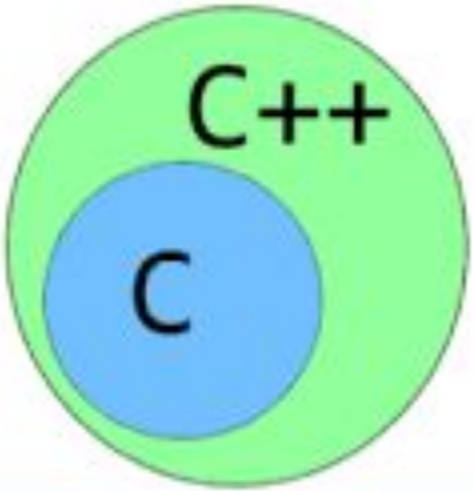
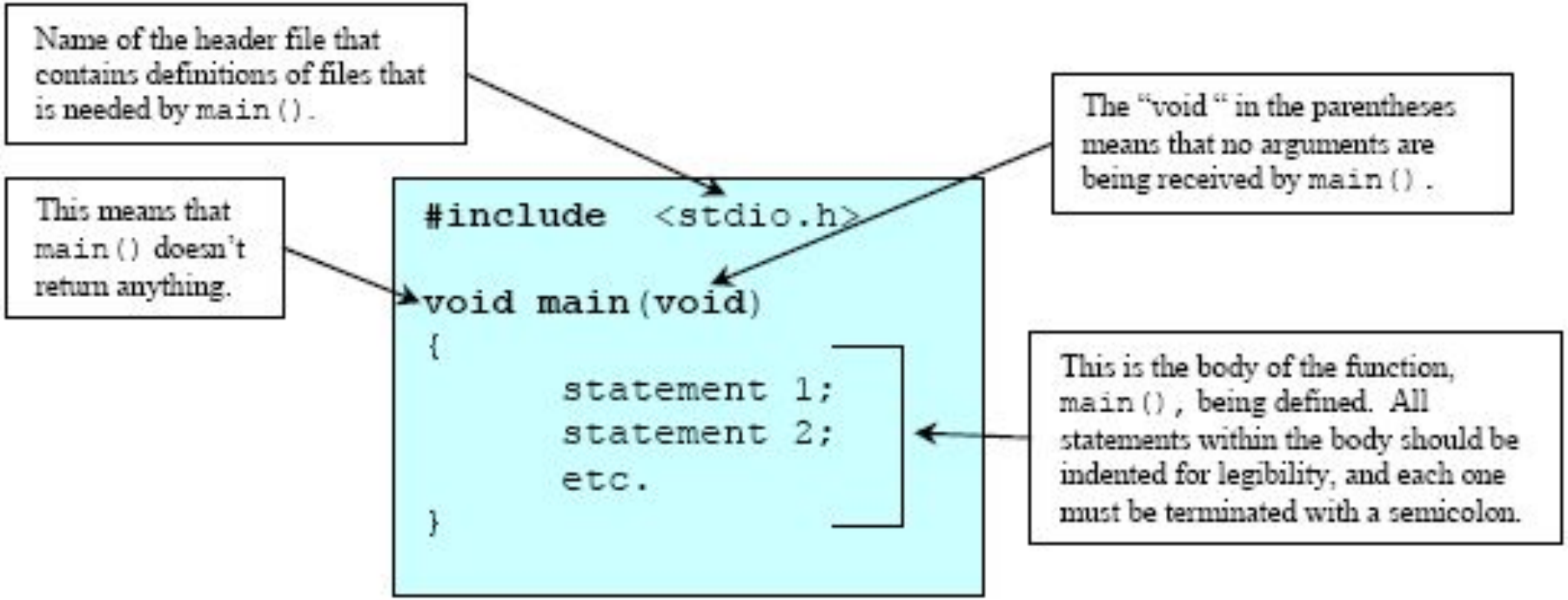
# Introduction to C / C++ Programming

Calling C functions from R Programming
[UNIT - 5 ]

Asst. Prof. Ashwini Mathur

C++ is a superset of C






Program 1: Basic Structure of the C/C++ program


CPP
10101101
11010101
0001010
Source Code
Compile
Machine Code

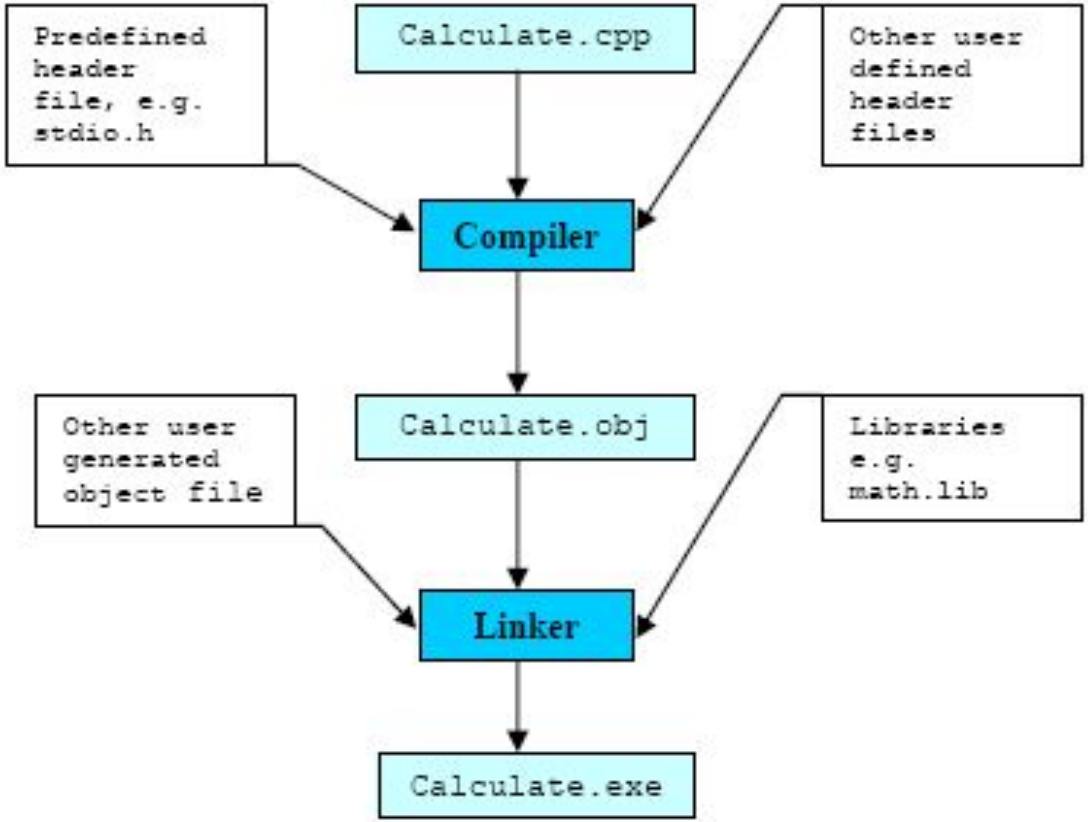
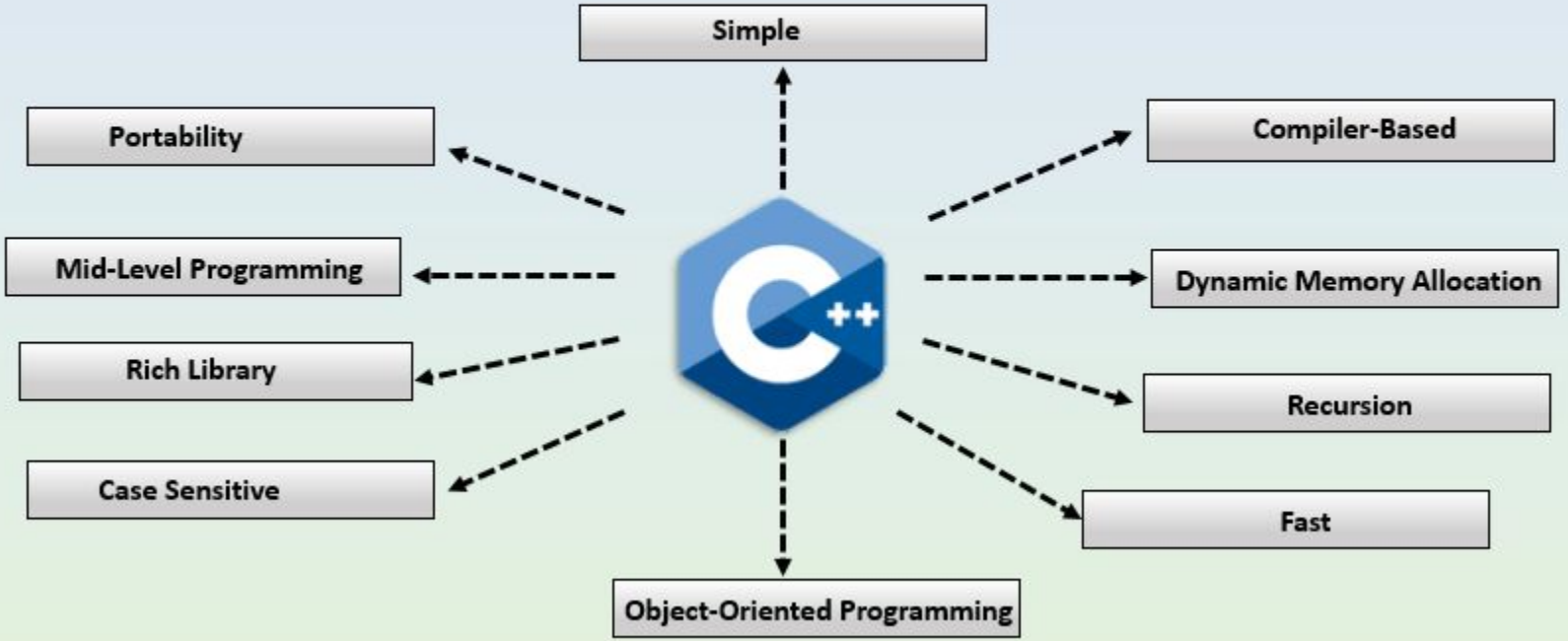



Figure 1.1: Typical program development steps

# Features of C++




www.educba.com


Compatible with other tools
Open Source and Free
Huge community
R
Great Machine learning capability
Awesome Graphical Capabilities
Thousand of packages

# Calling C Functions using R Programming

- R is a programming language which provides statistical and graphical techniques for data analysis.

- C functions can be called from R programming.

# R Programming

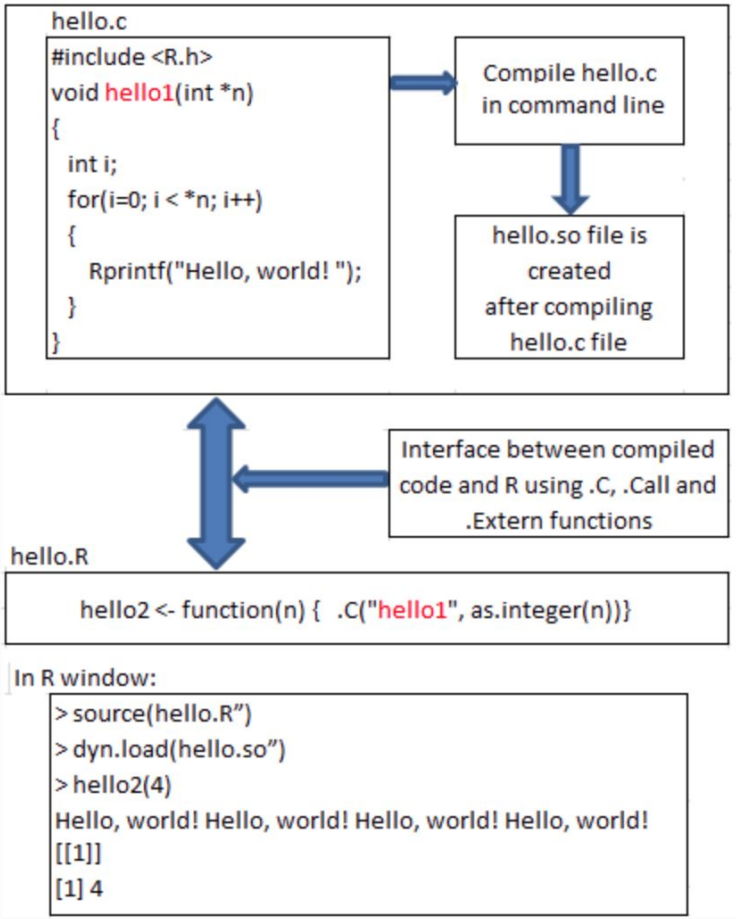
- R is a programming language which provides statistical, graphical techniques for data analysis.
- R is an interactive, open source and object oriented programming language designed for statisticians.
- R is used to perform data analysis by using scripts and functions which is available in R programming language.
- Complete data analysis could be done  in few lines of code using R programming.

**Purpose of calling C from R programming language:**

- R programming language is slow in iterative algorithm. Iterating very large data sets leads to poor performance in R.

- To improve the performance of R, C functions are called to make use of high performance of C.

PICTORIAL REPRESENTATION FOR CALLING C FROM R PROGRAMMING:



## Methods of calling C functions from R:

The link/interface between C compiled code and R language is made by below 3 functions.

These functions are used to call a C function from R. They are,

1. .C
2. .Call
3. .External

**Creating, Compiling and Dynamic Loading:**

Create a C function with below prerequisite.
Prerequisite:

a) Data type of the function should be void.
b) Compiled code should not return any value except its arguments.

Below is the command to compile C code from command prompt. Once function is written in C, use below command to compile.

```
R CMD SHLIB -lgsl –lgslcblas test.c
```

After compiling, we get the compiled code file name as test.so

Then, this compiled code should be loaded in running R session using the command dyn.load(“test.so”)

Once above dynamic load is done, all the functions written in C file will be available for R to call.

# Writing the C code

If we want to interface C code with R, the functions we write in C need to have a few important properties:

1. C functions called by R must all return void, which means they need to return the results of the computation in their arguments.

2. All arguments passed to the C function are passed by reference, which means we pass a pointer to a number or array.

3. Each file containing C code to be called by R should include the R.h header file. The top of the file should contain the line

# Header include

#include<R.h>

If you are using special functions (e.g. distribution functions), you need to include the Rmath.h header file

When compiling your C code, you use R to do the compilation rather than call the C compiler directly. This makes life much easier since R already knows where all of the necessary header files and libraries are located

# Your First Program

```
hello1 <- function(n) {
                for(i in 1:n) {
                        cat("Hello, world!\n")
                }

        }
```

OUTPUT - ??? hello1(4) ..

Simple. Now what if we wanted to run the `for` loop in C code rather than use pure R? We can do this by writing the relevant C code and calling it from R using `.C`.

First, one must create a separate file containing the C code which will do the work. We will call that file `hello.c`. The file `hello.c` might look something like this:

```
#include <R.h>

void hello(int *n)
{
  int i;

  for(i=0; i < *n; i++) {
    Rprintf("Hello, world!\n");
  }
}
```

As you can see, we have simply taken the loop in R and translated it into C. Notice that there is no `main` function. Since we are not really writing a C *program*, just a C *function*, there is no need for a separate `main` function. The function `Rprintf` is exactly like the standard `printf` function in C except that `Rprintf` sends its output to the R console so that you can see it when running R.

The corresponding R program would be as follows:

```
hello2 <- function(n) {
  .C("hello", as.integer(n))
}
```

The first argument to `.C` is a quoted string containing the name of the C function to be called. Recall that in the file `hello.c` we named our C function `hello`. Therefore,

The first thing you must do is write the code! We put the R code in a file called `hello.R` and the C code in a file called `hello.c`. Having done that, we must then compile the C code. At the command line (in your Terminal window), we can type

```
R CMD SHLIB hello.c
```

Running this command will produce a file called `hello.so`. Now, startup R. In R we type

```
> source(‘‘hello.R’’)
> dyn.load(‘‘hello.so’’)
> hello2(5)
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
[[1]]
[1] 5
```

# Introduction to the components of Python

Next Session..