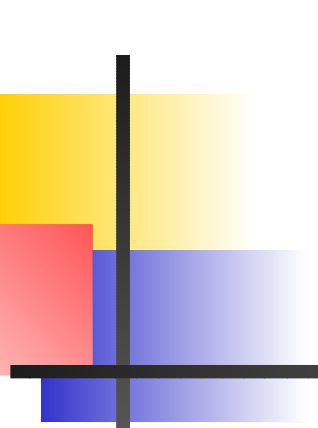




VOID FUNCTION IN C++

...

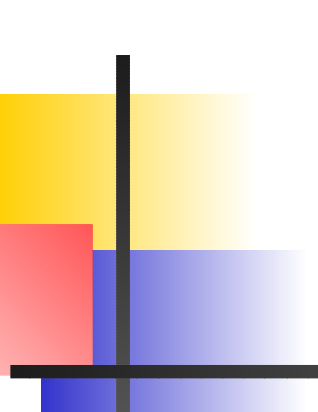
S. Rajasekar

School of Physics

Bharathidasan University

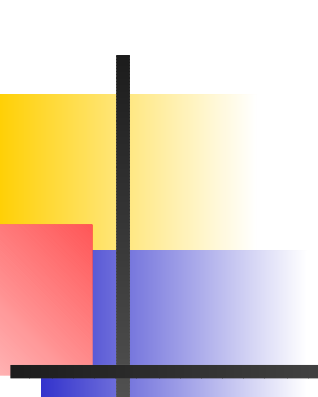
Tiruchirapalli - 620 024

email: rajasekar@cnld.bdu.ac.in
srj.bdu@gmail.com



Definition

When a program is large it will be difficult to understand it. For simplicity, easy understanding and minimize the length of a program, the job of the program can be divided into a number of subtasks. These subtasks may be performed independently in separate subprograms. Then the subprograms can be linked together to do the entire job. For example, we want to write a program to compute a mean value, variance, moments and probability distribution of a set of numbers. We can write separate programs (subprograms) for each of these tasks and then combine them into a single program. *These subprograms are called void functions.*



General Format

The name of a void function cannot assume a value. However, any number of values calculated in the void function subprogram can be brought into the main program. A main program itself can be written as a void function. The following is the general form of a program with a void function.

```
void name (arguments);
```

```
main ()
```

```
{
```

```
    -----
```

```
    name (arguments);
```

```
    -----
```

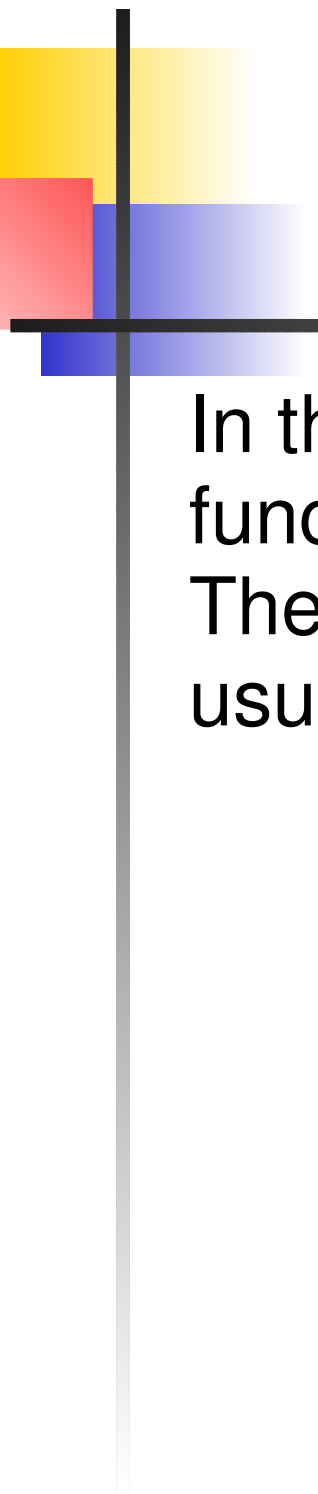
```
}
```

```
void name (arguments)
```

```
{
```

```
    -----
```

```
}
```



General Format

In the above name is the name of the void function and arguments are the list of variables. The task of a void subprogram is specified usually at the end of the program.

Example

Write a program using a void function for arranging the numbers a and b in ascending order.

// Program for arranging a and b in ascending order using a void function

```
#include <iostream.h>
```

```
#include <math.h>
```

```
#include <conio.h>
```

```
void ascending (double a, double b);
```

```
main ( )
```

```
{
```

```
    double a, b;
```

```
    cin>>a>>b;
```

```
    ascending (a, b);
```

```
    return 0;
```

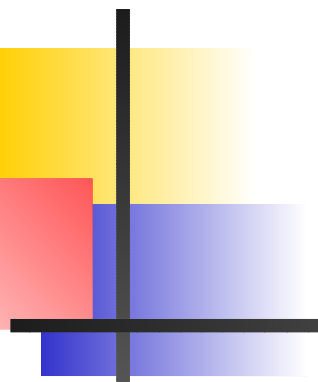
```
}
```

```
void ascending (double a, double b)
```

```
{
```

```
    double tem;
```

```
    if (a>b)
```



Example

```
{  
    tem = a;  
    a = b;  
    b = tem;  
}  
cout<<"a = "<<a<<" b = "<<b<<endl;  
}
```

Input 5 3

Output 3 5

Void function with passing-by-value and passing-by-reference

Consider the following program for arranging the two numbers a and b in ascending order: // Program for arranging a and b in ascending order using a void function

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
void ascending (double a, double b);
main ( )
{
    double a, b;
    cin>>a>>b;
    ascending (a, b);
    cout<<"a = "<<a<<" b = "<<b<<endl;
    return 0;
}
void ascending (double a, double b)
{
    double tem;
    if (a>b)
```

Void function with passing-by-value

```
{  
    tem = a;  
    a = b;  
    b = tem;  
}  
cout<<"a = "<<a<<" b = "<<b<<endl;  
}
```

In the above program for the two numbers a=5 and b=3 the output statement in the void function prints the result as

a=3 b=5

which is correct. While the cout in the main program prints a=5 and b= 3. That is, the changes happened to a and b in the void function subprogram are not transferred to the main function. The changes happened to a and b are restricted to the void function only. The values of a and b in the main program are unaltered. This means the variables a and b are **local variables** to the void function. In the main function the variables a and b are **read-only-variables**. This is known as **pass-by-value** mechanism.

Void function with passing-by-reference

Suppose we wish to update the values of a and b in the main program. This can be done by passing the values by **reference** and is called **passing-by-reference**. To pass a value of a variable by reference simply attach the ampersand symbol '&' to the type specifier in the function. In this case the argument is **read-write** instead of read only. Then any change to the local variables in the void function subprogram will be passed to the corresponding variables in the main program.

Example:

// Program for arranging a and b in ascending order using a void function

```
#include <iostream.h>
```

```
#include <math.h>
```

```
#include <conio.h>
```

```
void ascending (double &a, double &b);
```

```
main ( )
```

```
{
```

```
    double a, b;
```

```
    cin>>a>>b;
```

```
    ascending (a, b);
```

```
    cout<<"a = "<<a<<" b = "<<b<<endl;
```

Void function with passing-by-reference

```
    return 0;
}
void ascending (double &a, double &b)
{
    double tem;
    if (a>b)
    {
        tem = a;
        a = b;
        b = tem;
    }
}
```

Input 5 3

Output 3 5

Difference between passing-by-value and passing-by-reference

The difference between passing-by-value and passing-by-reference are listed below.

S.No.	Passing-by-value	Passing-by-reference
1	Ex: double a;	double &a;
2	a is a local variable	a is a local reference
3	a is a duplicate of the variable.	a is a synonym for the variable.
4	It cannot change the value of the variable in the main function.	It can change the value of the variable in the main function.
5	The variable is read-only.	The variable is read-write.