Object-Oriented Programming

Lecture No. 2

(Recall - Basic programming concepts)

Division of science and Technology

University of Education, Lahore

Basic structure of C++ program

- Preprocessor directive
- main() function
- Program body (C++ statements)

main () function

- Execution of C++ program starts from main()
- Each program must contains a main() function
- Syntax of main()

```
void main ()
{
// body of main
}
```

First C++ program

```
#include<iostream>


main()
{
cout<<“Hello World!”;
}
```

Pre-requisite recall (Practice problem)

- C++ programs to add two integers with and without functions

```cpp
//addition of two integers
#include <iostream>
using namespace std;
int main()
{     int x, y, z;
      cout<<"enter first number: ";
      cin>> x;
      cout<<"enter second number: ";
      cin>>y;
      z=x+y;
      cout<<"Sum of these two :"<<z;
       return 0;
}
```

```cpp
//addition of two integers using function
#include <iostream>
using namespace std;
int sum(int x ,int y);
int main()
{     int x, y;
      cout<<"enter first number: ";
      cin>> x;
      cout<<"enter second number: ";
      cin>>y;
      cout<<"Sum of these two :"<<sum(x,y);
       return 0;
}
int sum(int a, int b)
{
    int c = a+b;
   return c;
}
```

```
/*C++ Program that defines a function which performs
arithmetic operation on two numbers on the basis of
the operator and operand entered by user and passed
to function as parameters.*/

void mathop (int, int, char);
int main()
{
 int a,b;
 char op;
      cout<<"Enter First Number:";
      cin>>a;
      cout<<"Enter Second Number:";
      cin>>b;
      cout<<"Enter Any Arithmetic Operator: ";
      cin>>op;
      mathop(a,b,op);
 return 0;
 }
```

```
void mathop(int a, int b, char op)
{
 switch (op)
  {
 case '+':
     cout<<a << op << b <<"+" << a+b;
     break;
 case '-':
    cout<<a << op << b <<"-" << a-b;
    break;
case '*':
    cout<<a << op << b<<"*" << a+b;
    break;
 case '/':
    cout<<a << op << b<<"/" << a/b;
    break;
 default:
    cout<<"Wrong Arithmetic Operator Entered";
    break;
  }
}
```

Call by reference vs Call by value

- The **call by reference** method of passing arguments to a function copies the reference of an argument into the formal parameter. Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.

- The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument

```
/*C++ program to demonstrate call by value vs call by
reference.*/

#include <iostream>
using namespace std;

void func(int &x, int y);

void func(int &x, int y) {
  x=x+1;
  y=y+1;
  cout<<"value of x is "<<x<<endl;
  cout<<"value of y is "<<y<<endl;
  return;
}
```

```
int main () {

  int x = 100;
  int y = 200;
  cout << "Before func call, value of x is " << x << endl;
  cout << "Before func call, value of y is " << y << endl;
  func(x, y);
  cout << "After func call, value of x is " << x << endl;
  cout << "After func call, value of y is " << y << endl;

  return 0;
}
```

Output:

```
Before func call, value of x is 100
Before func call, value of y is 200
value of x is 101
value of y is 201
After func call, value of x is 101
After func call, value of y is 200
```

Actual parameters vs Formal parameters

- **Actual Parameters**: The values/variables passed while calling a function are called actual parameters.

- **Formal Parameters**: These are the variables written/declared in function definition/prototype, and receive their values when a call to that function is made.

Variable scoping

- Variable is visible within the scope in which it is declared.
- Variable with same name cannot be declared within same scope; however, this can be done if scope is different.
- Formal parameters have functional scope.

```
main()
{
int a=0;
int a=2; //error
}
```

Variable scoping –example 1

```
#include<iostream>
using namespace std;
//global variable
int x=4;
main()
{
        int x=2;  //local variable
        cout<<x;
}
```

Output? How to access global variable?

Variable scoping –example 1

```
#include<iostream>
using namespace std;
//global variable
int x=4;
main()
{
        int x=2;  //local variable
        cout<<x;
        cout<<::x;

}
```

Output:

24

Variable scoping – example 2

```
#include<iostream>
using namespace std;

void func(int);

main()
{
        int x=5;
        cout<<x;
        func(x);


}
```

```
void func(int a)
{
        int b=2;
        cout<<a;
        cout<<b;
        {
                cout<<a;
                int a=6;
                int b=3;
                cout<<a;
                cout<<b;
        }
}
```

Output:

52563

Array

```
#include <iostream>
using namespace std;
int main()
{
   int arr[] = {11, 22, 33, 44, 55};
   int n=0;
 while(n<=4)
    {
      cout<<arr[n]<<endl;
       n++;
    }
return 0;
}
```

Passing array to function

/*C++ program that declares and initializes an array of integer of size 5, pass it to function which calculates the average and displays the result in main*/

```
#include <iostream>
using namespace std;
double getAverage(int arr[], int size);
int main ()
{
int balance[5] = {1000, 2, 3, 17, 50};
double avg;
avg = getAverage( balance, 5 ) ;
cout << "Average value is: " << avg << endl;
return 0;
}
```

```
double getAverage(int arr[], int size)
{
int i, sum = 0;
double avg;

for (i = 0; i < size; ++i)
  {
    sum += arr[i];
  }
avg = double(sum) / size;
return avg;
}
```

References

- C++ How to Program
  By Deitel & Deitel

- The C++ Programming Language
  By Bjarne Stroustrup

- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza

- https://www.tutorialspoint.com

- http://ecomputernotes.com

- http://www.cplusplus.com