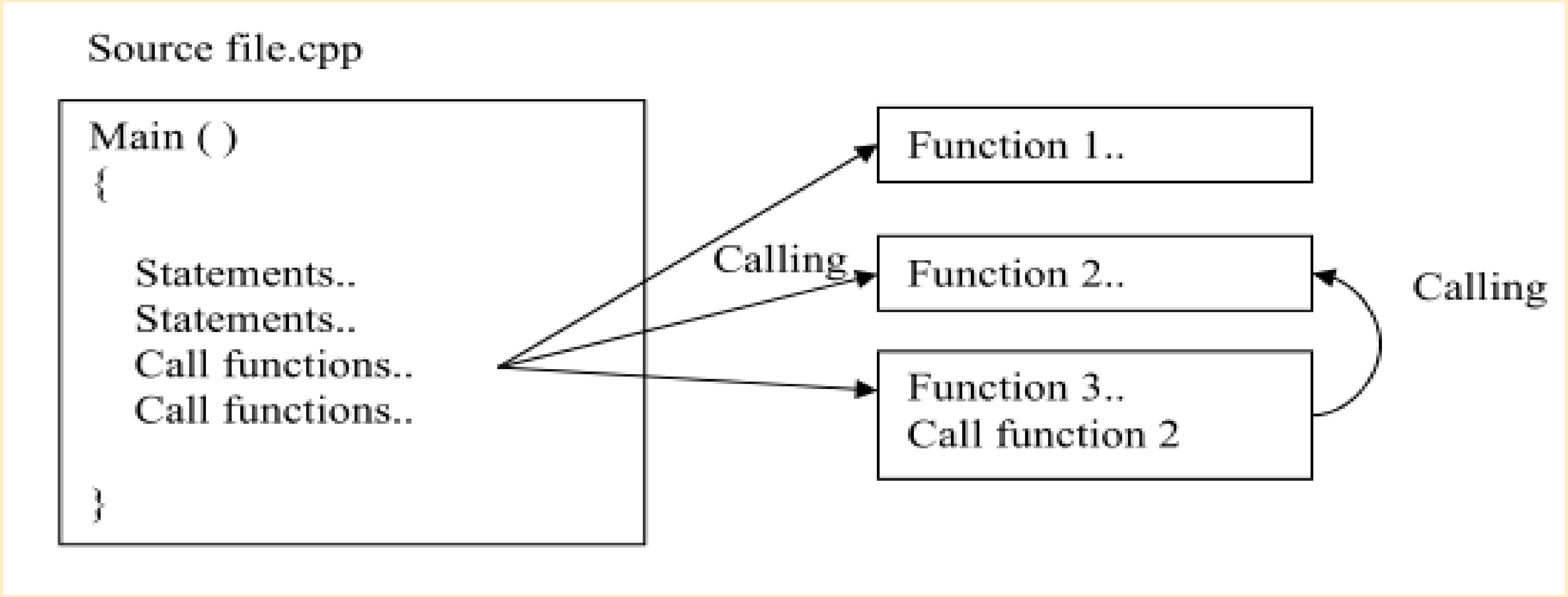
Procedure



Dr. Zaid Ameen

Defining a Procedure:

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

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

Using (Formal) Parameters in C++

- In C++, there are two types of (formal) parameters: **value** parameters, and **reference** parameters.
- A **value parameter** is used by a function to receive a value from the calling function.
- A **reference parameter** is used by a function to supply and/or receive a value from the calling function.
- When a **function** is called, the arguments (or actual parameters) must be specified as follows:

– For a value parameter, you must specify a value (or an expression):

A memory location is created for each value parameter, and it is initialized with that value.

– For a **reference parameter**, you must specify a variable name:

This variable name replaces the reference parameter in the body of the function.

– The data types of the arguments and the parameters must be compatible.

Example :Using value and reference parameters:

```
#include <iostream>
using namespace std;
void  funct1(int vnum1, int rnum2, int &rnum3)
{
vnum1 += 2; //6
rnum2 += 3; //8
//cout << " The sum is= " << vnum1 + rnum2 << endl;
rnum3 = vnum1 + rnum2;
cout << endl << "vnum1 = " << vnum1 << "\trnum2 = " << rnum2 << "\trnum3 = " << rnum3;
}
void main()
{
int num1 = 4, num2 = 5, num3;
funct1(num1, num2, num3);// by Reference
cout << endl << "num1 = " << num1 << "\tnum2 = " << num2 << "\tnum3 = " << num3;
}
```

The output of this program is: num1= 6, num2= 8,   num3 = 14

Example : Using value and pointer parameters

```
#include <iostream>// Using value and pointer parameters
using namespace std;
void  funct1(int vnum, int *pt1, int *pt2)
{
vnum += 2;
*pt1 += 3;
*pt2 = vnum + *pt1;
cout << "*pt1=  " << *pt1 << "    pt1=  " << pt1 << endl;
pt1++;
cout << "    pt1++= " << pt1 << endl;

}
int main()
{
int num1 = 4, num2 = 5, num3;
funct1(num1, &num2, &num3);
cout << endl << "num1 = " << num1 << "\tnum2 = " << num2
<< "\tnum3 = " << num3;
cout << "&num2=  " << &num2 << endl;
num2++;
cout << "num2=  " << num2 << endl;
return 0;
}
```

Overloaded functions: In C++ two different functions can have the same name if their parameter types or number are different. That means that you can give the same name to more than one function if they have either a different number of parameters or different types in their parameters. For example:

```
#include <iostream>// Overloaded function
using namespace std;
int operate(int a, int b)
{
   return (a * b);
}
float operate(float a, float b)
{
   return (a / b);
}
int main()
{
   int x = 5, y = 2;
   float n = 5.0, m = 2.0;
   cout << operate(x, y);
   cout << "\n";
   cout << operate(n, m);
   cout << "\n";
   return 0;
}
```

Example) Write a program to enter n integer numbers and then check each one if odd or even number based on using procedure principle.
// declaring functions prototypes.

```
#include <iostream>
using namespace std;
void odd(int);
void even(int);
int main() {
    int n, no;
    cout << "Enter n" << endl;
    cin >> n;
    for (int i = 0;i < n;i++)
    {
        cout << "Enter number " << endl;
        cin >> no;
        odd(no);
    }
    return 0;
}
void odd(int a)
{
    if ((a % 2) != 0)
        cout << "Number is odd.\n";
    else
        even(a);
}
void even(int a)
{
    if ((a % 2) == 0)
        cout << "Number is even.\n";

}
```

Example) Write a program to compute and print summation of the following equation:

5+10+15+…….+n

```
#include <iostream>
using namespace std;
void sum(int &, float &);
int main()
{
    int  n;
    float s;
    cout << "Enter n" << endl;
    cin >> n;
    s = 0;
    cout << "s=  " << s << endl;
    sum(n, s);
    cout << "s = :" << s << endl;
    cout << "n = :" << n << endl;

    return 0;
}
void sum(int &n, float &s)
{
    for (int i = 1;i <= n; i++)
        s = s + 5 * i;


    n += n;
}
```