Programming Fundamentals Using C

Nimisha Manan,
Assistant Professor,
Department of Computer Science

INTRODUCTION TO C

- C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories
- C was initially developed for writing system software
- Today, C has become a popular language and various software programs are written using this language.
- Many other commonly used programming languages such as C++ and Java are also based on C

Characteristics of C

- A high level programming language
- Small size. C has only 32 keywords. This makes it relatively easy to learn
- Makes extensive use of function calls
- C is well suited for structured programming. In this programming approach,
- Unlike PASCAL it supports loose typing (as a character can be treated as an integer and vice versa)
- Stable language.
- Quick language
- Facilitates low level (bitwise) programming
- Supports pointers to refer computer memory, array, structures and functions.
- C is a core language
- C is a portable language.
- C is an extensible language

USES OF C

- **C language is primarily used for system programming. The portability, efficiency, the ability to access specific hardware addresses and low runtime demand on system resources makes it a good choice for implementing operating systems and embedded system applications.**
- **C has been so widely accepted by professionals that compilers, libraries, and interpreters of *other* programming languages are often implemented in C.**
- **For portability and convenience reasons, C is sometimes used as an intermediate language by implementations of other languages. Example of compilers which use C this way are BitC, Gambit, the Glasgow Haskell Compiler, Squeak, and Vala.**
- **C is widely used to implement end-user applications**

STRUCTURE OF A C PROGRAM

```
main()
{
            Statement 1;
            Statement 2;
            ……
            ……
            Statement N;
}
Function1()
{
            Statement 1;
            Statement 2;
            ……
            ……
            Statement N;
}
Function2()
{
            Statement 1;
            Statement 2;
            ……
            ……
            Statement N;
}
……
……
FunctionN()
{
            Statement 1;
            Statement 2;
            ……
            ……
            Statement N;
}
```

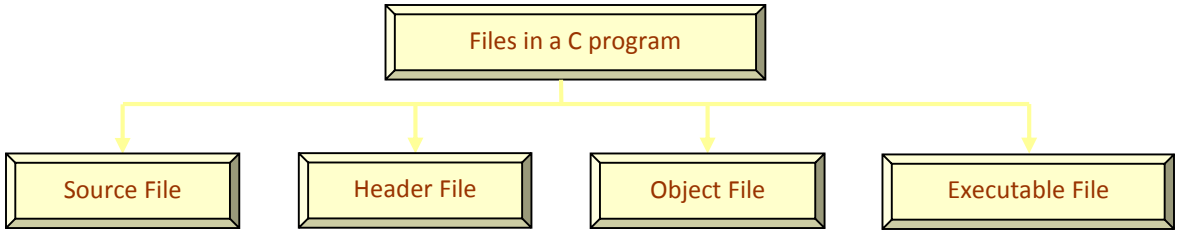
A C program contains one or more functions
The statements in a C program are written in a logical sequence to perform a specific task.
Execution of a C program begins at the main() function
You can choose any name for the functions. Every program must contain one function that has its name as main().

YOUR FIRST C PROGRAM

```
// This is my first program in C

#include<stdio.h>
int main()
{
      printf("\n Welcome to the world of C ");
      return 0;
}
```

FILES USED IN A C PROGRAM



Source code file

- The source code file contains the source code of the program. The file extension of any C source code file is ".c". This file contains C source code that defines the *main* function and maybe other functions. The main() is the starting point of execution when you successfully compile and run the program. A C program in general may include even other source code files (with the file extension .c).

Header Files

- When working with large projects, it is often desirable to make sub-routines and store them in a different file known as header file. The advantage of header files can be realized when

a) The programmer wants to use the same subroutines in different programs.

b) The programmer wants to change, or add, subroutines, and have those changes be reflected in all other programs.

- Conventionally, header files names ends with a ".h" extension and its name can use only letters, digits, dashes, and underscores.
- While some standard header files are available in C, but the programmer may also create his own user defined header files

FILES USED IN A C PROGRAM contd.

Object Files

- Object files are generated by the compiler as a result of processing the source code file. Object files contain compact binary code of the function definitions. Linker uses this object file to produce an executable file (.exe file) by combining the of object files together. Object files have a ".o" extension, although some operating systems including Windows and MS-DOS have a ".obj" extension for the object file.

Binary Executable File

- The binary executable file is generated by the linker. The linker links the various object files to produce a binary file that can be directly executed. On Windows operating system, the executable files have ".exe" extension.

COMPILING AND EXECUTING C PROGRAMS

The compilation process in the figure 2.5 is done in two steps. In the first step, the preprocessor program reads the source file as text, and produces another text file as output. Source code lines which begin with the hash symbol are actually not written in C but in the preprocessor language. The output of the preprocessor is a text file which does not contain any preprocessor statements. This file is ready to be processed by the compiler. The linker combines the object file with library routines (supplied with the compiler) to produce the final executable file.

USING COMMENTS

- It is a good programming practice to place some comments in the code to help the reader understand the code clearly.
- Comments are just a way of explaining what a program does. It is merely an internal program documentation.
- The compiler ignores the comments when forming the object file. This means that the comments are non-executable statements.

C supports two types of commenting.

- // is used to comment a single statement. This is known as a line comment. A line comment can be placed anywhere on the line and it does not require to be specifically ended as the end of the line automatically ends the line.
- /* is used to comment multiple statements. A /* is ended with */ and all statements that lie within these characters are commented.

KEYWORDS

- C has a set of 32 reserved words often known as keywords. All keywords are basically a sequence of characters that have a fixed meaning. By convention all keywords must be written in lowercase (small) letters.
- **Example:** for, while, do-while, auto break, case, char, continue, do, double, else, enum, extern, float, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile

IDENTIFIERS

- Identifiers are names given to program elements such as variables, arrays and functions.

Rules for forming identifier name

- it cannot include any special characters or punctuation marks (like #, $, ^, ?, ., etc) except the underscore"_".
- There cannot be two successive underscores
- Keywords cannot be used as identifiers
- The names are case sensitive. So, example, “FIRST” is different from “first” and “First”.
- It must begin with an alphabet or an underscore.
- It can be of any reasonable length. Though it should not contain more than 31 characters.

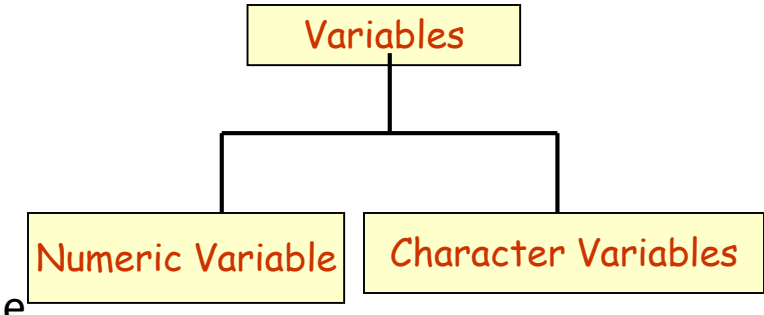
Example: roll_number, marks, name, emp_number, basic_pay, HRA, DA, dept_code

DATA TYPE	SIZE IN BYTES	RANGE
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed short int	2	-32768 to 32767
signed int	2	-32768 to 32767
short int	2	-32768 to 32767
unsigned short int	2	0 to 65535
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
signed long int	4	-2147483648 to 2147483647
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

DATA TYPES IN C

VARIABLES IN C

- A variable is defined as a meaningful name given to the data storage location in computer memory.
- When using a variable, we actually refer to address of the memory where the data is stored. C language supports two basic kinds of variables.
- Numeric variables can be used to store either integer values or floating point values.
- While an integer value is a whole numbers without a fraction part or decimal point, a floating point number, can have a decimal point in them.
- Numeric values may also be associated with modifiers like short, long, signed and unsigned.
- By default, C automatically a numeric variable signed..
- Character variables can include any letter from the alphabet or from the ASCII chart and numbers 0 – 9 that are put between single quotes.



To declare a variable specify data type of the variable followed by its name.
Variable names should always be meaningful and must reflect the purpose of their usage in the program.
Variable declaration always ends with a semicolon. Example,

```
int emp_num;
float salary;
char grade;
double balance_amount;
unsigned short int acc_no;
```

CONSTANTS

- Constants are identifiers whose value does not change.
- Constants are used to define fixed values like PI or the charge on an electron so that their value does not get changed in the program even by mistake.
- To declare a constant, precede the normal variable declaration with const keyword and assign it a value. For example,

  const float pi = 3.14;
- Another way to designate a constant is to use the pre-processor command *define*.

  #define PI 3.14159

  When the preprocessor reformats the program to be compiled by the compiler, it replaces each defined name with its corresponding value wherever it is found in the source program. Hence, it just works like the Find and Replace command available in a text editor.

Rules that needs to be applied to a #define statement which defines a constant.

- Constant names are usually written in capital letters to visually distinguish them from other variable names which are normally written in lower case characters. Note that this is just a convention and not a rule.
- No blank spaces are permitted in between the # symbol and define keyword
- Blank space must be used between #define and constant name and between constant name and constant value
- #define is a pre-processor compiler directive and not a statement. Therefore, it does not end with a semi-colon.