C Fundamentals

Structure of a program

Every C program consists of one or more modules called functions. One of these functions is called main. The program begins by executing main function and access other functions, if any. Functions are written after or before main function separately. A function has **(1)** heading consists of name with list of arguments (optional) enclosed in parenthesis, **(2)** argument declaration (if any) and **(3)** compound statement enclosed in two braces { } such that each statement ends with a semicolon. Comments, which are not executable statement, of necessary can be placed in between /* and */.

Example

```
/* program to find the area pf a circle */
#include<stdio.h>
#include<conio.h>
main( )
{
float r, a;
printf("radius");
scanf("%f", &r);
a=3.145*r*r;
printf("area of circle=%f", area);
}
```

The character set

C used the upper cases A,B,…….,Z, the lower cases a ,b,…..,z and certain special characters like **+ - * / = % & # ! ? ^ " ' ~ \ < > ( ) = [ ] { } ; : . , _** blank space @ $. also certain combinations of these characters like \b, \n, \t, etc…

Identities and key words

Identities are names given to various program elements like variables, arrays and functions. The name should begin with a letter and other charact4rs can be letters and digits and also can contain underscore character (_) **Exapmle**: area, average, x12 , name_of_place etc……

Key words are reserved words in C language. They have predicted meanings and are used for the intended purpose. Standard keywords are **auto, break, case, char, const, continue, default, do, double, else enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while. (**Note that these words should not be used as identities**.)**

Data type

The variables and arrays are classified based on two aspects- first is the data type it stores and the second is the type of storage. The basic data types in C language are int, char, float and double. They are respectively concerned with integer quantity, single character, numbers, with decimal point or exponent number and double precision floating point numbers (ie; of larger magnitude). These basic data types can be augmented by using quantities like short, long, signed and unsigned. (ie; long int, short int, long double etc).

CONSTANTS

There are 4 basic types of constants . they are int4ger constants, floating-point constants, character constants and string constants.

(a) **integer constants**: It is an integer valued numbers, written in three different number system, decimal (base 10) , octal(base8), and hexadecimal(base 16).

A decimal integer constant consists of 0,1,…..,9..

Example : 75 6,0,32, etc…..
5,784, 39,98, 2-5, 09 etc are **not** integer constants.

An octal integer constant consists of digits 0,1,…,7. with 1st digit 0 to indicate that it
is an octal integer.

Example : 0, 01, 0756, 032, etc…..
32, 083, 07.6 etc..... are **not** valid octal integers.

A hexadecimal integer constant consists of 0,1, …,9,A, B, C, D, E, F. It begins with 0x.

Example: 0x7AA2, 0xAB, etc……
0x8.3, 0AF2, 0xG etc are **not** valid hexadecimal constants.

Usually negative integer constant begin with (-) sign. An unsigned integer constant is identified by appending U to the end of the constant like 673U, 098U, 0xACLFU etc. Note that 1234560789LU is an unsigned integer constant.

(b) floating point constants : It is a decimal number (ie: base 10) with a decimal point or an exponent or both. Ex; 32.65, 0.654, 0.2E-3, 2.65E10 etc.
These numbers have greater range than integer constants.

(c) character constants : It is a single character enclosed in single quotes like 'a'. '3', '?', 'A' etc. each character has an ASCII to identify. For example 'A' has the ASCII code 65, '3' has the code 51 and so on.

(d) escape sequences: An escape sequence is used to express non printing character like a new line, tab etc. it begin with the backslash (\ ) followed by letter like a, n, b, t, v, r, etc. the commonly used escape sequence are

\a : for alert	\n : new line	\0 : null
\b : backspace	\f : form feed	\? : question mark
\f : horizontal tab	\r : carriage return	\' : single quote
\v : vertical tab	\" : quotation mark	

(e) string constants : it consists of any number of consecutive characters enclosed in double quotes .Ex : " C program" , "mathematics" etc……

Variables and arrays

A variable is an identifier that is used to represent some specified type of information. Only a single data can be stored in a variable. The data stored in the variable is accessed by its name. before using a variable in a program, the data type it has to store is to be declared.

Example : int a, b, c,
a=3; b=4;
c=a+b

]

Note : A statement to declare the data types of the identifier is called declaration statement. An array is an identifier which is used to store a collection of data of the same type with the same name. the data stored is an array are distinguished by the subscript. The maximum size of the array represented by the identifier must be mentioned.

Example : int mark[100] .

With this declaration n, mark is an array of size 100, they are identified by nark[0], mark[1],..... ,mark[99].

Note : along with the declaration of variable, it can be initialized too. For example

```
int x=10;
```

with this the integer variable x is assigned the value 10, before it is used. Also note that C is a case sensitive langauge. i.e. the variables d and D are different.

DECLARATIONS

This is for specifying data type. All the variables, functions etc must be declared before they are used. A *declaration* tells the compiler the name and type of a variable you'll be using in your program. In its simplest form, a declaration consists of the type, the name of the variable, and a terminating semicolon:

Example : int a,b,c;

```
Float mark, x[100], average;
char name[30];
char c;
int i;
float f;
```

You may wonder *why* variables must be declared before use. There are two reasons:

1. It makes things somewhat easier on the compiler; it knows right away what kind of storage to allocate and what code to emit to store and manipulate each variable; it doesn't have to try to intuit the programmer's intentions.
2. It forces a bit of useful discipline on the programmer: you cannot introduce variables willy-nilly; you must think about them enough to pick appropriate types for them. (The compiler's error messages to you, telling you that you apparently forgot to declare a variable, are as often helpful as they are a nuisance: they're helpful when they tell you that you misspelled a variable, or forgot to think about exactly how you were going to use it.)

EXPRESSION

This consists of a single entity like a constant, a variable, an array or a function name. it also consists of some combinations of such entities interconnected by operators.

Example : a, a+b, x=y, c=a+b, x<=y etc……..

STATEMENTS

Statements are the ``steps'' of a program. Most statements compute and assign values or call functions, but we will eventually meet several other kinds of statements as well. By default, statements are executed in sequence, one after another

A statement causes the compiler to carry out some action. There are 3 different types of statements – expression statements compound statements and control statements. Every statement ends with a semicolon.

Example:
```
(1)  c=a + b;
(2)      {
         a=3;
         b=4;
         c=a+b;
         }

(3)      if (a<b)
         {
         printf("\n a is less than b");
         }
```
Statement may be single or compound (a set of statements).

Most of the statements in a C program are *expression statements*. An expression statement is simply an expression followed by a semicolon. The lines

```
        i = 0;
        i = i + 1;
and     printf("Hello, world!\n");
are all expression statements
```

SYMBOLIC CONSTANTS

A symbolic constant is a name that substitutes for a sequence of characters, which represent a numeric, character or string constant. A symbolic constant is defined in the beginning of a program by using #define, without: at the end.

Example :
```
#define      pi    3.1459
#define     INTEREST     P*N*R/100
```

With this definition it is a program the values of p, n ,r are assigned the value of INTEREST is computed.

Note : symbolic constants are not necessary in a C program.