Basics of C

Learning Objectives

- History of C
- Salient Features of C
- Structure of a C Program
- A Simple C Program
- Compiling a C Program
- Link and Run the C Program.

==========

C is a general-purpose programming language that is extremely popular, simple and flexible. It is machine-independent, structured programming language which is used extensively in various applications.

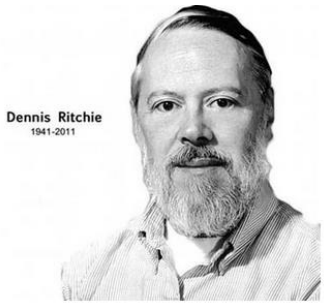
C was the basic language to write everything from operating systems (Windows and many others) to complex programs like the Oracle database, Git, Python interpreter and more.

==========

History of C

1960	**ALGOL**	The base or father of programming languages
1967	'BCPL' which stands for Basic Combined Programming Language	BCPL was designed and developed by Martin Richards, especially for writing system software. This was the era of programming languages
1970	'B'	introduced by Ken Thompson that contained multiple features of 'BCPL.' This programming language was created using UNIX operating system at AT&T and Bell Laboratories.

==========



In 1972, a great computer scientist Dennis Ritchie created a new programming language called 'C' at the Bell Laboratories. It was created from 'ALGOL', 'BCPL' and 'B' programming languages. 'C' programming language contains all the features of these languages and many more additional concepts that make it unique from other languages.

==========

'C' is a powerful programming language which is strongly associated with the UNIX operating system. But today 'C' runs under a variety of operating systems and hardware platforms. To assure that 'C' language will remain standard, American National Standards Institute (ANSI) defined a commercial standard for 'C' language in 1989. Later, it was approved by the International Standards Organization (ISO) in 1990. 'C' programming language is also called as 'ANSI C'.

==========

Salient Features of C

1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed
8. Pointers
9. Recursion
10. Extensible

Simple

C is a simple language in the sense that it provides a **structured approach** (to break the problem into parts), **the rich set of library functions**, **data types**, etc.

Machine Independent or Portable

Unlike assembly language, c programs **can be executed on different machines** with some machine specific changes. Therefore, C is a machine independent language.

Mid-level programming language

Although, C is **intended to do low-level programming**. It is used to develop system applications such as kernel, driver, etc. It **also supports the features of a high-level language**. That is why it is known as mid-level language.

Structured programming language

C is a structured programming language in the sense that **we can break the program into parts using functions**. So, it is easy to understand and modify. Functions also provide code reusability.

Rich Library
C provides a lot of inbuilt functions that make the development fast.

Memory Management

It supports the feature of **dynamic memory allocation**. In C language, we can free the allocated memory at any time by calling the **free()** function.

Speed

The compilation and execution time of C language is fast since there are fast and efficient inbuilt functions and hence the lesser overhead.

Pointer

C provides the feature of pointers. We can directly interact with the address of the memory by using the pointers. We **can use pointers for memory, structures, functions, array**, etc.

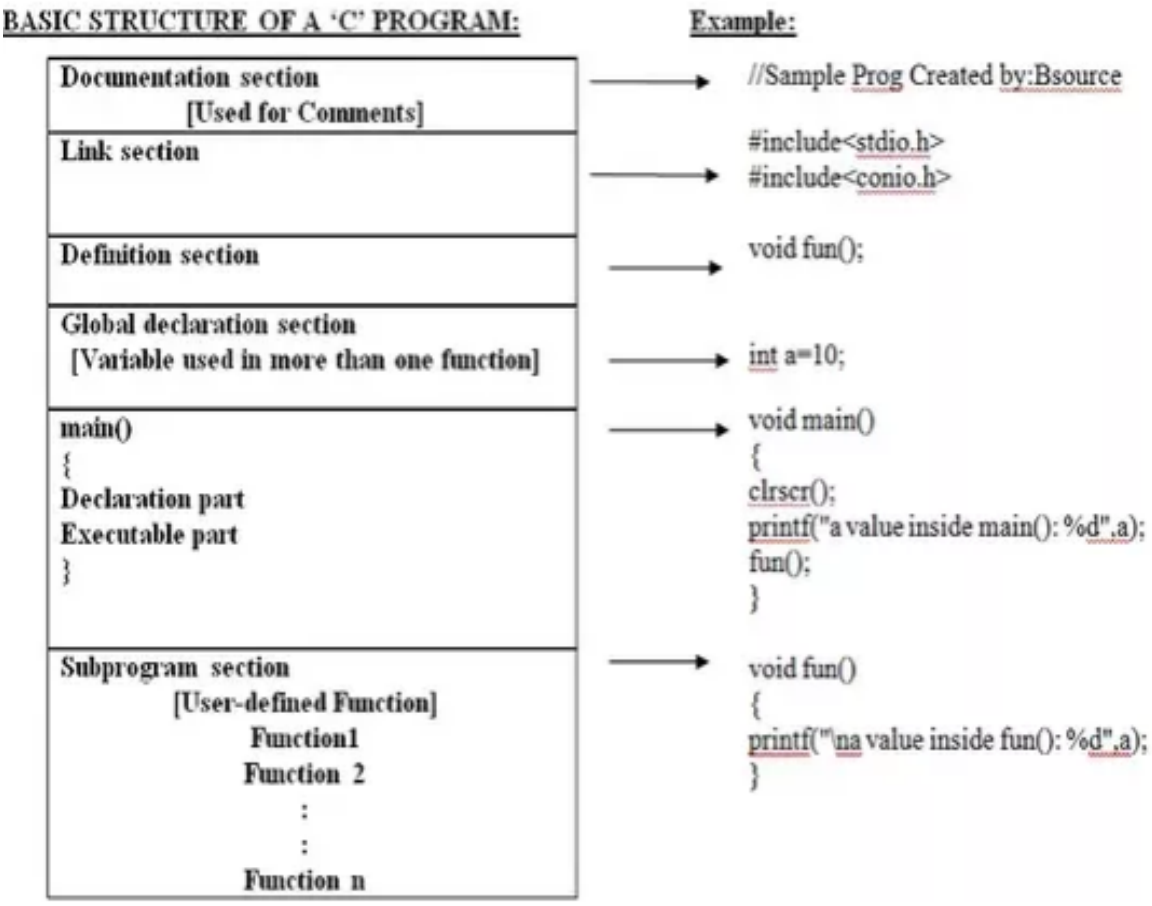
Recursion

Recursion is a function when it calls itself, therefore it is **a technique in which a problem is solved in-terms of itself**". Here we call the same problem by reducing the set to its subset.

Extensible : c language is extensible that is designed so that users or developers can expand or add to its capabilities of a program, programming language, or a protocol,

==========

Structure of a C Program



==========

- **Documentation section :** The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.
- **Link section :** The link section provides instructions to the compiler to link functions from the system library.
- **Definition section :** The definition section defines all symbolic constants.
- **Global declaration section :** There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.
- **main () function section :** Every C program must have one main function section. This section contains two parts; declaration part and executable part**Declaration part :** The declaration part declares all the variables used in the executable part.**Executable part :** There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.
- **Subprogram section :** The subprogram section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order. Note:All section, except the main () function section may be absent when they are not required.

==========

A Simple C Program

Write a program to print area of a circle.

Problem is to find the area of a circle for a given radius 10cm.

Formula

The formula to compute the area of a circle is πr^2 where π is PI = 3.1416 (approx.) and r is the radius of the circle.

==========

Lets write the C code to compute the area of the circle.

```
/**
 * file: circle.c
 * author: XYZ
 * date: 2020-07-10
 * description: program to find the area of a circle
 *              using the radius r
 */

#include <stdio.h>

#define PI 3.1416

float area(float r);

int main(void)
{
  float r = 10;
  printf("Area: %.2f", area(r));
  return 0;
}

float area(float r) {
   return PI * r * r;
}
```

The above code will give the following output.

```
Area: 314.16
```

==========

Documentation

This section contains a multi line comment describing the code.

```
/*
 * file: circle.c
 * author: XYZ
 * date: 2020-07-10
 * description: program to find the area of a circle
 *              using the radius r
 */
```

In C, we can create single line comment using two forward slash // and we can create multi line comment using /* */.

Comments are ignored by the compiler and is used to write notes and document code.

==========

Link

This section includes header file.

#include <stdio.h>

We are including the stdio.h input/output header file from the C library.

Definition

This section contains constant.

#define PI 3.1416

In the above code we have created a constant PI and assigned 3.1416 to it.

The #define is a preprocessor compiler directive which is used to create constants. We generally use uppercase letters to create constants.

The #define is not a statement and must not end with a ; semicolon.

==========

Global declaration

This section contains function declaration.

float area(float r);

We have declared an area function which takes a floating number (i.e., number with decimal parts) as argument and returns floating number.

==========

main( ) function

This section contains the main() function.

```
int main(void)
{
  float r = 10;
  printf("Area: %.2f", area(r));
```

```
  return 0;
}
```

This is the main() function of the code. Inside this function we have created a floating variable r and assigned 10 to it.

Then we have called the printf() function. The first argument contains "Area: %.2f" which means we will print floating number having only 2 decimal place. In the second argument we are calling the area() function and passing the value of r to it.

==========

Subprograms

This section contains a subprogram, an area() function that is called from the main() function.

```
float area(float r) {
   return PI * r * r;
}
```

This is the definition of the area() function. It receives the value of radius in variable r and then returns the area of the circle using the following formula PI * r * r.

==========

Compiling a C Program

C is a high-level language and it needs a compiler to convert it into an executable code so that the program can be run on our machine.

How to compile and run the c program

There are 2 ways to compile and run the c program, by menu and by shortcut.

Window version

By menu

Now **click on the compile menu then compile sub menu** to compile the c program.
Then **click on the run menu then run sub menu** to run the c program.

By shortcut

Or, press ctrl+f9 keys compile and run the program directly.
You can view the user screen any time by pressing the **alt+f5** keys.
Now **press Esc** to return to the turbo c++ console.

==========

Linux version

to compile: gcc f1.c (file name) –o f1(outfile)

to execute: ./f1(outfile)

The .c file name is the source file to be compiled
The option -o is used to specify the output file name. If we do not use this option, then an output file with name a.out is generated.

After compilation executable is generated and we run the generated executable using below command.
to execute ./filename (outfile)

==========

What goes inside the compilation process?

Compiler converts a C program into an executable. There are four phases for a C program to become an executable:

1. Pre-processing
2. Compilation
3. Assembly
4. Linking

By executing below command, We get the all intermediate files in the current directory along with the executable.

now what is intermediate files

Let us one by one see what these intermediate files contain.

==========

Pre-processing

This is the first phase through which source code is passed. This phase include:

- Removal of Comments
- Expansion of Macros
- Expansion of the included files.
- Conditional compilation

The preprocessed output is stored in the **filename.i**.

In the above output, source file is filled with lots and lots of info, but at the end our code is preserved.
Analysis:

- printf contains now a + b rather than add(a, b) that's because macros have expanded.
- Comments are stripped off.
- **#include<stdio.h>** is missing instead we see lots of code. So header files has been expanded and included in our source file.

Compiling

The next step is to compile filename.i and produce an; intermediate compiled output file **filename.s**. This file is in assembly level instructions. Let's see through this file using **$vi filename.s**

Assembly

In this phase the filename.s is taken as input and turned into **filename.o** by assembler. This file contain machine level instructions. At this phase, only existing code is converted into machine language, the function calls like printf() are not resolved.

Linking

This is the final phase in which all the linking of function calls with their definitions are done. Linker knows where all these functions are implemented. it adds some extra code to our program which is required when the program starts and ends.

The program has reached the final stage for execution.

Link and Run the C Program.

Linking - **Linking** refers to the creation of a single executable file from multiple object files. The link phase is implemented by the linker. The linker is a process that accepts as input object files and libraries to produce the final executable program. The object code is combined with required supporting code to make an executable program. This step typically involves adding in any libraries that are required.