

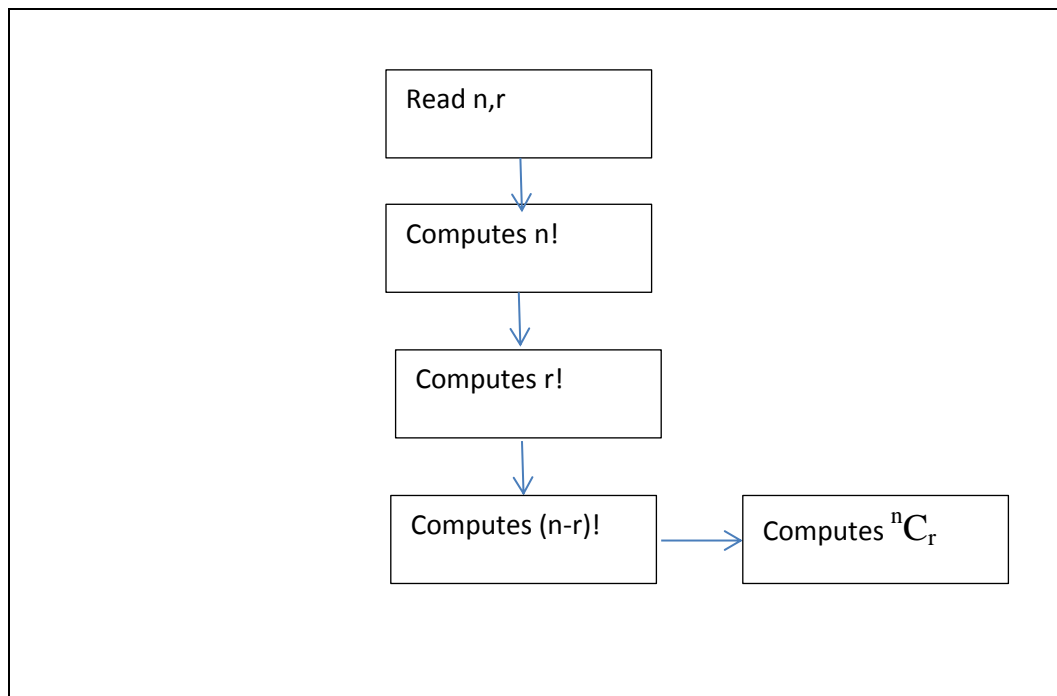
Chapter 5: Functions

5.1 Introduction

To make large programs manageable, programmers modularize them into subprograms. These subprograms are called functions. A function is defined as follows: (1) it is an independent section of C++ code that performs a specific task. (2) it is a block of instructions that is executed when it is called from some other point of the program. In mathematics, a function “f” is defined as $f(x)=y$, where “f” is the function name, “x” is an argument, and y is the result. The objectives of functions are given in terms of the following:

1. If a part of a program is need in more than one place in a program, or in different programs, you can write it once and use it many times.
2. Using functions greatly enhances the program’s readability because it reduces the complexity of the function main
3. While working on one function, you can focus on just that part of the program and construct it, debug it, and perfect it.

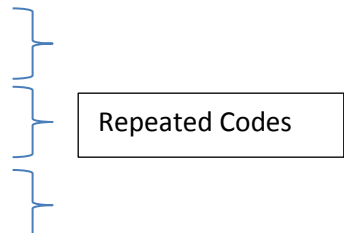
The following flowchart illustrates the idea for the function for the program that computes ${}^nC_r = n!/r!(n-r)!$



```

#include <iostream.h>
Main()
{
    Int n,r;
    Cin>>n>>r;
    Long factn=1;
    For (int i=1;i<=n;i++) factn*=I;
    Long factr=1;
    For (int i=1;i<=r;i++) factr*=I;
    Long factn_r=1;
    For (int i=1;i<=n-r;i++) factn_r*=I;
    Cout<<n<<"C'<<r<<"='<<factn/(factr*factn_r);
}

```



4.2 Types of functions and Its Declaration

There are two types of functions: pre-defined functions and user-defined functions. The following table illustrates the differences between each of them:

	Pre-defined functions	User-defined functions
Definition	Built in compiler and the programmer is not define it	Defined by programmer
Objective	Support the programmer	Do a specific task
Existence	Included in the header file such as [math.h] and [iostream.h]	Not necessary included in the header file
Example	Rand() function defined in <cstdlib.h>, sqrt() function defined in <math.h>	Fact(n), search(array,number)

The definition of function consists of two parts:

1. Header of function: it consists of (1) type return, (2) name of function, (3) arguments enclosed by parenthesis
2. Body of function: it consists of statements enclosed by parenthesis as follows {statements}. One of such statements is the return statement which is written as: **return expression;**

An example for a header is as follows:

Float mean(int a, int b, int c), Where

The name of the function is “mean”, the return type is “float”, and the arguments are a, b, and c

The general form to define a function is:

```
typeReturn NameOfFunction(argument1, argument2, ...argumentn) // header
{
Statements
Return statement
} // body
```

where:

- typeReturn: is the type of data returned by the function. If no value returned then the type is “**void**”.
- NameOfFunction : is the name by which it will be possible to call the function. The name of the function is an identifier and it is used to call the function. We can't define function without name.
- Arguments : arguments are variable declaration, where each argument consists of a type of data followed by its identifier, like in a variable declaration (e.g. int x). They allow passing parameters to the function when it is called. The different parameters are separated by commas. The argument of the function can be empty as follows: nameOfFunction(); or nameOfFunction(void); where void indicates that the function has no parameters. We can use the array as argument in the function as follows: type name[], e.g. float mean(int x[], int y). When a two dimensional array is passed to a function, the first dimension is not specified, while all the remaining dimensions are specified. For example: type name[][dim2] For example: type name[][dim2]
- typeReturn, NameOfFunction, and arguments are called function header. When we add a semicolon after the arguments we call it function prototype.
- Statement is the function's body. It can be a single instruction or a block of instructions.
- One of the statements in the body of the function is the return statement which is written as: **return expression;**
Where

- Return is a reserved word.
- The type of the expression must be the same as the “typeReturn” in the header file. Example: the following example is invalid:
`Int mean(float x, float y) {return (x+y)/2;}`
- The return statement should be at the end of the function. The statements after return are not executed
- The body of the function can contain more than one return statement
 Example: `int max(int x, int y) {if (x>=y) return x; else return y; }`
- A return statement returning only one value
 Example: `int swap(int x, int y) {int z=x; x=y; y=z; return x; return y;}` The output is x.
- If the function has return type then the body of it must contains a return statement.
- When the return statement is executed the following are done:
 - Returns a value that computed inside the function.
 - Terminating the execution of the function.

4.2 Function Headers Examples

Write a header of a function in the following cases:

1. A function that computes the maximum of two real numbers.
2. A function that writes the message “C++ is a programming language”.
3. A function that searches for an element x in the array “a” of n elements.

Solution:

1. `Float maximum(float x, float y)`
2. `Void display()`

3. Int search (int x, int a[], int n)

4.3 Declaration of Function

You must declare the function before it use. There are two ways:

1. Prototype definition: is the function heading with added semicolon and we put it before using it and then place the complete definition after the main.

The general form is as follows:

```
#include<>
```

```
typeReturn NameOfFunction(argument1, argument2, ...argumentn);
```

```
.....
```

```
Void main()
```

```
{ Some statements for the main function }
```

```
typeReturn NameOfFunction(argument1, argument2, ..., argumentn)
```

```
{ Statements }
```

Example:

```
#include<>
```

```
Float mean(int x, int y);
```

```
.....
```

```
Void main() { ..... }
```

```
Float mean(int x, int y)
```

```
{ return (x+y)/2.0; }
```

2. Complete definition: place the complete definition before the main.

Example: #include<>

```
Float mean(int x, int y) {return (x+y)/2.0;}
```

.....

```
Void main() { statements with a function call }
```

4.4 How to use function

We use the function by calling it. The general form of calling is as follows:

nameOfFunction(actual parameters)

Where the actual parameters are the values passed to function. The actual parameters can be constant or variable.

- ❖ **Program#1:** The following program uses a function called max with two parameters and returns the larger of the two values when we call it.

```
#include <iostream>
```

```
Using namespace std;
```

```
Int max(int x, int y) {if (x<y) return y; else return x;}
```

```
Int main()
```

```
{
```

```
int m,n;
```

```
do {
```

```
Cin>>m>>n;
```

```
 Cout << "\t max("<<m<<","<<n<<")="<<max(m,n)<<endl; }
```

```
While (m!=0); }
```

```
system ("pause");
```

```
}
```

- ✓ **Sample Run:**

```
5 8      4 -3    0 0
```

```
Max(5,8)=8    max(4,-3)=4    max(0,0)=0
```

```
Press any key to continue ...
```

- ❖ **Program#2:** The following program is the same as the above one But here

the function's declaration appears above the main program using prototype function.

```
#include <iostream>
Using namespace std;
Int max(int x, int y); automatically assigned with zero value.
Int main()
{
int m,n;
do {
Cin>>m>>n;
Cout << "\t max("<<m<<","<<n<<")="<<max(m,n)<<endl; }
While (m!=0); }
system ("pause");}
Int max(int x, int y) {if (x<y) return y; else return x;}
```

- ❖ **Program#3:** The following program find the value of y, where $y=a!+b!-c!$

```
#include <iostream>
Using namespace std;
Int fact(int n) { int f,I;
F=1;
For (i=2;i<=n;i++) {f=f*I; return f;}
Int main()
{
int y,a,b,c;
Cout << "Enter three numbers\n";
Cin>>a>>b>>c;
Y=fact(a)+fact(b)-fact(c);
Cout << "y="<<y<<endl;
system ("pause");
}
```

✓ **Sample Run:**

```
Enter three numbers
3 4 2
Y=14
Press any key to continue ...
```

- ❖ **Program#4:** The following program find the value of y, where $y=1!+2!+\dots+10!$

```
#include <iostream>
```

```

Using namespace std;
Int fact(int n) { int f,I;
F=1;
For (i=2;i<=n;i++) {f=f*I; return f;}
Int main()
{
int y=0;i;
for(i=1;i<=10;i++) {y=y+fact(i);}
Cout << "y="<<y<<endl;
system ("pause");
}

```

✓ **Sample Run:**

Y=230

Press any key to continue ...

4.5 Review Questions

Name:

Group#:

Results:

Instructor Signature:

1. Where can the declaration of a function be placed?
2. What is the difference between a function's declaration and its definition?
3. What are the advantages of using functions in the program?
4. Write and test the following average() function that returns the average of four float numbers: float average (float x, float y, float z, float m)
5. Write a program that reads an integer array with 7 integers and use a function that searches for an element x in such array.
6. Write a program that uses a function that writes the message "C++ is a programming language".
7. Write a program to calculate:

$$Y = (2+3+4+\dots+10) / (1*2*3*\dots*20) * (1+4+7+\dots+22) / (2*4*8*\dots*30)$$