Object Oriented Programming in C++

Nepal College of Information Technology

Course Instructor: Er. Rabina Chaudhary

Output Operator/ Insertion Operator /put Operator:

- The operator << (two less than sign) is called insertion operator
- It inserts data that follows it into the computer screen
- cout is predefined object of ostream class
- The object cout is used with insertion operator to display desired data in the screen

Syntax :

cout<< data or message to be displayed

Example:

```
cout<<“Hello world”;
cout <<x;   // x is a variable
cout<<500
```

Output Operator/ Insertion Operator :

- We can display multiple data using single insertion operator and single cout object, which is called cascaded output operation4

Syntax:

cout<< data or message to be displayed<< data or message to be displayed;

Example:

Int x=10,y=100;

cout<<“ The value of x is “<<x<<“ and the value of y is “<<y;

Input Operator/Extraction Operator/get Operator:

- The operator >> (two greater than sign) is known as extraction operator
- It extracts data from input device and store into a variable
- cin is predefined object of istream class
- The object cin is used with extraction operator to read data from input device

Syntax:

```
cin>>variable;
```

Example:

```
cin>>a ;
```

Input Operator/Extraction Operator:

- We can use cin to take more than one value as input.

Syntax:

```
cin>>variable1>>variable2>>variable3;
```

- iostream header file contains the declarations for cout ,cin,>> operator ,<< operator
- Header file iostream must be included in the program that use input/ output statements

Example:

```
#include<iostream.h>
#include<conio.h>
void main()
{
        cout<<“Hello World”;
        getch();
}
```

Example Program:

```
#include<iostream.h>
#include<conio.h>
        void main()
        {
        int x,y,sum;
        cout<<“Enter two numbers”;
        cin>>x;              //cin>>x>>y;
        cin>>y;
        sum=x+y;
        cout<<“The sum of  “<<x<<“and”<<y<<“ is “<<sum;
        getch();
        }
```

Enumeration:

- C++ allows programmers to create their own data type called enumerated type

Syntax:

```
enum enumerated_datatype
{
        Enumerator1, Enumerator2,….
};
```

Example:

```
enum days
{
        sunday, monday, tuesday, wednesday, thursday, friday
};
```

Enumeration:

```
#include<iostream.h>
#include<conio.h>

void main()
{
        enum days
                {
                sun,mon,tues,wed,thur,fri,sat
                };
  days day1,day2;
  day1=sun;
  day2=thur;

  cout<<"day 1= "<< day1 <<endl<<"day 2= "<< day2;
  getch();
}
```

Practice:

Q1. Write a program to define a enumerated data type Month with name of 12 months. Assign first month as 1 and display the integer value assigned to the months.

Inline Function:

- Inline function is a function that is expanded in line when it is called
- When the inline function is called whole code of the inline function gets inserted at the point of inline function call

Syntax:

```
inline returnType functionName( parameters)
{
        function body;
}
```

Program to calculate area of rectangle using inline function.

```
#include<iostream.h>
#include<conio.h>
        inline int area(int l,int b){
                return (l*b);
         }
      void main()
      {
                int len,bre,result;
                cout<<“Enter values of length and breadth”;
                cin>>len>>bre;
                result=area(len,bre);
                cout<<endl<<“The area of rectangle is “<<result;
                getch();
      }
```

Inline function:

Compiler can ignore the request for inlining in following cases:

- If function contains loop
- If function contains static variables
- If function is recursive
- If the function is large

Practice:

Q1. Write a program using inline function to calculate the square of a number.

Q2. Write a program to calculate volume of a cube. (vol=side*side*side)

Q3. Write a program using inline functions to calculate the multiplication and division of two user input numbers.

Default Argument:

- In C++, we can call function without specifying all its arguments by assigning default values for some arguments

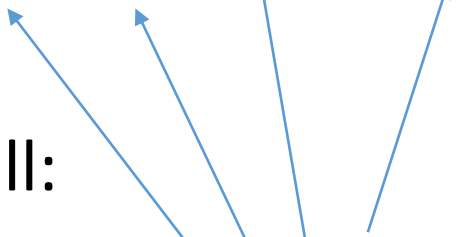
Syntax for declaring function with default argument:

returnType functionName(argument1, argument2=default_value, argument3=default_value);

Default argument

Function prototype:

void test(int a,int b,int c=0,int d=10);

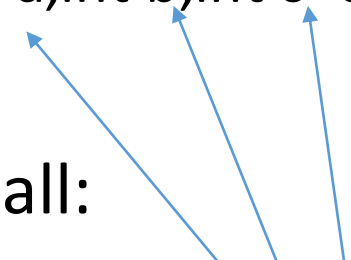


Function call:

test(w,x,y,z);

Function prototype:

void test(int a,int b,int c=0,int d=10);



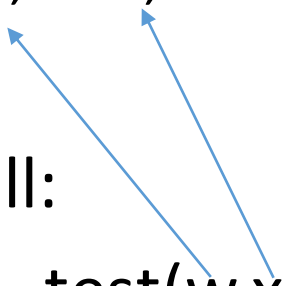
Function call:

test(w,x,y);

Default Argument:

Function prototype:

```
void test(int a,int b,int c=0,int d=10);
```



Function call:

```
test(w,x);
```

During calling of function, arguments from calling function to called function are copied from left to right

Default Argument:

Function prototype:

```
void test(int a=5,int b=2,int c=0,int d=10);
```

Function call:

```
test();
```

Default Argument:

Function Prototype:

returntype functionName (argument1, argument2=defaultValue, argument3=defaultValue);

default values must be defined from right to left

- int testFunction(int a=5, int b=10, int c=20) ; //valid
- int testFunction(int a, int b=10, int c=20) ; //valid

- int testFunction(int a=5, int b, int c=20) ; //invalid
- int testFunction(int a, int b=10, int c) ; //invalid
- int testFunction(int a=5, int b, int c) ; //invalid

Example:

```
void add(int a, int b, int c=0, int d=0);
void main()
        {
                int x,y,z,w;
                /*ask user and input values
              for x, y and z*/
                add(x,y);
                add(x,y,z);
                add(x,y,z,w);

        getch();
        }
void add( int a, int b, int c, int d)
        {
                int sum=a+b+c+d;
                cout<<endl<<"The sum
                is "<<sum;
        }
```

Practice:

Q1. Write a program to calculate simple interest using default value of r=1.5%. Ask the user for principal amount and time [SI=PTR/100]

Function Overloading:

- Function overloading is a feature in C++ where two or more functions can have the same name but can do different operations
- Two or more functions can have same name but different in their argument list
- We can overload function by either making
    i. type of arguments different
    ii. making number of argument different

- The correct function is invoked whose argument list matches the arguments in the function call

Function Overloading:

Example:

```
int calculate(int ); //function1
 float calculate(float ); //function2
 void calculate( int, int ); //function3
 int calculate(int, int, int); //function4
 float calculate(int ,float); //function5
 float calculate(float, int); //function6
```

Function Overloading:

- Example:
- Declaration:

```
int add(int a,int b); //function1
int add(int a, int b, int c); //function2
double add(double x , double y); //function3
double add(int p ,double q); //function4
double add(double a, int b); //function5
```

- Function call:

```
cout<<add(5,10);
cout<<add(100,100.5);
cout<<add(5.5,10);
cout<<add(12,23,56);
cout<<add(10.5,5.6);
```

```
#include<iostream.h>
#include<conio.h>

int add(int a,int b)
{
        cout<<"Function adds two integer"<<endl;
        return (a+b);
}

int add(int a,int b, int c)
{
        cout<<endl<<"Function adds three integer"<<endl;
        return (a+b+c);
}
double add(double a,double b)
{
        cout<<endl<<"Function adds two double values"<<endl;
        return (a+b);
}
```

```
double add(double a,int b)
{
        cout<<endl<<"Function adds double and int"<<endl;
        return (a+b);
}
double add(int a,double b)
{
        cout<<endl<<"Function adds int and double"<<endl;
        return (a+b);
}
void main()
{
        cout<<"Result is "<<add(2,10)<<endl<<endl;
         cout<<"Result is "<<add(10,10,20)<<endl;
        cout<<"Result is "<<add(10.5,20.52)<<endl;
        cout<<"Result is "<<add(10.5,30)<<endl;

        cout<<"Result is "<<add(35,70.5);

  getch();
}
```

Practice:

Q1. Write a program to find maximum of 2 numbers and maximum of 3 numbers using same function name, maximum().

Q2. Write a program to find the volume of 3 objects :cube, cylinder and

Rectangular box using same function name, volume().

Q3. Write a program to find the area of cube, cylinder and rectangle using concept of function overloading.

Reference Variable:

- Reference variable is another name for an already existing variable
- When a variable is declared as reference, it becomes an alternative name for an existing variable
- A variable can be declared as reference by putting '&' in the declaration

Syntax:

```
datatype &referenceVariableName= VariableName;
```

```
int a;
int &ref=a;
```

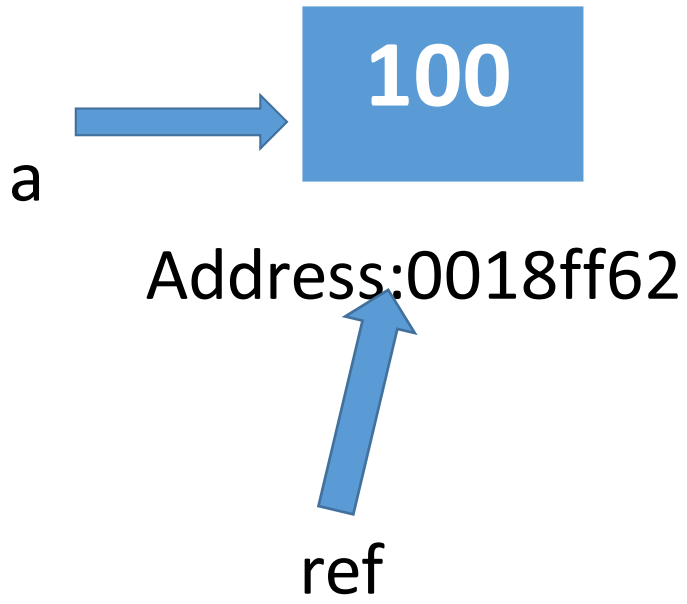
Reference Variable:

Syntax:

datatype &referenceVariableName= VaraibleName;

Example:

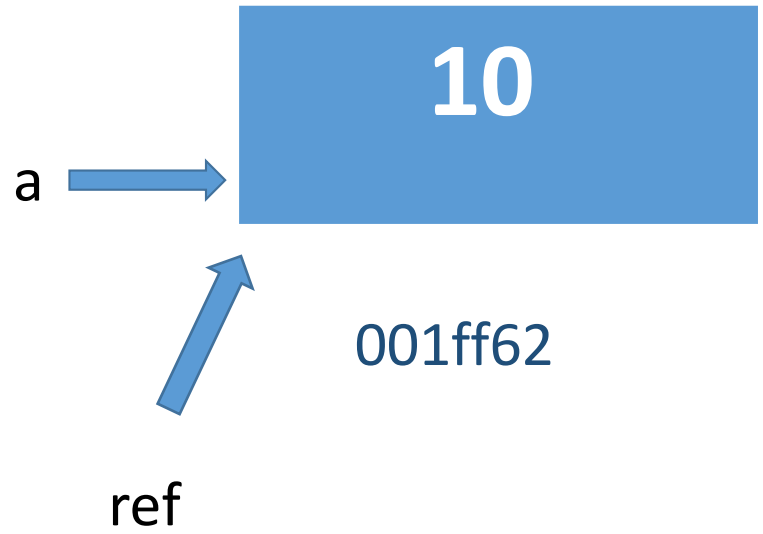
int a=100;



int &ref=a;

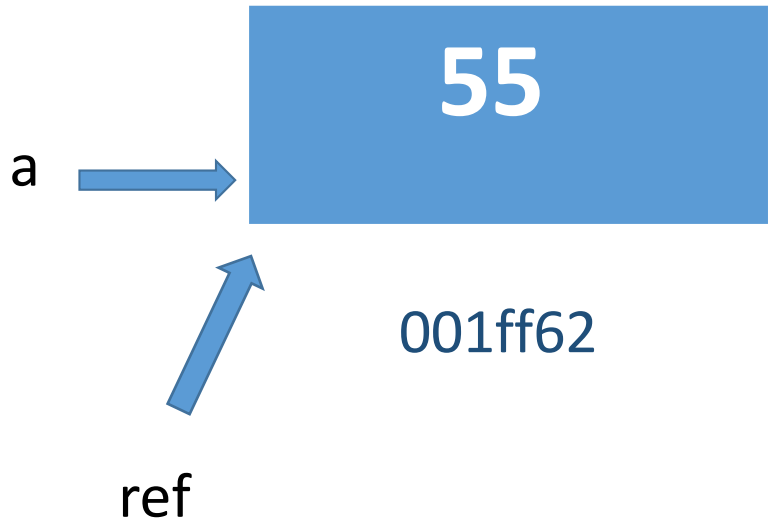
Example:

```
#include<iostream.h>
void main()
{
        int a=10;
        int &ref=a;
        cout<<"a="<<a<<endl;
         ref=55;
         cout<<"a="<<a<<endl;
        a=99;
         cout<<"ref="<<ref<<endl;
        cout<<"address of ref ="<<&ref<<endl;
         cout<<"address of a ="<<&a<<endl;

}
```



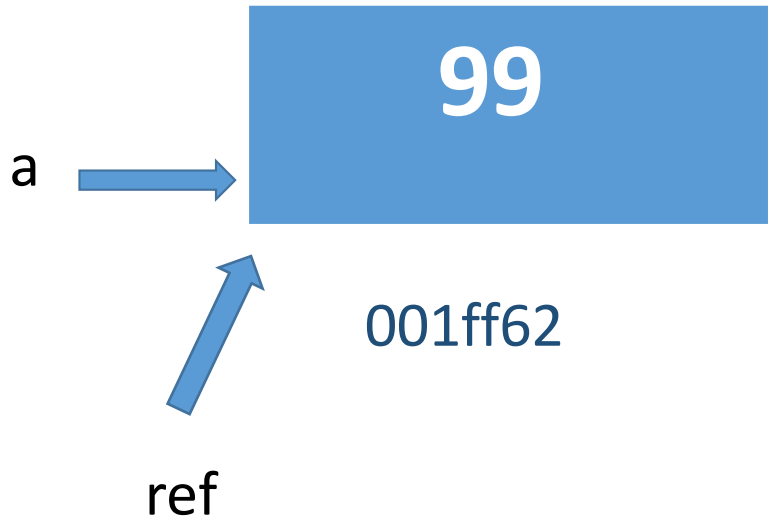
Example:

```
void main()
{
        int a=10;
        int &ref=a;
        cout<<"a="<<a<<endl;
         ref=55;
         cout<<"a="<<a<<endl;
        a=99;
         cout<<"ref="<<ref<<endl;
        cout<<"address of ref ="<<&ref<<endl;
         cout<<"address of a ="<<&a<<endl;

}
```



Example:

```
void main()
{
        int a=10;
        int &ref=a;
        cout<<"a="<<a<<endl;
         ref=55;
         cout<<"a="<<a<<endl;
        a=99;
         cout<<"ref="<<ref<<endl;
        cout<<"address of ref ="<<&ref<<endl;
         cout<<"address of a ="<<&a<<endl;

}
```



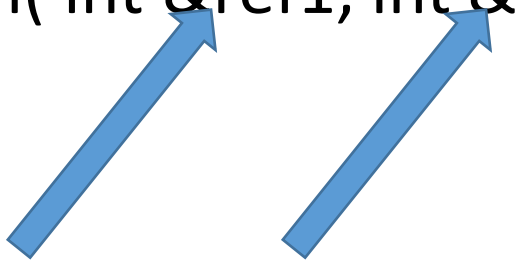
Call by Reference:

Function Prototype:

```
void testFunction( int &ref1, int &ref2);
```

Function Call:

```
testFunction  (   a       ,     b  );
```



When you pass parameters by reference, a new storage location is not created for these parameters. The reference parameters represent the same memory location as the actual parameters that are supplied to the method.

Here, ref1 will be alternative name of a and ref2 will be alternative name of b.

Example Program:

```
#include<iostream.h>
#include<conio.h>
void swap(int &x, int &y);
void main ()
{
        int a = 10;
         int b = 20;
        cout << "Before swap, value of a :" <<a << endl;
        cout << "Before swap, value of b :" <<b << endl;
        swap(a, b);
        cout << "After swap, value of a :" << a << endl;
         cout << "After swap, value of b :" << b << endl;
        getch();
}
```

```
void swap(int &x, int &y)
{
        int temp;
         temp = x;
        x = y;
        y = temp;
}
```

Practice:

Q1. Write and test the following computeSphere() function that returns the volume “v” and surface area “s” of a sphere with the given radius.

void computeSphere(float &v, float &s, float r)