Leaning Objective:

In this Module you will be learning the following:

- Introduction to C
- How to use input and output statements
- How to use different types of variables
- Different type of operators
- Expressions

Introduction:

The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis Ritchie. C programming language features were derived from an earlier language called "B" (Basic Combined Programming Language – BCPL). C language was invented for implementing UNIX operating system. In 1978, Dennis Ritchie and Brian Kernighan published the first edition "The C Programming Language" and commonly known as K&R C. In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or "ANSI C", was completed late 1988.

Material:

C programming basics to write a c program:

Below are few commands and syntax used in C programming to write a simple C program. Let's see all the sections of a simple C program line by line.

C Basic commands	Explanation
#include <stdio.h>	This is a preprocessor command that includes standard input output header file(stdio.h) from the C library before compiling a C program
int main()	This is the main function from where execution of any C program begins.
{	This indicates the beginning of the main function.
/*_some_comments_*/	whatever is given inside the command "/* */" in any C program, won't be considered for compilation and execution.
printf("Hello_World! ");	printf command prints the output onto the screen.
getch();	This command waits for any character input from keyboard.
return 0;	This command terminates C program (main function) and returns 0.
}	This indicates the end of the main function.

A Simple C Program:

```
#include <stdio.h>
int main(){
  /* Our first simple C basic program */
  printf("Hello World! ");
  getch();
  return 0;
}
```

OUTPUT: Hello World!

What is variable?

Variables in C have the same meaning as variables in algebra. A variable in C is a storage unit, which sets a space in memory to hold a value and can take different values at different times during program execution.

Rules to construct a valid variable name

1. A variable name may consists of letters, digits and the underscore (_) characters.

2. A variable name must begin with a letter. Some system allows to starts the variable name with an underscore as the first character.

3. ANSI standard recognizes a length of 31 characters for a variable name. However, the length should not be normally more than any combination of eight alphabets, digits, and underscores.

4. Uppercase and lowercase are significant. That is the variable Totamt is not the same as totamt and TOTAMT.

5. The variable name may not be a C reserved word (keyword).

Some valid variable names

Total	Amount	ctr	name1
n1	M_age	AMOUNT	

Some invalid variable names

13th	(name)	111	%nm

Naming Conventions

Generally, C programmers maintain the following conventions for naming variables.

- Start a variable name with lowercase letters.
- Try to use meaningful identifiers
- Separate "words" within identifiers with mixed upper and lowercase (for example empCode) or underscores (for example emp_code).
- For symbolic constants use all uppercase letters (for example #define LENGTH 100, #define MRP 45).

Keywords and Identifiers

Every C word is classified as either a keyword or an identifier. Each keyword has a specific meaning and these meanings cannot be changed. Keywords serve as basic building blocks for program statements. There are only 32 keywords in C. The list of all keywords in ANSI C is listed in the following table. All keywords must be written in lowercase.

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

Constants

Constants in C refer to a specific quantity that doesn't change during the execution of a program.

Types of Constants

- Integer Constants
- Real Constants
- Single Character Constants
- String Constants

Rules to construct Integer Constant

- An integer constant refers to a sequence of digits. The three types of integers are decimal, octal and hexadecimal
- No embedded blanks, commas, and non-numeric character are permitted within an integer constant.
- An integer constant must contain one digit
- Decimal integers consist set of digits, 0 to 9 without any decimal point and can be preceded by an optional +ve or -ve sign.
- An octal integer constant contains any combination of digits between 0 and 7 with a leading 0.
- A hexadecimal constant contains any combination of digits between 0 and 9 and can also hold alphabets between A and F or a and f with prefix 0x or 0X. Alphabets A or a represents number 10 and F or f represents 15.
- The largest integer value for the 16-bit machine is 32767 and 2147483647 for a 32-bit machine.

Example of various valid numeric constants

Constant	Type	Constant	Type
241	Decimal Integer	047	Octal Integer
-973	Decimal Integer	053	Octal Integer
0	Decimal Integer	0X59	Hexadecimal Integer
+4177	Decimal Integer	0x47F	Hexadecimal Integer

Example of some invalid numeric constants

Invalid Constant
05 241
7,412
$120

Rules to construct Real Constant

- A real constant is a number that may have a fractional part.
- It could be either +ve or -ve.
- No embedded blanks, commas, and non-numeric character are permitted within a real constant.
- A real number may also be expressed in exponential notation. An exponent is an integer number with an optional plus or minus sign. Exponential is useful for representing the number which is very large or very small in magnitude.

Example of various valid real constants

0.0045	-.71
+45.203	0.45e3
-0.547	0.78e-4
337.	2.79E4
.478.	-4.69E-4

Data Types

Each program needs a certain kind of data for displaying a meaningful result. This certain kind of data are known as a data type.

ANSI C supports four classes of data types:

- Primary data types
- User-defined data types
- Derived data types
- Empty data set

All C compilers support four fundamental data types

Type	Range of values	Description
char (Characters)	-128 to 127	a single byte(8 bits) and can store one character type data
int Integers (whole numbers)	-32768 to 32767	an integer type is used to represent whole numbers within a specified range of values.
Float Floating point (real numbers)	3.4e-38 to 3.4e+38	single-precision floating point
Double (Double)	1.7e-308 to 1.7e+308	double-precision floating point

Character Types:

A single character can be defined as a character type data. Characters are usually stored in 8 bits of internal storage. The qualifier signed or unsigned may be explicitly applied to char. While unsigned chars have values between 0 and 255, signed chars have values from -128 to 127.

Integer Types:

C has three classes of integer storage, namely short int, int, and long int, in both signed and unsigned forms.

The keywords signed and unsigned are the two sign qualifiers which specify whether a variable can store positive or negative or both numbers.

The keyword signed uses one bit for a sign and 15 bits for the magnitude of the number in a 16-bit machine.

The keyword unsigned uses to store all the bits for the magnitude of the number and always positive.

Floating Point Types:

Floating point numbers are stored in 32 bits with 6 digits of precision. Floating point numbers are defined in C by the keyword float. When the accuracy provided by a float number is not sufficient, the type double can be used to define the number.

Double Point Types:

A double data type number uses 64 bits giving a precision of 14 digits. These are known as double precision numbers. Remember that double type represents the same data type that float represents but with a greater precision. To extend the precision further, we may use long double which uses 80 bits.

The following table shows the size and range of the type-specifiers on most common implementations:

Type	Size(bits)	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 255
int or signed int	16	-32768 to 32767
unsigned int	16	0 to 65535
short int or signed short int	8	-128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
float	32	3.4E-38 TO 3.4E+38
double	64	1.7E-308 TO 1.7E+308
long double	80	3.4E-4932 TO 1.1E+4932

Initialization of Variables

Variables are given initial values, or initialized, when declared. See the following examples:

```
char abc = 'X';

int marks =77;

float amount = 45.23;
```

Write initialized variables on a separate line and a comment beside the variable name is a good idea. See the following examples:

```
int qty; /* quantity of an item */

float value = 12.10; /* purchase value of an item */

int marks; /* marks of a subject */
```

Operators

An Operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables.

C operators can be classified into a number of categories. They are:

- Arithmetic operators.
- Relational Operators.
- Logical Operators.
- Assignment Operators.
- Increment and Decrement Operators.
- Conditional Operators.
- Bitwise Operators.
- Special Operators.

Arithmetic Operators:

Arithmetic operators include +, -, *, /, %, which performs all mathematical manipulations. These operators can operate on any built-in data type allowed in C. Table shows the function of Arithmetic Operators.

Operators	Meaning
+	Addition or unary plus
-	Subtraction of unary minus
*	Multiplication
/	Division
%	Modulo Division

Relational Operators:

Relational operators are used to compare two operands, and depending on their relation, certain decisions are taken. For example, it can be used to compare the age of two persons, price of two items and so on. There are 6 types of relational operators. They are:

Operators	Meaning
<	Is less than
>	Is greater than

<=	Is less than or equal to
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to

Logical Operators:

C supports three Logical Operators. They are:

Operators	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Assignment Operators:

Assignment operators are used to assign the result of an expression to a variable. Usually, '=' operator is used. There is an additional 'shorthand' assignment operators of the form

V op = exp;

Here, V= variable, exp = expression and op = a binary arithmetic operator.

The Operator op= is known as the shorthand assignment operator.

Shorthand Assignment Operators

Statement with simple assignment operator	Statement with shorthand operator
a = a + 1	a += 1
a = a - 1	a -= 1
a = a * (n+1)	a *= n + 1
a = a / (n+1)	a /= n + 1
a = a % b	a % = b

Increment and Decrement Operators:

C supports two unique operators that are not found in other languages. They are: ++ and -- (increment and decrement operators respectively).

The operator ++ adds 1 to the operand, while – subtracts 1. Both are unary operators and take the following form:

++m; or m++;

--m; or m--;

++m is equivalent to m = m + 1;

--m is equivalent to m = m – 1;

Conditional Operator:

A ternary operator pair "? :" is available in C to construct conditional expressions of the form:

Exp1? Exp2: Exp3

Exp1, Exp2 and Exp3 are expressions. The operator ?: works as follows:

Exp1 is evaluated first. If it is nonzero (true), then exp2 expression is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression. Here only one of the expressions is evaluated.

Bitwise Operators:

Bitwise operators are special operators that are used for manipulation of data at the bit level. These operators are used for testing the bits, or shifting them to right or left. Bitwise operators may not be applied to float or double.

Operators	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift Left
>>	Shift Right
~	One's Complement

Special Operators:

C supports some special operators such as :

- comma operator
- sizeof operator
- pointer operators(& and *)
- member selection operators.

Formatted Output and the printf function

One of the common task in every program is the printing of output. We use the output to request input from a user and later display the status/result, computations etc. In C programming there are several functions for printing formatted output. Here we discuss the printf() function, which writes output to the computer monitor. To use the printf() function we must include the stdio library in the source code. To do this just place the following code at the beginning of your program.

#include <stdio.h>

To print a simple message in computer screen you might call printf() function as follows :

```
#include <stdio.h>
main(){
printf ("You are learning printf() function");
}
```

Output:

You are learning printf() function

In the above examples, the cursor will remain at the end of the printed output. If you repeat the above code in the following way the second message would appear immediately after the first one.

```
#include <stdio.h>
main(){
printf("You are learning printf() function");
printf("You are learning printf() function");
}
```

Output:

You are learning printf() function You are learning printf() function

If we want to print the second output in a new line we must need a different way, which has discussed in the following section.

Escape sequences in C

An escape sequence is a series of characters that represents a special character. It begins with a backslash character (\), which indicates that the character(s) that follow the backslash character should be treated in a special way. C uses escape sequences within a format string to print certain special characters. For example \n moves the output position to the beginning of the next line. The following is a list of escape sequences.

Escape sequence	Action
\n	prints a new line
\b	backs up one character
\t	moves the output position to the next tab stop
\\	prints a backslash
\"	prints a double quote
\'	prints a single quote

You will get an idea of using the above escape sequences from the following example.

```
#include<stdio.h>
    main(){
        printf("Create a new line\n");
        printf("Print a double quotes (\") within a string\n");
        printf("Print a single quotes (\') within a string\n");
        printf("Print a Backslash\\ within a string\n");
        printf("Using Backspace\b within a string\n");
        printf("Using\tTab within a string\n");
    }
```

Output:

Create a new line

Print a double quotes (") within a string

Print a single quotes (') within a string

Print a Backslash\ within a string

Using Backspace within a string

Using Tab within a string

Using printf() to print values

In the above section, we have discussed how to print a new line, single quote, double quote etc. In this section, we will discuss how to print the value of a variable. There are various format available to print different type values. Let start with these codes:

```
int x = 150;

printf("Number of students in Class V is %d",x);
```

In the printf() function the format parameter '%d' is replaced with the value of the parameter x. Therefore the output looks like : Number of students in Class V is 150. In addition to %d, there are quite a number of format specifiers, each having a different meaning. Here are the basic ones (several others exist):

Parameter	Meaning
%d	Print an integer number printed in decimal (preceded by a minus sign if the number is negative).
%f	Print a floating point number (in the form dddd.dddddd).
%E	Print a floating point number (in scientific notation: d.dddEddd).
%g	Print a floating point number (either as f or E, depending on value and precision).
%x	Print an integer number in hexadecimal with lower case letters.
%X	Print an integer number printed in hexadecimal with upper case letters.
%c	Print a character.
%s	Print a string.

The following code prints two numbers one is positive another is negative. #include<stdio.h>

```
main(){
        int x,y;
        x = 5;
        y= -5;
        printf("The value of x is %d and value of y is %d",x,y);
}
```

Output:

The value of x is 5 and value of y is -5

You can do some calculation within printf. See the following example:

```
#include<stdio.h>
main(){
        int x;
        x = 5;
        printf ("%d + %d = %d\n",x,y,x+y);
        printf ("%d - %d = %d\n",x,y,x-y);
        printf ("%d x %d = %d\n",x,y,x*y);
        printf ("%d / %d = %d\n",x,y,x/y);
}
```

Output:

5 + 5 = 10

5 - 5 = 0

5 x 5 = 25

5 / 5 = 1

The following code uses the other parameters of printf mentioned in the parameter list:

```
#include<stdio.h>
main(){
        char a;
        float x,y;
        a = 'G';
        x = 5.23;
        y = 76000000.00;
        printf("%c %f %g %E %s\n", a, x, y, y,"String");
        printf("Hexadecimal(lower case) of 12 is %x\n",12);
        printf("Hexadecimal(upper case) of 12 is %X\n",12);
}
```

Output:

G 5.230000 7.6e+007 7.600000E+007 String

Hexadecimal (lower case) of 12 is c

Hexadecimal (upper case) of 12 is C

Problem sets

1. Write a C program to print your name, date of birth, and mobile number.
2. Print each of the following patterns. Use one printf() statement for each line of outputs. End each line by printing a newline (\n).

```
* * * * *    * * * * *   * * * * *
 * * * * *   *      *    *    *
* * * * *    *      *     *  *
 * * * * *   *      *      * *
* * * * *    * * * * *      *
```

(a) (b) (c)

3. Print the above patterns using one printf() statement.
4. Write a C program to enter two numbers and perform all arithmetic operations.
5. Write a program to prompt user for 5 integers and print their product. Use an int variable product to store the product and operator * for multiplication.
6. Write a program to print the area and perimeter of a rectangle. Your program shall prompt the user for the length and width of the rectangle, in doubles.
7. Write a C program to enter marks of five subjects and calculate total, average and percentage.
8. Write a C program that converts kilometers per hour to miles per hour.
9. Write a program in C that takes minutes as input, and display the total number of hours and minutes.
10. Write a C program to display following variables.
    a+ c, x + c, dx + x, ((int) dx) + ax, a + x, s + b, ax + b, s + c, ax + c, ax + ux

Variable declaration:

```
int a = 125, b = 12345;        long ax = 1234567890;
short s = 4043;                float x = 2.13459;
double dx = 1.1415927;         char c = 'W';
unsigned long ux = 2541567890;
```