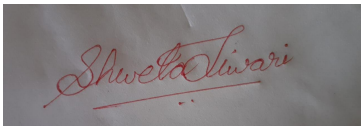
Covered Topics Under UNIT-1 , UNIT-2 of "PPS-PROGRAMMING FOR PROBLEM SOLVING (BCS101 / BCS201)"

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com



Class Notes

Published Date: November, 2022

PPS: UNIT-1

- **Programming Basics**

PPS: UNIT-2

- **Arithmetic Expression and Precedence**
- **Conditional Branching**

FALL SEMESTER, YEAR (I/II sem, 1st yr)

FALL SESSION (2022-23)
(PPS)
MS. SHWETA TIWARI
Published Date: November, 2022

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

—

By SHWETA TIWARI
Under On: UNIT-1 , UNIT-2

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Units Classified into Chapters

PPS: UNIT-1:

Chapter-3 START PROGRAMING IN 'C' LANGUAGE

PPS: UNIT-2:

Chapter-4 OPERATORS

Chapter-5 CONDITION CONTROL STATEMENTS

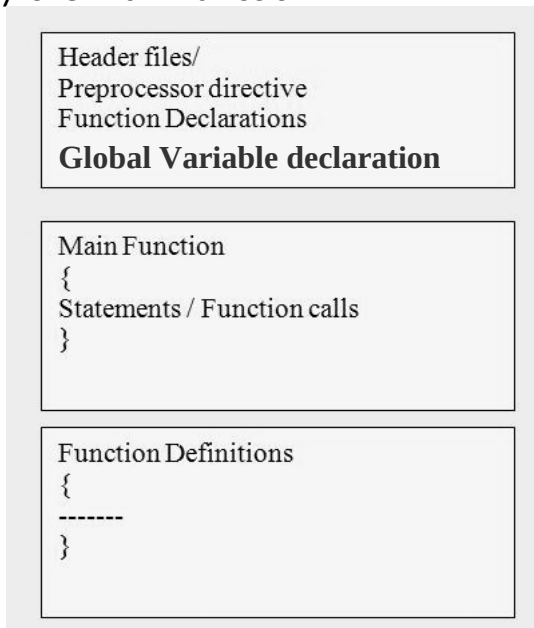
Chapter 3

START PROGRAMING IN 'C' LANGUAGE

As we discuss we can use various approaches to develop software. Today languages are based on two main concepts Procedural Programming and Object Oriented Programming. 'C' is procedural language and 'C++, Java' are Object Oriented languages. In procedural language functions are created to perform various tasks. Functions can be inbuilt or user defined and in object oriented language classes are created. Classes are the combination of functions and data.

'C' language is a procedural language where every thing is done in the form of functions. In 'C' language we have two main types of functions
1. Inbuilt
2. User defined functions.

Inbuilt functions are pre defined in the 'C' compiler and we use them according to the requirement, work of inbuilt functions are fix. With the help of inbuilt functions and statements we create user define functions. 'C' program can contain any number of user define functions. But one of them must be **main** function. Main function is the entry point of the program, compiler run the working of program from main function, all the other functions are directly and indirectly called by the main function.



Basic Structure of 'C' program

Explaination of Basic structure of 'C'

HEADER FILES:These include all the library functions,macros, it contains inbuilt functions like printf , scanf , getch etc.In C there are 24 header files.

PREPROCESSOR DIRECTIVE:- It instructs the C compiler to add contents of an include file into program during compilation. Include files should have .H extension
e.g #include<stdio.h>
GLOBAL VARIABLE DECLARATION:- There are some variables which are used in different function.It is better to declare such variables outside of all the functions.These variables are called global variables.
MAIN FUNCTION SECTION:- Every program of C must have one main() function. All statements written under this function must appear between the opening & closing braces.
FUNCTION DEFINITION:- This section contains all user defined function.These user-defined functions are always called in the main function.

Rules to create a program in 'C' language.

1. Each statement mostly ends with a semicolon (**;)**.
2. The statements in a program must be written in the same order in which we want to execute them.
3. All statements are written in small letters.
4. Every program must have main function.
5. clrscr() function should be written after variable declaration.
6. Program should be saved with the extension **.c** .

Some important functions
An ISO C standard libraries is made of 24 header files, which can be used into a program. Each header file contains one or more functions and macros, it contain inbuilt functions. In beginning we mainly use only two header files.

1. **stdio.h** : standard input output header file, it contains functions to input data in the program and take output from the program. Eg printf(), scanf() functions etc
2. **conio.h** : console input output header file, it also contains some additional input output functions.

Eg getch(),putch(),clrscr() etc

- printf():- printf function is used to take output from the program.
- scanf():- scanf function is used to give input in the program.
- clrscr():- used to clear the screen.
- getch():- To input a character.

What is a Program?
It is defined as the set of instructions that we give to computer as input through keyboard. Program is also known as passive entity and program in execution is called active entity.
Before writing program in 'C' language, it is always better that we write an algorithm of it. [Especially when we are new in programming]. In this book we write algorithms with the some programs so that beginners can understand programs easily.
Write algorithm and program to print “hello world”

Step1 Start
Step2 Print "hello world"
Step3 Stop
To write a program from algorithm, we step by step convert the steps into program.

Algorithm	**Program**
Step1 Start	void main() {
Step2 Print "hello world"	printf("hello world");
Step3 Stop	}

```
#include<stdio.h>
void main()
{
printf("hello world");
}
```

When we use inbuilt function in the program it is better to include header files of that functions like printf function is define in stdio.h so we include this file in our program. The program become like this.
We compile the program with alt+F9. It displays errors. After removing the errors we can run the program using Ctrl+F9.Complier runs the program and control come back to the editor screen, we can use Alt+F5 to view result.

Output:- hello world
If a program work on some variables then we need to declare these variables with the help of data types. We need to follow some rules to declare variables.

- The variables should be declared before clrscr() function.
- Use variable names that mean something. Single-letter variables are difficult to understand so it is better to use long names like **index** is better than **i**, **count** is better than **c**, and **name** is better than **n**.
- Variables are typically in lowercase. They can contain letters and numbers.
- Uppercase letters can be used in variable names.
- You should not begin a variable name with a number. They can contain numbers, but you begin it with a letter.
- "underscore" character can be used in variablenames like first_name, roll_no etc.
- We can not use keywords as variable names eg variable name like int,if,float are invalid.
- We can not create more than one variable with same name in same scope.

Write algorithm and program to print sum of two numbers

Step1 Start
Step2 declare a,b,c
Step3 a=10
Step4 b=20
Step5 c=a+b
Step6 Print c
Step7 Stop

.**Algorithm**	**Program**
Step1 Start	void main() {
Step2 declare a,b,c	int a,b,c;
Step3 a=10	a=10;
Step4 b=20	b=20;
Step5 c=a+b	c=a+b;
Step6 Print c	printf("%d",c);
Step7 Stop	}

In printf we need format specifier to print value of the variable like we use %d to print value of variable c. Some times we use getch() to avoid pressing Alt+F5. and clrscr() function if we want to clear the screen. clrscr() function should be used after the variable declaration.

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a,b,c;
clrscr();
a=10;
b=20;
c=a+b;
printf("%d",c);
getch();
}
```

Output:- 30

If we write printf function like :- printf("sum=%d",c);
Output:- **sum=30**

Write algorithm and program to swap two numbers

Step1 Start
Step2 declare a,b,c
Step3 a=10
Step4 b=20

Step5 c=a
Step5 a=b
Step5 b=c
Step6 Print a
Step6 Print b
Step7 Stop

Algorithm	**Program**
Step1 Start	void main() {
Step2 declare a,b,c	int a,b,c;
Step3 a=10	a=10;
Step4 b=20	b=20;
Step5 c=a	c=a;
Step5 a=b	a=b;
Step5 b=c	b=c;
Step6 Print a	printf("%d",a);
Step6 Print b	printf("%d",b);
Step7 Stop	}

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a,b,c;
a=10;
b=20;
c=a;
a=b;
b=c;
printf("a=%d ",a);
printf("b=%d ",b);
getch();
}
```

Output:- a=20 b=10

scanf function:- It is used to take input from the user. It contain format specifiers and variables. Eg:- scanf("%d",&a);
%d is format specifier, & is address operator and a is a variable name which store value that we enter.We can also enter more than one variable in scanf function like
scanf("%d%d",&a,&b);

Qualifier :-It give properties to an identifier. The **const** type qualifier declares an identifier to be fixed [not changeable]. The **volatile** qualifier declares an identifier whose value can rightfully be changed.

Write algorithm and program to enter two numbers and print sum of these numbers.

Step1 Start
Step2 Declare a,b,sum
Step3 Print "Enter value of a"
Step4 Input a
Step5 Print "Enter value of b"
Step6 Input b
Step7 sum=a+b
Step8 Print sum
Step9 Stop

.**Algorithm**	**Program**
p1 Start	void main() {
Step2 declare a,b,sum	int a,b,sum;
Step3 Print "Enter value of a"	printf("Enter value of a");
Step4 Input a	scanf("%d",&a);
Step5 Print "Enter value of b"	printf("Enter value of b");
Step6 Input b	scanf("%d",&b);
Step5 sum=a+b	sum=a+b;
Step6 Print sum	printf("%d",sum);
Step7 Stop	}

In this program we use format specifier %d for int in scanf function.

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a,b,sum;
printf("Enter value of a ");
scanf("%d",&a);
printf("Enter value of b ");
scanf("%d",&b);
sum=a+b;
printf("sum=%d",sum);
getch();
}
```

Output:- Enter value of a 10
Enter value of b 20
Sum=30

**10 and 20 will be entered by the user through keyboard*

We can use float, long, double data types according to the requirement. And if we want to work on characters we use char data type.

Write a program to input and print a character.

```
#include<conio.h>
#include<stdio.h>
void main()
{
char ch;
clrscr();
printf("Enter a character ");
scanf("%c",&ch);
printf("character=%c",ch);
getch();
}
```

Output:

Enter a character G

character =G

We can also assign a character without scanf function, but character should be in single quotes like 'a' 'F' etc

```
#include<conio.h>
#include<stdio.h>
void main()
{
char ch;
clrscr();
ch='A';
printf("character=%c",ch);
getch();
}
```

Output:- character=A

<u>The Escape Sequences</u>

Escape Sequences are used for formatting the output string mainly written in printf(). It consists of backslash followed by a character. The character sequence is always enclosed in the control string. Various escape sequences are

\n the next line character
\t tab position
\a used for sound
\r carriage return
\b shifts cursor one position left

Without any Escape sequence:
printf("hellobye");
Output:
hellobye
With \n Escape sequence:
Using \n the output will be shifted to next line
printf("hello\nbye");
Output:
hello
bye
With \t Escape sequence:
Using \t some space[TAB]will be displayed before printing next information
printf("hello\tbye");
Output:
hello bye
With \b Escape sequence:
After \b one character from the left of \b is deleted
printf("hello\bbye");
Output:
hellbye
With \r Escape sequence:
Using \r position of the cursor come to the beginning of the line.
printf("hello\rbye");
Output:
byelo
With \a sequence:
It produce beep sound.
printf("hello\abye");
Output:
hello[beep sound]bye

I/O functions in 'C' language

Input/Output means to receive data from any input device and send data to the output device respectively.

I/O functions can be classified into two types:

1. **Disk I/O functions**: These are the functions that perform I/O operations on secondary storage device like floppy, hard disk etc.
2. **Console I/O functions:** These are the functions that receive input from keyboard and write output to screen. These functions are further classified into two types:
    a. Unformatted I/O Functions
    b. Formatted I/O Functions

Unformatted I/O Function

1. getch(): When we type a character, the character would not be displayed on the screen but it is assigned to a variable immediately without pressing enter button from the keyboard.

For e.g.

char A:

A=getch();

2. getche(): When we type a character, the character is displayed on the screen and assigned to a variable immediately without pressing enter button from the keyboard.

For e.g.

char A:

A=getche();

3. getchar(): The typed character is displayed on the screen but it is assigned to a variable when we press enter button from the keyboard.

For e.g.

char A:

4. A=getchar();

5. gets(): It is used to input string. We can also input spaces between the various words of the string.

6. putch() or putchar(): These functions are exactly the same and use to print a character to screen.

7. puts: It is used to print the string on the monitor.

Formatted I/O Functions

1. scanf(): In scanf we can input integer, float, char, string simultaneously. The limitation of this function is that we cannot enter space between the words of the string.

2. printf(): This function is used to display values of int, float or plain messages to the user. Syntax is:

printf("format string",variablelist);

Type casting Or Type conversion

It is used to convert the one data type into another temporarily. It can be implicit or explicit. Implicit type conversion is automatically performed by the computer and explicit type conversion is performed by the programmer.

```
void  main()
{
int a=5;
float b;
b=a/2;
printf("b= %f",b);
}
```

Output:- b= 2.000000

```
void  main()
{
int a=5;
float b;
  b=(float)a/2;
printf("b= %f",b);
}
```

Output: b=2.500000

Important Questions and programs[Short Questions]

1. Explain printf and scanf functions.
2. Explain the use of clrscr function.
3. Write a program to convert centigrade temperature into Fahrenheit temperature.
4. Write a program to swap two numbers with third variable.
5. Write a program to swap two numbers without third variable.
6. Write rules to create a simple 'C' program.

Important Questions and programs[Long Questions]

1. Write a program to print sum of digits of two digit number.
2. Write a program to reverse a three digit number.
3. Write a program to input cost and quantity. Give 10% discount on total bill
4. Input marks in four subjects of a student. Print total and average marks.
5. Explain escape sequences.
6. Explain various input output functions.
7. Explain Structure of the 'C' program.
8. Write various rules for variable declaration.

** *Perform Programs from 1 to 15 from program list*

Chapter 4

OPERATORS

Operators are the symbols that are used to perform various operations on data, like arithmetic operations, comparison or logical operations etc.Operator refers to a symbol that tells the computer to perform certain mathematical or logical tasks on operands. Operators can be Unary and Binary. Unary Operators work on one operand like –a, b++ etc and Binary operators work on two operators like a+b, a*c etc.

C provides different types of operators as follow:-

1) Arithmetic operators
2) Logical operators
3) Assignment operators
4) Relational operators
5) Bit wise operators
6) Unary operators
7) sizeof operator
8) Comma operator
9) Conditional operator

These operators are performed on operands and operands are the variables on which these operations are performed.

e.g x+y

In this x,y are operands and + is operator and x+y is called expression.

1). Arithmetic operator:- These operators are used in arithmetic operations. C provides all the basic arithmetic operators like +,-,*,/,%. The modulus(%) operator tells us what would be the remainder after integer division.

NAME	**SYMBOL IN C**
Addition	+
Subtraction	-
Multification	*
Division	/
Modulus	%

Arithmetic operators are classified into 3 categories :-

a) Integer arithmetic:-When arithmetic operators are performed on two integers i.e a,b are integers , then the expression is known as an integer expression and this operation is called integer arithmetic.

e.g. if a & b are integer like a=9,b=4

then

a+b=13

a-b=5

a*b=36

a/b=2 (decimal ignored as it is an integer arithmetic)

a%b=1 (remainder)

b) Real arithmetic:- The Operators which are performed only on real operands, called real arithmetic. Real operands may include decimal or exponential notation.

e.g a=9.0 and b=4.0

then a/b=9.0/4.0=2.250000

NOTE:- The operation % cannot be used with real operands.

c) Mixed mode arithmetic:- It is the combination of real operand & integer operand. In this case if either operand is of real type then result will always be is a real no. e.g 25/10.0=2.5

whereas 25/10=2

2) Logical operators: - The logical operators && and || are used when we want to test more than one condition to make decisions. The result of logical operator AND (&&) will be true only if both the conditions are true, whereas the result of a logical operator OR (||) will be true if one or more conditions are true. In other words, the result of a logical OR operation will be false only if all conditions are false.

logical operators

NAME	SYMBOL IN 'C'
D	&&
OR	||
NOT	!

e.g. 1) (age>18 && age<=25)

2) (number<0 || number>10)

LOGICAL AND (&&)

AND (&&) is a logical operator that will give result if and only if all of its conditions are true.

Condition1	Condition2	Result	Value
False	False	False	0
False	True	False	0
True	False	False	0
True	True	True	1

LOGICAL OR (||)

OR (||) is a logical operator that will give result if minimum one of its condition is true.

Condition1	Condition2	Result	Value
False False True True	False True False True	False True True True	0 1 1 1

LOGICAL NOT (!)

NOT (!) is a logical operator that convert true into false and false into true.

Input	Output	Value
False True	True False	1 0

3) **Assignment Operators: -** It is used to assign value to the variable. Value can be any constant value or evaluation of an expression. It evaluates the expression on right hand side and then assigns the value to a variable on the left hand side. Some assignment operators are:

OPERATOR	EXAMPLE	MEANING
+=	X+=4	x=x+4
-=	y-=3	y=y-3
*=	Z*=4	z=z*4
/=	p/=9	p=p/9
%=	i%=2	i=i%2

e.g a=4 //assigns value 4 to variable a
x=a // assigns value of a to x
z=a+x //performs addition & assigns result to z

4) **Relational Operators:-** The relational operators are used to check the relation between two operands. These are used to compare two different quantities and then to take decisions e.g we may compare salary of two employees , age of two students and so on. It can be done by using relational operators. These operators are used with decision making statements such as if-else, for , while etc.

e.g. let a=1,b=5
then a<b is true
but
b<a is false

It means the result of relational operators is returned as either true or false i.e.1 or 0.

NAME	'C' SYNBOL
Greater than	>
Less than	<
Equal to	==
Not equal to	! =
Greater than or equal to	>=
Less than or equal to	<=

5) **Bit Wise Operators: -**These operators are used for testing the bits, or shifting them right or left. These operators are used for manipulation of data at bit level.

1. Bit wise logical operators
   a. Bit wise and (&)Bit wise or (|)
   b. Bit wise xor (^)
2. Bit wise shift operators
   a. Bit wise left(<<)
   b. Bit wise right(>>)
3. Complement operator(~)

Operator	Meaning
&	Bit wise Logical AND
|	Bit wise OR
^	Bit wise XOR
<<	Left shift
>>	Right shift
~	Complement

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,m,and,or,com,rs,ls,xor;
n=11;
m=5;
and=n&m;
or=n|m;
xor=n^m;
com=~n;
ls=n<<2;
rs=n>>2;
printf("\n n&m=%d",and);
printf("\n n|m=%d",or);
```

```
printf("\n n^m=%d",xor);
printf("\n ~n=%d",com);
printf("\n rs>>2=%d",rs);
printf("\n ls<<2=%d",ls);
}
```

Output:-
n&m=1
n|m=15
n^m=14
~n=-12
rs>>2=2
ls<<2=4

How Bitwise operators produce this result?
[in description we assume int is of 8 bit or 4 bit. But in 'C' language integer is of 16 bits]
The binary of n(11)= 00001011
The binary of m(5)= 00000101
a. n&m
1011
0101
0001 which is=1 so n&m=1
b. n|m
1011
0101
1111 which is=15 so n|m=15
c. n^m
1011
0101
1110 which is=14 so n^m=14
d. ~n
00001011
complement of this is 11110100 which =-12 in 2's compliment system
e. n<<2
00001011 when we shift 2 bits left then value=00101100
Which is=44 so n<<2=44
f. n>>2
00001011 when we shift 2 bits right then value=00000010
Which is=2 so n>>2=2

6) **Increment / Decrement operators:-** The increment operators increase the value of an operand by 1 and decrement operator decrease the value of an operand by 1.It is of types: -
a) Increment (++)
b) Decrement (- -)

Example

```
int a=5;
a++:
printf("%d",a);
```

Output is 6 means ++ operator increase the value by 1.

Example

```
int a=5;
a--:
printf("%d",a);
```

Output is 4 means -- operator decreases the value by 1.

Increment or Decrement operators are further two types: -

1) Pre increment operator/Pre Decrement operator
2) Post increment operator/post Decrement operator.

Both operators increment or decrement the variables but at different times. The statement ++x increments x before using its value, while x++ increments x after its value has been used.

The x++ or x-- reads the value & store a copy of it at temporary location. The computer increments or decrements the variables, & the temporary copy is used in expression.

e.g.

```
int x,y;
x=5;
y=x++;
printf("x=%d,y=%d\n",x,y);
```

o/p :-

```
x=6,y=5
```

Example of pre increment operator

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5,b;
b=++a;
printf("\na=%d",a);
printf("\nb=%d",b);
getch();
}
```

output:- a=6
b=6

Example of post increment operator

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5,b;
```

```
b=a++;
printf("\na=%d",a);
printf("\nb=%d",b);
getch();
}
```

output:- a=6
b=5

Example of pre decrement operator

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5,b;
b=--a;
printf("\na=%d",a);
printf("\nb=%d",b);
getch();
}
```

output:- a=4
b=4

Example of post decrement operator

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5,b;
b=a--;
printf("\na=%d",a);
printf("\nb=%d",b);
getch();
}
```

output:- a=4
b=5

7) sizeof operator: - The sizeof operator returns the size , in bytes , of the given operand. A sizeof operator is a unary operator that returns the number of bytes required to store a variable or a data type.

Syntax is:-

sizeof(datatype or variable)

Example:-

sizeof (int)

Write a program to find the size of int , float and char.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
float b;
char c;
printf("\n size of int =%d bytes",sizeof(a));
printf("\n size of float=%d bytes",sizeof(b));
printf("\n size of char =%d byte",sizeof(c));
getch();
}
```

Output

```
size of int =2 bytes
size of float=4 bytes
size of char=1 byte
```

8) Comma operator: - The comma operator (,) is used to declare more than one variable in single line. It is also known as separater as it separates one variable name from other.

Example

```
int a,b,c;
```

9) Ternary operator: - Ternary operator is also known as conditional operator or immediate if. These are **?** and **:** ternary operator. An operator that takes three statements to work. The syntax of this operator is

<Condition>? <Expression1>: <expression2>

The <condition> is evaluated first. If the result is true then expression 1 is executed, otherwise expression 2 is executed.

If <condition> is false ,expression3 is evaluated & its value becomes the result of expression.

e.g. if a=15,b=5

```
x=(a>b)?a:b
```

The above example means that if a is greater than b then a will be assigned to x, otherwise b will be assigned to x.

It can be achieved using if-else statements as follows:

```
if(a>b)
x=a;
else
x=b;
```

Note: The detail of if-else is discussed in next chapter.

Operators precedence and Associativity

Operators	Associativity
([- .	Left to right
! - ++ -{- + * & (type-cast) sizeof	Right to left
* / %	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
||	Left to right
?:	Left to right
= += -= *= /= %=	Right to left
&= ^= |= <<= >>=	Left to right

Important Questions[Short]

1. Define relational operators.
2. Explain Advance assignment operator.
3. Define Ternary operators.
4. Define sizeof operator.
5. Write difference between logical AND and OR operators.
6. Write difference between ++I and I++.

Important Questions[Long]

1. Explain Bitwise operators.
2. Explain increment and decrement operators with example.
3. Write precedence of the operators.

Chapter 5

CONDITION CONTROL STATEMENTS

Control Statements

Normally program is executed in the sequential order, and when we want to change the normal execution of the program then we need to use control statements or we can say control statements are used to change the normal execution of the program.

Control statements are mainly divided into three types

1. Condition control statements
2. looping or iterative statements
3. branching statements

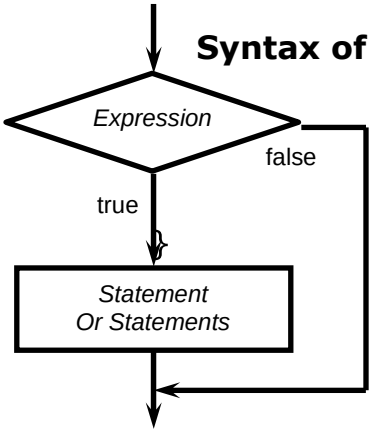
1. Condition control statements

These are used when we want to execute the statement or block of statements according to some condition. Conditional execution can be controlled by :-

a. if
b. if else
c. if else if
d. Nested if
e. switch case
f. Conditional operator or ternary operator

If Statement

It is the basic condition control statement. When we have single condition then we mostly use it. The if statement allows a programmer to test the value of a conditional expression and to select or reject the execution of the statement or block of statements depending on this value.



Syntax of if:

```
if (condition)
{
statement1;
statement2;
```

Statement1 and statement2 will only execute if the condition is true

RULES TO USE IF STATEMENT

- The IF condition ends with a pair of parenthesis
- There is no semicolon after the parenthesis

1. Write an algorithm and program to Input age and print "you are child" if age <10.

Algorithm	Program
Step1 Start	void main() {
Step2 declare age	int age;
Step3 Print "Enter age"	printf("Enter age");
Step4 Input age	scanf("%d",&age);

Step5 if age< 10
then
Print "You are child"
End
	if(age<10)
	{
	printf("You are child");
	}

Step7 Stop	}

2. Write an algorithm and program to Input a number and print its absolute value.[eg absolute value of 10 and -10 is 10, it is non negative value]

Algorithm	Program
Step1 Start	void main() {
Step2 declare a	int a;
Step3 Print "Enter value of a"	printf("Enter value of a");
Step4 Input a	scanf("%d",&a);

Step5 if a< 0
then
a=-a.
End
	if(a<0)
	{
	a=-a;
	}

Step6 print a	printf("%d",a);
Step7 Stop	}

Output:-

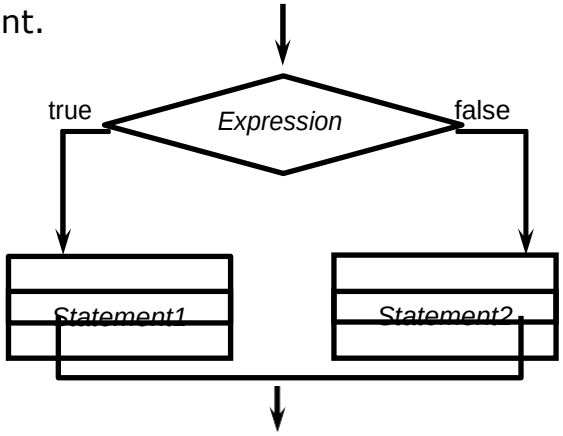
1. Enter value of a 22
   22
2. Enter value of a -22
   22

* *when we enter 22 if block is not executed and when we enter -22 if block convert this -22 to 22.*

In above program if we enter negative number then if block will executed and convert negative number to positive number.

IF ELSE

If is a single selection structure and if else is a double-selection structure because it selects from two different actions. If else is one of the most commonly used control statement.



Syntax

```
if (Expression)
        statement1
   else
        statement2
```

If Expression is true then it executes statement1 otherwise it executes statement2

Write an algorithm and program to Input two number and print greatest from them.

Algorithm	Program
Step1 Start	void main() {
Step2 declare a,b	int a,b;
Step3 Print "Enter value of a"	printf("Enter value of a");
Step4 Input a	scanf("%d",&a);
Step5 Print "Enter value of b"	printf("Enter value of b");
Step6 Input b	scanf("%d",&b);

Step5 if a> b
begin
Print "a is big"
End
Else
Begin
Print "b is big"
End
	if(a>b)
	{
	printf("a is big");
	}
	else
	{
	printf("b is big");
	}

Step7 Stop	}

Output:-

Enter value of a 20
Enter value of b 10
a is big

In above program “a is big” will be printed if a is greater than b else result will be ”b is big”.

What happen if a=b?

In above program if we input same values in both the variables then output will be “b is big”. To overcome this we need to create three blocks.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
if(a>b)
{
            printf("a is big");
}
if(b>a)
{
            printf(" b is big");
}
if(a==b)
{
            printf(" a is equal to b ");
}
getch();
}
```

If Else If ladder

If we have more than two conditions then we use if else if statement . In this we can create multiple blocks of statements and statements will execute according to the condition.

Syntax

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

Example of if-else-if statement.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
if(a>b)
{
            printf("a is big");
}
else if(b>a)
{
            printf(" b is big");
}
else
{
            printf(" a is equal to b ");
}
getch();
}
```

In this example if a is greater than b then first block is executed and if b>a then second block will execute else if none of the above conditions are true then else part will execute. If –else-if is faster than multiple if block because in this if first block is executed then all the other blocks are ignored by the compiler.

Nested if

We can write if statement within the scope of another if or else block, it is known as nested if.

Syntax of nested if:

```
if (condition)
        {
                if (condition)
                {
        do this;
        }
                else
                {
```

do this;

```
                and this;
            }
        }
        else
{
            do this;
}
```

In this syntax if condition 1 is true then compiler check the condition 2 and if condition 2 is also true then it will execute the statements written in inner if block.

Example of nested if.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b,c;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
printf("enter the value of c");
scanf("%d",&c);
if(a>b)
{
   if(a>c)
   {
   printf("a is big");
   }
}
if(b>a)
{
   if(b>c)
   {
          printf("b is big");
   }
}
if(c>a)
{
   if(c>b)
   {
          printf("c is big");
   }
}
getch(); }
```

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b,c;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
printf("enter the value of c");
scanf("%d",&c);
if(a>b)
{
    if(a>c)
    {
            printf("a is big");
    }
    else
    {
    printf("c is big");
    }
}
else
{
    if(b>c)
    {
            printf("b is big");
    }
    else
    {
    printf("c is big");
     }
   } }
```

Conditional operator or ternary operator

The conditional operator is sometimes called ternary operator since they take three arguments. These are the replacement for if else statements.

expression 1 ? expression 2 : expression 3

If expression 1 is true then the value returned will be expression 2 otherwise the value returned will be expression 3.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
(a>b)?printf("a is big"):printf(" b is big");
getch();
}
```

Nested ternary operators.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int a,b,c;
clrscr();
printf("enter the value of a");
scanf("%d",&a);
printf("enter the value of b");
scanf("%d",&b);
printf("enter the value of c");
scanf("%d",&c);
(a>b&&a>c)?printf("a is big"):(b>c)? printf(" b is big"):printf("c is big");
getch();
}
```

Switch statement

When we have multiple if statements then we can use switch case statement. The variable used in switch is known as switch variable. The variable in the switch statement can be of int or char type only. The switch statement can also contain the default statements that will be executed if none of the case is executed, default is optional.

syntax of Switch

```
switch (expression)
        {
                case 1:
                   ...block of statements1...
                break;
                case 2:
                   ...block of statements2...
                break;
                ----------------
```

```
                ----------------
                ----------------
                default:
                    ...block of statements for default..
            break;
            }
```

Program using if else if statements.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
char c;
clrscr();
printf("enter the value of c");
scanf("%c",&c);
if(c=='r')
{
    printf("colour is red");
}
else if(c=='b')
{
    printf("colour is blue");
}
else if(c=='g')
{
    printf("colour is green");
}
else
{
    printf("colour is white");
}
}
```

Program using switch case statement.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
char c;
clrscr();
printf("enter the value of c");
scanf("%c",&c);
switch(c)
{
case 'r':
    printf("colour is red");
```

```
break;
case ‘b’:
    printf("colour is blue");
break;
case ‘g’:
    printf("colour is green");
break;
default:
    printf("colour is red");
}
}
```

Applications of switch:

1) It makes program easy to read and understand.
2) It is the replacement of multiple if statements.
3) Whole switch block is act like a single block.

Limitations of switch:

1) We can only use int and char type data.
2) We can not write relational operators in case.

Like case > 10 is wrong.

Write an algorithm and program to input two numbers if choice a add it if choice s subtract it if choice d divide it otherwise give a message wrong key pressed. Algorithm:-

Step 1: START
Step 2: DECLARE ch,a,b,c
Step 3: PRINT “Enter your choice”
Step 4: INPUT ch
Step 5: PRINT “Enter two numbers”
Step 6: INPUT a,b
Step 7: if(ch==’a’) then GOTO STEP 8 else GOTO STEP 9
Step 8: c=a+b PRINT c GOTO STEP 14
Step 9: If(ch==’s’) then GOTO STEP 10 ELSE GOTO STEP 11
Step 10: c=a-b PRINT c GOTO STEP 14
Step 11: if(ch==’d’) then GOTO STEP 12 else GOTO STEP 13
Step 12: c=a/b PRINT c GOTO STEP 14
Step 13: PRINT “wrong key pressed”
Step 14: STOP

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
char ch;
printf(“enter two numbers”);
scanf(“%d%d”,&a,&b);
printf(“enter any character”);
```

```
fflush(stdin);
scanf("%c",&ch);
switch(ch)
{
    case 'a':
    c=a+b;
    printf("\naddition=%d",c);
    break;
    case 's':
    c=a-b;
    printf("\nsubtraction=%d",c);
    break;
    case 'd':
    c=a/b;
    printf("\ndivision=%d",c);
    break;
    default:
    printf("\nwrong key pressed");
}
getch();
}
```

** fflush(stdin); is used to clear the buffer. It is always better to use this function before entering character.*

Important Questions [Short]

1. What are control statements?
2. Write syntax of if-else-if
3. Write difference between if and if-else.
4. Define ternary operators.
5. What is the use of default statement in switch?

Important Question [Long]

1. Explain switch case statement with example.
2. Explain working of nested if statements.
3. Write syntax of various control statements used in 'C'.

** Perform Programs from 12 to 26 from program list