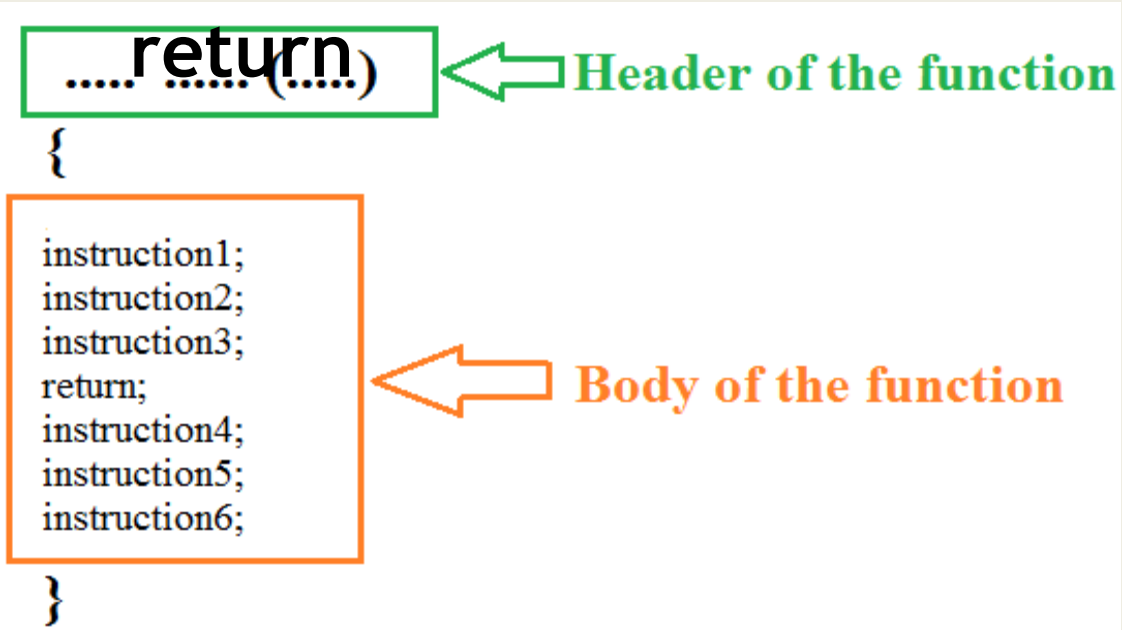
3. Instruction return

The Function will be **terminated** in one of the two cases:

1. Arriving to the closed brace



2. Execution of instruction return



3. Instruction return

a) **Terminates** the function without passing value to the caller (no output value from the function)

Example: **return;**

- ✓ *In this case the returned type of the function is void.*
- ✓ *No output value is providing to the caller*

Example 1:

```
void test( int a)
{
Instruction1;
Instruction2;
return;
}
```

3. Instruction return

a) **Terminates** the function without passing value to the caller (no output value from the function)

Example: **return;**

- ✓ *In this case the returned type of the function is void.*
- ✓ *No output value is providing to the caller*

Example 1:

```
void test( int a)
{
Instruction1;
Instruction2;
return;
}
```

```
void test( int a)
{
Instruction1;
Instruction2;

}
```

3. Instruction return

a) **Terminates** the function without passing value to the caller (no output value from the function)

Example: **return;**

- ✓ *In this case the returned type of the function is void.*
- ✓ *No output value is providing to the caller*

Example 2:

```
void test( int a)
{instruction1;
if (….)
    return;
Instruction2;
Instruction3;
}
```

3. Instruction return

b) **Terminates** the function **and returns** the value of *expression* (output) to the caller

Example: **return** *expression*;

- *expression* is the value returned to the caller (main function or the other functions);

3. Instruction return

a) **Terminates** the function **and returns** the value of *expression* (output) to the caller

Example: **return** *expression*;

- *expression* is the value returned to the caller (main function or the other functions);
- *expression* could be :
  - ✓ **Constant value**

```
int test( int a)
{
int b;
……
return 0;// return 1; …
}
```

3. Instruction return

a) **Terminates** the function **and returns** the value of *expression* (output) to the caller

Example: **return** *expression*;

- *expression* is the value returned to the caller (main function or the other functions);
- *expression* could be :
  - ✓ Constant value
  - ✓ **Variable**

```
int test( int
a)
{
int b;
……
return b;
}
```

4. Instruction return

a) **Terminates** the function **and returns** the value of *expression* (output) to the caller

Example: **return** *expression*;

- *expression* is the value returned to the caller (main function or the other functions);
- *expression* could be :
  - ✓ Constant value
  - ✓ Variable
  - ✓ **Expression**

```
int test( int
a)
{
int b;
……
return b+2*a;
}
```

3. Instruction return

a) **Terminates** the function **and returns** the value of *expression* (output) to the caller

Example: **return** *expression*;

- *expression* is the value returned to the caller (main function or the other functions);
- *expression* could be :
  - ✓ Constant value
  - ✓ Variable
  - ✓ Expression
  - ✓ **Call of a function**

```
int test( int a)
{
int b;
……
return pow(a,b);
}
```

Headlines

1. Introduction
2. Syntax to define a function in C
3. Instruction return
4. **Examples**
5. Files locations of the functions
6. Call a function
7. Exercises

4. Examples

<u>Example 1:</u>

Function that returns the factorial of an integer number

4. Examples

Example 1:

Function that returns the factorial of an integer number

```
int factorial(int n)
{
int i,f;
for(i=1,f=1;i<=n;i++)
    f=f*i;
return f;
}
```

4. Examples

Example 1:

Function that returns the factorial of an integer number

```
int factorial(int n)
{
int i,f;
for(i=1,f=1;i<=n;i++)
    f=f*i;
return f;
}
```

Example 2:

Function that returns the maximum of two float numbers.

```
float max(float a, float b)
{
float m;
if(a>b)
    m=a;
else
    m=b;
return m;
}
```

4. Examples

Example 3:

Function that returns a positive integer value read from the keyboard.

```
int get_positive()
{
  int x;
      do
    { cout<<“enter a positive value”;
      cin>>x;
      }while(x<=0);
return x;
}
```

Example 4:

Function that prints a line of n * separated by one space

```
void print_line(int n)
{
int i;
for(i=1;i<=n;i++)
cout<<“* “;
return;
}
```

Example 5:

Function that prints the sum of two integers.

```
void print_sum(int a, int b)
{
cout<<“Sum of “<<a<<“
and“<<b<<“=“<<a+b;
return;
}
```

4. Examples

<u>Example 6</u>:

Write a program that calculates the following sum:

$$S = \frac{x!}{x} + \frac{(2 * x)!}{x^2} + \cdots + \frac{(n * x)!}{x^n}$$

Where n>0 and x>0 are entered by the user.

```
#include<iostream>
using namespace std;
int main()
{
    int x,n,i,j,f;
    float S,p;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>n;
    }while(n<=0);

    do
    {
        cout<<"enter a positive integer number : ";
        cin>>x;
    }while(x<=0);

    for(i=1,S=0;i<=n;i++)
    {
        for(j=1,f=1;j<=i*x;j++)
            f=f*j;

        for(j=1,p=1;j<=i;j++)
            p=p*x;

        S=S+f/p;
    }
    cout<<"S="<<S;
}
```

Example 6:

$$S = \frac{x!}{x} + \frac{(2*x)!}{x^2} + \cdots + \frac{(n*x)!}{x^n}$$

Example 6:

$$S = \frac{x!}{x} + \frac{(2*x)!}{x^2} + \cdots + \frac{(n*x)!}{x^n}$$

```
#include<iostream>
using namespace std;
int main()
{
    int x,n,i,j,f;
    float S,p;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>n;
    }while(n<=0);

    do
    {
        cout<<"enter a positive integer number : ";
        cin>>x;
    }while(x<=0);

    for(i=1,S=0;i<=n;i++)
    {
        for(j=1,f=1;j<=i*x;j++)
            f=f*j;

        for(j=1,p=1;j<=i;j++)
            p=p*x;

        S=S+f/p;
    }
    cout<<"S="<<S;
}
```

Same need «reading a positive number» → repetition

Calculation of factorial of $i*x$ (i.e. $(i*x)!$)

Calculation of x^i

Example 6:

$$S = \frac{x!}{x} + \frac{(2*x)!}{x^2} + \cdots + \frac{(n*x)!}{x^n}$$

```
#include<iostream>
using namespace std;
int main()
{
    int x,n,i,j,f;
    float S,p;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>n;
    }while(n<=0);

    do
    {
        cout<<"enter a positive integer number : ";
        cin>>x;
    }while(x<=0);

    for(i=1,S=0;i<=n;i++)
    {
        for(j=1,f=1;j<=i*x;j++)
            f=f*j;

        for(j=1,p=1;j<=i;j++)
            p=p*x;

        S=S+f/p;
    }
    cout<<"S="<<S;
}
```

Using Functions

```
#include<iostream>
#include <math.h>
#include"myfunctions.h"
using namespace std;
int main()
{
    int x,n,i,j,f;
    float S,p;
    n=get_positive();
    x=get_positive();
    for(i=1,S=0;i<=n;i++)
        S=S+factorial(i*x)/pow(x,i);
    cout<<"S="<<S;
    return 0;
}
```

4. Examples

Example 7:

Write a program that reads a positive integer n and prints the following form:

```
enter a positive integer number : 10
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *

* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Example 7:

```
#include<iostream>
using namespace std;
int main()
{
    int n,i,j;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>n;
    }while(n<=0);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
            cout<<"* ";
        cout<<endl;
    }
    cout<<endl;
    for(i=n;i>=1;i--)
    {
        for(j=1;j<=i;j++)
            cout<<"* ";
        cout<<endl;
    }
return 0;
}
```

«Reading a positive number» we can use *get_positive function*

«Drawing a line of * with number i»
We can use *print_line* function

```
enter a positive integer number : 10
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *

* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Example 7:

```
#include<iostream>
using namespace std;
int main()
{
    int n,i,j;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>n;
    }while(n<=0);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
            cout<<"* ";
        cout<<endl;
    }
    cout<<endl;
    for(i=n;i>=1;i--)
    {
        for(j=1;j<=i;j++)
            cout<<"* ";
        cout<<endl;
    }
return 0;
}
```

Using Functions

```
#include<iostream>
using namespace std;
int main()
{
    int n,i;
    n=get_positive();
    for(i=1;i<=n;i++)
        print_line(i);
    cout<<endl;
    for(i=n;i>=1;i--)
        print_line(i);
    return 0;
}
```

Headlines

1. Introduction
2. Syntax to define a function in C
3. Instruction return
4. Examples
5. **Files locations of the functions**
6. Call a function
7. Exercises

5. Files locations of the functions

1. In the same file of the main function:
   *a) **before** the main function*
   *b) **After** the main function*
2. In a separate file other than the main function

5. Files locations of the functions

1. In the same file of the main function
   ***a) before** the main function*

```
int factorial (int n)
{
....
}
#include<iostream.h>
int main()
{
....
return 0;
}
```

OR

```
#include<iostream.h>
int factorial (int n)
{
....
}
int main()
{
....
return 0;
}
```

5. Files locations of the functions

1. In the same file of the main function
   ***a) before** the main function*

```
int factorial (int n)
{
....
}
#include<iostream.h>
int main()
{
....
return 0;
}
```

OR

```
#include<iostream.h>
int factorial (int n)
{
....
}
int main()
{
....
return 0;
}
```

5. Files locations of the functions

1. In the same file of the main functi
   ***a) before** the main function*

Saved as MyProgName.cpp

```
#include<iostream>
using namespace std;
//#include the other needed
// librairies

// Define all your functions to
//be used in the program

int func1( int s, int b)
{
....
}
void func2(float k)
{
....
}

....
//Write your main function
int main()
{
....
//call the functions
....
return 0;
}
```

5. Files locations of the functions

1. In the same file of the main function

   ***a) before** the main function*

<u>Example 1:</u>

Function that returns the max between two float numbers.

```
#include<iostream>
using namespace std;
float max(float a, float b)
{
    float m;
    if(a>b)
        m=a;
    else
        m=b;
    return m;
}
int main()
{
    float x,y,maximum;
    cout<<"enter values for x and y: ";
    cin>>x>>y;
    maximum=max(x,y);
    cout<<" max of "<<x<<" and "<<y<<" is ="<<maximum;
        return 0;
}
```

5. Files locations of the functions

1. In the same file of the main function
   *a) **before** the main function*

<u>Example 2:</u>
Function that prints the phrase:
Hello World

```
#include<iostream>
using namespace std;
void print()
{
    cout<<"Hello World";
    return;
}
int main()
{
    print();
    return 0;
}
```

```
Hello World
--------------------------------
Process exited after 0.04296 seconds with return value 0
Press any key to continue . . .
```

5. Files locations of the functions

1. In the same file of the main function

b) ***After the main function***:

in this case we MUST **add the prototype** of the function before the main function.

Prototype of the function contains the information of the header of the function. It indicates:

- ✓ the type of the data transmitted (output) by the function «returned type»
- ✓ The type(s) of the data received (inputs) by the function
- ✓ Name of the function

5. Files locations of the functions

1. In the same file of the main function

b) After the main function:

in this case we MUST **add the prototype** of the function before the main function.

Example 1:

test-0.cpp

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      int n;
6      cin>>n;
7      cout<<factorial(n);
8  return 0;
9  }
10 int factorial (int x)
11 {
12     int i,f=1;
13     for(i=1;i<=x;i++)
14         f=f*i;
15 return f;
16 }
17
```

Compiler (2) | Resources | Compile Log | Debug | Find Results | Close

Line	Col	File	Message
		C:\Users\Rima\Desktop\I1101\session 9\test-0.cpp	In function 'int main()':
7	19	C:\Users\Rima\Desktop\I1101\session 9\test-0.cpp	[Error] 'factorial' was not declared in this scope

Example 1:

test-0.cpp

```cpp
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    cout<<factorial(n);
return 0;
}
int factorial (int x)
{
    int i,f=1;
    for(i=1;i<=x;i++)
        f=f*i;
return f;
}
```

Compiler (2) | Resources | Compile Log | Debug | Find Results | Close

Line	Col	File	Message
		C:\Users\Rima\Desktop\I1101\session 9\test-0.cpp	In function 'int main()':
7	19	C:\Users\Rima\Desktop\I1101\session 9\test-0.cpp	[Error] 'factorial' was not declared in this scope

test.cpp

```cpp
#include<iostream>
using namespace std;
int factorial(int);
int main()
{
    int n;
    cin>>n;
    cout<<factorial(n);
return 0;
}
int factorial(int x)
{
    int i,f=1;
    for(i=1;i<=x;i++)
        f=f*i;
return f;
}
```

ources | Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Rima\Desktop\I1101\session 9\test.exe
- Output Size: 1.83193111419678 MiB
- Compilation Time: 0.64s
```

5. Files locations of the functions

1. In the same file of the main function

b) After the main function:

in this case we MUST **add the prototype** of the function before the main function.

Saved as MyProgName.cpp

```
#include<iostream>
using namespace std;
//Write the prototypes of all
//the defined functions written
//below the main function
.....
int func1(float);
//Write your main function
int main()
{
....
//call the functions
....
return 0;
}
// Define all your functions to
//be used in the program
int func1(float a)
{
.....
}
....
```

5. Files locations of the functions

1. In the same file of the main function

b) After the main function:

in this case we MUST **add the prototype** of the function before the main function.

```
1  #include<iostream>
2  using namespace std;
3  float max(float,float);
4  int main()
5  {
6      float x,y,maximum;
7      cout<<"enter values for x and y: ";
8      cin>>x>>y;
9      maximum=max(x,y);
10     cout<<" max of "<<x<<" and "<<y<<" is ="<<maximum;
11         return 0;
12 }
13 float max(float a, float b)
14 {
15     float m;
16     if(a>b)
17         m=a;
18     else
19         m=b;
20     return m;
21 }
22
```

5. Files locations of the functions

1. In the same file of the main function:
   *a) **before** the main function*
   *b) **After** the main function*
2. In a separate file other than the main function

5. Files locations of the functions

2. In a separate file other than the main function file

```
int factorial (int n)
{
....
}
```

myfunc.h
(saved as header file type)

5. Files locations of the functions

2. In a separate file other than the main function file

```
int factorial (int n)
{
....
}
```

myfunc.h
(saved as header file type)

```
#include<iostream.h>
 using namespace std;
# include "myfunc.h"
  int main()
  {
  ....  //call the function factorial
  return 0;
  }
```

test.cpp

5. Files locations of the functions

2. In a separate file other than the main function file

myfunction.h | test-0.cpp

```cpp
#include<iostream>
#include"myfunction.h"
using namespace std;
int main()
{
    int n;
    cin>>n;
    cout<<factorial(n);
return 0;
}
```

myfunction.h | test-0.cpp

```cpp
int factorial(int x)
{
    int i,f;
    for(i=1,f=1;i<=x;i++)
        f=f*i;
return f;
}
```

5. Files locations of the functions

2. In a separate file other than the main function file

example3.cpp | myfunctions.h

```cpp
#include<iostream>
#include"myfunctions.h"
using namespace std;
int main()
{
    int n,i;
    n=get_positive();
    for(i=1;i<=n;i++)
        print_line(i);
    cout<<endl;
    for(i=n;i>=1;i--)
        print_line(i);
return 0;
}
```

myfunctions.h

```cpp
#include<iostream>
using namespace std;
float max(float a, float b)
{
    float m;
    if(a>b)
        m=a;
    else
        m=b;
    return m;
}
int get_positive()
{
    int m;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>m;
    }while(m<=0);
    return m;
}
void print_line (int m)
{
    int j;
    for(j=1;j<=m;j++)
            cout<<"* ";
    cout<<endl;
    return;
}
int factorial(int n)
{
    int i,f;
    for(i=1,f=1;i<=n;i++)
    {
        f=f*i;
    }
    return f;
}
```

5. Files locations of the functions

2. In a separate file other than the main function file

```
#include<iostream>
#include <math.h>
#include"myfunctions.h"
using namespace std;
int main()
{
    int x,n,i,j,f;
    float S,p;
    n=get_positive();
    x=get_positive();
    for(i=1,S=0;i<=n;i++)
        S=S+factorial(i*x)/pow(x,i);
    cout<<"S="<<S;
    return 0;
}
```

myfunctions.h

```
#include<iostream>
using namespace std;
float max(float a, float b)
{
    float m;
    if(a>b)
        m=a;
    else
        m=b;
    return m;
}
int get_positive()
{
    int m;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>m;
    }while(m<=0);
    return m;
}
void print_line (int m)
{
    int j;
    for(j=1;j<=m;j++)
        cout<<"* ";
    cout<<endl;
    return;
}
int factorial(int n)
{
    int i,f;
    for(i=1,f=1;i<=n;i++)
    {
        f=f*i;
    }
    return f;
}
```

Headlines

1. Introduction
2. Syntax to define a function in C
3. Instruction return
4. Examples
5. Files locations of the functions
6. **Call a function**
7. Exercises

6. Call a function

i. The function is called using its name.

ii. When we call the function we needs to provide values (arguments) to its parameters (inputs values).

6. Call a function

i. The function is called using its name.

ii. When we call the function we needs to provide values (arguments) to its parameters (inputs values). These arguments could be:

- ✓ *Constant values*
- ✓ *Variables*

```
#include<iostream>
#include "myfunctions.h"
using namespace std;

int main()
{
    float x,y,maximum;
    cout<<max(1.2,10e6);
    cout<<"enter values for x and y: ";
    cin>>x>>y;
    maximum=max(x,y);
    cout<<" max of "<<x<<" and "<<y<<" is ="<<maximum;
    cout<<max(2*x-1,x-y+2);
    return 0;
}
```

myfunctions.h

```
#include<iostream>
using namespace std;
float max(float a, float b)
{
    float m;
    if(a>b)
        m=a;
    else
        m=b;
    return m;
}
int get_positive()
{
    int m;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>m;
    }while(m<=0);
    return m;
}
void print_line (int m)
{
    int j;
    for(j=1;j<=m;j++)
            cout<<"* ";
    cout<<endl;
    return;
}
int factorial(int n)
{
    int i,f;
    for(i=1,f=1;i<=n;i++)
    {
        f=f*i;
    }
    return f;
}
```

6. Call a function

iii. The returned value will replace the name of the function in the caller.

```
#include<iostream>
#include "myfunctions.h"
using namespace std;

int main()
{
    float x,y,maximum;
    cout<<max(1.2,10e6);
    cout<<"enter values for x and y: ";
    cin>>x>>y;
    maximum=max(x,y);
    cout<<" max of "<<x<<" and "<<y<<" is ="<<maximum;
    cout<<max(2*x-1,x-y+2);
    return 0;
}
```

maximum .cpp | myfunctions.h

```
#include<iostream>
using namespace std;
float max(float a, float b)
{
    float m;
    if(a>b)
        m=a;
    else
        m=b;
    return m;
}
int get_positive()
{
    int m;
    do
    {
        cout<<"enter a positive integer number : ";
        cin>>m;
    }while(m<=0);
    return m;
}
void print_line (int m)
{
    int j;
    for(j=1;j<=m;j++)
            cout<<"* ";
    cout<<endl;
    return;
}
```

Headlines

1. Introduction
2. Syntax to define a function in C
3. Instruction return
4. Examples
5. Files locations of the functions
6. Call a function
7. **Exercises**

Exercise 1

a. Write a function that displays the dividers of an integer n

b. Write a program that uses this function to display the dividers of the numbers $\leq n$

Exercise 1: Solution

myfunction.h | Exercise1.cpp

```cpp
#include<iostream>
#include"myfunction.h"
using namespace std;
int main()
{
    int a,i;
    a=get_positive();
    for(i=1;i<=a;i++)
        display_dividers(i);
return 0;
}
```

myfunction.h | Exercise1.cpp

```cpp
#include<iostream>
using namespace std;
void display_dividers(int n)
{
    int i;
    cout<<endl<<"Dividers of "<<n<<" are: ";
    for(i=1;i<=n;i++)
        if(n%i==0)
            cout<<i<<", ";
    return;
}
int get_positive()
{
    int x;
    do
    {
        cout<<" enter a positive value: ";
        cin>>x;
    }
    while(x<=0);
    return x;
}
```

```
 enter a positive value: 10

Dividers of 1 are: 1,
Dividers of 2 are: 1, 2,
Dividers of 3 are: 1, 3,
Dividers of 4 are: 1, 2, 4,
Dividers of 5 are: 1, 5,
Dividers of 6 are: 1, 2, 3, 6,
Dividers of 7 are: 1, 7,
Dividers of 8 are: 1, 2, 4, 8,
Dividers of 9 are: 1, 3, 9,
Dividers of 10 are: 1, 2, 5, 10,
--------------------------------
Process exited after 1.501 seconds with return value 0
Press any key to continue . . .
```

Exercise 2:

a. Write a function named Print_Line that draws a line of n characters c, where n and c are given as parameters

b. Write a program that uses the function get_positive() to read a positive integer number n, then prints the triangle of n lignes of character c using the function Print_Line

Exercise 2: Solution

a. Write a function named Print_Line that draws a line of n characters c, where n and c are given as

myfunction.h | Exercise2.cpp

```cpp
#include<iostream>
using namespace std;
void Print_Line(char c, int n)
{
    int i;
    for(i=1;i<=n;i++)
        cout<<c;
    return;
}
int get_positive()
{
    int x;
    do
    {
        cout<<" enter a positive value: ";
        cin>>x;
    }
    while(x<=0);
    return x;
}
```

Exercise 2: Solution

myfunction.h | Exercise2.cpp

```cpp
#include<iostream>
using namespace std;
void Print_Line(char c, int n)
{
    int i;
    for(i=1;i<=n;i++)
        cout<<c;
    return;
}
int get_positive()
{
    int x;
    do
    {
        cout<<" enter a positive value: ";
        cin>>x;
    }
    while(x<=0);
    return x;
}
```

```cpp
#include<iostream>
#include"myfunction.h"
using namespace std;
int main()
{
    char c;
    int n,i;
    cout<<" enter the character to display the triangle :";
    cin>>c;
    n=get_positive();
    for(i=1;i<=n;i++)
    {
        Print_Line(' ',n-i);
        Print_Line(c,2*i-1);
        cout<<endl;
    }
return 0;
}
```

```
 enter the character to display the triangle :*
 enter a positive value: 10
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************

--------------------------------
Process exited after 8.621 seconds with return value 0
Press any key to continue . . .
```

```
 enter the character to display the triangle :+
 enter a positive value: 6
     +
    +++
   +++++
  +++++++
 +++++++++
+++++++++++

--------------------------------
Process exited after 9.98 seconds with return value 0
Press any key to continue . . .
```

```
 enter the character to display the triangle :-
 enter a positive value: 10
         -
        ---
       -----
      -------
     ---------
    -----------
   -------------
  ---------------
 -----------------
-------------------

--------------------------------
Process exited after 8.499 seconds with return value 0
Press any key to continue . . .
```

Exercise 3

Consider the general term U_n defined by:

$$U_0 = 1, U_1 = -1, U_n = 2 * U_{n-1} + U_{n-2} + 3 \ (n \geq 2)$$

a. Write the function U(int n) that calculates the value of the term U_n of the sequence (U_n)

b. Write a function that uses the first function U(int n) to calculates the following sum :

$$S = \sum_{i=0}^{n} U_n$$

c. Write a simple program to test these functions.

Exercise 3: solution

$U_0 = 1, U_1 = -1,$

$$U_n = 2 * U_{n-1} + U_{n-2} + 3 \ (n \geq 2)$$

myfunction.h | exercise3.cpp

```cpp
#include<iostream>
using namespace std;
int U(int n, int U0, int U1)
{
    int i, A=U0, B=U1, R;
    if (n==1)
        R=U1;
    else
        if(n==0)
            R=U0;
        else
            for(i=2;i<=n;i++)
            {
                R=2*B+A+3;
                A=B;
            B=R;
            }
        return R;
}
int Sum(int n,  int U0, int U1)
{
    int i, S;
    for(i=0,S=0;i<=n;i++)
        S=S+U(i,U0,U1);
    return S;
}
```

myfunction.h | exercise3.cpp

```cpp
#include<iostream>
#include"myfunction.h"
using namespace std;
int main()
{
    int n;
    n=get_positive();
    cout<<"U"<<n<<"="<<U(n,1,-1);
    cout<<" and S="<<Sum(n,1,-1);
    return 0;
}
```

```
 enter a positive value: 0
U0=1 and S=1
--------------------------------
Process exited after 2.286 seconds with return value 0
Press any key to continue . . .
```

```
 enter a positive value: 1
U1=-1 and S=0
--------------------------------
Process exited after 1.272 seconds with return value 0
Press any key to continue . . .
```

```
 enter a positive value: 3
U3=6 and S=8
--------------------------------
Process exited after 2.914 seconds with return value 0
Press any key to continue . . .
```

```
 enter a positive value: 5
U5=43 and S=68
--------------------------------
Process exited after 3.577 seconds with return value 0
Press any key to continue . . .
```