Programming Fundamentals

Course Code: CS-111

Course Instructor: Isra Naz

Learning Objectives

- Passing parameters to functions
  - Returning value from function

Learning Objectives

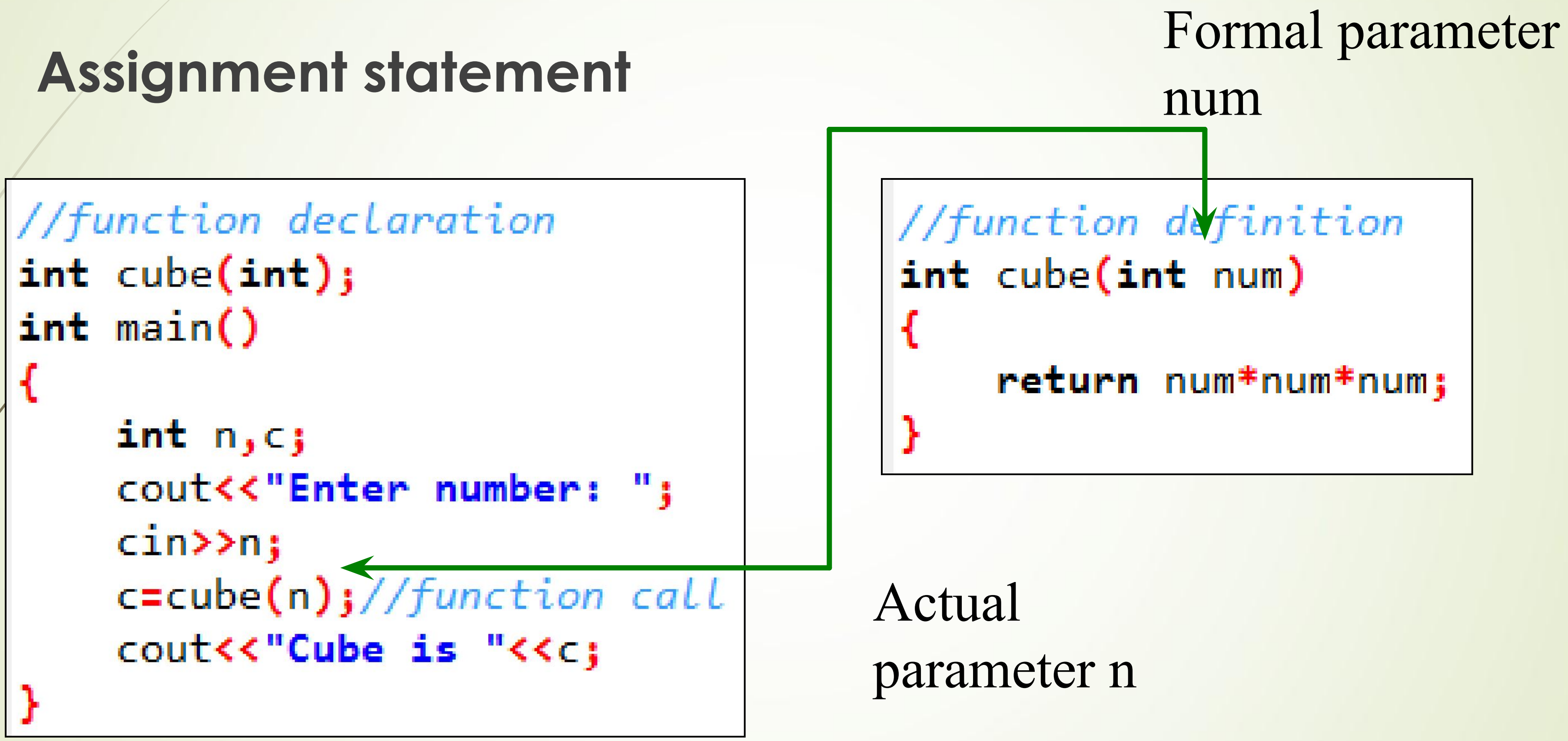
- Passing Array as Parameter/Argument
- Declaring function with array as parameter
- Function definition with array as parameter
- Calling function with array as parameter
- Function Overloading

Returning Values From Functions

- When a function completes its execution, its **returns a single value to the calling function**.
- A function can return only one value
- **It can be any type except string and array**
- The type of data that a user-defined function returns is declared in function declaration.
- For example,
- **int** is used as **return type** if the function **returns integer value**
- If function **returns no value**, the void keyword is used as **return type**.

Returning Values From Functions

- **Assignment statement**



```
//function declaration
int cube(int);
int main()
{
    int n,c;
    cout<<"Enter number: ";
    cin>>n;
    c=cube(n);//function call
    cout<<"Cube is "<<c;
}
```

```
//function definition
int cube(int num)
{
    return num*num*num;
}
```

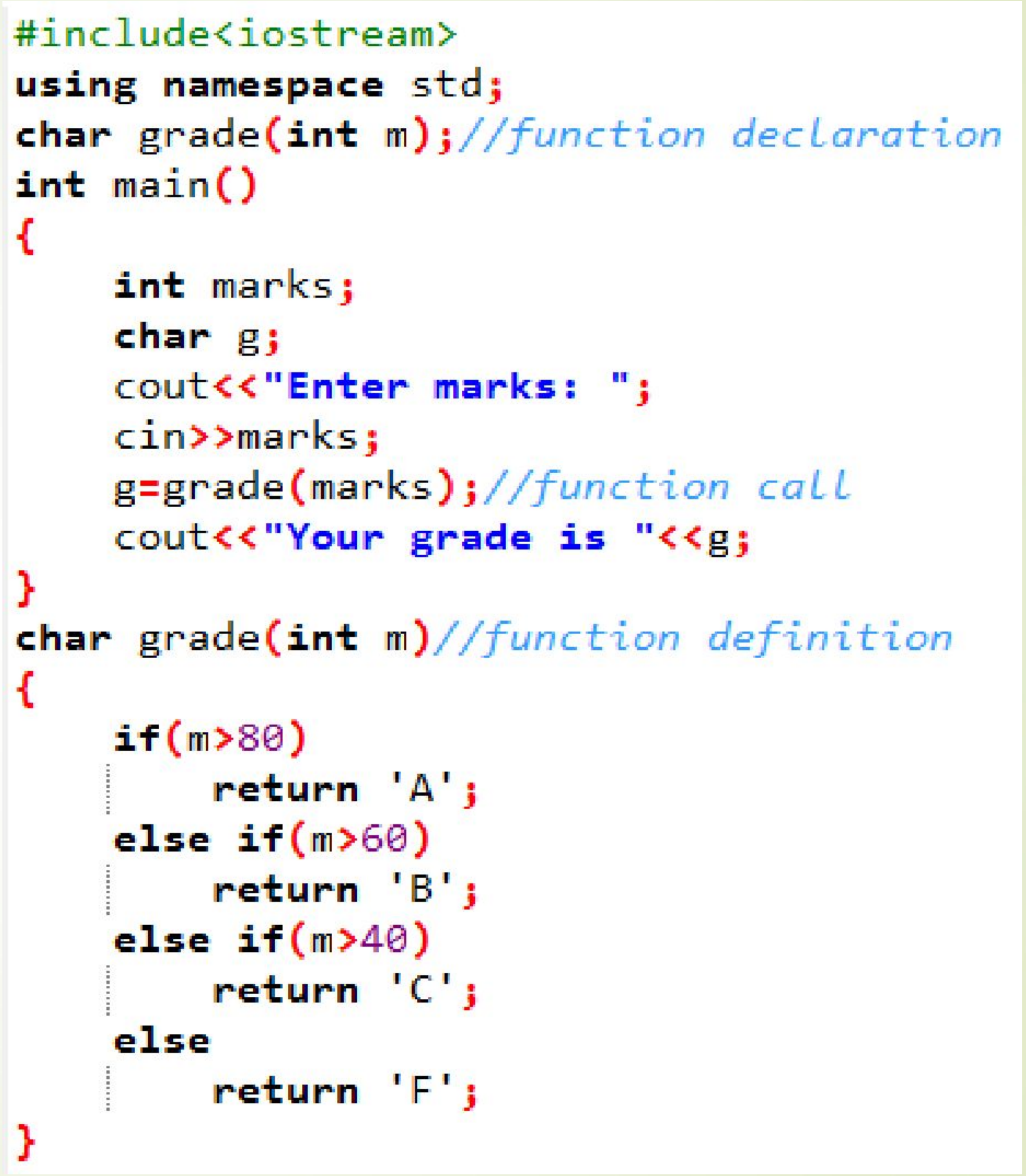
Example

- Write a program that inputs marks in main function and passes these marks to a function. The function finds grade of student on the basis of the following criteria:

Grade A	80 or above marks
Grade B	60 to 79 marks
Grade C	40 to 59 marks
Grade F	below 40 marks

The function returns grade back to main function where it is displayed on the screen.

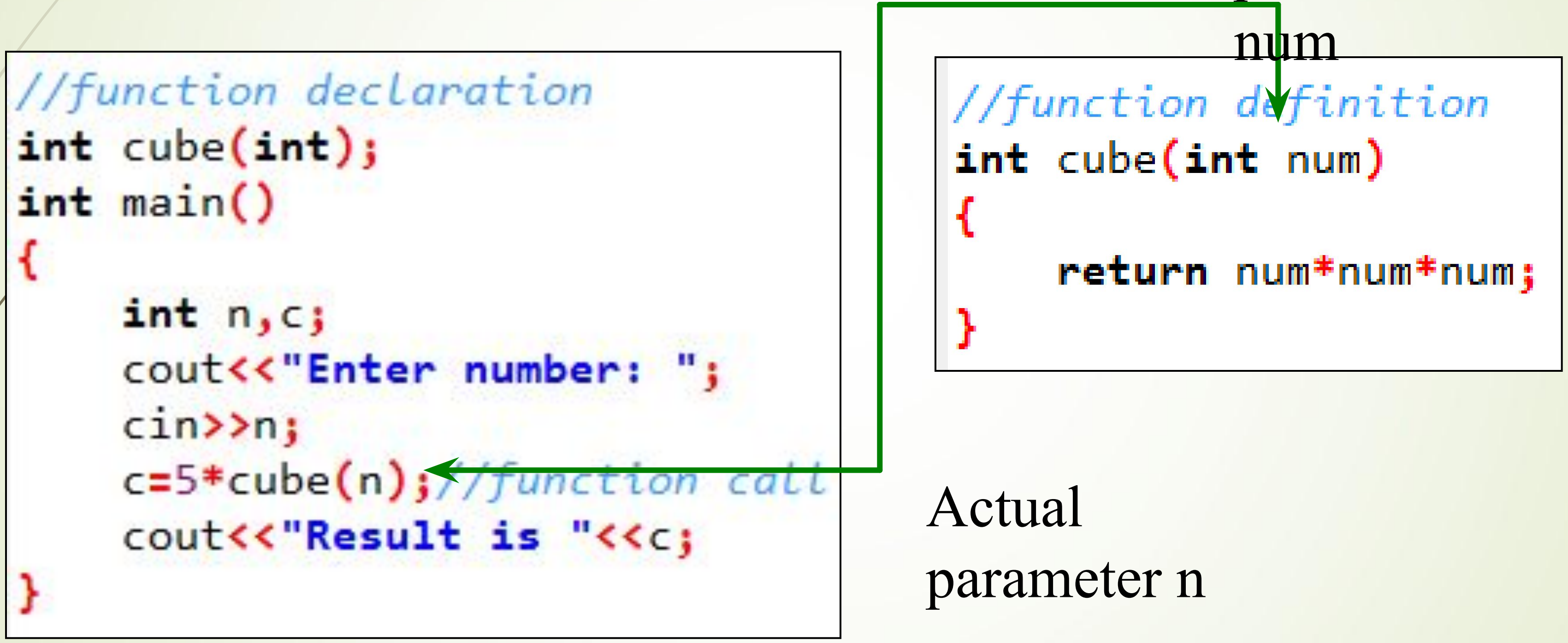
Example



```
#include<iostream>
using namespace std;
char grade(int m);//function declaration
int main()
{
    int marks;
    char g;
    cout<<"Enter marks: ";
    cin>>marks;
    g=grade(marks);//function call
    cout<<"Your grade is "<<g;
}
char grade(int m)//function definition
{
    if(m>80)
        return 'A';
    else if(m>60)
        return 'B';
    else if(m>40)
        return 'C';
    else
        return 'F';
}
```

Returning Values From Functions

- **Arithmetic Expression**



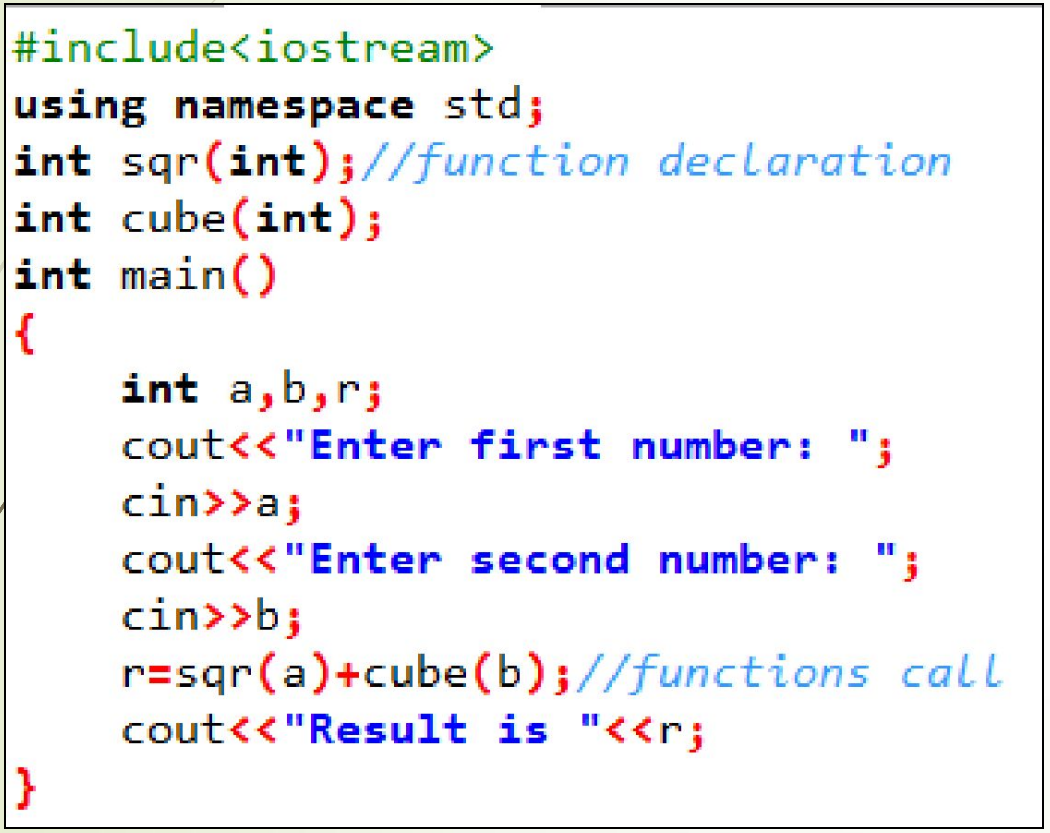
```
//function declaration
int cube(int);
int main()
{
    int n,c;
    cout<<"Enter number: ";
    cin>>n;
    c=5*cube(n);//function call
    cout<<"Result is "<<c;
}
```

```
//function definition
int cube(int num)
{
    return num*num*num;
}
```

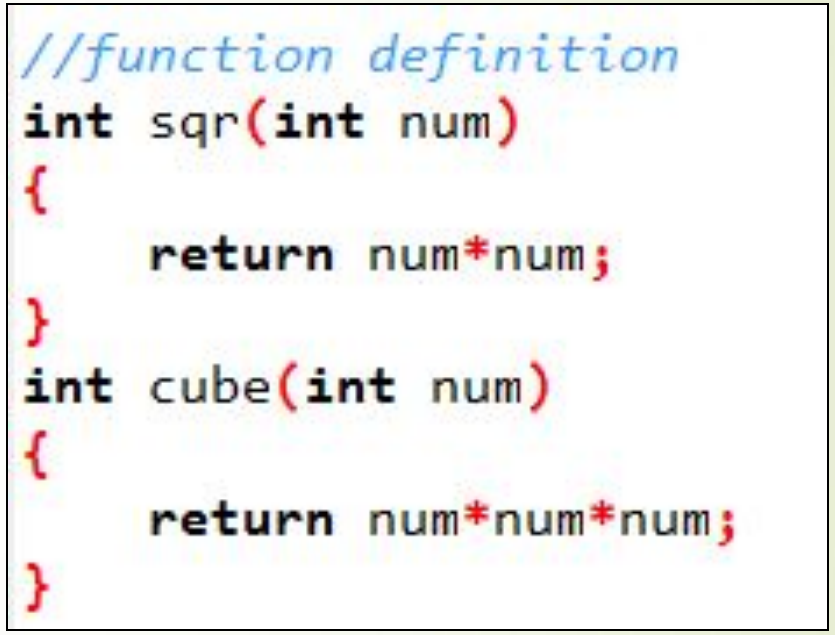
Example

- Write a program that inputs two integers. It passes first integer to a function that calculates and returns its square. It passes second integer to another function that calculates and returns its cube. The main () function adds both returned values and displays the result.

Example

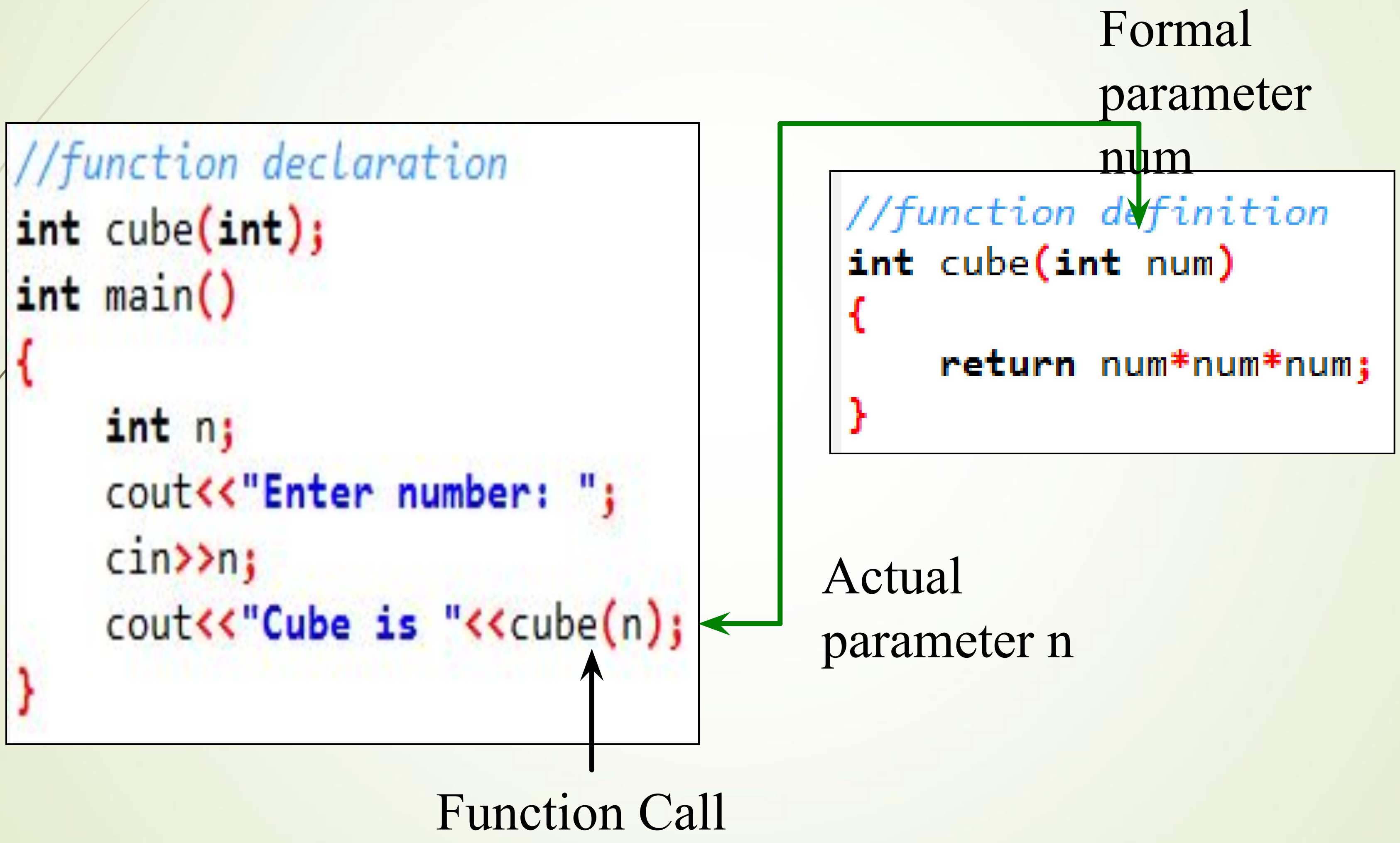


```cpp
#include<iostream>
using namespace std;
int sqr(int);//function declaration
int cube(int);
int main()
{
    int a,b,r;
    cout<<"Enter first number: ";
    cin>>a;
    cout<<"Enter second number: ";
    cin>>b;
    r=sqr(a)+cube(b);//functions call
    cout<<"Result is "<<r;
}
```



```cpp
//function definition
int sqr(int num)
{
    return num*num;
}
int cube(int num)
{
    return num*num*num;
}
```

Returning Values From Functions



```
//function declaration
int cube(int);
int main()
{
    int n;
    cout<<"Enter number: ";
    cin>>n;
    cout<<"Cube is "<<cube(n);
}
```

```
//function definition
int cube(int num)
{
    return num*num*num;
}
```

Passing Array as Parameter

- An array can be passed to a function as parameter.
- When an array is passed as parameter to a function, only the address of first element of the array is passed.
- **An array is passed by reference not by value.**
- A **separate copy of the array is not created** in the function.

Declaring function with Array as Parameter/Argument

- **If a 1-D array is passed.**
- void myfunction(int []);
- void max(int [], float []);

- **If a 2-D array is to be passed**
- void myFunction(int [][]);
- void max(int [][], float [][]);

Function Definition with Array Parameters/Arguments

- The name of array and its type is given in the declarator.
- E.g.

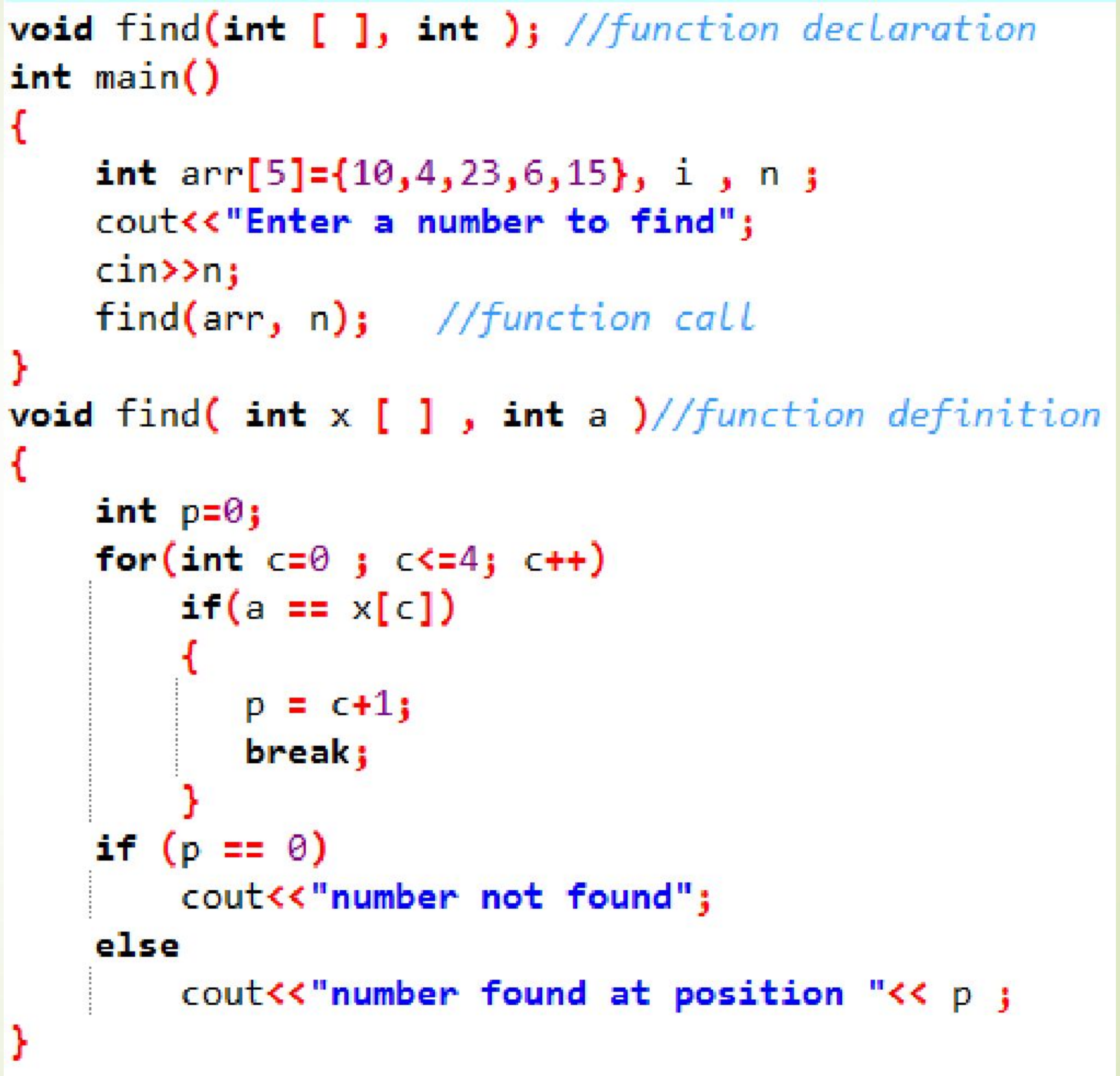
```
void myFunction( int a[ ])
{
    body of the function
}
```

- Size of the array can be given in the function declaration and in the function definition.

Calling function with Array Parameters/Arguments

- A function with array parameter is called by giving the name of the array as actual parameter.
- The index or subscript of the array is not used in function call.
- The name of the array refers to the memory address of its first element.
- The memory address is passed to the function.
- The function then accesses the array by using the same memory address.
- Name of array used in the function call and function definition may be same or different.

Example (passing array as parameter)



```
void find(int [ ], int ); //function declaration
int main()
{
    int arr[5]={10,4,23,6,15}, i , n ;
    cout<<"Enter a number to find";
    cin>>n;
    find(arr, n);   //function call
}
void find( int x [ ] , int a )//function definition
{
    int p=0;
    for(int c=0 ; c<=4; c++)
        if(a == x[c])
        {
            p = c+1;
            break;
        }
    if (p == 0)
        cout<<"number not found";
    else
        cout<<"number found at position "<< p ;
}
```

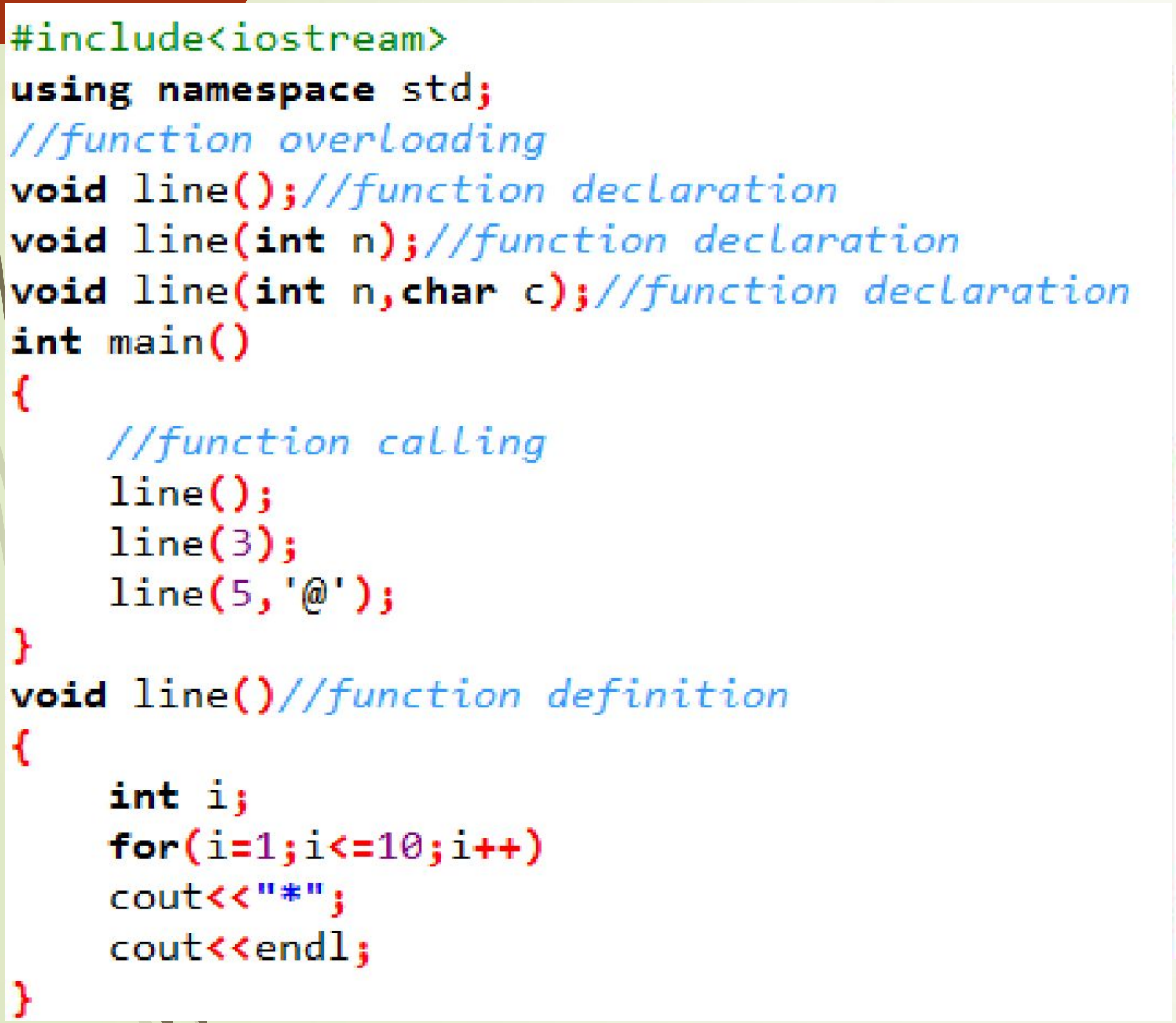
Function Overloading

- The process of declaring multiple functions with same name but different parameters is called function overloading.
- The function with same name must differ in one of the following ways:
- Number of parameters
- Type of parameter
- Sequence of parameters.
- **For Example**

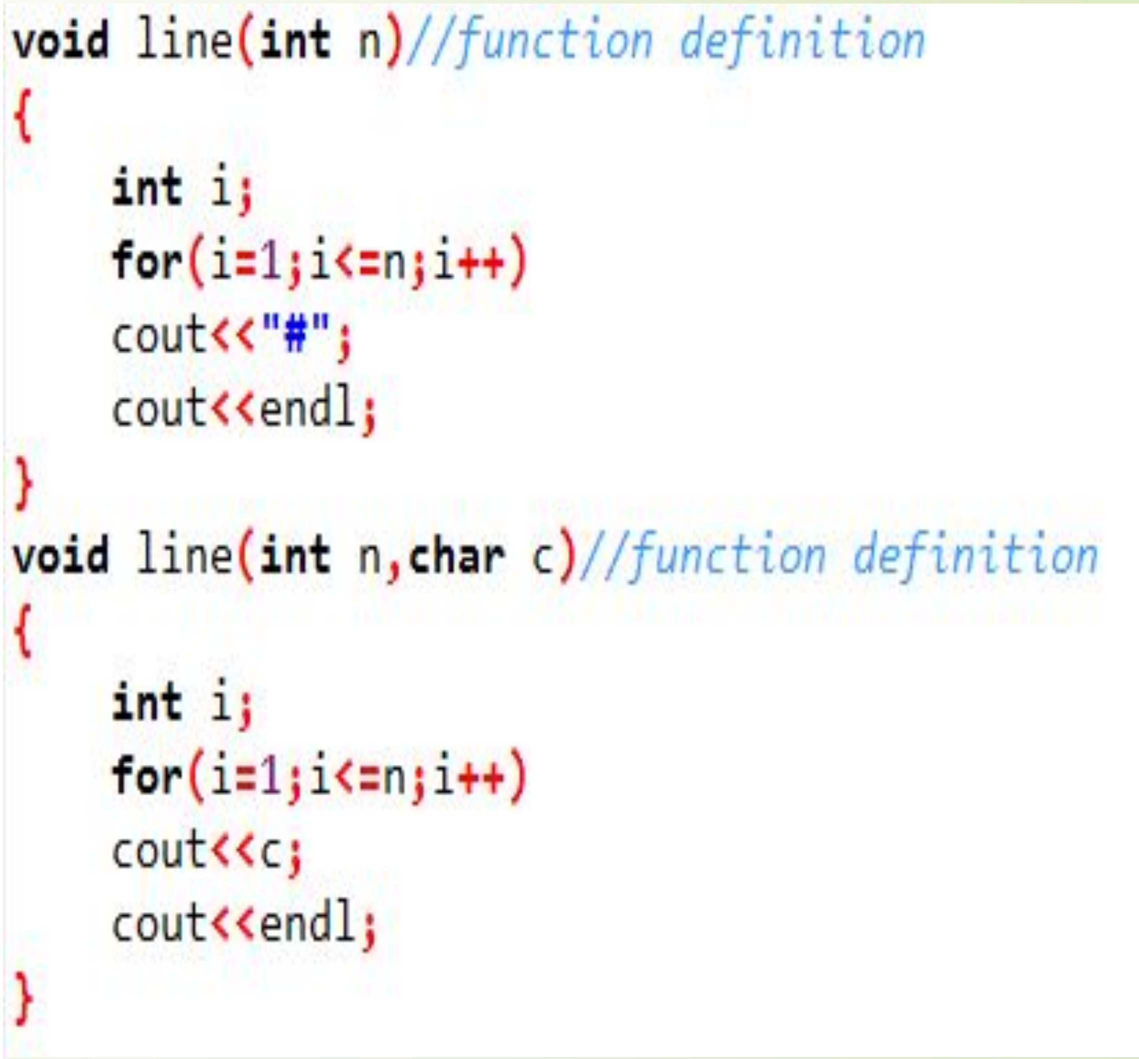
```
void sum();
void sum(int, int);
void sum(float, float);
```

  - In order to avoid confusion compiler check the number of arguments, and type of arguments.

Example



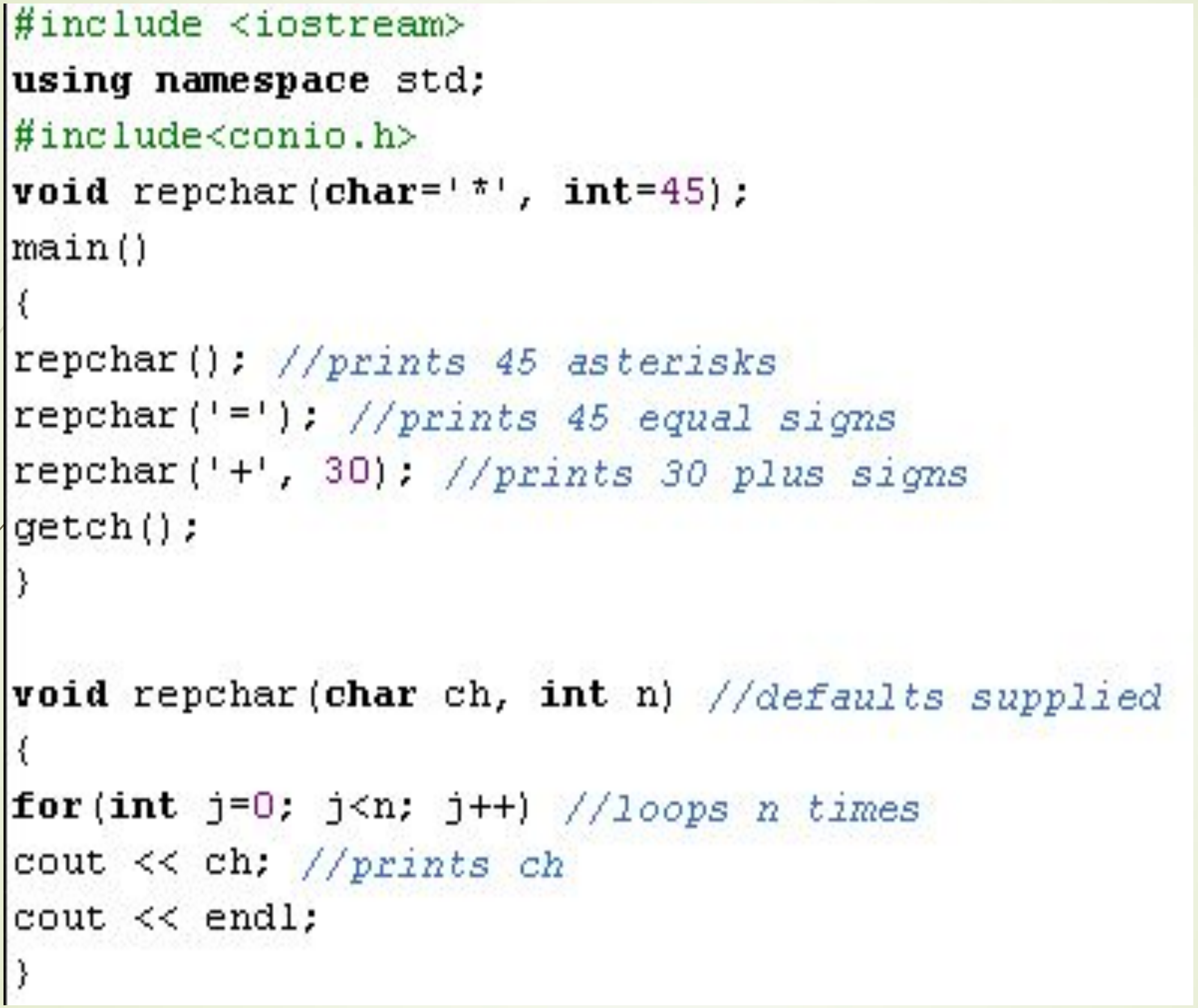
```cpp
#include<iostream>
using namespace std;
//function overloading
void line();//function declaration
void line(int n);//function declaration
void line(int n,char c);//function declaration
int main()
{
    //function calling
    line();
    line(3);
    line(5,'@');
}
void line()//function definition
{
    int i;
    for(i=1;i<=10;i++)
    cout<<"*";
    cout<<endl;
}
```



```cpp
void line(int n)//function definition
{
    int i;
    for(i=1;i<=n;i++)
    cout<<"#";
    cout<<endl;
}
void line(int n,char c)//function definition
{
    int i;
    for(i=1;i<=n;i++)
    cout<<c;
    cout<<endl;
}
```

Default Arguments

- A function can be called without specifying all its arguments.
- This won’t work on just any function: The function declaration must provide default values for those arguments that are not specified.



```
#include <iostream>
using namespace std;
#include<conio.h>
void repchar(char='*', int=45);
main()
{
repchar(); //prints 45 asterisks
repchar('='); //prints 45 equal signs
repchar('+', 30); //prints 30 plus signs
getch();
}

void repchar(char ch, int n) //defaults supplied
{
for(int j=0; j<n; j++) //loops n times
cout << ch; //prints ch
cout << endl;
}
```

Practice Task

You are tasked with creating a simple tool for a math competition. The tool must perform the following operations based on participant input using functions, and return the results to the main function for display:

- A participant wants to know how many ways they can arrange 5 items.
- Another participant is asked to designing a lottery system that only accepts prime numbers for entry. Using the program, how would you verify whether the entered number qualifies as a prime number for a special bonus round.
- Participant wants to calculate first number raised to the power of second number.
- After completing all the tasks, participant need to close the tool. How would you do this?

Questions?