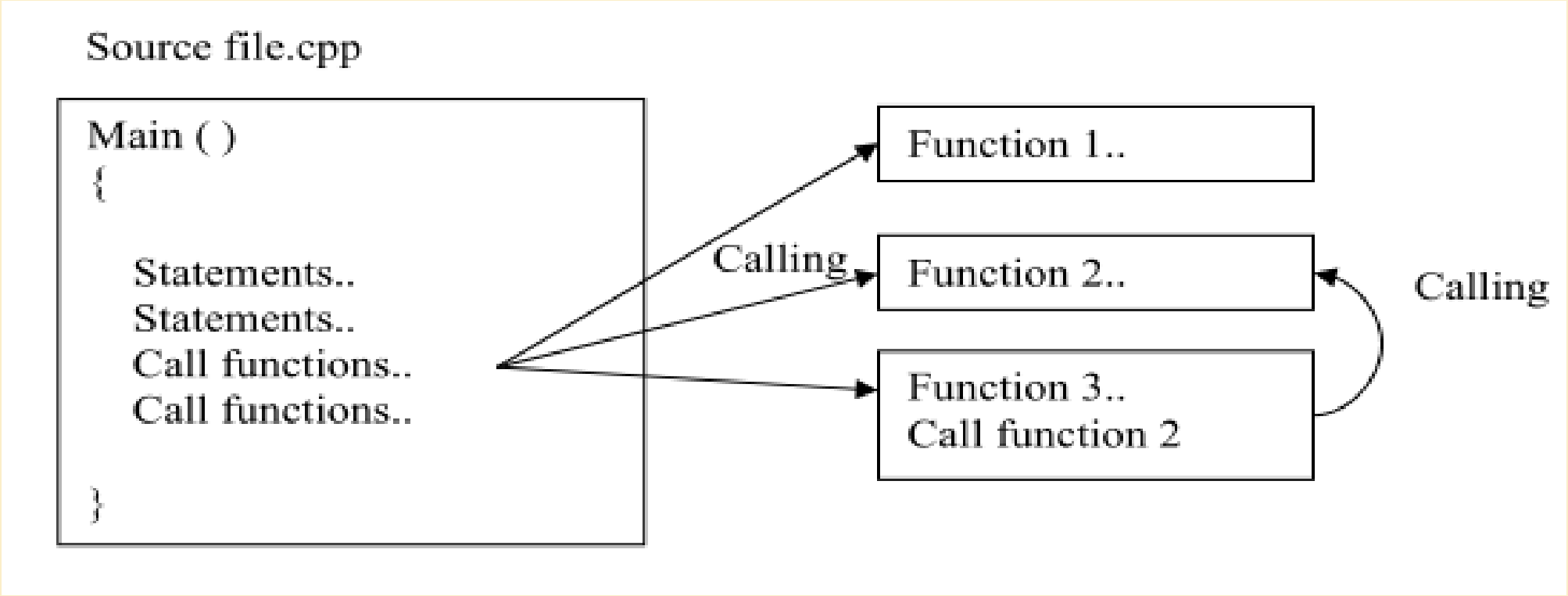
Functions

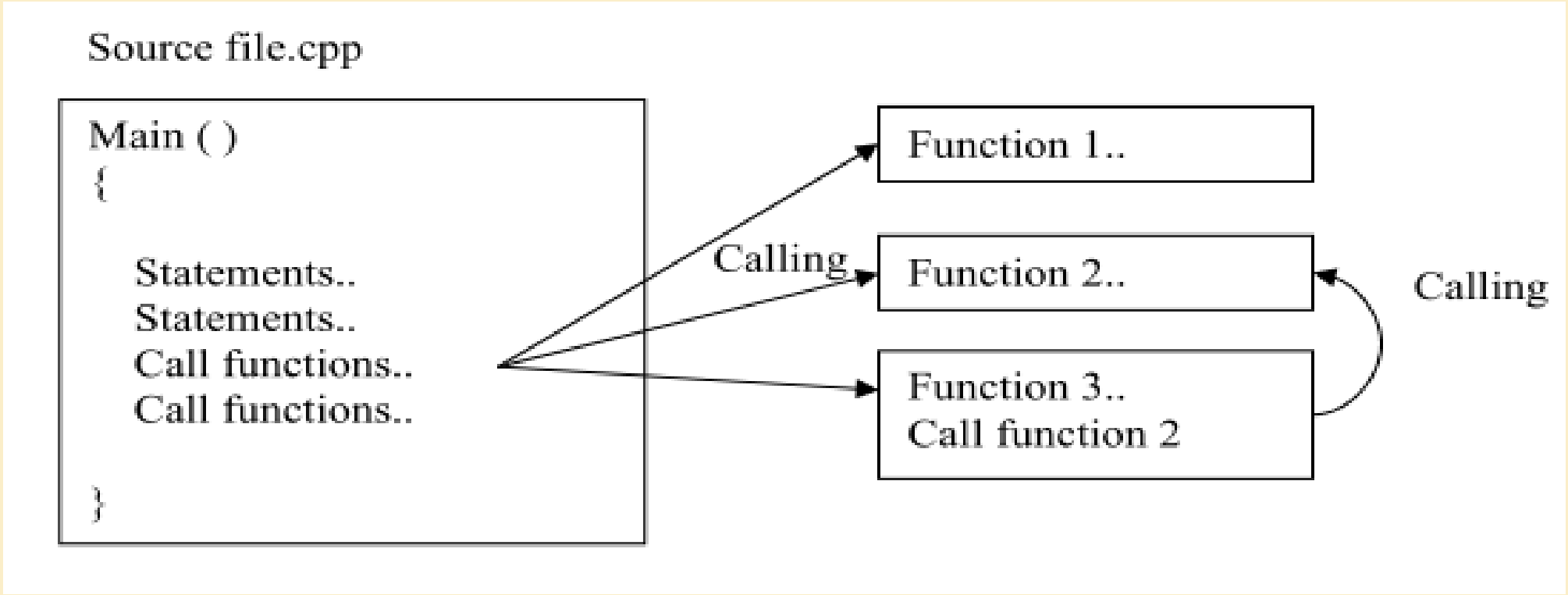


Dr. Zaid Ameen

Functions

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is **main(),** and all the most trivial programs can define additional functions. You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is so each function performs a specific task. **A function declaration** tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

Procedures and functions (subprograms) may have parameters. These represent the objects from our program that are used in the subprogram.



Defining a Function:

Defining a Function:

The general form of a C++ function definition is as follows:

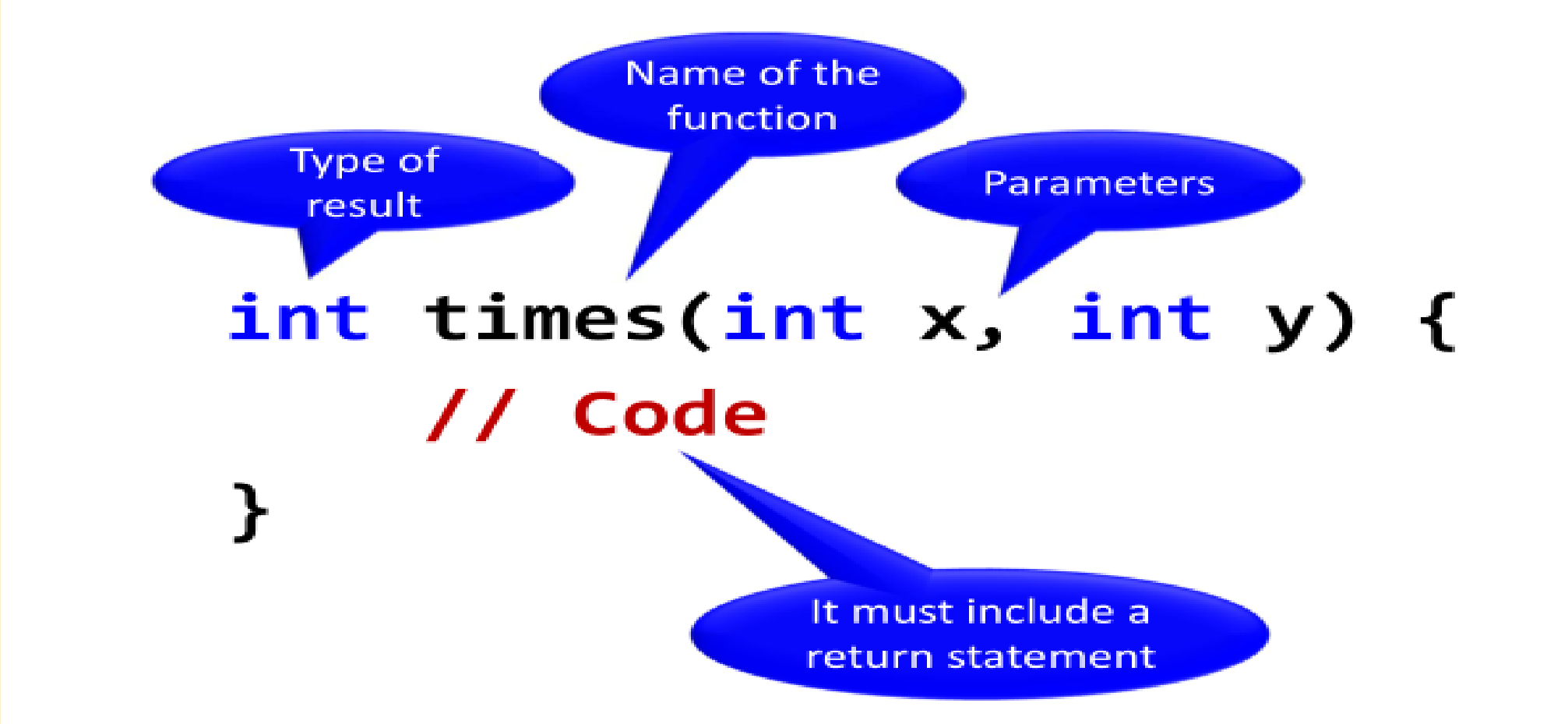
```
return_type function_name ( parameter list )
   {
      body of the function
    }
```

A C++ function definition consists of a function header and a function body. Here are all the parts of a function:

- **Return Type**: A function may return a value. The return_type is the data type of the value the function returns. Procedure performs the desired operations without returning a value. In this case, the return_type is the keyword void.

- **Function Name:** This is the actual name of the function. The function name and the parameter list together constitute the function signature.

- **Parameters**: A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- **Function Body:** The function body contains a collection of statements that define what the function does.
- **Calling Place:** The calling place refers to calling point which is located in main function or subprograms functions (function/procedure).



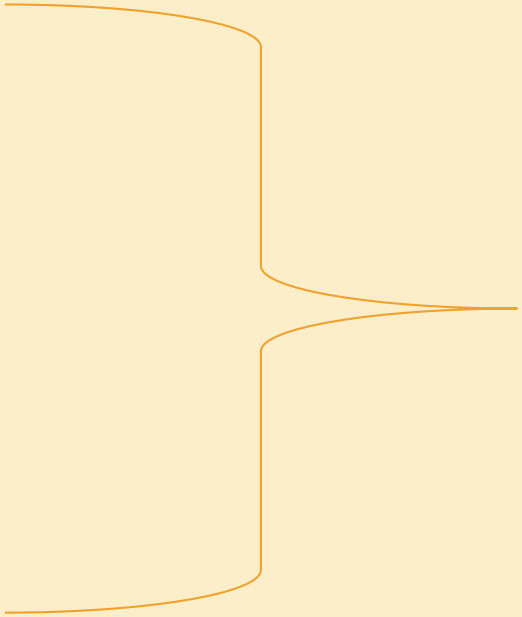
Example:
Following is the source code for a function called **max()**. This function takes two parameters num1 and num2 and returns the maximum between the two:

```
// function returning the max between two numbers

int max(int num1, int num2)
{
   // local variable declaration
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```



Function Declarations:

A function **declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts:

return_type function_name (parameter list);

For the above defined function max(), following is the function declaration:

int max (int num1, int num2);

Parameter names are not important in function declaration only their type is required, so following is also valid declaration:

int max (int, int);

Function declaration is required when you define a function in one source file and you call that function in another file. In such case, you should declare the function at the top of the file calling the function.

Calling a Function:

While creating a C++ function, you give a definition of what the function has to do. To use a function, you will have to call or invoke that function.

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value. For example:

```cpp
#include <iostream>
using namespace std;
// function declaration
int max(int num1, int num2);
int main()
{
    // local variable declaration:
    int a;
    int b;
    int ret;
    cout << "Enter a & b" << endl;
    cin >> a >> b;
    // calling a function to get max value.
    ret = max(a, b);
    cout << "Max value is : " << ret << endl;
    return 0;
}// function returning the max between two numbers
int max(int num1, int num2)
{
    // local variable declaration
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
```

2) Write a program to compute average of n integer numbers by using subprograms principle.

```
#include <iostream>
using namespace std;
// function declaration
float computeAvge(int);
int main()
{
    // local variable declaration:
    int n;
    float avg;
    cout << "n" << endl;
    cin >> n;
    // calling a function to get Avg value.
    avg = computeAvge(n);
    cout << "Average is : " << avg << endl;
    return 0;
}
// body of function
float computeAvge(int n)
{
    float total = 0, value;
    for (int j = 0; j < n; j++)
    {
        cout << "enter value" << endl;
        cin >> value;
        total += value;
    }
    return (total / n);
}
```

3) Write a program to compute Maximum of n integer numbers by using subprograms principle.

```
#include <iostream>
using namespace std;
// function declaration
int computeMax(int);
int main()
{  // local variable declaration:
    int n, Max;
    cout << "Enter n" << endl;
    cin >> n;
    // calling a function to get Max value.
    Max = computeMax(n);
    cout << "Maximum is : " << Max << endl;
    return 0;
}
// body of function
int computeMax(int n)
{
    int M, value;
    cout << "enter value" << endl;
    cin >> value;
    M = value;
    for (int j = 1; j < n; j++)
    {
        cout << "enter value" << endl;
        cin >> value;
        if (value > M)
            M = value;
}
  return M;
}
```

4) Write a program to compute factorial of integer number no based on subprogram principles.

```
#include <iostream>
using namespace std;
// function declaration
int factorial(int);
int main()
{
    // local variable declaration:
    int no, F;
    cout << "Enter no" << endl;
    cin >> no;
    // calling a function to get factorial value.
    F = factorial(no);
    cout << " factorial is : " << F << endl;
    return 0;
}
// body of function
int factorial(int no)
{
    int Fact;
    Fact = 1;
    for (int j = 1; j <= no; j++)
        Fact = Fact * j;
    return Fact;
}
```

cin>> no=4
Fact= 1;
Loop1: j=1 <= no=4 T
Fact= 1 * 1= 1; j++ ;j=2

Loop2: j=2 <= no=4 T
Fact= 1 * 2= 2; j++ ;j=3

Loop3: j=3 <= no=4 T
Fact= 2 * 3= 6; j++ ;j=4

Loop4: j=4 <= no=4 T
Fact= 6 * 4= 24; j++ ;j=5

Loop5: j=5 <= no=4 F

Return Fact= 24
F= factorial (no=4) = Fact=24
Cout<< F =24

5) Write a program to compute the following mathematic series based on subprogram principles.
2/3!+4/6!+6/9!+.....n

```
#include <iostream>
using namespace std;
// function declaration
int factorial(int);
int main()
{
    // local variable declaration:
    int n;
    float F, s=0;
    cout << "Enter n" << endl;
    cin >> n;
    for (int i = 1;i <= n;i++)
    {// calling a function to get factorial value.
        F = factorial(3 * i);
        s = s + (2 * i / F);
    }
    cout << " Sum is : " << s << endl;
    return 0;
}
// body of function
int factorial(int no)
{
    int Fact;
    Fact = 1;
    for (int j = 1; j <= no; j++)
        Fact = Fact * j;
    return Fact;
}
```

```
S=0
For (int i=1; i<=n; i++)
{
 F= factorial(3*i)
S = s +( 2*i)/ F;
}
//no= 3*i

Int factorial(int no)
{
 int Fact;
Fact= 1;
for (int j=1; j <= no; j++)
  Fact= Fact  * j;
return Fact;
}
```