UNIT - II

Features of **C:**

* It is a structured programming language.
* It is highly portable.
* It is a middle level language.
* It is a case sensitive language.
* It has rich set of libraries.
* It uses Top-Down approach.

**C** Tokens

* Identifiers:
* Identifiers are names given to various program elements such as variables, functions and arrays etc,.

Rules for naming identifier:

* First character must be alphabetic or underscore.
* Must consist only of alphabetic characters, digits, or underscores.
* Only the first 31 characters of an identifier are significant and are recognized by the compiler.
* Cannot use a ***keywords*** or ***reserved word*** (e.g. main, include, printf & scanf etc.).
* No space are allowed between the identifiers etc,.
* Variable:
* Variable is an identifier that is used to represent some specified type of information.

Eg: x=3

* Here x is variable.
* Keywords:
* It is a reserved words.
* Cannot be used for anything else.
* Examples:
  + int
  + while
  + for etc,.
* Constants:
* It is an entity whose value does not changes during the execution

Types:

* Numeric Constant
* Character Constant

Numeric Constant:

Integer constants

* It is formed using a sequence of digits.

Decimal - 0 to 9 .

Octal - 0 to 7.

Hexa - 0 to 9 ,A to F

Eg: 10,75 etc.

Real Constatnts:

* It is formed using a sequence of digits but it contain decimal point.
* length, height, price distance measured in real number

Eg: 2.5, 5.11, etc.

Character Constant

* A character constant is a single character they also represented with single digit or a single special symbol which is enclosed in single quotes.

Eg: ‘a’, ‘8’,’\_’etc.

String constants:

* String constant are sequence of characters enclosed with in double quote.

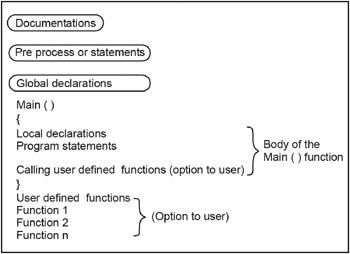
Eg: “Hello” ,”444”,”a” etc,.

Executing a C program

STRUCTURE OF A **C** PROGRAM:

A C program basically has the following form:

* Preprocessor Commands
* Functions
* Variables
* Statements & Expressions
* Comments



**Documentations**

The documentation section consist of a set of comment lines giving the name of the program, the another name and other details, which the programmer would like to use later.

**Preprocessor Statements**

The preprocessor statement begins with # symbol and are also called the preprocessor directive. These statements instruct the compiler to include C preprocessors such as header files and symbolic constants before compiling the C program. Some of the preprocessor statements are listed below.

* #include <stdio.h>
* #include <math.h>
* #include <conio.h>
* #define X 30

**Global Declarations**

The variables are declared before the main ( ) functions as well as user defined functions are called global variables. These global variables can be accessed by all the user defined functions including main ( ) function.

int x, y; // Global declaration

void main()

{

int a, b; // Local Variables

….

….

}

**The main ( ) function**

Each and Every C program should contain only one main ( ). The C program execution starts with main ( ) function. No C program is executed without the main function. The main ( ) function should be written in small (lowercase) letters and it should not be terminated by semicolon. Main ( ) executes user defined program statements, library functions and user defined functions and all these statements should be enclosed within left and right braces.  
  
**Braces**

Every C program should have a pair of curly braces ({, }). The left braces indicates the beginning of the main ( ) function and the right braces indicates the end of the main ( ) function. These braces can also be used to indicate the user-defined functions beginning and ending. These two braces can also be used in compound statements.

**Local Declarations**

The variable declaration is a part of C program and all the variables are used in main ( ) function should be declared in the local declaration section is called local variables. Not only variables, we can also declare arrays, functions, pointers etc. These variables can also be initialized with basic data types.

void main()

{

Int a, b; // Local Variables

}

# User defined functions

These are subprograms, generally, a subprogram is a function and these functions are written by the user are called user defined functions. These functions are performed by user specific tasks and this also contains set of program statements. They may be written before or after a main () function and called within main () function. This is an optional to the programmer.

void main()

{

Subprogram(); // function call

}

subprogram( ) // User defined function definition

{

}

Data Types

|  |
| --- |
| Data type Size(bytes) Range Format string |
| Char 1 128 to 127 %c |
| Unsigned char 1 0 to 255 %c |
| Short or int 2 -32,768 to 32,767 %i or %d |
| Unsigned int 2 0 to 65535 %u |
| Long 4 -2147483648 to 2147483647 %ld |
| Unsigned long 4 0 to 4294967295 %lu |
| Float 4 3.4 e-38 to 3.4 e+38 %f or %g |
| Double 8 1.7 e-308 to 1.7 e+308 %lf |
| Long Double 10 3.4 e-4932 to 1.1 e+4932 %lf |

**Operators**

* Arithmetic operator
* Relational operator
* Logical operator
* Assignment operator
* Increment or decrement operator(unary)
* Bitwise operator
* Conditional operator
* Arithmetic Operator
* It is used to carry out arithmetic operations like addition, subtraction etc,

Eg: + , - , \* , / etc,

Program:

#include<stdio.h> // Header File

#include <conio.h>

int b=10; //Global Declaration

void main ( ) /\* main is the starting of every c program \*/

{

int a,c; //Local Declaration

clrscr( );

scanf(“%d”,&a);

printf(“ \n The sum of the two values:”);

c = a+b;

printf(“%d”,c);

getch( );

}

* Relational Operator:
* It is used to **compare** two or more operands.

Eg : < , > , <= , >=, !=

* Logical Operator:
* It is used to **combine** the result of two or more condition.

AND(&&)

OR (||)

NOT (!) are Logical operators.

Eg: (i>10)&&(j>5).

(i>10)||(j>5) etc,.

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a=10,b=3,c=5,e;

clrscr( );

if(a>b) // relational operator

{

printf(" \n a is bigger than b");

}

if((a>b)&&(a>c)) //Logical operator

{

printf(" \n a is biggest");

}

getch( );

}

* Assignment operator:
* It is used to **assign a value or expression** etc to a variable.
* Eg: a =10.

a = b

a = b + c etc.,

**Compound operator**

It is also used to assign a value to a variable.

Eg: x + = y means x = x + y

**Nested operator**

It is used for multiple assignments.

Eg: i = j = k = 0;

Program:

#include<stdio.h>

#include <conio.h>

int b=10;

void main ( )

{

int a=3,b=5;

clrscr( );

a+=b; // a= a+b

printf(" \n The sum of the two values:%d",a);

getch( );

}

* Increment or Decrement Operator:
* It is used to **Increment or decrement** an operand.

* Eg: ++x (Pre Increment),

x++ (Post Increment),

--x (Pre Decrement),

x-- (Post Decrement).

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a=5;

clrscr( );

printf(" \n Post increment Value:%d",a++);

printf(" \n Pre increment Value:%d",++a);

printf(" \n Pre decrement Value:%d",--a);

printf(" \n Post decrement Value:%d",a--);

getch( );

}

* Bitwise Operator:
* It is used to manipulate data at bit level.

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a=5,b=4,c;

//char a=5,b=4,c;

clrscr( );

c = a&b;

printf(" \n value a&b is:%d",c);

getch( );

}

* Conditional Operator (or) Ternary Operator:
* It is used to checks the condition and execute the statement depending on the condition.

Eg: C = a > b ? a:b

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a=5,b=8,c;

clrscr( );

c = a>b?a:b; //Conditional operator

printf(" \n The Larger Value is%d",c);

getch( );

}

* Special Operator:
* comma operator ( , )
* sizeof operator
* pointer operator (& , \*) etc,.

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int c;

clrscr( );

printf(" \n size of int is:%d", sizeof( c));

getch( );

}

Expression:

* An expression represent data item such as variable, constant are interconnected using operators.
* Eg:

|  |  |
| --- | --- |
| Expression | C Expression |
| a + b + c | a + b + c |
| a2+b2 | a\*a + b\*b |

Operator Precedence & Associativity:

* The arithmetic expressions evaluations are carried out based on the precedence and associativity.
* The evaluations are carried in two phases.
  + **First Phase:** High Priority operators are evaluated.
  + **Second Phase:** Low Priority operators are evaluated.

|  |  |
| --- | --- |
| Precedence | Operator |
| High | \* , / , % |
| Low | + , - |

Type Conversion:

* Converting the type of an expression from one type to another type.

Eg: x = (int)10.45

Program:

#include<stdio.h>

#include <conio.h>

void main ( )

{

int c;

clrscr( );

c=(int)10.45;

printf("\nOutput is:%d",c);

getch( );

}

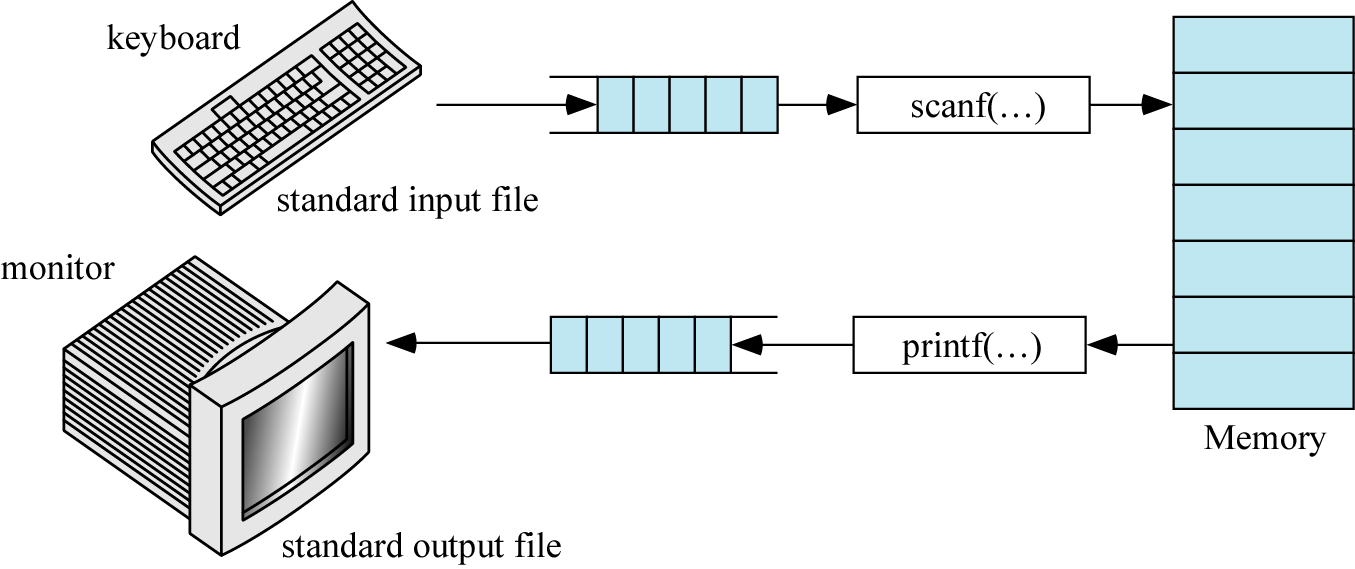
**OUTPUT**

Output is: 10

* **Input / Output Functions:**

Formatted Input/Output:

* **Formatted input** : reads formatted data from the keyboard.
* **Formatted output** : writes formatted data to the monitor.



* **Character Test Function**

It is used to test the character taken from the input.

* isalpha(ch)
* isdigit(ch)
* islower(ch)
* isupper(ch)
* tolower(ch)
* toupper(ch)

**Unformatted I/O Functions:**

* getc(char) :

It is used to get a single character.

Example:

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

void main()

{

char x;

printf("enter the character:");

x=getc(stdin);

if(islower(x))

putc(toupper(x),stdout);

else

putc(tolower(x),stdout);

getch();

}

* getchar() :

It accepts one character type data from the keyboard.

Example

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

void main()

{

char x;

printf("enter the character:");

x=getchar();

if(islower(x))

putchar(toupper(x));

else

putchar(tolower(x));

getch();

}

**Output:**

enter the character:A

a

* getche() :

It displays the entered character in the screen.

Example:

#include <stdio.h>

#include <conio.h>

void main()

{

char c ;

clrscr();

printf("\nInput a string:");

c = getche();

printf("\nstring is:");

putch(c);

getch();

}

* getch():

getch() accepts only single character from keyboard. The character entered through getch() is not displayed in the screen (monitor).

* gets(char):

It accepts any line  of string including spaces from the standard Input device (keyboard). gets() stops reading character from keyboard only when the enter key is pressed.

Example:

#include <stdio.h>

#include<conio.h>

void main()

{

char c[80];

clrscr();

printf("Input a string:");

gets(c);

printf("The string is:");

puts(c);

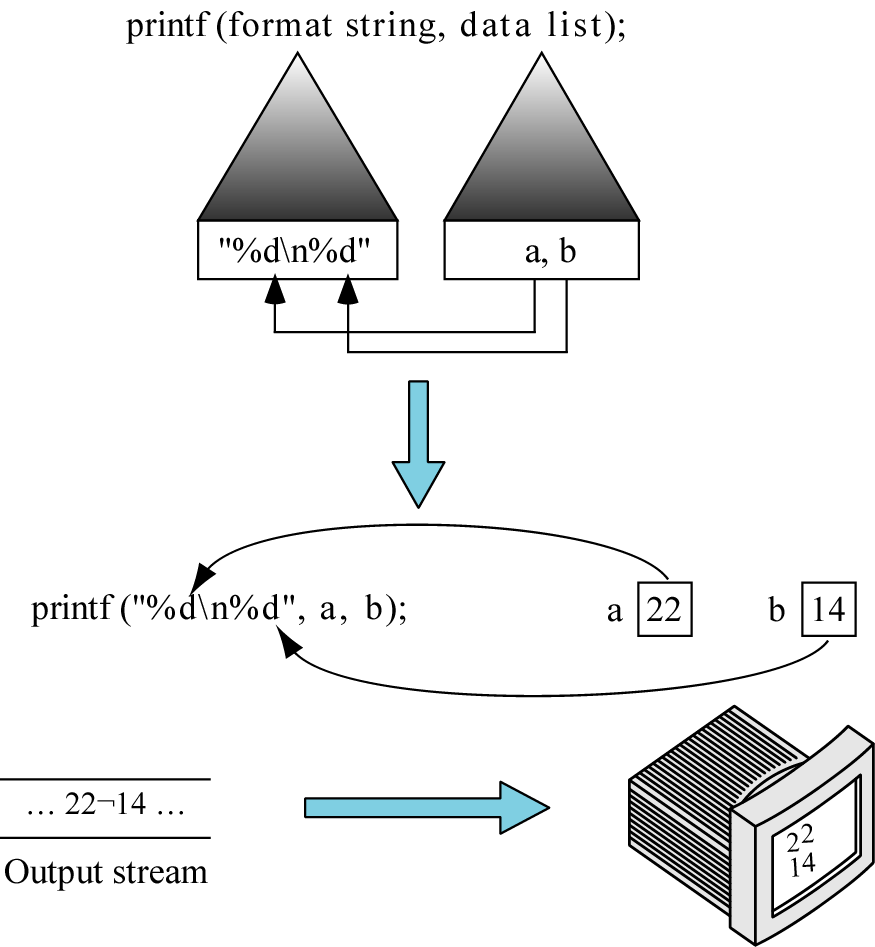
getch();

}

**Standard Output**

* The standard output file is the monitor.
* Like the keyboard, it is a *text* file.
* When you need to display data that is not text, it must be converted into to the text before it is written to the screen.

Format of **printf** Statement:



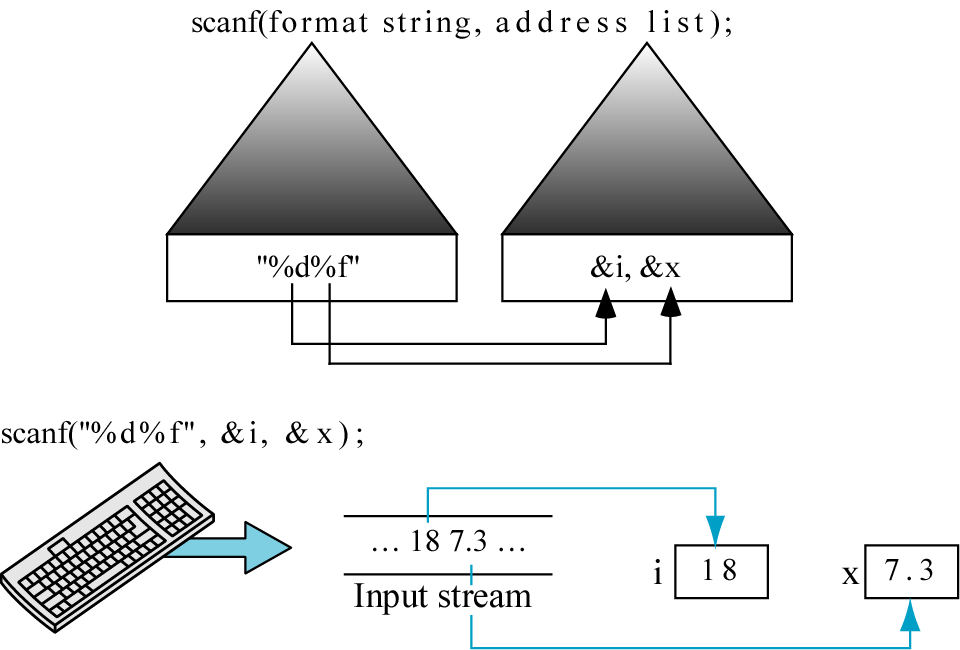
Formatted Input (***scanf***):

* The standard formatted input function in C is *scanf* (scan formatted).
* ***scanf*** consists of :
  + A format string .
  + An address list that identifies where data are to be placed in memory.

**scanf ( format string, address list );**

(“%c….%d…..%f…..”, &a,….&i,…..,&x…..)

Format of **scanf** Statement



**Control Structure**

Control structures are blocks of code that dictate the flow of control.

* **Categories:**
  + Sequential structure

In which instructions are executed in sequence.

* + Selection structure

In which instruction are executed based on the result of some condition

* + Iteration structure

In which instruction are executed repeatedly.

Selection Structure:

* It allows the program to make a choice from alternative paths.
* C provide the following selection structures
  + IF statement
  + IF … ELSE statement
  + Nested IF … ELSE statement
  + IF … ELSE ladder

* **IF Statement**

**Example:**

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a;

clrscr( );

printf("\nEnter the number:");

scanf("%d",&a);

if(a>10)

{

printf(" \n a is greater than 10");

}

getch( );

}

* **IF…ELSE Statement**

**Example:**

#include<stdio.h>

#include <conio.h>

void main ( )

{

int a;

clrscr( );

printf("\nEnter the number:");

scanf("%d",&a);

if(a>10)

{

printf(" \n a is greater than 10");

}

else

{

printf(" \n a is less than 10");

}

getch( );}

* **NESTED IF… ELSE**

Syntax

IF (condition1)

{

IF (condition2)

{

True statements;

}

ELSE

{

False statements;

}

}

ELSE

{

False statements;

}

* **IF…ELSE LADDER**

**Syntax**

IF (condition1)

{

statements;

}

else if (condition2)

{

statements;

}

else if (condition3)

{

statements;

}

else

{

statements;

}

**Example:**

#include<stdio.h>

#include<conio.h>

void main()

{

int m1,m2,m3;

float avg;

printf("\nEnter the marks:");

scanf("%d%d%d",&m1,&m2,&m3);

avg=(m1+m2+m3)/3;

printf("\n The average is:%f",avg);

printf("\n The Grade is:");

if(avg>=60)

{

printf("First class");

}

else if(avg>=50)

{

printf("Second class");

}

else if(avg>=35)

{

printf("Thrid class");

}

else

{

printf("Fail");

}

getch();

}

* **Looping structure**
* It is used to execute some instructions several time based on some condition.
  + WHILE
  + Do…WHILE
  + For
* **WHILE Loop**

**Example:**

#include<stdio.h>

#include<conio.h>

void main()

{

int i=1,fact=1,n;

printf("\nEnter the Number:");

scanf("%d",&n);

while(i<=n)

{

fact =fact \*i;

i++; // To increment i=i+1

}

printf("\n The value of %d! is:%d",n,fact);

getch();

}

* **DO…WHILE Loop**
* **FOR Loop**

**Syntax**

for (initialization; test condition; Increment/Decrement)

{

Body of the loop

}

**Example:**

#include<stdio.h>

#include<conio.h>

void main()

{

int i,fact=1,n;

printf("\nEnter the Number:");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

fact =fact \*i;

}

printf("\n The value of %d! is:%d",n,fact);

getch();

}

* **Nested for loop**

**Syntax**

for (initialization; condition; Inc/Dec)

{

for (initialization; condition; Inc/Dec)

{

Body of the loop

}

}

* **CASE structure**

**Syntax**

switch (expression)

{

case constant 1:

block1;

break;

case constant 2:

block2;

break;

.

.

default :

default block;

break;

}

Example:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,n;

printf("\nEnter the Number:");

scanf("%d",&n);

switch(n)

{

case 1:

{

printf("\n Its in case 1");

break;

}

case 2:

{

printf("\n Its in case 2");

break;

}

default:

{

printf("\n Its in default");

break;

}

}

getch();

}

break Statement

* It is used to terminate the loop
* When a ***break*** statement is encountered inside a loop, then the loop is terminated.
* Loops with break Statement

While

For:

* Continue Statement
* When a ***continue*** statement is encountered inside a loop, the control is transferred to the beginning.

Loops with continue Statement

While:

do while:

For: