

Programming Fundamentals

Course Code: CS-111

Course Instructor: Isra Naz





Learning Objectives

- Functions
 - Scope of Function
- Passing parameters to functions
 - Pass by value
 - Pass by reference



CLO Covered

- CLO1: Describe fundamental problem-solving techniques and logic constructs. GA 1
- CLO2: Apply basic programming concepts. GA2
- CLO3: Analyze and solve the real-world problems by using programming constructs. GA3

Scope of Functions

- The area in which a function can be accessed is known as the scope of a function.
- The scope of any function is determined by the place of function declaration.
- **Types of function on the basis of scope are:**
- **Local Functions**
- It is declared within another function.
- Can be called within the function in which it is declared
- **Example**

```
int main( ) {  
    void show(void); // local functions  
    show( );  
    return 0;  
}
```

Scope of Functions

❑ Global Functions

- ❑ A type of function that is declared outside any function.
- ❑ It can be called from any part of the program.

❑ Example

```
void show(void); // Global functions
```

```
int main( ) {  
    show( );  
    return 0;  
}
```

Passing Parameters to Functions

- ❑ **Parameters/ Arguments are the values that are provided to a function when the function is called.**
- ❑ Parameters are given in the parentheses.
- ❑ If there are many parameters, these are separated by commas.
- ❑ If there is no parameter, empty parentheses are used.
- ❑ Both ***variables and constants*** can be passed to a function as parameters.

Passing Parameters to Functions

- ❑ The sequence and types of parameters in function call must be similar to the sequence and types of parameters in function declaration.
- ❑ Parameters in function call are called actual parameters
- ❑ Parameters in function declaration are called formal parameters
- ❑ When a function call is executed, the values of actual parameters are copied to formal parameters
- ❑ The methods to pass parameters/arguments
 - ❑ Constants
 - ❑ By value/variable
 - ❑ By reference

Passing Parameters to Functions

Example (Passing Constants)

```
#include<iostream>
using namespace std;
void sum(int , int); //function declaration
int main()//start of main function
{
    sum(2, 2); //passing constant values during function call
    cout<<"exit";
}

void sum( int x, int y) //function definition
{
    int s;
    s = x + y ;
    cout<<"sum is = "<< s << endl;
}
```




Pass by value

- ❑ A parameter passing mechanism in which the value of **actual parameter** is copied to **formal parameters** of called function is known as pass by value.
- ❑ If the function makes any change in formal parameter, it does not affect the values of actual parameter.
- ❑ It is the default mechanism for passing parameters to functions.

Pass by value

Example

```
// function declaration
void show(int);
int main()
{
    int n;
    cout<<"Enter number.";
    cin>>n;
    // function call
    show(n);
    cout<<"End of Program";
}
```

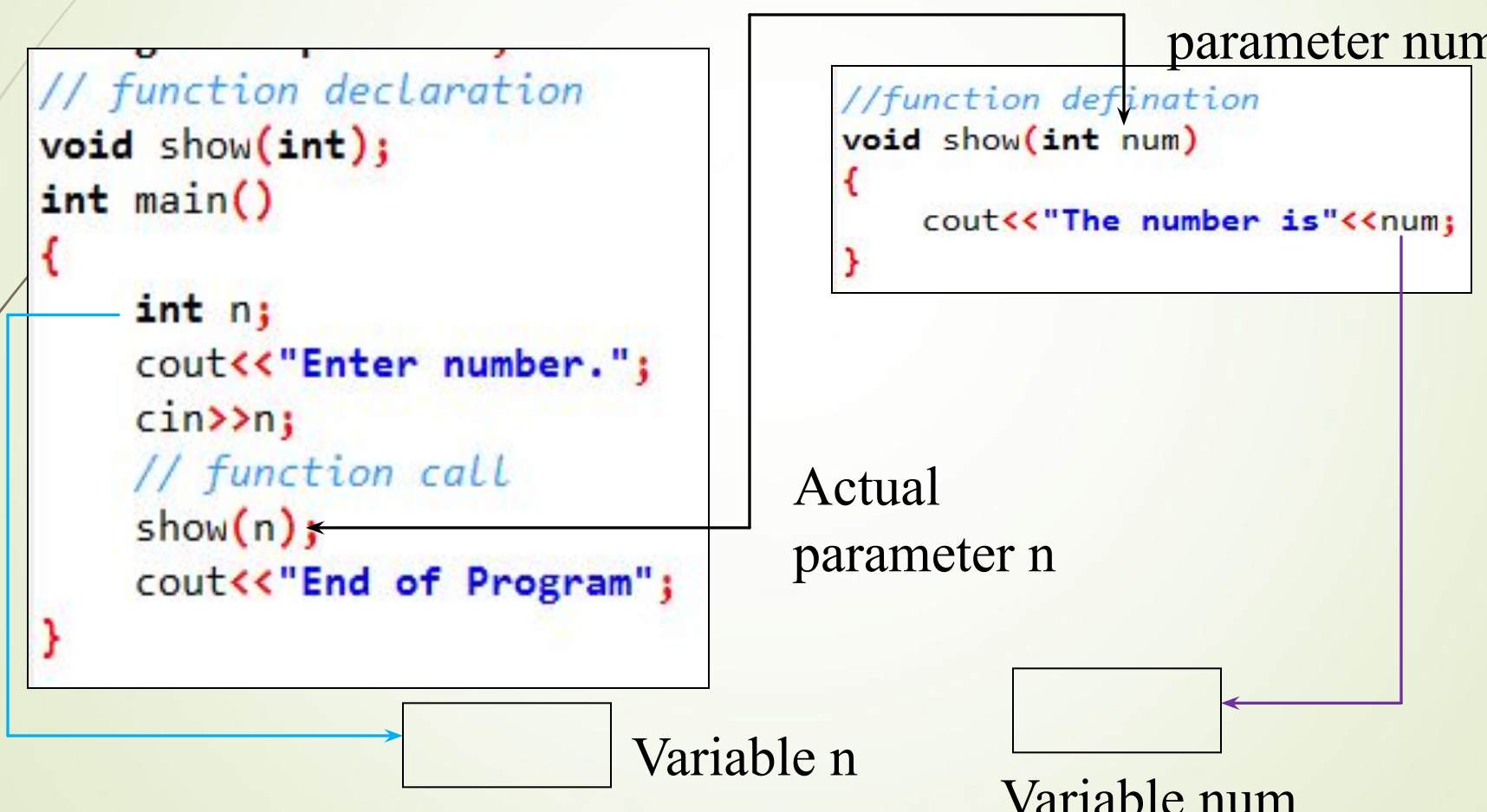
```
//function definition
void show(int num)
{
    cout<<"The number is"<<num;
}
```

Formal
parameter num

Actual
parameter n

Variable n

Variable num



Passing Parameters to Functions

Example (Passing Variables)

```
#include<iostream>
using namespace std;
void sum(int , int); //function declaration
int main()
{
    int a , b;
    cout<<"enter 1st value";
    cin>> a;
    cout<<"enter 2nd value";
    cin>> b;
    sum(a, b); //passing variables during function call
    cout<<"exit";
}

void sum( int x, int y) //function definition
{
    int s;
    s = x + y ;
    cout<<"sum is = "<< s << endl;
}
```



Example (pass by value)

- A program that inputs two numbers in main() function, passes these numbers to a function. The function displays the maximum number.


Example (pass by value)

- A program that inputs two numbers in main() function, passes these numbers to a function. The function displays the maximum number.

```
#include<iostream>
using namespace std;
void max(int a,int b);
int main()
{
    int x,y;
    cout<<"Enter two numbers: ";
    cin>>x>>y;
    max(x,y);
}
void max(int a, int b)
{
    if(a>b)
        cout<<"Maximum number is "<<a;
    else
        cout<<"Maximum number is "<<b;
}
```



Example (pass by value)

- A program that inputs a number and displays its predecessor and successor numbers using functions
- 

Example (pass by value)

- A program that inputs a number and displays its predecessor and successor numbers using functions

```
#include<iostream>
using namespace std;
void value(int);//function declaration
int main()
{
    int x;
    cout<<"Enter ans integer: ";
    cin>>x;
    value(x);//function call
}
void value(int x)
{
    int p,n;
    p=x-1;
    n=x+1;
    cout<<"The number before "<<x<<"is "<<p<<endl;
    cout<<"The number after "<<x<<"is "<<n<<endl;
}
```

Example

- A program that inputs two numbers and one arithmetic operator in main function and passes them to a function. The function applies arithmetic operation on two numbers on the basis of the operator entered by user using switch statement.

```
#include<iostream>
using namespace std;
void cal(int a, int b, char op);//function declaration
int main()
{
    int x,y;
    char c;
    cout<<"Enter first number, operator and second number: ";
    cin>>x>>c>>y;
    cal(x,y,c);//function call
}
```


Example

```
void cal(int a, int b, char op)//function definition
{
    switch(op)
    {
        case '+':
            cout<<a<<" + "<<b<<" = "<<a+b;
            break;
        case '-':
            cout<<a<<" - "<<b<<" = "<<a-b;
            break;
        case '*':
            cout<<a<<" * "<<b<<" = "<<a*b;
            break;
        case '/':
            cout<<a<<" / "<<b<<" = "<<a/b;
            break;
        case '%':
            cout<<a<<" % "<<b<<" = "<<a%b;
            break;
        default:
            cout<<"Invalid operator";
    }
}
```

Pass by Reference

- ❑ The parameter passing mechanism in which the **address of actual parameter is passed to the called function** is known as pass by reference.
- ❑ Reference is represented by **ampersand (&)**.
- ❑ The formal parameter is not created separately in the memory.
- ❑ Formal parameter becomes a second name of actual parameter. It means that single memory location is shared between actual and formal parameter.
- ❑ If the called function makes any change in formal parameter, the change is also available in calling function.

Pass by Reference

□ Example

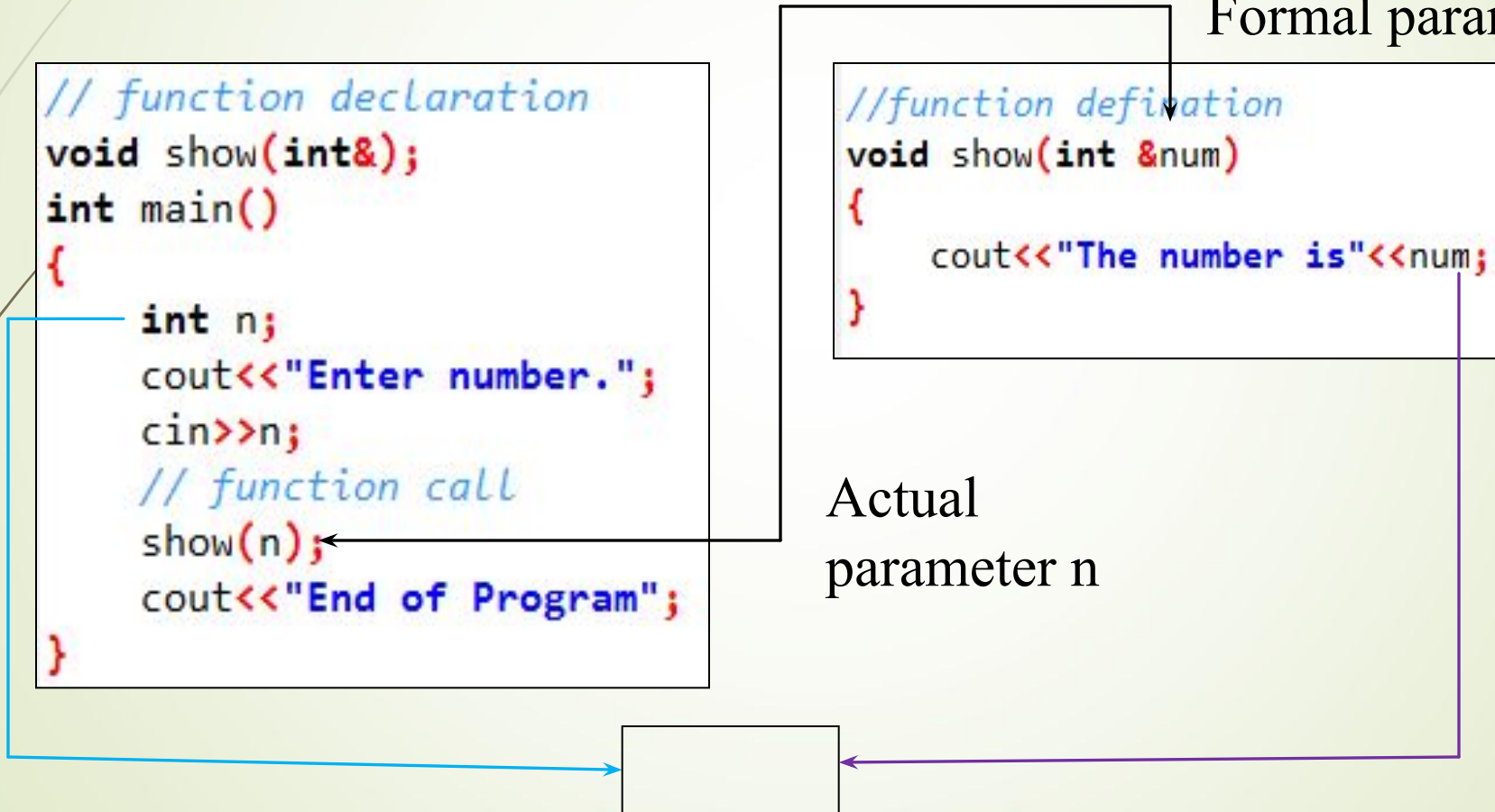
```
// function declaration
void show(int&);
int main()
{
    int n;
    cout<<"Enter number.";
    cin>>n;
    // function call
    show(n);
    cout<<"End of Program";
}
```

```
//function definition
void show(int &num)
{
    cout<<"The number is"<<num;
}
```

Formal parameter num

Actual
parameter n

Shared memory location



Example (Swapping of numbers)

```
#include<iostream>
using namespace std;
void swap(int & , int &);//function declaration
int main()
{
    int a, b;
    cout<<"enter value of A";
    cin>>a;
    cout<<"enter value of B";
    cin>>b;
    swap(a,b);//function call
    cout<<"values after swapping"<<endl;
    cout<<"value of A is "<<a<<endl;
    cout<<"value of B is "<<b<<endl;
}
void swap( int & x , int & y)//function definition
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

Difference b/w Call by value and call by reference

Call by Value	Call by Reference
Call by value passes the value of actual parameter to formal parameter.	Call by reference passes the address of actual parameter to formal parameter.
The actual and formal parameters refer to different memory locations	The actual and formal parameters refer to the same memory location.
Any change made by function in formal parameter does not affect the value of actual parameter.	Any change made by function in formal parameter actually changes the value of actual parameter.
It requires more memory.	It requires less memory.
It is less efficient.	It is more efficient.