
بسم الله الرحمن الرحيم

Introduction

M.Nabeel Khan	(61)
Daud Mirza	(57)
Danish Mirza	(58)
Fawad Usman	(66)
Aamir Mughal	(72)
M.Arslan	(17)

Programming Fundamentals

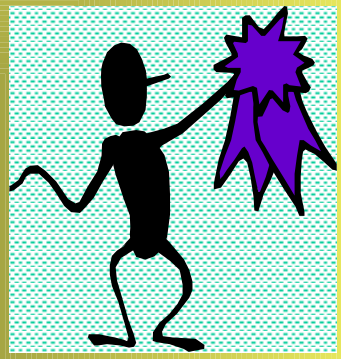
Introduction to Functions

Importance of Function

- A program may need to repeat the same piece of code at various places.
- It may be required to perform certain task repeatedly.
- The program may become very large if functions are not used.
- The real reason for using function is to divide program into different parts.

Advantages of Functions

- Easier to Code
- Easier to Modify
- Easier to Maintain
- Reusability
- Less Programming Time
- Easier to Understand



C++ Functions

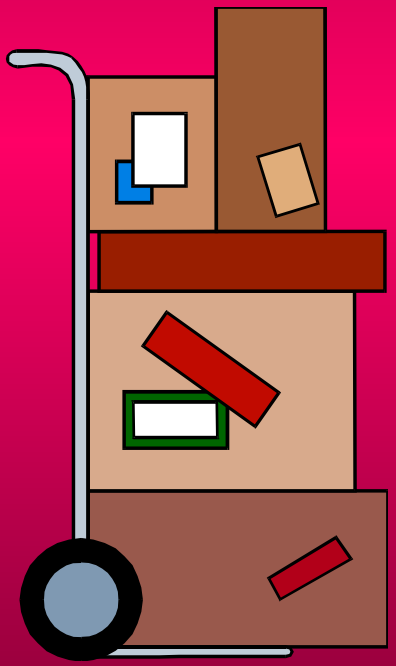
- C++ allows the use of both internal (user-defined) and external (**BUILT IN**) functions.

- **EXTERNAL FUNCTIONS** are usually grouped into specialized libraries (e.g., **`iostream`**, **`stdlib`**, **`math`**, etc.)

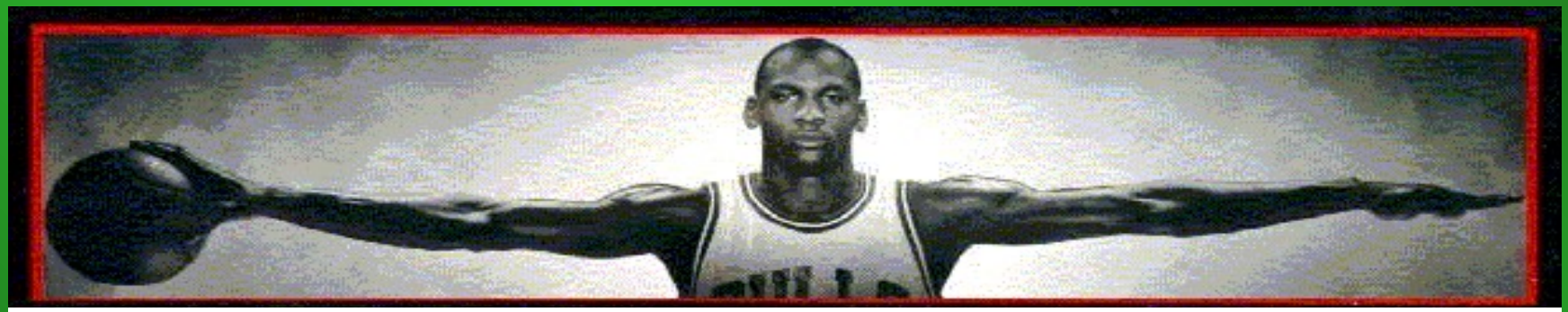
User-Defined Functions

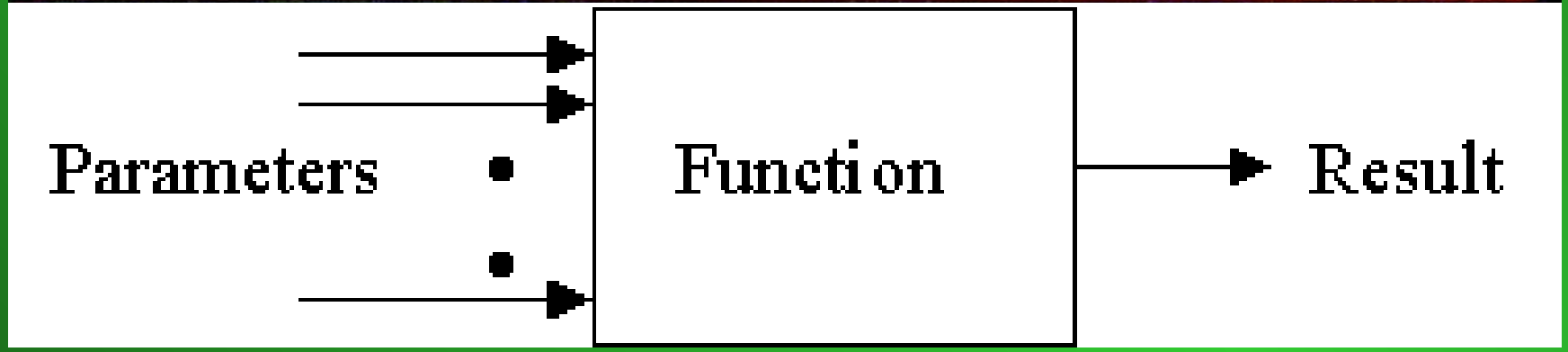
C++ programs usually have the following form:

```
// include statement
// function prototype
 // main() function
 // function definitions
```



Function Input and Output





Function Definition

A set of statements that explains what a function does is called **FUNCTION Definition**

A function definition can be written at:

- Before main() function
- After main() function
- In a separate file

Syntax of Function Definition

Return-type Function-name (parameters)

Function header

```
{
    statement  1;
    statement  2;
          :
          :
          :
    statement  N;
}
```

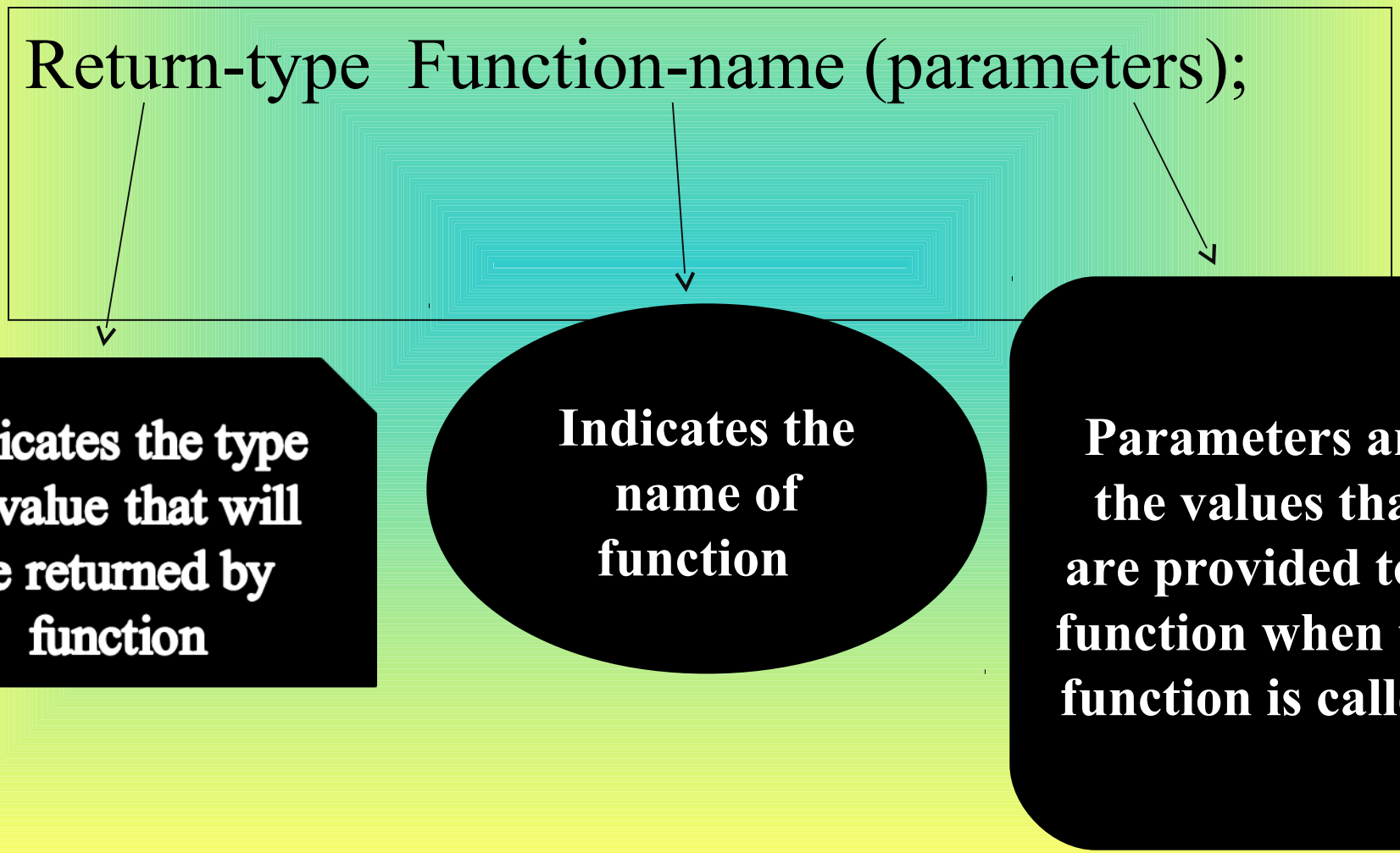
Function body

Function Declaration

Function declaration is the model of a function. It is also known as **FUNCTION PROTOTYPE**. It provides information to compiler about the structure of function to be used in program. It ends with semicolon (;). It consists of:

- FUNCTION NAME
- FUNCTION RETURN TYPE
- NUMBERS & TYPES OF PARAMETERS

Syntax of Function Declaration



Example of Function Declaration & Function Definition

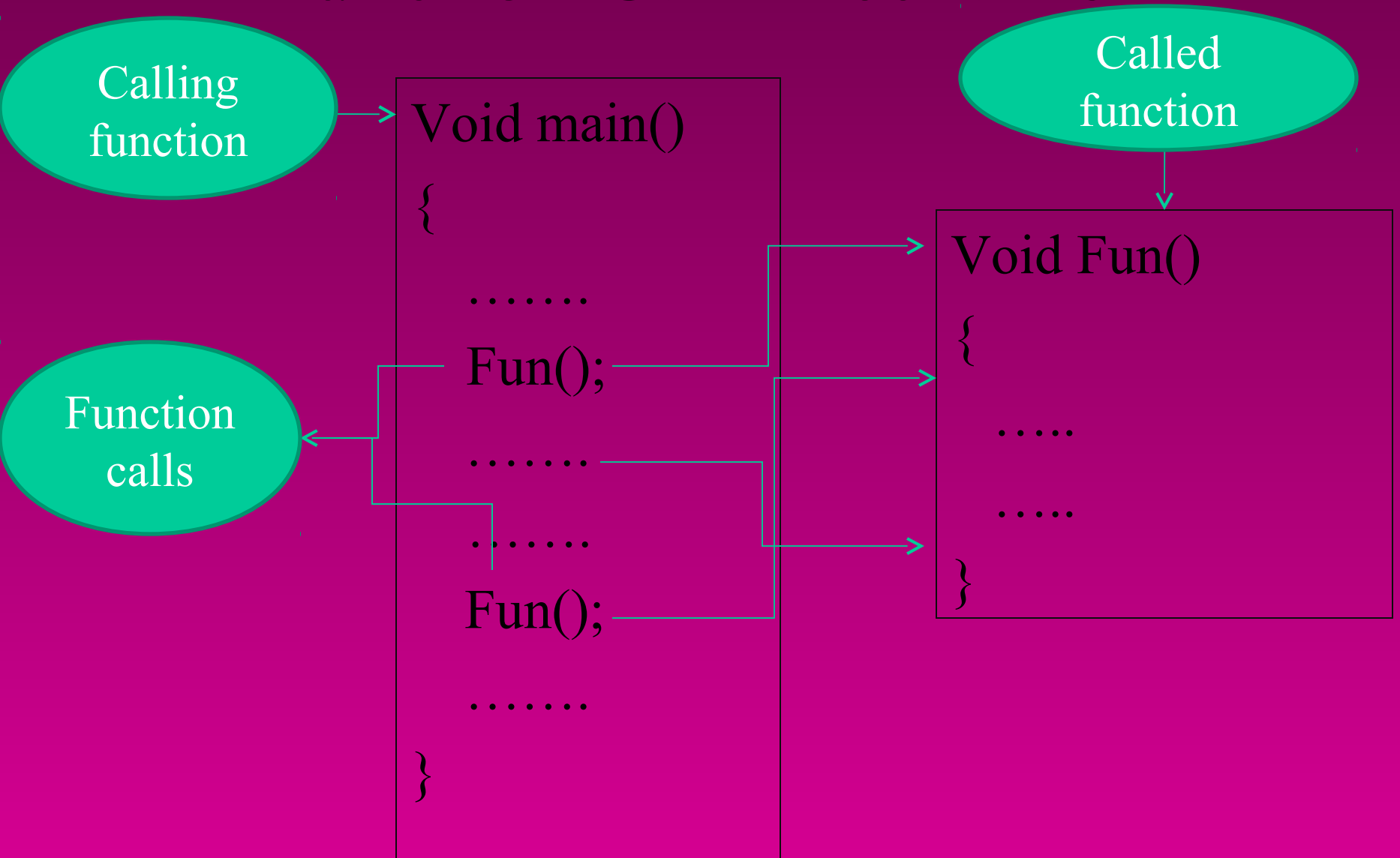
```
1   #include<iostream>
2   using namespace std;
3
4   int sum (int x , int y);          // function declaration
5   int sum(int x , int y)            // funciotn definition
6   {
7      int result;
8     result = x  + y;
9     return (result);
10  }
11  int main()
12  {
13     int x , y , output ;
14    x = 20;
15    y = 100;
16    output = sum(x,y);          /* calling a function and storing the
17                                value from funciton to variable output*/
18   cout<<output;
19
20  return 0;
21  }
```

Function Call

The statement that activates a function is known as **FUNCTION CALL**.The following steps take place when a function is called:

1. The control moves to the function that is called.
2. All statements in function body are executed.
3. Control returns back to calling function.

Function Call Mechanism



Example of Function Call

```
int main()
{
    double num1, num2, maxNum;
    cout<<"Please enter a number :";
    cin>> num1;
    cout<<"Great!\nPlease enter a second number :";
    cin>> num2;
    FindMax(num1, num2, maxNum); //calling

    system ("PAUSE");
    return 0;
}
```

Scope of Functions

Area in which a function can be accessed is known as **SCOPE OF FUNCTION**.

These are two types:

1. **Local Function**

   A function that is declared in another function is called Local Function.

1. **Global Function**

   A function that is declared outside any function is called Global Function.

Call by Value

- Call by value passes the value of actual parameter to formal parameter.
- Actual & formal parameter refer to different memory location.
- It requires more memory.
- It is less efficient.

Call by Reference

- Call by reference passes the address of actual parameter to formal parameter.
- Actual & formal parameter refer to same memory location.
- It requires less memory.
- It is more efficient.

Local Variable

- Local variables are declares within a function.
- It can be used only in function in which they declared.
- Local variable are destroyed when control leave the function.
- Local variables are used when the vales are to be used within a function.

Global Variable

- Global variables are declares outside any function
- Global variables can be used in all function.
- Global variable are destroyed when the program is terminated.
- Global variables are used when values are to be shared among different functions.

A program that calls a function for five times using loop & also using static(local) variable.

A **local variable** declared with keyword **STATIC** is called **STATIC VARIABLE**. It is used to increase the lifetime of **local variable**.

- This program declare function **Fun().**
- Function declare **static variable**
- Initializes it to **0**.
- The **main()** function calls five time.
- Using **FOR LOOP**.
- Last statement display value of "**n**".

```
#include <iostream>
Using namespace std;
Int fun();
Int main()
{   int I;
    for(  i=0;i<=5;i++)
    fun();
    return 0;      }
Int fun()
{   static int n=0;
    n++
    cout<<"value of n
    ="<<n<<endl;
}
```

OUTPUT

Value of n=1
Value of n=2
Value of n=3
Value of n=4
Value of n=5

Register Variable

A variable declared with keyword register is known as **Register variable**. The value of register is stored in **Registers** instead of **RAM** because **Registers** are faster than **RAM** so value stored in **Registers** can be accessed faster than **RAM**. **Syntax** for declaring is:

REGISTER DATA-TYPE VARIABLE-NAME

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:

1. Numbers of parameters
2. Type of parameter
3. Sequence of parameters

Thank You

THANK YOU FOR WATCHING MY PRESENTATION, I HOPE YOU ENJOYED IT...