UNIT 3 VARIABLES AND CONSTANTS

Structure

3.0 INTRODUCTION

As every natural language has a basic character set, computer languages also have a character set, rules to define words. Words are used to form statements. These in turn are used to write the programs.

Computer programs usually work with different types of data and need a way to store the values being used. These values can be numbers or characters. C language has two ways of storing number values—**variables and constants**—with many options for each. Constants and variables are the fundamental elements of each program. Simply speaking, a program is nothing else than defining them and manipulating them. A variable is a data storage location that has a value that can change during program execution. In contrast, a constant has a fixed value that can't change.

This unit is concerned with the basic elements used to construct simple C program statements. These elements include the C character set, identifiers and keywords, data types, constants, variables and arrays, declaration and naming conventions of variables.

3.1 OBJECTIVES

After going through this unit, you will be able to:

- define identifiers, data types and keywords in C;
- know name the identifiers as per the conventions;
- describe memory requirements for different types of variables; and
- define constants, symbolic constants and their use in programs.

3.2 CHARACTER SET

When you write a program, you express C source files as text lines containing characters from the character set. When a program executes in the target environment,

it uses characters from the character set. These character sets are related, but need not have the same encoding or all the same members.

Every character set contains a distinct code value for each character in the **basic C character set**. A character set can also contain additional characters with other code values. The C language character set has alphabets, numbers, and special characters as shown below:

1. Alphabets including both lowercase and uppercase alphabets - A-Z and a-z.

2. Numbers 0-9

3. Special characters include:

;	:	{	,	‘	“	|
}	>	<	/	\	~	_
[	]	!	$	?	*	+
=	(	)	-	%	#	^
@	&	.				

3.3 IDENTIFIERS AND KEYWORDS

Identifiers are the names given to various program elements such as constants, variables, function names and arrays etc. Every element in the program has its own distinct name but one cannot select any name unless it conforms to valid name in C language. Let us study first the rules to define names or identifiers.

3.3.1 Rules for Forming Identifiers

Identifiers are defined according to the following rules:

1. It consists of letters and digits.
2. First character must be an alphabet or underscore.
3. Both upper and lower cases are allowed. Same text of different case is not equivalent, for example: **TEXT** is not same as **text**.
4. Except the special character underscore (_), no other special symbols can be used.

For example, some valid identifiers are shown below:

```
 X
X123
_XI
temp
tax_rate
```

For example, some invalid identifiers are shown below:

123	First character to be alphabet.
“X.”	Not allowed.
order-no	Hyphen allowed.
error flag	Blankspace allowed.

3.3.2 Keywords

Keywords are reserved words which have standard, predefined meaning in C. They cannot be used as program-defined identifiers.

The lists of C keywords are as follows:

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

Note: Generally all keywords are in lower case although uppercase of same names can be used as identifiers.

3.4 DATA TYPES AND STORAGE

To store data inside the computer we need to first identify the type of data elements we need in our program. There are several different types of data, which may be represented differently within the computer memory. The data type specifies two things:

1. Permissible range of values that it can store.
2. Memory requirement to store a data type.

C Language provides four basic data types viz. int, char, float and double. Using these, we can store data in simple ways as single elements or we can group them together and use different ways (to be discussed later) to store them as per requirement. The four basic data types are described in the following table 3.1:

Table 3.1: Basic Data Types

DATA TYPE	TYPE OF DATA	MEMORY	RANGE
int	Integer	2 Bytes	– 32,768 to 32,767
char	character	1 Byte	– 128 to 128
float	Floating point number	4 bytes	3.4e – 38 to 3.4e +38
double	Floating point number with higher precision	8 bytes	1.7e – 308 to 1.7e + 308

Memory requirements or size of data associated with a data type indicates the range of numbers that can be stored in the data item of that type.

3.5 DATA TYPE QUALIFIERS

Short, long, signed, unsigned are called the data type qualifiers and can be used with any data type. A ***short int*** requires less space than ***int*** and ***long int*** may require more space than ***int***. If ***int*** and ***short int*** takes 2 bytes, then ***long int*** takes 4 bytes.

Unsigned bits use all bits for magnitude; therefore, this type of number can be larger. For example ***signed int*** ranges from –32768 to +32767 and ***unsigned int*** ranges from 0 to 65,535. Similarly, ***char*** data type of data is used to store a character. It requires 1 byte. ***Signed char*** values range from –128 to 127 and ***unsigned char*** value range from 0 to 255. These can be summarized as follows:

Data type	Size (bytes)	Range
Short int or int	2	–32768 to 32,767
Long int	4	–2147483648 to 2147483647

Signed int	2	–32768 to 32767
Unsigned int	2	0 to 65535
Signed char	1	–128 to 127
Unsigned char	1	0 to 255

3.6 VARIABLES

Variable is an identifier whose value changes from time to time during execution. It is a named data storage location in your computer's memory. By using a variable's name in your program, you are, in effect, referring to the data stored there. A variable represents a single data item i.e. a numeric quantity or a character constant or a string constant. Note that a value must be assigned to the variables at some point of time in the program which is termed as assignment statement. The variable can then be accessed later in the program. If the variable is accessed before it is assigned a value, it may give garbage value. The data type of a variable doesn't change whereas the value assigned to can change. All variables have three essential attributes:

- the name
- the value
- the memory, where the value is stored.

For example, in the following C program ***a, b, c, d*** are the variables but variable ***e*** is not declared and is used before declaration. After compiling the source code and look what gives?

```
main( )
{
   int    a, b, c;
   char  d;
         a = 3;
         b = 5;
         c = a + b;
         d = 'a';
         e=d;
         …….
         …….
         }
```

After compiling the code, this will generate the message that variable ***e*** not defined.

3.7 DECLARING VARIABLES

Before any data can be stored in the memory, we must assign a name to these locations of memory. For this we make declarations. Declaration associates a group of identifiers with a specific data type. All of them need to be declared before they appear in program statements, else accessing the variables results in junk values or a diagnostic error. The syntax for declaring variables is as follows:

data- type variable-name(s);

For example,

```
int  a;
short int   a, b;
```

```
int  c, d;
long  c, f;
float  r1, r2;
```

3.8 INITIALISING VARIABLES

When variables are declared initial, values can be assigned to them in two ways:

a) Within a Type declaration

The value is assigned at the declaration time.

For example,

```
int       a = 10;
float     b = 0.4 e –5;
char      c = ‘a’;
```

b) Using Assignment statement

The values are assigned just after the declarations are made.

For example,

```
a = 10;
b = 0.4 e –5;
c = ‘a’;
```

Check Your Progress 1

1) Identify keywords and valid identifiers among the following:

hello	function	day-of-the-week
student_1	max_value	“what”
1_student	int	union

……

……

……

2) Declare type variables for roll no, total_marks and percentage.

……

……

……

3) How many bytes are assigned to store for the following?

a) Unsigned character b) Unsigned integer c) Double

……

……

……

3.9 CONSTANTS

A constant is an identifier whose value can not be changed throughout the execution of a program whereas the variable value keeps on changing. In C there are four basic types of **constants**. They are:

1. Integer constants
2. Floating point constants
3. Character constants
4. String constants

Integer and Floating Point constants are numeric constants and represent numbers.

Rules to form Integer and Floating Point Constants

- No comma or blankspace is allowed in a constant.
- It can be preceded by – (minus) sign if desired.
- The value should lie within a minimum and maximum permissible range decided by the word size of the computer.

3.9.1 Integer Constants

Further, these constant can be classified according to the base of the numbers as:

1. **Decimal integer constants**

   These consist of digits 0 through 9 and first *digit should not be 0.*

   For example,

   1 443 32767
   are valid decimal integer constants.

2. **Invalid Decimal integer Constants**

   12 ,45 , not allowed
   36.0 Illegal char.
   1 010 Blankspace not allowed
   10 – 10 Illegal char –
   0900 The first digit should not be a zero

3. **Octal integer constants**

   These consist of digits 0 through 7. The first digit must be zero in order to identify the constant as an octal number.

   Valid Octal INTEGER constants are:

   0 01 0743 0777

   Invalid Octal integer constants are:

   743 does not begin with 0
   0438 illegal character 8
   0777.77 illegal char .

4. **Hexadecimal integer constants**

These constants begin *with 0x or OX* and are followed by combination of digits taken from hexadecimal digits 0 to 9, a to f or A to F.

Valid Hexadecimal integer constants are:

OX0 OX1 OXF77 Oxabcd.

Invalid Hexadecimal integer constants are:

OBEF	x is not included
Ox.4bff	illegal char (.)
OXGBC	illegal char G

Maximum values these constants can have are as follows:

Integer constants	**Maximum value**
Decimal integer	32767
Octal integer	77777
Hexadecimal integer	7FFF

Unsigned interger constants: Exceed the ordinary integer by magnitude of 2, they are not negative. A character U or u is prefixed to number to make it unsigned.

Long Integer constants: These are used to exceed the magnitude of ordinary integers and are appended by L.

For example,

50000U	decimal unsigned.
1234567889L	decimal long.
0123456L	otal long.
0777777U	otal unsigned.

3.9.2 Floating Point Constants

What is a base 10 number containing decimal point or an exponent.

Examples of valid floating point numbers are:

0.	1.
000.2	5.61123456
50000.1	0.000741
1.6667E+3	0.006e-3

Examples of Invalid Floating Point numbers are:

1	decimal or exponent required.
1,00.0	comma not allowed.
2E+10.2	exponent is written after integer quantity.
3E 10	no blank space.

A Floating Point number taking the value of 5×10^4 can be represented as:

5000.	5e4
5e+4	5E4
5.0e+4	.5e5

The magnitude of floating point numbers range from 3.4E –38 to a maximum of 3.4E+38, through 0.0. They are taken as double precision numbers. Floating Point constants occupy 2 words = 8 bytes.

3.9.3 Character Constants

This constant is a single character enclosed in apostrophes ‘ ’ .

For example, some of the character constants are shown below:

‘A’, ‘x’, ‘3’, ‘$’

‘\0’ is a null character having value zero.

Character constants have integer values associated depending on the character set adopted for the computer. ASCII character set is in use which uses 7-bit code with 2^7 = 128 different characters. The digits 0-9 are having ASCII value of 48-56 and ‘A’ have ASCII value from 65 and ‘a’ having value 97 are sequentially ordered. For example,

‘A’ has 65, blank has 32

ESCAPE SEQUENCE

There are some non-printable characters that can be printed by preceding them with ‘\’ backslash character. Within character constants and string literals, you can write a variety of **escape sequences**. Each escape sequence determines the code value for a single character. You can use escape sequences to represent character codes:

- you cannot otherwise write (such as \n)
- that can be difficult to read properly (such as \t)
- that might change value in different target character sets (such as \a)
- that must not change in value among different target environments (such as \0)

The following is the list of the escape sequences:

Character	Escape Sequence
"	\"
'	\'
?	\?
\	\\
BEL	\a
BS	\b
FF	\f
NL	\n
CR	\r
HT	\t
VT	\v

3.9.4 String Constants

It consists of sequence of characters enclosed within double quotes. For example,

“ red ” “ Blue Sea ” “ 41213*(I+3) ”.

3.10 SYMBOLIC CONSTANTS

Symbolic Constant is a name that substitutes for a sequence of characters or a numeric constant, a character constant or a string constant. When program is compiled each occurrence of a symbolic constant is replaced by its corresponding character sequence. The syntax is as follows:

#define name text

where ***name*** implies symbolic name in caps.
text implies value or the text.

For example,

```
#define  printf   print
#define  MAX   100
#define  TRUE  1
#define  FALSE 0
#define   SIZE   10
```

The # character is used for preprocessor commands. A ***preprocessor*** is a system program, which comes into action prior to Compiler, and it replaces the replacement text by the actual text. This will allow correct use of the statement printf.

Advantages of using Symbolic Constants are:

- They can be used to assign names to values
- Replacement of value has to be done at one place and wherever the name appears in the text it gets the value by execution of the preprocessor. This saves time. if the Symbolic Constant appears 20 times in the program; it needs to be changed at one place only.

Check Your Progress 2

1) Write a preprocessor directive statement to define a constant PI having the value 3.14.

……

……

2) Classify the examples into Interger, Character and String constants.

'A'	0147	0xEFH
077.7	"A"	26.4
"EFH"	'\r'	abc

……

……

3) Name different categories of Constants.

……

……

3.11 SUMMARY

To summarize we have learnt certain basics, which are required to learn a computer language and form a basis for all languages. Character set includes alphabets, numeric characters, special characters and some graphical characters. These are used to form words in C language or names or identifiers. Variable are the identifiers, which change their values during execution of the program. Keywords are names with specific meaning and cannot be used otherwise.

We had discussed four basic data types - int, char, float and double. Some qualifiers are used as prefixes to data types like signed, unsigned, short, and long.

The constants are the fixed values and may be either Integer or Floating point or Character or String type. Symbolic Constants are used to define names used for constant values. They help in using the name rather bothering with remembering and writing the values.

3.12 SOLUTIONS / ANSWERS

Check Your Progress 1

1. **Keywords:** int, union
   Valid Identifiers: hello, student_1, max_value

2. int rollno;
   float total_marks, percentage;

3. a) 1 byte b) 2 bytes c) 8 bytes

Check Your Progress 2

1. # define PI 3.14

2. **Integer constant**: 0147
   Character constants: 'A', '\r'
   String constants: "A", "EFH"

3.13 FURTHER READINGS

1. The C Programming Language, *Kernighan & Ritchie*, PHI Publication.
2. Computer Science A structured programming approach using C*, Behrouza A. Forouzan, Richard F. Gilberg*, Second Edition, Brooks/Cole, Thomson Learning, 2001.
3. Programming with C*, Gottfried*, Second Edition, Schaum Outlines, Tata Mc Graw Hill, 2003.