Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

(Week 13) Lecture 25 and 26

Objectives: **Learning objectives of this lecture are**

- **Functions**
- **Why functions?**
- **Advantages of functions.**
- **Parameterized and function return type.**

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

Lecture # 25 Functions

So far we have discussed different type of program involving conditional statement, loops, arrays and structures. We shall be discussing function in this lecture. Following functions come into mind. Why we need functions? How to make them? How to use them? What are advantages of functions? To get answer of this question let us look at a simple program that does not involves functions. Given below is the statement of program.

Statement.

Consider a scenario where you are supposed to make a menu driven program for package description of cellular company. You have to display menu for subscription of call, SMS and internet packages. Cellular company wants you to display following menu in a sequence.

- Press C for call subscription.
- Press S for SMS subscription.
- Press I for internet subscription.

After user have entered a valid input from above menu. Display menu given below.

- Press d for daily package.
- Press w for weekly package.
- Press m for monthly package.

After both inputs are valid from above choice display following statement.

you have subscribed monthly package for call.

Note that above message is dependent on user input.

Solution:

```
#include "iostream"
using namespace std;
void main(){
	char opt;
```

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
        cout << "Press C for call\n";
        cout << "Press S for SMS\n";
        cout << "Press I for internet\n";
        cin >> opt;
 if (opt == 'C')
 {
  cout << "Press D for daily";
  cout << "Press M for monthly";
  cout << "Press I for internet";
  cin >> opt;
    if (opt == 'D')
    cout<<"Subscribe daily call";
    else if (opt == 'M')
    cout<<"Subscribe monthly call";
    else if (opt == 'W')
    cout<<"Subscribe weekly call";
    else
     cout << "Wrrong";
 }
else if (opt == 'S')
{
 cout << "Press D for daily";
 cout << "Press M for monthly";
 cout << "Press I for internet";
 cin >> opt;
   if (opt == 'D')
   cout<<"Subscribe daily SMS";
   else if (opt == 'M')
   cout<<"Subscribe monthly SMS";
   else if (opt == 'W')
   cout<<"Subscribe weekly SMS";
   else
    cout << "Wrrong";
}
else if (opt == 'I')
{
 cout << "Press D for daily";
 cout << "Press M for monthly";
 cout << "Press I for internet";
 cin >> opt;
   if (opt == 'D')
   cout<<"Subscribe daily net";
   else if (opt == 'M')
   cout<<"Subscribe monthly net";
   else if (opt == 'W')
   cout<<"Subscribe weekly net";
   else
    cout << "Wrrong";
}
else
    cout << "Wrrong";

}//end of main
```

```
cout << "Press D for daily\n";
cout << "Press W for weekly\n";
cout << "Press M for monthly\n";
```

```
if (opt == 'C')
{

}
else if (opt == 'S')
{

}
else if (opt == 'W')
{

}
```

```
cout << "Press D for daily";
cout << "Press M for monthly";
cout << "Press I for internet";
```

Analyze that above three statement is written 9 times in three different if matching C, S and W. Why not write it once? And use it in each block. How it is done? It is done through functions. Let us discuss functions.

Components of Function:

1. Declaration of function.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

2. Definition of function.
3. Calling function.

Declaration of function:

A function declaration consist of further three portions

1. Return type of function.
2. Name of function.
3. Parameters.

return type name of finction parameter.

void display ( ) ;

Return type	Meaning
void	Function will not return any value
int	Function will return an integer value.
float	Function will return a float value.
double	Function will return boolean value
long	Function will return long type value.
short	Function returns a short type value.
string	Function returns string type value.
User defined type	Function will return a user defined type. (Mentioned in return type portion)

name of function must be meaningful and same rule should be applied on name as that of variable. Parameter of function is mention in parenthesis ( ) if parenthesis is empty it means function is parameter less. Inside these parenthesis variable can be declared which we will see in lecture number 26. Function declaration is also known as function proto type or signature of a function. We will declare function above main. Figure given below further explains function declaration.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

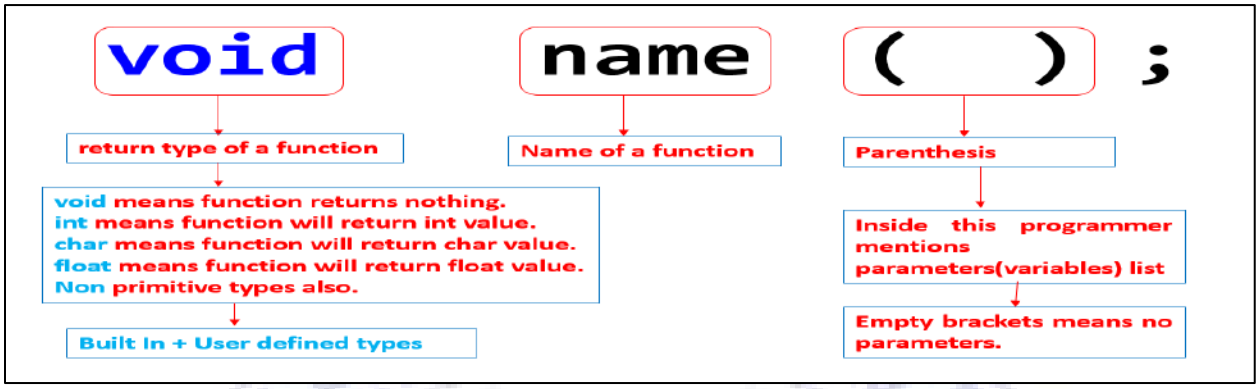


Figure 1 Function Declaration

Definition of Function:

So far we have written definition of main function. Now let us define a function name display that contains three cout statements as given below.

```
cout << "Press D for daily";
cout << "Press M for monthly";
cout << "Press I for internet";

void display ()
{
 cout << "Press D for daily";
 cout << "Press M for monthly";
 cout << "Press I for internet";

}
```

Definition of functions contains body and inside body of function code/functionality of function is written. Function can be define after main (if declared) and above main (if not declared).

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
void display()
{
  cout << "Press D for daily\n";
  cout << "Press W for weekly\n";
  cout << "Press M for monthly\n";
}
```

If declared than define after main.

If not declared than define above.

Figure 2 Function definition

Calling Function:

Function is called by its name for example if we call display function we will be calling as

display ( );

Note: When function is called control shifts to the body of function. After executing function it returns to the point from where function is called. Figure given below explains function call.

```
display();
```

Must be defined before calling.

When function is called control shifts into body of a function, where it is defined.

After executing body function returns back from where it is called.

Figure 3 Function call

Now let us solve above program with function.

Solution of program with function:

```
#include "iostream"
using namespace std;
void display(); // declaration
void main(){
       char opt;
       cout << "Press C for call\n";
       cout << "Press S for SMS\n";
       cout << "Press I for internet\n";
       cin >> opt;
  if (opt == 'C')
 {
    display();
    cin >> opt;
    if (opt == 'D')
```

```
else if (opt == 'S')
{
   display();

  cin >> opt;
```

```
else if (opt == 'I')
{
   display();

 cin >> opt;
```

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**


```
    cout<<"Subscribe daily call";
    else if (opt == 'M')
    cout<<"Subscribe monthly call";
    else if (opt == 'W')
    cout<<"Subscribe weekly call";
    else
     cout << "Wrrong";
 }
```

```
    if (opt == 'D')
    cout<<"Subscribe daily SMS";
    else if (opt == 'M')
    cout<<"Subscribe monthly SMS";
    else if (opt == 'W')
    cout<<"Subscribe weekly SMS";
    else
     cout << "Wrrong";
}
```

```
    if (opt == 'D')
    cout<<"Subscribe daily net";
    else if (opt == 'M')
    cout<<"Subscribe monthly net";
    else if (opt == 'W')
    cout<<"Subscribe weekly net";
    else
     cout << "Wrrong";
}
```

```
else
    cout << "Wrrong";

}//end of main
void display()
{
 cout << "Press D for daily";
 cout << "Press M for monthly";
 cout << "Press I for internet";

}
```

Note that in above program instead of writing statements again and again it's written in a function and is called. If in above program definition of display is moved above main than there is no need of declaration. Instead if declaration we can just write definition of a function.

Advantages of Function:

1. Code reusability (Writing code once and using it again)
2. Organized code (Code written in function is organized as compared to written without function.)
3. Increase Program readability. (Easy in reading and tracing problem)
4. Easy maintenance (Code is easily in maintenance because if error occurs we can direct to function having problem.

Local Variables of a function

When a function is called a memory in stack is allocated to it and all variables inside body of function are considered its local variable. Local variable are not directly accessible in a another function. And when function execution is finished memory is taken from function and its local variables are destroyed. Figure given below further explains it.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
#include "iostream"
using namespace std;
void add()
{
 int num1, num2;
 cin >> num1 >> num2;
 int sum = num1 + num2;
}

void main()
{
 int num1, num2;
 cin >> num1 >> num2;
 int sum = num1 + num2;
 add();

}
```

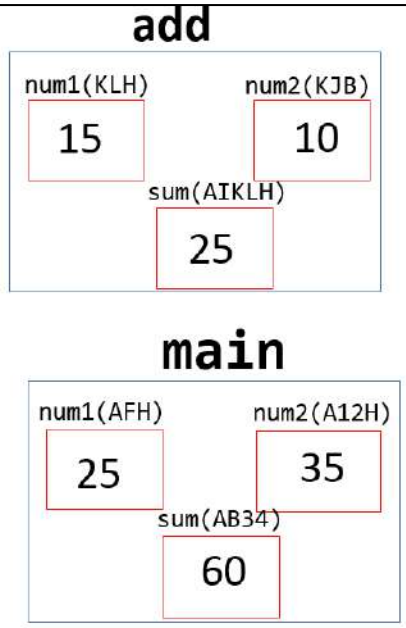


Figure 4 Local variable of function.

In above figure there are two functions main and add both of them having three variables with name num1, num2 and sum but note that num1 of main has different memory address than num1 of add function inside add function when cin >> num1 >>num2; is executed it takes input in KLH and KJB memory respectively and same statement when executed in main it takes input in AFH and A12H. Note that every time a function is called new memory is allocated to its variables.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

Lecture # 26 Functions with return type and Parameters

In lecture number 23 we started studying functions, and have discussed function having void return type (that is function not returning any value) and has no parameters. In this lecture we shall be discussing about function having return type also we will be discussing parametrized function. Let us discuss a key word.

Key word return: when a return statement is executed a function is returned to the point from where it is called. We will be looking at different figures for getting better understanding of return keyword.

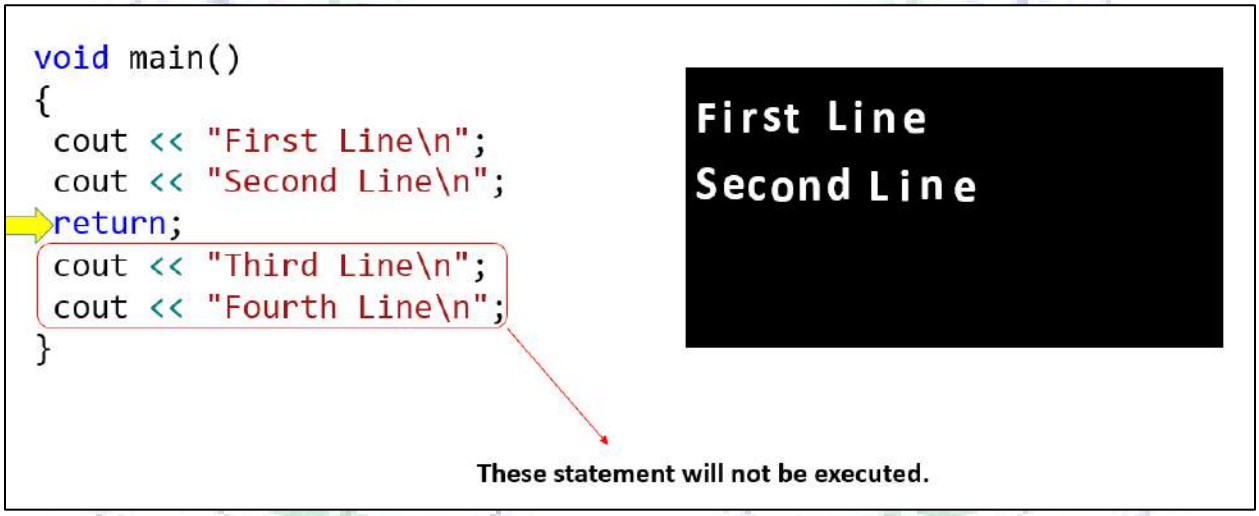


Figure 1 return with void return type

When a return statement is executed function goes back to the point from where it is called in above case program will be terminated.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
void print()
{
 cout << "PF\n";
 cout << "C++\n";
return;
 cout << "Programing\n";
 cout << "JAVA\n";
}
```

```
void main()
{
 cout << "Calling function\n";
 print();
 cout << "After function call\n";
}
```

In above program firs line of main is executed and Calling function is printed after that there is a call to function print in function print first two statements are executed and a third statement is a return statement when return statement is executed function goes back to point from where it is called in this case function will return to its call in main and after that next line will be executed.

```
int main()
{
 cout << "First Line\n";
 cout << "Second Line\n";
 cout << "Third Line\n";
 cout << "Fourth Line\n";
}
```

A function having return type other than void must return a value. Value return must be same as return type. In this case value return will be integer.

Now in above case main is having a return type int for a function having return type other than main its compulsory for function to return a value. Now if we write return as given below.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
int main()
{
 cout << "First Line\n";
 cout << "Second Line\n";
 cout << "Third Line\n";
 cout << "Fourth Line\n";
   return;
}
```

Still error because this function is returning an integer value. however in above case with return no value is provided

Still an error as simple return statement is not enough because simple return is written that means function is not returning any value. in this case function must return an integer as given below.

```
int main()
{
 cout << "First Line\n";
 cout << "Second Line\n";
 cout << "Third Line\n";
 cout << "Fourth Line\n";
   return 15;
}
```

Now it is correct as with return statement value is provided. Note this value must be same as return type of a function.

Now it is correct as function is returning a value 15. Instead of 15 there could be any value.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
void main()
{
    return 20;
}
```

Error because function having return type void should not return a value.

Above statement is an error because a function having return type void cannot return a value.

```
int func1()
{
    return 15;
    return 1;
    return 2;
}
```

```
void main()
{
    func1();  // 15 is returned
}
```

Note: A function returns one and only one value.

In above program func1( ) returns 15, 1 and 2 but the first return value is returned so no matters how many return statements are written when return statement is executed it gets back to the point from where it is called.

```
int func1()
{
    return 15;
    return 1;
    return 2;
}
```

```
void main()
{
    func1() + func1();
}
```

15 15

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

In main two function are added do not read it like that actually it is a call to two functions having + operator in between them. In this case func1 is called two times. Operand of + will be values returned by func1.

```
int func1()
{
     return 15;
     return 1;
     return 2;
}
```

```
void main()
{
  int x = func1();   // returned 15;

}
```

x

15

Value returned by function can be assigned to variable also as mentioned above.

```
void func2()
{
   // some code
}
```

```
int y = func2 ( ) ;

x= func1( ) + func2( );
}
```

Error as function is not returning any value

Error as func2 is not returning any value for addition.

If return type of function is void it cannot be assigned to a variable of any type. However if return type is mentioned it is assigned to the data type compatible with return type. Sam no arithmetic operation is applied if return type is void or not compatible with arithmetic operator.

Parametrized function

Parameter can be as value and reference. In this lecture we will just discuss about by value.

Write a function pintsquare that receives int type parameter.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

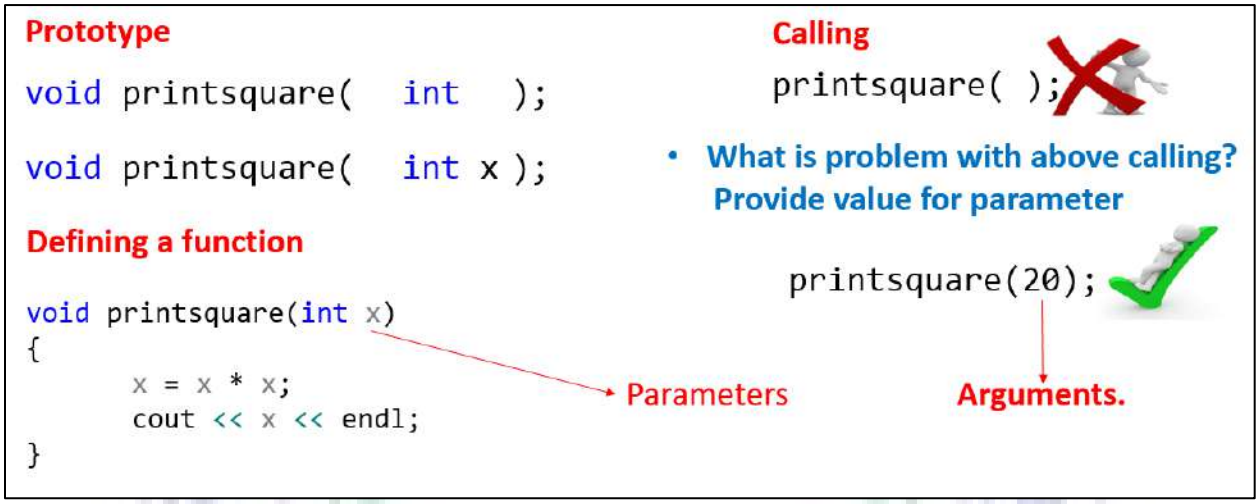


Figure 2 Parameterized Function

Above function explains parameterized function how to call it and what is parameter and argument? Value provided to parameter while calling is called argument note that parameter pass by value is a local variable of a function.

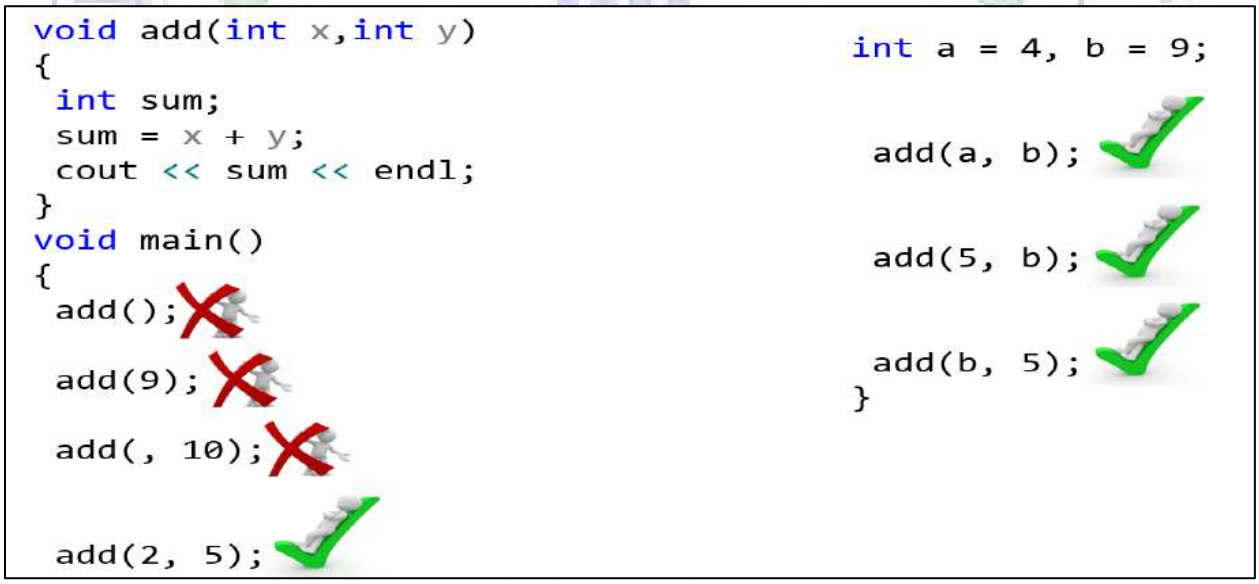


Figure 3 Call of parameterized function

Program 1: Make a function with name getsquare that receives integer type parameter and returns its square.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

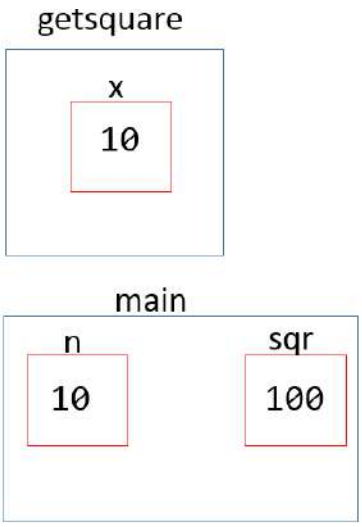
Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

Solution:

```
int getsquare(int x)     //int x=10
{
  return   x * x;//10 * 10;➔ 100;
}
void main()
{
 int n , sqr;

 cin >> n;

 sqr =  getsquare( n );// 100 is returned;
}
                       10 is passed to x
```



Solution with memory picture.

Program 2: Input total marks and obtained marks of pf and ds. While taking total marks as input ensure that total marks must be less than or equal to 100 and greater than equal to 30. If total marks entered are greater than 100 or less than 30 reenter marks, also ensure that obtained marks are less than equal to total marks and are not negative.

Solution without function:

```
#include "iostream"
using namespace std;
void main()
{
	int pftotal, dstotal;
	int pfobtain, dsobtain;
	cout << "Enter pf total ";
	cin >> pftotal;
	while (pftotal < 30 || pftotal >100)
	{
		cout << "Invalid re-enter ";
		cin >> pftotal;
	}
	cout << "Enter ds total ";
	cin >> dstotal;
	while (dstotal < 30 || dstotal >100)
	{
		cout << "Invalid re-enter ";
```

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**


```
		cin >> dstotal;
	}
	cout << "Entre pf obtain ";
	cin >> pfobtain;
	while (pfobtain<0 || pfobtain>pftotal)
	{
		cout << "Invalid re-enter ";
		cin >> pfobtain;
	}
	cout << "Entre ds obtain ";
	cin >> dsobtain;
	while (dsobtain<0 || dsobtain>dstotal)
	{
		cout << "Invalid re-enter ";
		cin >> dsobtain;
	}
}
```

Now analyze logic of total marks

PF Total	**DS Total**

`cin >> pftotal;`
`while (pftotal < 30 || pftotal >100)`
`{`
`cout << "Invalid re-enter ";`
`cin >> pftotal;`
`}`
	`cin >> dstotal;`
	`while (dstotal < 30 || dstotal >100)`
	`{`
	`cout << "Invalid re-enter ";`
	`cin >> dstotal;`
	`}`

Logic of both function is same why not implement this logic in a function and then return a value and assign to relevant variable as given below.

```
int gettotalmarks()
{
	int total;
	cin >> total;
	while (total < 30 || total >100)
	{
		cout << "Invalid re-enter ";
		cin >> total;
	}
	return total;
}
```

```
pftotal= gettotalmarks();
dstotal= gettotalmarks();
```

If you look at above call you can see code is reduced and in case of same logic we can write function.

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

Let us look at code obtain marks logic.

PF Total	**DS Total**

cin >> pfobtain;
while (pfobtain<0 || pfobtain>pftotal)
{
cout << "Invalid re-enter ";
cin >> pfobtain;
}
	cin >> dsobtain;
	while (dsobtain<0 || dsobtain>dstotal)
	{
	cout << "Invalid re-enter ";
	cin >> dsobtain;
	}

Difference lies in validation with relevant total variable why not pass this variable to a function get validated and assign its value to relevant variable as given below.

```
int getobtainmarks(int total)
{
	int obtain;
	cin >> obtain;
	while (obtain<0 || obtain>total)
	{
		cout << "Invalid re-enter ";
		cin >> obtain;
	}

}
```

```
pfobtain= getobtainmarks(pftotal);

dsobtain= getobtainmarks(dstotal);
```

Solution with function:

```
#include "iostream"
using namespace std;
int gettotalmarks()
{
	int total;
	cin >> total;
	while (total < 30 || total >100)
	{
		cout << "Invalid re-enter ";
		cin >> total;
	}
	return total;
}
int getobtainmarks(int total)
```

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**

```
{
	int obtain;
	cin >> obtain;
	while (obtain<0 || obtain>total)
	{
		cout << "Invalid re-enter ";
		cin >> obtain;
	}

}
void main()
{
	int pftotal, dstotal;
	int pfobtain, dsobtain;
	cout << "Enter pf total ";
	pftotal = gettotalmarks();
	cout << "Enter ds total ";
	dstotal = gettotalmarks();
	cout << "Enter pf obtain ";
	pfobtain = getobtainmarks(pftotal);
	cout << "Enter ds obtain ";
	dsobtain = getobtainmarks(dstotal);
}
```

Northern University, Nowshera

Programming Fundamentals (ECS-121)

Muhammad Athar **email id: athar@northern.edu.pk,** **Whatsapp# 0333-5077664**