Chapter-1

(*) Structure of C:-

C structure contains following five main things.

i) **Documentation** :- Documentation is the topic of program or a thing that represent that who designed the program and for what purpose.

ii) **Link** :- It is the preprocessor command which tells C compiler to include pre-processed files such as:-

#include<stdio.h>, #include<conio.h>, #include<string.h>.

iii) **Main function / function**:- These are building blocks of any C program. C program will have one or more function and there is one compulsory included function which is called main function and written as "main()" while programming.

iv) **Statements and Expressions**:- Statements are expressions, function calls, or control flow statements which combine to form C program. Expressions combine variables and constants to create new values. Statements and Expressions are written after function inside curly bracesses.

N) User declearation/comments:- User creates some space for input data as varible name which is called user declearation. User declearation is done by writing as: input data_type variable;

Comments are used to give additional information inside a C program. For single line comment we use double slash and then we write comment but for multi-line comment we use ~~back~~slash asterik then we write comments and finally we use asterik with ~~back~~slash.

Note:- C is case sensative, semicolon at end of each statement, multiple statement can be on same line, while spaces are ignored and statements can continue over multiple lines.

Chapter - 2

Elements of C

RK ISC SCO Navneet

CS CS S.D 21, 21

Elements of C

i) Character set:-

C characters denote any alphabet, digit, or special symbol used to represent information. C uses upper case letters "A-Z" and lower case letters "a-z", the digits "0-9" and certain special characters ~~like consonants~~ like comma, semi-colon, colon, backslash etc.

ii) C tokens:-

C tokens are each and every smallest individual units in a C program. C tokens are basic building blocks in C ~~program~~ language which are constructed together to write a C program. C tokens are of 6 types as follows:-

ⓐ Reserved keywords → There are certain words that are reserved for doing specific tasks, these words are known as keywords. for e.g. int, float, char etc.

ⓑ Identifiers → Identifiers are userdefined words and are used to give names to arrays, functions, structures etc. for e.g. "main", "number", "name"

ⓒ Constants → Constant is a value that can not be changed during execution of program. These fix values are also called literals. for e.g. 2, 5, 10, 20 etc.

(d) Strings → Sequence of characters is called string. for e.g. "total", "mul", "Ram", "school" etc.

(e) Special characters/symbols → Some special characters and symbols are used in C program like (), {}, &, !, = etc.

(f) Operators → Operators like +, /, −, *, % are used to perform differnt operations.

iii) Scape sequence :- Some characters such as new line, tab and back space can not be printed like other normal characters. C uses backslash (\) and some other characters to print those characters. These character combinations are known as scape sequence characters. for e.g. \b is used for backspace, \n is used for new line.

iv) Delimetres :- A delimeter is a unique character or ~~set~~ series of characters that indicate the begining or end of a specific statement, strings or function body. colon (:), semicolon (;), #etc are some delimeters.

d. Double → Doubles are the data types who stores a longer range of number.

The size and range of data types used in C programming are as follows:-

	Data type	Size	Range
a)	Int	2 byte	-32768 to 32767
b)	Char	1 byte	-128 to 127
c)	Float	4 byte	3.4e-38 to 3.4e+38
d)	Double	8 byte	1.7e-308 to 1.7e+308

Unit-3

Input and Output:-

Reading Input Data:-

scanf("control string", address 1, address 2...);

Writing Output data:-

printf("control string", variable 1, variable 2, ...);

Unit-4

Operators and Expressions:-

⑧ Different types of operators available in C

(Ari, Ass, ++, --, Re, Lo, Co)

1) Arithmetic operator:

Arithmetic operators are used for numeric calculations. Arithmetic operators are of following two types:-

a) Unary operator → Unary operator require only one operand for e.g. +x, -y etc.

b) Binary arithmetic operator → Binary arithmetic operator require two operend. for e.g. a+b, a*b etc. There are five binary arithmetic operands which are as in the below table.

Operators	Purpose
+	Addition.
-	Subtraction
*	Multiplication
/	Division.
%	Remainder (Modulus)

2) Assignment operator:

The operator which is used to store value in a variable is called assignment operator. The assignment operator '=' is used in assignment expression and assignment statement. The operend on the left hand side should be a variable, while the operent on the right hand side can be any variable, value or expression.

3) **Increament / Decreament operator:**

C has two useful operators (++) and (--). These are unary operators. The increament operator (++) increments the value of variable by 1 and decreament operator (--) operator decrements the value of variable by 1.

for e.g. ++x is equivalent to $x = x+1$

--x is equivalent to $x = x-1$.

These operators are used only with the variables. These are of two types as follows:-

(a) Prefix increament / decreament :- Here, first the value of variable is increamented and decreamented then the new value is used in the operation.

(b) Post fix increament / decreament :- Here, first the value of variable is used in the operation and then increment / decreament is performed.

4) **Relational operator :-**

These operators are used to compare values of two expressions. An expression that contains relational operator is called relational expression. If the relation is true then the value of relational expression is 1 and if the relation is false, the value of expression is 0. The relational operators are as follows:-

Operator	Meaning
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to

Truth table:

Let us suppose two variables, a=9 and b=5

Expression	Relation	Value of expression.
a<b	False	0
a>b	True	1
a<=b	False	0
a>=b	True	1
a==b	False	0
a!=b	True	1
b!=0	True	1
a>8	True	1

5) Logical operator:-

An expression that combines two or more expressions is termed as a logical expression. For combing these expressions we use some operators which are called logical operators. These operators return 0 for false and 1 for true. C has three

logical operators which are as follows:-

Operator	Meaning
&&	AND
||	OR
!	NOT

Here logical NOT is a unary operator while the other two are binary operators.

Types with truth table:

(a) AND (&&) operator → This operator gives the net result true only if both the conditions are true, otherwise the result is false.

Truth table:

Condition 1	Condition 2	Result
False	False	False
False	True	False
True	False	False
True	True	True

(b) OR (||) operator → This operator gives the net result false only if both the conditions are false, otherwise the result is true.

Truth table:

Condition 1	Condition 2	Result
False	False	False
False	True	True
True	False	True
True	True	True

© Not (!) operator →

This is unary operator and neglects the value of the condition. If the value of condition is false then it gives the result true. If the value of the condition is true then it gives the result false.

Condition	Result
False	True
True	False.

6) Conditional operator :→

Conditional operator is a ternary operator which requires three expressions as operant, which is written as:

Test expression? expression 1 : expression 2.

For eg. obtained marks >=35? Pass : Fail

First the test expression is evaluated, if the test expression is true, the expression 1 is evaluated and it becomes the value of overall conditional expression.

But if the test expression is false, then expression 2 is evaluated and it becomes the value of conditional expression.

Unit-5

Control Statements:

The statements that alter normal flow of program execution are known as control statements. It help us to execute only a part of program as we need. C language supports four types of control statements which are as follows:-

1) **if... else statement:**

If... else statement is further divided into three sub statements as follows:-

(a) **if...else :-** if... else statement is bidirectional conditional control statement. This statement is used to test a condition and take of the two possible actions. If the condition is true then a single statement or a block of statements is executed otherwise another single statement or a block of statements is executed. Any non-zero value is regarded as true while zero is regarded as false.

Syntax:-

for single line

```
if (condition)
statement;
else
statement;
```

For multi line

```
if (condition) {
  statement 1;
  statement 2;
  :
else
  statement 1;
  statement 2;
  :
}
```

(b) Nested if... else statement:-

If we decleare another if... else statement in the block of if or else block, this is known as nested if... else statement.

Syntax:-

```
if (condition)
  {
    if (condition)
    statement;
    else
    statement;
    }
else
    {
      if (condition)
      statement;
      else
      statement;
      }
```

(c) else... if ladder:-

This is the type of if... else statement in which there is an else if statement in every else part except the last else part.

Syntax:-

```
if (condition 1)
statement 1;
else if (condition 2)
statement 2;
else if (condition 3)
statement 3;
  . . .
  . . .
  . . .
else
  statement N;
```

2) goto statement:-

This is an unconditional control statement that transfer the flow of control to another part of the program. The goto statement can be used as;

```
goto label;
  . . .
  . . .
label:
statements;
```

3) Switch statement:- Sometimes there is a need to make choice among number of alternatives, for making this choice, we use the switch statement. This is multi-directional condition control statement.

Syntax:-

```
switch (expression)
  {
    Case constant1:
       statement1;
       break;
    Case constant2:
       statement2;
       break;
    Case constant3:
       statement3;
       break;
    . . .
    . . .
    . . .
    default:
    statement;
      }
```

4) Loop :- Loop or looping is the repeatation of executing the same section of code more than once until the given condition is true. C provides 3 looping statements as below:

@ For loop :- It makes programming more convinent to count iterations of a loop and works well where the number of interations of the loop is known before the loop entered.

Syntax:-

```
for(initilization; condition; loop update)
  {
    statement (s);
  }
```

(b) While loop:- A while loop statement in C programming language repeatedly executes a target statement as long as the given condition is true.

Syntax:-

```
initilization;
while (condition)
  {
    statement (s);
    loop update;
  }
```

(c) do... while loop:- In while loop test expression is checked at first but in do... while loop, code is executed at first then the condition is checked. So, the code are executed at least once in do... while loops.

Syntax:-

```
initilization;
do{ statement (s);
    loop update;
  }
while (condition);
```

(*) Differences between while and do...while statements.

S.N.	while statement	do... while statement
i)	It is entry control loop.	It is an exit control loop.
ii)	Test condition is evaluated before the loop is executed.	Test condition is evaluated after the loop is executed.
iii)	It use keyword while.	It use keyword do and while.
iv)	loop is not terminated with semicolon.	Loop is terminated with semicolon.

v)	Syntax:
	initialization;
	while (condition)
	{
	statement (s);
	loop update;
	}
		Syntax:
		initialization;
		do
		{
		statement (s);
		loop update;
		}
		while (condition);

vi)	Example:
	int i;
	i=0;
	while (i<=10)
	{
	printf("%d", i);
	i = i+1;
	}
		Example:-
		int i;
		i=0;
		do{
		printf("%d", i);
		i=i+1;
		}
		while (i<=10);

⑧ Differences between break and continue statement.

SN	Break statement		Continue statement.
i)	Break statement is used to terminate the control from the switch case as well as in loop.	i)	It is used to by pass the execution of the further statements.
ii)	When it is encountered it terminates the execution of entire loop or switch case.	ii)	When it is en-countered it by-passes single pass of the loop.
iii)	It use keyword break;	iii)	It use keyword continue;
iv)	Syntax:- break;	iv)	Syntax:- continue;

v)	Example
	void main(){
	int i;
	for(i=0; i<=10; i++)
	{ if(i==5)
	break;
	printf("%d", i); }
	getch();
	}
		v)	Example
			void main(){
			int i;
			for(i=0; i<=10; i++)
			{ if(i==5)
			continue;
			printf("%d", i); }
			getch();
			}

vi)	Output: 0 1 2 3 4	vi)	Output: 1 2 3 4 6 7 8 9 10

Unit-7

Functions:

user-defined function

Navneet

Date
Page

(*) Functions, its types with syntaxes and examples

Types of user defined functions:-

User defined functions can be classified into four categories on the basis of the arguments and return values as follows:-

1) Function with no arguments and no return type:-

When a function has no arguments it does not receive any data from the calling function. Similarly when it does not return a value, the calling function does not receive any data from the called function.

Syntax:

```
Function declaration: void function();
function call: function();
function defination: void function()
                     { local variable decleration;
                     statements(s);
                     }
```

e.g. Program to add two numbers:

```
#include<stdio.h>
#include<conio.h>
 void sum();
 Void main()
 { clrscr();
   sum();
   getch(); }
```

Note:- i> Argument list ले भने called function ले calling function बाट value receive गर्न सक्छ।
ii> Return type ले भने calling function ले called function बाट value receive गर्न सक्छ।

```
void sum()
  {
    int x,y,z;
    printf("Enter two numbers \n");
    scanf("%d %d", &x, &y);
      z=x+y;
    printf("Sum is %d", z);
      }
```

2> Function with no argument and a return value:-

There could be occassions where we may need to design functions that may not take any arguments but returns a value to the calling function. A example for this is getchar(). It has no parameters but it returns an integer type data that represents a character.

Syntax:-

```
function declearation: int function();
function call: function();
function defination: int function()
                     {
                       local variable declearation;
                       statement(s);
                     return x;
                      }
```

e.g. Program to add two numbers:

```
#include<stdio.h>
#include<conio.h>
int sum();
void main()
{
  int c;
  clrscr();
  c=sum();
  printf("sum=%d",c);
  getch();
}
int sum()
{
  int x,y,s;
  printf("Enter two numbers\n");
  scanf("%d %d",&x,&y);
  s=x+y;
  return(s);
}
```

3) Function with arguments and no return value

When a function has arguments it receive any data from the calling function but it returns no values.

Syntax:-

function declearation: void function (int);
function call: function (x);
function defination: void function (int x)

```
{
    statement (s);
}
```

e.g. Program to add two numbers

```
#include<stdio.h>
#include<conio.h>
void sum (int, int);
void main ()
{
    int a, b;
    clrscr ();
    printf ("Enter two numbers \n");
    scanf ("%d %d", &a, &b);
    sum (a, b);
    getch ();
}
void sum (int x, int y)
{
    int s;
    s = x+y;
    printf ("sum = %d", s);
}
```

4) Function with arguments and return value:-

This function can receive any data from the calling function to called function and can also return value to the calling function.

Syntax:-

function declaration: void function (int);
function call: function (x);
function defination: void function (int x)
{
statement (s);
}

e.g. Program to add two numbers:-

```
#include<stdio.h>
#include<conio.h>
int sum (int, int);
void main ( )
{
int a,b,c;
clrscr ();
printf ("Enter two numbers \n");
scanf ("%d %d", &a, &b);
c = sum (a,b);
printf ("sum = %d", c);
getch ();
}
int sum (int x, int y)
{ int s;
s = x+y;
return (s);
}
```