CHAPTER 2: CONSTANTS, VARIABLES AND DATA TYPES

Dr.A.PRAVEEN, M.Sc.,M.Phil.,

Introduction:

A *programming language* is designed to help process certain kinds of data consisting of numbers, characters and strings and to provide useful output known as information.

Definitions:

Program:

The task of processing of data is accomplished by executing a sequence of precise instructions called a *program*.

Grammar (or) Syntax rules:

The instructions are formed using certain symbols and words according to some rigid rules known as *syntax rules* or *grammar*.

CHARACTER SET:

The character set in C can be classified into the following types:

1. Letters
2. Digits
3. Special characters
4. White spaces.

Letters:

Uppercase A.....Z
Lowercase a.....z

Digits:

All decimal digits 0.....9

Special characters:

, Comma	! Exclamation mark	% Percent sign
.Period	| Vertical bar	& Ampersand
; semicolon	/ Slash	^ Caret
: Colon	\ Backslash	* asterisk
? Question mark	~ Tilde	- minus sign
' Apostrophe	_ Under score	+ Plus sign
$ Dollar sign	< Opening angle bracket(or less than sign)	

> Closing angle bracket (or greater than sign)] right bracket

(left parenthesis { left brace) Right parenthesis

} Right brace [left bracket # Number sign

WHITE SPACES:

i)Blank space ii)New Line iii)Horizontal tab iv)Form feed v)Carriage return.

TRIGRAPH CHARACTERS:

Trigraph sequences provide a way to enter certain characters that are not available on some keyboards.

A trigraph sequence consists of three characters. (two question marks followed by another character)

Trigraph sequence	Transalation
??=	# Number sign
??(	[Left bracket
??)	] Right bracket
??<	{ Left brace
??>	} Right brace
??!	| Vertical bar
??/	\ Back slash
??-	~ Tilde

C TOKENS:

The smallest individual units of a C program are known as C tokens. C has six types of tokens as shown below.

Note:

A C program is written using these tokens and the syntax of the language.

Every C word is classified as either a keyword or an identifier.

Keywords:

- All keywords have fixed meaning and these meanings cannot be changed.
- Keywords serve as basic building blocks foe program.
- All keywords must be written in lowercase only

auto	double	int	struct	break
else	long	switch	case	enum
register	typedef	char	extern	return
union	const	float	short	unsigned
continue	for	signed	void	default
goto	sizeof	volatile	do	if
static	while			

Identifiers:

Identifiers refer to the name of variables, functions and arrays. These are user – defined names and consist of a sequence of letters and digits, with a letter as a first character. Both upper case and lower case letters are permitted, although lowercase letters are commonly used. Underscore character is also permitted in identifiers.

Note:

1. A keyword cannot be used as an identifier.
2. An identifier must not contain white spaces.

Constants:

Constants in C refer to fixed values that do not change during the execution of a program.

Integer Constants:

An integer constant refers to a sequence of digits. There are three types of integers namely decimal integer, octal integer, and hexadecimal integer.

Decimal integers consist of a set of digits, o through 9, preceded by an optional – or + sign.

Example: 123 -321 0 +78

Note: Spaces, Commas and non – digit characters are not permitted between digits.

An octal integer consists of any character of digit from the set 0 through 7 with a leading 0.

Example: 037 0 0553

Hexadecimal integer consists of a sequence of digits preceded by0x or 0X. They may also include alphabets A through F (or) a through f . The letter A through F represent the number 10 to 15.

Example: 0X2 0x9F 0Xbcd 0X

Real Constants:

Numbers containing fractional parts are called as real constants or floating point constants.

Example: 0.0083 -0.75 435.86 +247.0

The numbers above are represented in decimal notation, having a whole number followed by a decimal point and the fractional part.

- It is possible to omit digits before (or) after the decimal point.

Example : 215. .95

- A real number may also be expressed in exponential or Scientific notation.

Example: 215.65 may be written as **2.1565e2 (e2 means multiply by 10^2).**

The general form is **mantissa e exponent**

The mantissa is either a real number expressed in decimal notation or an integer.

Example: 0.65e4 65e2

The letter e separating the mantissa and the exponent part can be written in either lowercase or uppercase.

Single character constants:

A single character constant contains a single character enclosed within a pair of single quote marks.

Example: '5' 'X' ';' ' '

Character constants have integer values known as ASCII values .

For example, the statement **printf("%d",'a')** would print the number **97.**

Similarly the statement **printf("%c",'97')** would print the character a

String constants:

A string constant is a sequence of characters enclosed in double quotes. The characters may be letters, numbers, special characters and blank spaces.

Example: **"hello"** **"1997"** **"5+3"** **"?....!"**

Backslash character constants:

C supports special backslash character constants that are used in output functions.

Constant	Meaning
'\a'	**audible alert(bell)**
'\b'	**back space**
'\n'	**new line**
'\r'	**carriage return**
'\t'	**horizontal tab**
'\v'	**vertical tab**
'\''	**single quote**
'\"'	**double quote**

Variables:

A variable is a data name that may be used to store a data value. A variable may take different values at different times during execution.

Variable names may consist of letters, digits and the underscore character subject to the following conditions

1. They must begin with a letter.
2. The length of a variable name should not be more than eight characters.
3. Uppercase and lowercase letters are significant.

4. It should not be a character.
5. White space is not allowed.

Examples: **John** **ph_value** **sum1**

Invalid examples: **123** **(area)%**

Declaration of variables:

After designing suitable variable names, they must be declared to the compiler. Declaration does two things:

1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.

Note:

Declaration of variables must be done before they are used in the program.

The syntax for declaring a variable is as follows:

data-type v1,v2,....vn;

v1,v2,.....vn are the names of variables. Variables are separated by commas. A declaration statement must end with a semicolon.

Example:

int amount;

double ratio;

float sum;

char a,d;

Assigning values to variables:

The variables that are used in expressions (on the right side q equal (=) sign of a computational statement) must be assigned values before they are encountered in the program. Values can be assigned to variables using the assignment operator = as follows:

The general format is

Variable – name = Constant;

Examples: initial – value = 0;

Balance = 75.84;

Yes = 'x' ;

'c' permits multiple assignments in one line.

Example: initial – value = 0; final – value = 100

An assignment statement implies that the value at the variable on the left of the 'equal sign' is set equal to the value of the quantity (or the expression on the right.

Example: year = year + 1;

means that the 'new value' of year is equal to the 'old value' of year plus:

During assignment operation, C converts the type at value on the R.H.S to the type on the left. This may involve truncation when real value is converted to an integer.

It is possible to assign a value to a variable at the time the variable is declared. This take the following fom:

data-type variable-name = constant;

Examples:

int a = 100;

float balance = 75.84;

The process of giving initial value to variables is called initialization.

- C permits the initialization of more than one variables in one statement using multiple assignment operators.

Example:

p=q=r=s=0;

Reading Data from keyboard:

Values can be given to variables by inputting the data through keyboard using the scanf function. The general format of scanf is

Scanf("control string",&variable1,&variable2,....);

The control string contains the format of data being received. The ampersand symbol & before each variable name is an operator that specifies the address of the variable name.

When the scanf function is encountered by the computer, the execution stops and waits for the value of the variable to be typed in. After the value is typed in and the Return key is pressed, the computer then proceeds to te next statement.

Defining symbolic constants:

Certain unique constants are often used in a program. These constants may appear repeatedly in a number of places in the program. We face two problems in the subsequent use of such programs.

1. Problem in modification of the program.
2. Problem in understanding the program.

Assignment of such constants to a symbolic name frees us from these problems. Constant vales are assigned to these names at the beginning of the program. A constant is defined as follows:

#define symbolic-name value of constant

- Symbolic names are also called as constant identifiers.
- Symbolic names are constants (not variables). Hence they do not appear in declarations.

The following rules apply to a #define statement, which defines a symbolic statement.

1. Symbolic names have the same form as variable names. Symbolic names are written in uppercase letters to distinguish them from the normal variable names.
2. No blank space between the pound sing # and the word define is permitted
3. # must be the first character in the line.
4. A blank space is required between # define and symbolic name and between the symbolic name and the constant.
5. # define statements must not end with a semicolon.

6. After definition, the symbolic name should not be assigned any other value within the program by using an assignment statement.
7. Symbolic names are not declared for data types. Its data type depends on the type of constant.
8. # define statements may appear anywhere in the program but before it is references in the program. Usually it is placed in the beginning of the program.

MANAGING INPUT AND OUTPUT OPERATION

getchar function

Reading a single character can be done by using the function **getchar().** The general form of the getchar function is as follows:

Variable-name=getchar();

Variable-name is a valid C name of *char* data type.

When this statement is encountered, the computer waits until a key is pressed and then assigns this character as a value to getchar function.

The getchar function is used on the right-hand side of an assignment statement. The character value of getchar is assigned to the variable name on the left.

Character Test Functions:

Function	Test
isalnum(c)	Is c an alphanumeric character ?
isalpha(c)	Is c an alphabetic character?
isdigit(c)	Is c a digit?
islower(c)	Is c a lowercase letter?
isupper(c)	Is c an uppercase letter?
isspace(c)	Is c a white space character?
ispunct(c)	Is c a punctuation mark?

Putchar function

General form: putchar(variable-name);

Where variable-name is a type char variable containing a character. This statement displays the character contained in the variable-name of the terminal.

Scanf function:

The general form of the scanf function is

scanf("control sring',arg1,arg2,.....,argn);

The *control string* specifies the field format in which the data to be entered and the arguments arg1,arg2,.....argn specify the address of the locations where the data is stored.

Control string and arguments are separated by commas.

Printf function:

The general form of the printf function is

Printf("control string",arg1,arg2,.....argn);

The control string indicates how many arguments follow and what their types are. The arguments arg1,arg2,.....argn are the variables whose values are formatted and printed according to the specifications of the control string.

The arguments should match in number, order and type with the format specifications.