**C language**

C (pronounced as "*see*", like the letter C) is a general-purpose computer programming language  developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories for use with the UNIX operating system.

Although C was designed for implementing system software  it is also widely used for developing portable application software.

C is one of the most widely used programming languages of all time and there are very few computer architectures for which a C compiler does not exist. C has greatly influenced many other popular programming languages, most notably C++, which began as an extension to C.

”C” is a compiler it can convert high level language code to low level language ,because of system(processor) can understand only low level language and people know only high level language

So “C” is a mediator between programmer(high level language) and system(processor)

“C” is developed by Dennis ritchie in 1972

**Language:** It acts as interface (mediator) between the programmer and the system. It offers some rules and regulations for writing the programs. It also offers some library, which is required for writing the programs. Obviously, the collection of programs is called as “Software”.

C is a middle level language because it has both high , low level language features

So to write a “C” program also it has some rules and regulations

**Charecteristics of a c program:**

1) A ”C” program is a collection of one more functions

2) Every “C” program must have at least one function called “main” it indicates starting of the program

3) All c statements must be enclosed with in a pair of braces ( { } )

4) Left brace indicates starting of the program

5) Right brace indicates ending of the program

6) Every statement ends with semicolon (;)

7) There is no rule the program must start left side , we may start the program from right side also (or) from the center also

8) We may have any number of blank spaces in b/n two different words, but not with in a word

9) All c statements are entered in lower case letters only

From above concepts we learnt how to write c program

When we are writing c program we need to take some variables to store the data(constants)

So now lets discuss about DATATYPES

**DATATYPES:**

Data type defines size of the variable.

‘data types’ which are used to define a variable before its use.

Data types are used to store various types of data that is processed by program. Data type attaches with variable to determine the number of bytes to be allocate to variable and valid operations which can be performed on that variable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable type | Key word (or) key word | Format specifiers | Byte required | Range |
| Character | Char | %c | 1 | -128 to 127 ( -27  to 27-1) |
| Unsigned character | unsigned char | %c | 1 | 0 to 255 ( 0 to 28-1) |
| Integer | Int | %d or %i | 2 | -32768 to 32767 (-215  to 215-1) |
| Short integer | short int | %d or %i | 2 | -32768 to 32767 (-215  to 215-1) |
| Long integer | long int | %ld | 4 | -2,147,483,648 to 2,147,438,647 (-231  to 231-1) |
| Unsigned integer | unsigned int | %u | 2 | 0 to 65535 ( 0 to 216-1 ) |
| Unsigned short integer | unsigned short int | %u | 2 | 0 to 65535 (0 to 216-1) |
| Unsigned long integer | unsigned long int | %lu | 4 | 0 to 4,294,967,295 (0 to 232-1) |
| Float | Float | %f | 4 | (-231 to 232-1) |
| Double | Double | %lf | 8 | (-263  to 263-1) |
| Long double | long double | %Lf | 10 | (-279 to 279-1) |

Simply if we have 1 byte--------8 bits------- -27 to 27-1 (signed purpose)

If we have 1 byte--------8 bits------- 0 to 28-1 (unsigned purpose)

**Alphanumeric:**

(i)char(1 byte): (a)signed (b)unsigned

-27 to 27-1 0 to 28-1

-128 to 127 0 to 255

ASCII codes: There are 256 ASCII codes in “C”.

Ascii code character

\0 – – – – – – 0 black slash zero

A – – – – – – – 65

B – – – – – – 66

X – – – – – –– 90

a – – – – – – – 97

b – – – – – – –98

‘

z – – – – – – 122

|  |  |
| --- | --- |
| main()  {  short signed int sree;  sree=10;  printf(“%d”,sree);  %i  } | Alt+f9---------for compilation  Cltr+f9------- for execution  Alt+f5-------to see the result |

**Rules for constructing a variable name:**

(i)A variable name is a combination of alphabets and digits.

(II)Alphabets may be lower case or upper case.

(iii)first character should not be a digit.

(iv)No comas,blank spaces are allowed with in the variable name.

(v)No special character are allowed other than underscore( understrike, underbar, low line, or low dash).

(vi)A variable name must have atleast one charcter length.

|  |  |
| --- | --- |
| **Right**  sree;  sree 1 2;  s1ree;  sree;  sree12;  sree\_M;  -sree | **wrong**  1sree;  sree m;  sree \*;  sree-m; |

**Rules for constructing an integer constant:**

(I)An integer constant must name atleast one digit.

(ii)No zeros are placed before an integer constant.

(iii)No decimal points are placed within an integer constant.

(iv)No comas (,) blank spaces are allowed with in an integer constant.

(v)A +ve or –ve sign can be placed before an integer constant,the default sign is \_+ve.

|  |  |
| --- | --- |
| **Correct**  10;  +10;  10000;  -10; | **Wrong**  010  10.00;  10,000;  10.000; |

main()

{

float raju;

raju=2.5;

printf(“%f”,raju);

}

**Ouput: 2.500000**

**Rules for constructing a character constant:**

(i)A character constant must be enclosed in a single coat.

(ii)A character constant gets stored in the memory in its ASCII codes.

(iii)The maxium length of character constant is only one digit.

main()

{

signed char tech;

tech=’A’;

printf(“%c”,tech);

}

**Output: A**

**We may assign charecters and integers into char data type variable**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | void main() |

|  |  |
| --- | --- |
|  | { |

|  |  |
| --- | --- |
|  | char ch = 'A'; |

|  |  |
| --- | --- |
|  | printf("%c\n",ch); |

|  |  |
| --- | --- |
|  | ch = 65;// using integer representation |

|  |  |
| --- | --- |
|  | printf("%c\n",ch); |

|  |  |
| --- | --- |
|  | ch = '\x41';   // using hexadecimal representation |

|  |  |
| --- | --- |
|  | printf("%c\n",ch); |

|  |  |
| --- | --- |
|  | ch = '\101';   // using octal representation |

|  |  |
| --- | --- |
|  | printf("%c\n",ch); |

|  |  |
| --- | --- |
|  | } |

**Write a program to find area of a square of 2cm side.**

|  |  |
| --- | --- |
| main()  {  short signed int side;  short signed int area;  side=2;  area=side\*side;  printf(“%d”,area);  } | **Write a program to find area of a triangle of length 3cm and breadth 6cm.**  main()  {  short signed int length;  short signed int breadth;  short signed int area;  length=3;  breadth=6;  area=length \*breadth;  printf(“%d”,area);  }  **Output: 18** |
| **WAP to find area of a triangle of 3cm base and 5cm height.**  main()  {  short signed int base;  short signed int height;  float area;  base=3;  height=5;  area=0.5\*base\*height;  printf(“%f”,area);  }  **Ouput: 7.500000** | **WAP to find area of a circle of 2cm radius.**  main()  {  short signed int radius;  float area;  radius=2;  area=3.14\*radius\*radius;  printf(“%f”,area);  } |
| **WAP to find perimeter of a rectangle of length 3cm and breadth 4.1cm**  main()  {  short signed int length;  float breadth;  float perimerter;  length=3;  breadth=4;  perimeter=2\*(length+breadth);  printf (“%f”, perimeter);  }  **Output**:  **14.200000** | **WAP to find circumference of a circle of 3.1cm radius**.  main()  {  float radius;  float cir;  radius=3.1;  cir=2\*3.14\*radius;  printf(“%f”,cir);  } |
| **WAP to find area of a square of side 2cm.**  main()  {  int s=2,a;  a=s\*s;  printf(“%d”,a);  } | **WAP to find area of a triangle of 3cm length and 6cm breadth.**  main()  {  int l=3,b=6,a;  a=l\*b;  printf(“%d”,a);  } |
| **WAP to find area of a triangle of 3cm base and 5cm height.**  main()  {  int b=3,h=5;  float a;  a=0.5\*b\*h;  printf(“%f”,a);  }  **Output: 7.500000** | **WAP to find circumference of a triangle of 2.1cm radius.**  main()  {  float r=2.1,c;  c=2\*3.14\*r;  printf(“%f”,c);  }  **Output**: |

**printf**

**syntax:** printf(“string”,var1,var2,…….)

main()

{

int a=65;

printf(“%d\n”,a);---------65

printf(“%i\n,a);------------65

printf(“%0\n”,a);---------101

printf(“%x\n”,a);----------41

printf(“%u\n”,a);----------65

printf(“%c”,a);---------------A

}

**Format specifier:**

**Int:**  **Float:**

%d – decimal %f – float

%i – integer %e - engineering

%o – octal %g – no logging zeros

%x – hexa %lf - double

%u – unsigned %lf – long double

%c – char

%ld – long int

%lu – long unsigned

**Note:** you may print more than one value by using printf

main()

{

int a=5,b=6;

printf(“%d%d”,a,b);

}

**Note:**when we are printing the data we have to use relevant format specifiers

For exp if we want to print int ------then-----%d and float--------then------%f only don’t cross these limits

main()

{

int a=5,b=6;

float c=2.5;

printf(“%d%d%f”,a,b,c);

printf(“%d%f%d”,a,c,b);

printf(“%f%d%d”,c,a,b);

}

**Escape sequence: (or) Character constants:**

\n -------- new line

\t --------tab

\b -------- back space

\r --------carriage return

\a --------audible alert

\‘ ---------single quote

\? --------question mark

\\ --------back slash

|  |  |
| --- | --- |
| main()  {  int a=5,b=6;  printf(“%d/n%d”,a,b);  }  **Output:**  **5**  **6** | main()  {  int a=5,b=6;  printf(“%d\n”,a);  printf(“%d”,b);  }  **Output:**  **5**  **6** |
| main()  {  printf(“hello\balu”);  }  **Output:**  **hellalu** | main()  {  printf(“hello\rhai”);  }  **Output:**  **hailo** |

**Ordinary:**

Normal text will be treated as ordinary

Example:”hello” , “how” ,”enter a number”…………etc

These will be printed on output as it is

|  |  |
| --- | --- |
| main()  {  printf(“good boy”);  }  **Output:good boy** | main()  {  int a=5,b=6;  printf(“a=%d,b=%d”,a,b);  }  **Output: a=5,a=6** |
| main()  {  int a=5,b=6;  printf(“a=%d\n b=%d”,a,b);  }  **Output: a=5**  **b=6** | main()  {  int a=5,b=6;  printf(“%d”,a);  printf(“\n %d”,b);  }  **Output: a=5**  **b=6** |

**Common programs:**

|  |  |
| --- | --- |
| **WAP to find area of rectangle of length 3cm and breadth 5.2cm**  main()  {  int l=3;  float b=5.2,a;  a=l\*b;  printf(“area of rectangle=”%f”,a);  } | **WAP to find area of a rectangle of length 3cm and breadth 5.2cm.**  main()  {  int l=3;  float b=5.2,a;  a=l\*b;  printf(“area of rectangle=%f”,a);  } |
| **Write a program to print area of square:**  main()  {  float s,a;  clscr();  printf(“Enter side”);  scanf(“%f”,&s);  a=s\*s;  printf(“%f”,a);  getch();  } | **WAP to find circumference of a circle**  main()  {  float r,c;  clrscr();  printf(“enter radius”);  scanf(“%f”,&r);  c = 2\*3.14\*r;  printf(“%f”,c);  getch();  } |
| **WAP to find perimeter of a rectangle**  main()  {  float l,b,p;  clrscr();  printf(“enter length,breadth”);  scanf(“%f%f”,&l,&b);  p=2\*(l+b);  printf(“%f”,p);  getch();  } | **WAP to find avg of three integer**  main()  {  int a,b,c;  flaot avg;  printf(“enter a,b,c”);  scamf(“%d%d%d”,&a,&b,&c);  avg=(a+b+c)/3.0;  printf(“%f”,avg);  getch();  } |
| **WAP to find avg of five subject marks**  main()  {  int s1,s2,s3,s4,s5;  float avg;  clrscr();  printf(“enter 5 subject marks”);  scanf(“%d%d%d%d%d”,&s1,&s2,&s3,&s4,&s5);  avg=(s1+s2+s3+s4+s5)/5.0;  printf(“%f”,avg);  getch();  } | **WAP to find simple intrest**  main()  {  flaot p,t,r,si;  printf(“enter p,t,r”);  scanf(“%f%f%f”,&p,&t,&r”);  si=p\*t\*r/100;  printf(“%f”,si);  } |

**WAP to find discriminent of a general quadratic equation**

main()

{

int a,b,c,d;

printf(“enter a,b,c”);

scanf(“%d%d%d”,&a,&b,&c);

d=b\*b-4\*a\*c;

printf(“%d”,d);

}

**Operators:**

**Definition:** operator is a symbol it performs particular operation on operands.

Data itself is operand

**Types of operators.**

|  |  |
| --- | --- |
| 1.Arithmetic  2.Modulus  3.Increment and Decrement  4.Relational  5.Logical | 6.Bitwise  7.Conditional  8.Assignment  9.Special operations |

**1.Arithmetic:-(+,-,\*,/)**

+,-,\*,/ are called arithmetic operatiors. There are binary operations. We apply there operators on ints, floats, chars.

main( )

{

int x;

x = ‘A’ + ‘B’ ;

printf (“%d”,x);

}

**Output: 131**

**2.Modulus:-(%)**

% is called modulus operator. It is a binary operator. It gives remainder.

main( )

{

int x;

x = 5%2;

printf (“d”,x);

}

**Output: 1**

(i) If numerator is is less than denominator we get numerator as reminder.

main ( )

{

int x ;

x = 7%77;

printf (“%d”,x);

}

**Output: 7**

(ii) The sign of the results depends only on sign of numerator but not on denominator.

|  |  |
| --- | --- |
| main( )  {  printf (“%d\n”, 4%3); 1  printf(“%d\n”, 4% -3); 1  printf(“%d\n”,- 4%3); -1  printf(“%d\n”,- 4% -3); -1  } | main ( )  {  printf (“d%”, -33% 777); -33  } |

(iii) We should not apply modulus operator on floats eventhough we apply compiler gives error message.

|  |  |
| --- | --- |
| **Write a program to print number of days,weeks,months,years in given value:**  main ( )  {  int d =45, m,w;  m= d/30;  d = d%30  w = d/7;  d = d%7;  printf (“%d months, %d weeks, %d days,m,w,d);  }  **Output: 1 months , 15 weeks ,2 days , 1 day** | **Sum of digits , reverse a given number , sum of the squares of the digits**  main ( )  {  int n =253, m, s =0;  m = n%10;  s = s+m; \\s = s\*10+m; \\ s =s+ m\*m;  n = n /0;  m = n%10;  s =s+m; \\ s =s\*10+m; \\ s =s+ m\*m;  n = n/10;  m = n%10;  s = s + m; \\ s =s \*10+m;\\ s =s + m\*m;  n = n/10;  printf (“%d”, s);  } |

**3. Increment and decrement:-** ++, - - are called increment and decrement operators. These are unary operators. These are two notations prefix, postfix.

|  |  |
| --- | --- |
| **Prefix**  b = + + a ; a= a+ 1;  b =a;  main ( )  {  int a = 5,b;  b = + + a;  printf (“% d% d”, a, b);  }  **Output:** 6 6 | **Postfix**  b= a ++;  b = a;  a = a+ 1;  main ( )  {  int a =5, b;  printf (“%d %d”, a, b);  }  **Output:** 6 5 |

|  |  |
| --- | --- |
| main ( )  {  int a =1, b=5, c =3, d=8,e;  e = a++ + b++ + ++c + d++;  printf (“%d % d% d% d% d” , a, b, c, d, e);  }  **Output:** 2 6 4 9 19 | main ( )  {  int a =1, b =5, c =3, d =8, e;  e =a++ + +b-- + --c + ++d;  printf (“%d %d %d %d %d”, a, b, c, d, e);  }  **Output:**2 4 2 9 17 |

(i) In isolated statements there is no difference between prefix and postfix

main ( )

{

int a =5;

+ + a;//a++ //(there is no if you write a++ or ++a)

printf (“%d”, a);

}

**Output:**6

(ii) We should apply this operator only on variable;

but not on constants i. e x++ is valid, but 5++ is invalid.

**4. Relational operations :-**

Symbol Meaning

< less than

> Greater than

< = less than or equal to

> = greater than or equal to

= = equal to

! = not equal to

1. These are binary operaters.
2. Relational operators check the relation between two operations.
3. If the relation is true it gives true value i.e 1. If the relation is false it gives false value i.e 0.

main ( )

{

int a =5, b =7, c;

c = a<b

a <=b

a>b

a>=b

a= =b

a!=b

printf (“%d”, c);

}

**Output:** 1 1 0 0 0 1

**5. Logical operations** **:-**

|  |  |  |
| --- | --- | --- |
| (i) Logical And (&&) | (ii) Logical OR (||) | (iii)Logical not (!) |

|  |  |
| --- | --- |
| **(i) Logical and (&&) :-**  It is a binary operator.  **Syntax:-** oparend1 && oparend2  Table:-  OP1 OP2 OP1 && OP2  0 0 0  0 1 0  1 0 0  1 1 1 | **(ii) Logical or (||) :-**  It is a binary operator.  **Syntax:-** oparend1 || oparend2  Table:-  OP1 OP2 OP1 || OP2  0 0 0  0 1 1  1 0 1  1 1 1 |

**Note:**C language treats a non zero value as true and ‘0’ as false ;

but C language gives true value as 1 and false as 0.

|  |  |
| --- | --- |
| main ( )  {  int n;  n=5&&26;  printf (“%d”, n );  }  **Output: 1**  main ( )  {  int a =25, b =0,c;  c=a || b;  printf (“%d”,c);  }  **Output: 1** | main ( )  {  int a =7, b = -13, c;  c =a && b;  printf (“d”, c);  }  **Output: 1**  main ( )  {  printf (“%d”, 5 && ’0’);  }  **Output: 1** |

|  |  |
| --- | --- |
| **(iii)Logical not:-** ! (negation operator)  It is a unary operator. It changes true value to false and false value to true.  **Syntax:-** !oparend1  Table:-  Op! !Op1  1 0  0 1 | main ( )  {  int a = -13, b;  b = ! a;  printf (“%d”, b);  }  **Output:** **0** |

**6. Bitwise operator:-**

(i) Bitwise and (&)

(ii) Bitwise OR (|)

(iii) Bitwise XOR (^)

(iv) Once compliment (~) (telda)

(v) Left shift (<<)

(vi) Right shift (>>)

We should not apply bitwise operators on floats.

Bitwise operators always concentrate on bits

1. **Bitwise and (&) :- (ii) Bitwise OR (|)**

It is a binary operator.

|  |  |
| --- | --- |
| main ( )  {  int a =5, b =6, c;  c = a & b;  printf (“%d”, c);  }  **Output explanation:**  00000000000000000101  00000000000000000110  00000000000000000100--------4 | main ( )  {  int a =5, b = -6, c;  c = a | b;  printf (“%d”, c)  }  **Output explanation:**  00000000000000000101  10000000000000000110  10000000000000000100--------(-4)  -ve | +ve; -ve; |

**(iii)Bitwise XOR :-**

OP1 Op2 OP1 ^ OP2

0 0 0

0 1 1

1 0 1

1 1 0

main ( )

{

int a=5, b =6, c;

c = a ^ b;

printf (“%d”, c);

}

**Output explanation:**

000000000000000101

000000000000000110

000000000000000011-----------(3)

**(iv)Once compliment :-**

Syntax:- ~Op

It is a unary operator. It changes true bit to false and false bit to true.

main ( )

{

int a =5, b;

b =~a;

printf (“%d”, b);

}

**Output: -6**

**(v) Left shift operator :-**

Syntax:- OP1 << no. of bits to shift left.

It is a binary operator. If you shift “n” no. of bits to left then the no. o is multiple by 2n.

main ( )

{

int a =5, b;

b =a<<3;

printf (“%d”, b);

}

**Output explanation:**

00000000000000000101-----5

Shifting first shift 00000000000000001010-----10

Shifting second shift 00000000000000010100------20

**(iv) Right shift operators:-**

**Syntax:** OP1 >> no. of bits to shift right.

It is binary operator. If you shift ‘n’ no. of bits to right then the no. is divided by 2n.

main ( )

{

int a =40,b;

b=a >> 3;

printf (“%d”, b);

}

**6.CONDITIONAL OPERATORS:-**

“ ?: ” is called conditional operator

**SYNTAX:-** Exp1? Exp2: Exp3;

It is a tri nary operator

First Exp1 is evaluated if it gives true value then Exp2 is evaluated otherwise Exp3 is evaluated

|  |  |
| --- | --- |
| main( )  {  int a, b ;  printf(“enter a, b ”);  scanf(“%d % d”,&a, &b );  a>b ? printf(“a is big”): print f(“b is big”);  } | main( )  {  int a;  print f(“enter a no”);  scan f(“%d”, &a );  a%2= =0?print f(“even”) :print f (“odd”);  } |

c.) main( )

{

int a;

print f(“enter an year”);

scanf(“%d”, &a );

a%4= =0?printf(“leap”) :printf (“non leap”);

}

**ASSIGNMENT OPERATORS:-**

= , += , -= , \*= , /= , %= , != , &= , <<= , >>= , ^= are called assignment operators

These are binary operators

**SYNTAX:-** OP1 Operator OP2

main( )

{

int a=5;

a+=2

/\* -=

\*=

%=

/=

&=

!=

^=

<<=

>>= \*/

printf (“%d”, a );

}

**SPECIAL OPERATORS-**

**a) &-address**

**SYNTAX: –** &var

It is a unary operator. we apply this operator only on variables, we should not apply address operator on register storage clause

It gives address of a variable

|  |
| --- |
| main ( )  {  int a=5;  printf(“% u”, &a);  }  **output:** 2004(some address) |

**b)value at address( \* )**

**SYNTAX:-** \*(address)

It is a unary operator

|  |
| --- |
| main ( )  {  int a=5;  print f(“%d”,\*(&a));  }  **Ouput: 5** |

**c.)sizeof ((var) or(constant) or(data type))**

It is a unary operator . it gives size(of the variable) in bytes

|  |
| --- |
| main( )  {  int x=10;  printf(“%d\n”, sizeof(x));  printf(“%d\n”, sizeof(25));  printf(“%d\n”, sizeof(int));  printf(“%d\n”, sizeof(float));  printf(“%d\n”, sizeof(long));  printf(“%d\n”, sizeof(char));  printf(“%d\n”,sizeof(5.6));  printf(“%d”,sizeof(32768));  }  **output: 2 2 2 4 4 1 8 4** |

**d.)type cast-**

**SYNTAX:-** (type)op

It is a unary operator. It changes one data type to another data type .this change is temporary but not permanent

|  |
| --- |
| main( )  {  int a=5,b=2;  float c;  c=(float)a/b;  print f(“%f” , c);  }  **Output: 2.500000** |

Note:-

[\*] ( ) function operator

[\*] [ ] array

[\*] . structure

[\*] -> structure

**CTOKENS**

A smallest literal in a C language is called c token

**Following are the c tokens**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.Keywords | 2. Identifiers | 3. Constants | 4.Operators | 5.String | 6.Special symbols |

|  |  |
| --- | --- |
| Keywords | int , float, char,if |
| Identifiers | main( ), print f( ),scan f( ); a, b |
| Operators | =,&& ,\*,^ |
| Constants | 25,2.5,‘A’ |
| String | “abc”,“abc123” |
| Special symbols | { } |

|  |  |  |
| --- | --- | --- |
| **This Program converts the Temperature from Celcius and Fahrenheit** main( )  {  float c, f ;  printf(“enter c”);  scanf(“%f”,&c);  f=(c-32)/1.8;  print f(“%f”, f);  } | **Swapping of two variables using third one:-**  main ( )  {  int a=5,b=10,c;  c=a;  a=b;  b=c;  print f(“a=%d, b=%d”, a, b);  }  **Output: a=10,b=5** | **Swapping of two variables without using third one:-**  main( )  {  int a=5,b=10;  a= a + b;  b= a - b;  a= a - b;  print f(“a=%d, b=%d”, a, b );  } |

**Operator precedency table:**

|  |  |  |
| --- | --- | --- |
| **OPERATOR** | **MEANING** | **ASSOCIATIVITY** |
| ( )  [ ]  .  -> | Function  Array  Structure  structure | Left to right |
| -  ++,--  !  ~  \*  (type)  sizeof( )  & | Unary minus  increment and decrement  logical not  once compliment  Value at address type cast  sizeof  address | Right to left |
| \*  /  % | multiplication  division  modulus | Left to right |
| +  - | Plus  Minus | Left to right |
| <<  >> | Left shift  Right shift | Left to right |
| <  >  <=  >=  ==  != | less than  greater than  less than or equal to  greater than or equal to  equal to equal to  not equal to | Left to right |
| & | Bitwise and | Left to right |
| ^ | Bitwise xor | Left to right |
| | | Bitwise or | Left to right |
| && | Logical and | Left to right |
| || | Logical or | Left to right |
| ?: | Conditional | Left to right |
| =,+,==,\*=,/=,%=,  &=,!=,^=,<<=,>>= | Assignment operators | Right to left |
| , | Comma | Left to right |
| ; | Semi colon | Left to right |

**Examples:**

|  |  |
| --- | --- |
| main( )  {  int a=5,b=6,c=7,d=8,e=9,f=11,g;  g=a +b && c>d\*2%e-f/2;  print f (“%d”, g );  }  **Ouput::** g=5+6 && 7>8\*2%9-11/2;  =5+6&& 7>7-5;  =11 && 7>2;  =11 &&1;  =1 | main ( )  {  int x;  x=5-3\*5+2%3\*5/2+1;  print f(“%d”, x );  }  **Output:**x=5-3\*5+2%3\*5/2+1;  =5-15+5+1  =-10+6  =-4 |

**WAP to find sum of n natural numbers**

main( )

{

int n, s ;

printf(“enter n”);

scanf(“%d”,& n) ;

s=n\*(n+1)/2;

printf(“%d”, s );

}

**Note:-** if we want to get root values,logerithemics values we can’t get directly in c languge.so we have to use

#include<math.h> header file functions

**#include<math.h>**

|  |
| --- |
| sqrt(x)= |
| pow(x, y)= |
| exp(x)= |
| log(x)=log |
| log10(x)=log |
| sin(x)= |
| cos(x)= |
| tan(x)= |
| asin(x)= |
| acos (x)= |
| atan(x)= |
| fabs(x)-|x| |
| fmod(x,y)-x%y |
| ceil(x)-rounded upto nearest integer |
| floor(x)-rounded downto nearest integer |

**Program:**

**x^5+y^8+squareroot(|X^2+y^2|)+e^\x+Y|+log|X|+sin36 solve this equation**

#include<math.h>

main()

{

float x,y,r;

printf("enter x,y");

scanf("%f%f",&x,&y);

r=pow(x,5)+pow(y,8)=sqrt(fab(x\*x+y\*y))+exp(fab8(x+y))+log(x)+sin(36\*3.14/180);

printf(“%f”,r);

}

**control statements:**

**1.Sequential control statements**

**2.Branch control statements**

**3.switch control statements**

**4.Loop control statements**

**5.Jump control statements**

**2.Branch control statements:**

**i)if**

**ii)if else**

**iii)nested if**

**iv)else-if ladder**

**i)if:**

**syntax:** if(condition)

{

true block;

}

|  |  |
| --- | --- |
| main()  {  float a,d=0:  printf("enter amount");  scanf("%f",&a);  if(a>500)  {  d=0.1:  }  a=a-a\*d:  printf("%f",a);  } | main()  {  float a,b=0:  int gender;  printf("enter amount");  scanf("%f"'&a);  if(a>500)  {  b=0.1;  }  a=a+a\*d:  printf("%f",a);  } |
| main()  {  float a,b=0;  int gender;  printf("enter amount");  scanf("%f",&a);  printf("enter 1 for male 2 for female”);  scanf("%d",&gender);  if(gender==2)  {  b=0.1;  }  a=a+a\*b;  printf("%f",a);  } | main()  {  int a,b:  printf("enter a,b");  scanf("%d%d",&a,&b):  if(a>b)  {  printf("a is big");  }  if(b>a)  {  printf("b is big");  }  } |
| main()  {  int a;  printf("enter a");  scanf("%d",&a);  if(a%2==0)  {  printf("a is even"):  }  if(a%2!=0)  {  printf("a is odd");  }  } | main()  {  int a:  printf("enter a year");  scanf("%d",&a);  if(a%4==0)  {  printf("a is leap year");  }  if(a%4!==0)  {  printf(“non leap year”);  }  } |

**ii) if-else:-**

**syntax:-** if(con)

{

true block statements

}

else

{

false block statements

}

|  |  |  |
| --- | --- | --- |
| main()  {  int a,b;  printf("enter a,b");  scanf("%d%d",&a,&b);  if(a>b)  {  printf("a is big");  }  else  {  printf("b is big");  }  } | main()  {  int a:  printf("enter a”);  scanf("%d",&a);  if(a%2==0))  {  printf("a is even");  }  else  {  printf("a is odd");  }  } | |
| main()  {  int a:  printf("enter a year");  scanf("%d",&a);  if(a%4==0)  {  printf("leap");  }  else  {  printf("non leap");  }  } | main()  {  float a,d:  printf("enter a");  scanf("%f",Ra);  if(a>500)  {  d=0.1;    } | else  {  d=0;  }  a=a-a\*d:  printf("%f",a);  } |

**iii) nested if:-**

|  |  |  |
| --- | --- | --- |
| main()  {  int a,b,c:  printf("enter a,b,c");  scanf("%d%d%d",&a,&b,&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"):  }  } | main()  {  float a,b  int gender,age:  printf("enter amount");  scanf("%f",&a);  printf("enter gender");  scanf("%d",&gender):  printf("enter age");  scanf("%d",&age);  if(gen==1)  {  if(age>12)  {  b=0;  }  else  {  b=0.1;  }  }  else  {  b=0.1;  }  a=a+a\*b:  printf("%f"'a):  } | if(gen==2)  {  b=0.1;  }  else  {  if(age>12)  {  b=0;  }  else  {  b=0.1;  }  }  a=a+a\*b:  printf("%f"'a):  } |

**iv) else if ladder:-**

**syntax:-** if(con1)

{

\_;

\_;

}

else if(con2)

{

\_;

\_;

}

else if(con3)

{

\_;

\_;

}

else if(con n)

{

\_;

\_;

}

**Write a program to caleculate charge of the passenger:**

main()

{

float charge,d;

int ch;

printf("enter dist");

scanf("%f",&d);

printf("enter 1 for Male Adults 2 for Female Adults 3 for Male Child 4 for Female Child 5 for phy handi male female 6 for ph child");

scanf("%d",&ch);

if(ch==1)

charge=1;

else if(ch==2)

charge=0.9:

else if(ch==3)

charge=0.5:

else if(ch==4)

charge=0.45;

else if(ch==5)

charge=0.5:

else

charge=0;

charge=charge\*distance;

printf("%f",charge):

}

**Write a program to print day of the week if you enter a value in between 1 to 7**

main()

{

int ch:

printf("enter any number in between 1 to 7");

scanf("%d",&ch);

if(ch==1)

printf("sunday");

else if(ch==2)

printf("monday");

else if(ch==3)

printf("tuesday");

else if(ch==4)

printf("wednesday");

else if(ch==5)

printf("thursday");

else if(Ch==6)

printf("friday");

else if(ch==7)

printf("saturday"):

else

printf("enter1 to 7 only");

}

**Write a program to print roman letters if we give in b/n 1 to 10:**

main()

{

int a:

printf("entr 1 to 10"):

scanf("%d",&a);

if(a==1)

printf("i");

else if(a==2)

printf("ii");

else if(a==3)

printf("iii");

else if(a==4)

printf("iv");

else if(a==5)

printf("v");

else if(a==6)

printf("vi");

else if(a==7)

printf("vii");

else if(a==8);

printf("viii");

else if(a==9)

printf("ix");

else if(a==10)

printf("x");

else

printf("enter 1 to 10 only");

}

**Write a program to find average of 5 subjects marks:**

|  |  |
| --- | --- |
| main()  {  int s1,s2,s3,s4,s5,avg;  printf("enter 5 sub marks"):  scanf("%d",R avg);  scanf("%d%d%d%d%d",&s1,&s2,&s3,&s4,&s5):  avg=(s1+s2+s3+s4+s5)/5:  if(avg>=70)  printf("dist");  else if(avg>=60)  printf("first"):  else if(AVG>=50)  printf("second"):  else if(avg>=40)  printf("third");  else  printf("fail");  } | if(avg>=70)  printf("dist");  if(avg<70 && avg>=60)  printf("first");  if(avg<60 && avg>=50)  printf("second");  if(avg<50 && avg>=40)  printf("third");  if(avg<40)  printf("fail");  } |

**Discriminant:**

main()

{

int a,b,c,d:

printf("enter a,b,c");

scanf("%d%d%d",&a,&b,&c);

d=b\*b-4\*a\*c;

if(d==0)

printf("real and not equal");

else

printf("imaginary");

}

**Write a program to print biggest of three numbers:**

main()

{

int a,b,c,d:

printf("enter a,b,c,d);

scanf("%d%d%d%d",&a,&b,&c,&d);

if(a>b && a>c && a>d)

printf("a is big");

else if(b>c && b>d)

printf("b is big"):

else if(c>d)

printf("c is big");

else

printf("d is big");

}

**Sorting of three numbers:**

main()

{

int a,b,c:

printf("enter a,b,c");

scanf(“%d%d%d”,&a,&b,&c);

if(a>b && a>c)

{

if(b>c)

{

printf("%d%d%d",c,b,a);

}

else

{

printf("%d%d%d",b,c,a):

}

}

else if(b>c)

{

if(a>C)

{

printf("%d%d%d",c,a,b);

}

else

{

printf(%d%d%d",a,c,b);

}

}

else

{

if(a>b)

{

printf("%d%d%d,b,a,c);

}

else

{

printf("%d%d%d”,a,b,c);

}

}

}

**3.switch control statements:-**

**syntax:-**

switch(expression)

{

case constant expression1:statement1 ; stament2 ;

case constant expression2: statement3 ; stament4 ;

. .

. .

case con expn:\_;\_;

}

|  |  |
| --- | --- |
| **Write a program to print given number in words:**  main()  {  int ch:  printf("enter any number in between 1-5");  scanf("%d",&ch);  switch(ch)  {  case 1:printf("one"); break;  case 2:printf("two"); break;  case 3:printf("three"); break;  case 4:printf("four"); break;  case 5:printf("five"); break;  }  } | **Write a program to print day of the week if you enter a value in between 1 to 7**  main()  {  int ch:  printf("enter any number in between 1-7");  scanf("%d",Rch);  switch(ch)  {  case 1: printf("sunday"); break;  case 2: printf("monday"); break;  case 3: printf("tuesday"); break;  case 4: printf("wednesday"); break;  case 5: printf("thursday"); break;  case 6: printf("friday"); break;  case 7: printf("saturday"); break;  default: printf("enter 1-7 only");  }  } |
| **Write a program to print roman letters if we give in b/n 1 to 10:**  main()  {  int a;  printf("enter 1-9");  scanf("%d",&a);  switch (ch)  {  case 1: printf("i"); break;  case 2: printf("ii"); break;  case 3: printf("iii"); break;  case 4: printf("iv"); break;  case 5: printf("v"); break;  case 6: printf("vi"); break;  case 7: printf("vii"); break;  case 8: printf("viii"); break;  case 9: printf("ix"); break;  default: printf("enter 1-9 only");  }  } | **Write a program to addition , substraction ,multiplication , division of two numbers from the given choice(1,2,3,4)**  main()  {  float a,b,c;  int ch:  printf("enter 1 for add,2 for sub,3 for mul,4 for div);  scanf("%d",&ch);  printf(“enter two values”);  scanf(“%d%d”,&a,&b);  switch(ch)  {  case 1: c=a+b;printf("%f",c); break;  case 2: c=a-b;printf("%f",c); break;  case 3: c=a\*b;printf("%f",c); break;  case 4: c=a+b;printf("%f",c); break;  default: printf("enter 1-4 only");  }  } |

**Write a program to addition , substraction ,multiplication , division of two numbers from the given choice(+,-,\*,/)**

#include<studio.h>

main()

{

float a,b,c:

char ch;

printf("enter a,b");

scanf("%f%f",&a,&b);

fflush (stdin);//when we are taking a character after an intger or float we have to use fflush(stdin)

printf("enter + for add - for sub \* for mul / for div");

scanf("%c",&ch);

switch(ch)

{

case '+': c=a+b;printf("%f",c);break;

case '-': c=a-b;printf("%f",c);break;

case '\*': c=a\*b;printf("%f",c);break;

case '/': c=a/b;printf("%f",c);break;

default: printf("enter 1-4 only");

}

}

**NOTE:-**

i)case constants or lables may be ints or characters;but not float.

ii)case constants or lables may be ib ascending order may be descending order or may not be any particular order.

|  |  |
| --- | --- |
| **Write a program to print character equalent word:**  main()  {  char ch;  printf("enter a for apple b for boy c for cat");  scanf("%c",&ch);  switch(ch)  {  case 'a'  case 'a': printf("apple"); break;  case 'b'  case 'b': printf("boy"); break;  case 'c'  case 'c': printf("cat"); break;  default: printf("enter abc only");  }  } | **Write a program to print given character is vowel or consonant**  main()  {  char ch;  printf("enter a alphabet');  scanf("%c",&ch);  switch(ch)  {  case'a':  case'e':  case'i':  case'o';  case'u': printf("vowel"); break;  default: printf("consonant");  }  } |

**Write a program to print 0-4 if we give in b/n 0 to 4 , 5-9 if we give in b/n 5 to 9…………………………etc**

main()

{

int ch:

printf("enter a no.");

scanf("%d",&ch);

ch=ch/5;

switch(ch)

{

case 0: printf("0-4"); break;

case 1: printf("5-9"); break;

case 2: printf("10-14"); break;

default: printf("15 or above");

}

}

**Loop control statements or iterative or repeatative:-**

**i)while**

**ii)do-while**

**iii)for**

**I)while:-**

**syntax:-** initialization

while(con)

{

true block statemnts

inc or dec

}

**Programs:**

|  |  |
| --- | --- |
| **1)generation of first 'n' integers:-**  main()  {  int i,n;  printf("enter n value");  scanf("%d",&n);  i=1;  while(i<=n)  {  printf("%d",i);  i++;  }  } | **2)sum of first 'n' integers:-**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=n)  {  s=s+i;  i++;  }  printf("%d",s);  } |
| **3)product of first 'n' integers (or) factorial of a given number**  main()  {  int i,n,fa=1;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=n)  {  fa=fa\*i;  i++;  }  printf("%ld",fa);  } | **4)To find sum of digits of given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  while(n!=0)  {  m=n%10;  s=s+m;  n=n/10;  }  printf("%d",s);  } |
| **5)To reverse a given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  printf("%d",s);  } | **6)WAP to check whether a given number is palindrome or not?**  main()  {  int n,m,s=0,n1;  printf("enter n");  scanf("%d",&n);  n1=n;  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  if(n1==s)  printf("palindrome");  else  printf("not palindrome");  } |

**7)To check whether a given number is amstrong or not**

main()

{

int n,m,s=0,n1;

printf("enter n"):

scanf("%d",Rn);

n1=n;

while(n!=0)

{

m=n%10;

s=s+m\*m\*m;

n=n/10;

}

if(n1==s)

printf("Amstrong");

else

printf("not Amstrong");

}

|  |  |
| --- | --- |
| **8)To find G.C.D of given two number**  main()  {  int a,b;  printf("enter a,b);  scanf("%d%d",&a,&b);  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  printf("%d",a);  } | **9)L.C.M of two numbers**  main()  {  int a,b,LCM,c,d;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a;  d=b;  while (a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  LCM=c\*d/a;  printf("%d",LCM);  } |
| **10)WAP to print a multiplication table**  main()  {  int n,i;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=10)  {  printf("%d\*%d=%d\n",n,i,n\*i);  i++;  }  } | **11)write a program to print febinocci series**  main()  {  int n,min=0,max=1,i,sum;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=n)  {  printf("%d",min);  sum=min+max;  min=max;  max=sum;  i++;  }  } |
| **12) write a program to print given number is prime (or) not**  main()  {  int i,n,c=0;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=n)  {  if(n%i==0)  {  c++;  }  i++;  }  if(c==2)  printf("prime");  else  printf("not prime");  } | **12) write a program to check wheather given number is perfect number (or) not**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  i=1;  while(i<=n)  {  if(n%i==0)  {  s=s+i;  }  i++;  }  if(n==s)  printf("perfect number");  else  printf("not perfect number");  } |

**do-while:**

**syntax:**

initialisation

do

{

true block

inc or decr

}while(con);

**Programs:**

|  |  |
| --- | --- |
| **1)generation of first 'n' integers:-**  main()  {  int i,n;  printf("enter n value");  scanf("%d",&n);  i=1;  do  {  printf("%d",i);  i++;  } while(i<=n);  } | **2)sum of first 'n' integers:-**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  s=s+i;  i++;  } while(i<=n);  printf("%d",s);  } |
| **3)product of first 'n' integers (or) factorial of a given number**  main()  {  int i,n,fa=1;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  fa=fa\*i;  i++;  } while(i<=n);  printf("%d",fa);  } | **4)To find sum of digits of given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  do  {  m=n%10;  s=s+m;  n=n/10;  } while(n!=0);  printf("%d",s);  } |
| **5)To reverse a given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  do  {  m=n%10;  s=s\*10+m;  n=n/10;  } while(n!=0);      printf("%d",s);  } | **6)WAP to check whether a given number is palindrome or not?**  main()  {  int n,m,s=0,n1;  printf("enter n");  scanf("%d",&n);  n1=n;  do  {  m=n%10;  s=s\*10+m;  n=n/10;  } while(n!=0);  if(n1==s)  printf("palindrome");  else  printf("not palindrome");  } |

**7)To check whether a given number is amstrong or not**

main()

{

int n,m,s=0,n1;

printf("enter n"):

scanf("%d",Rn);

n1=n;

do

{

m=n%10;

s=s+m\*m\*m;

n=n/10;

} while(n!=0);

if(n1==s)

printf("Amstrong");

else

printf("not Amstrong");

}

|  |  |
| --- | --- |
| **8)To find G.C.D of given two number**  main()  {  int a,b;  printf("enter a,b);  scanf("%d%d",&a,&b);  do  {  if(a>b)  a=a-b;  else  b=b-a;  } while(a!=b);  printf("%d",a);  } | **9)L.C.M of two numbers**  main()  {  int a,b,LCM,c,d;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a;  d=b;  do  {  if(a>b)  a=a-b;  else  b=b-a;  } while(a!=b);  LCM=c\*d/a;  printf("%d",LCM);  } |
| **10)WAP to print a multiplication table**  main()  {  int n,i;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  printf("%d\*%d=%d\n",n,i,n\*i);  i++;  }while(i<=10);  } | **11)write a program to print febinocci series**  main()  {  int n,min=0,max=1,i,sum;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  printf("%d",min);  sum=min+max;  min=max;  max=sum;  i++;  }while(i<=n);  } |
| **12) write a program to print given number is prime (or) not**  main()  {  int i,n,c=0;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  if(n%i==0)  {  c++;  }  i++;  } while(i<=n);  if(c==2)  printf("prime");  else  printf("not prime");  } | **13) write a program to check wheather given number is perfect number (or) not**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  i=1;  do  {  if(n%i==0)  {  s=s+i;  }  i++;  } while(i<=n);  if(n==s)  printf("perfect number");  else  printf("not perfect number");  } |

**Differences between while and do while:**

|  |  |
| --- | --- |
| **while** | **do while** |
| syntax:  initialisation  while(con)  {  TBS  inc or dec  } | Syntax  initialisation  do  {  TBS  inc or dec  }while(con); |
| In while loop first condition is checked and after that true block statements are exicuted | In do while loop first true block statements  are exicuted and after condition is checked |
| we may not tell that true block statements are exicuted atleast once | we can say that true block statements are  executed atleast once |
| It is called as entry loop control statements | It is called as exhit loop control statement |
| It is called as pre conditional loop control statement | It is called as post conditional loop control statement |

**3)For\_loop:**

**syntax:**

for(initialization;con;inc or dec)

{

TBS

}

**Programs:**

|  |  |
| --- | --- |
| **1)generation of first 'n' integers:-**  main()  {  int i,n;  printf("enter n value");  scanf("%d",&n);  for(i=1;i<=n;i++)  {  printf("%d",i);  }  } | **2)sum of first 'n' integers:-**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  {  s=s+i;  }  printf("%d",s);  } |
| **3)product of first 'n' integers (or) factorial of a given number**  main()  {  int i,n,fa=1;  printf("enter n");  scanf("%d",&n);  i=1;  for(;i<=n;)  {  fa=fa\*i;  i++;  }  printf("%ld",fa);  } | **4)To find sum of digits of given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  for(;n!=0;)  {  m=n%10;  s=s+m;  n=n/10;  }  printf("%d",s);  } |
| **5)To reverse a given number**  main()  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  for(;n!=0;)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  printf("%d",s);  } | **6)WAP to check whether a given number is palindrome or not?**  main()  {  int n,m,s=0,n1;  printf("enter n");  scanf("%d",&n);  n1=n;  for(;n!=0;)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  if(n1==s)  printf("palindrome");  else  printf("not palindrome");  } |

**7)To check whether a given number is amstrong or not**

|  |  |
| --- | --- |
| main()  {  int n,m,s=0,n1;  printf("enter n"):  scanf("%d",&n);  n1=n;  for(;n!=0;)  {  m=n%10;  s=s+m\*m\*m;  n=n/10;  } | if(n1==s)  printf("Amstrong");  else  printf("not Amstrong");  } |

|  |  |
| --- | --- |
| **8)To find G.C.D of given two number**  main()  {  int a,b;  printf("enter a,b);  scanf("%d%d",&a,&b);  for(;a!=b;)  {  if(a>b)  a=a-b;  else  b=b-a;  }  printf("%d",a);  } | **9)L.C.M of two numbers**  main()  {  int a,b,LCM,c,d;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a;  d=b;  for(;a!=b;)  {  if(a>b)  a=a-b;  else  b=b-a;  }  LCM=c\*d/a;  printf("%d",LCM);  } |
| **10)WAP to print a multiplication table**  main()  {  int n,i;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=10;i++)  {  printf("%d\*%d=%d\n",n,i,n\*i);  }  } | **11)write a program to print febinocci series**  main()  {  int n,min=0,max=1,i,sum;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  {  printf("%d",min);  sum=min+max;  min=max;  max=sum;  }  } |
| **12) write a program to print given number is prime (or) not**  main()  {  int i,n,c=0;  printf("enter n");  scanf("%d",&n);  i=1;  for i=1;i<=n;i++)  {  if(n%i==0)  {  c++;  }  }  if(c==2)  printf("prime");  else  printf("not prime");  } | **13) write a program to check wheather given number is perfect number (or) not**  main()  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  {  if(n%i==0)  {  s=s+i;  }  }  if(n==s)  printf("perfect number");  else  printf("not perfect number");  } |

**Stars programs(printing output in different styles)**

|  |  |
| --- | --- |
| main()  {  int i,j;char x='A'  for(i=1;i<=5;i++,printf("\n"))  {  for (j=1;j<=5;j++)  <=i  <=6-i  {  printf("\*");  ("%d",i);  ("%d",j);  ("%c",x);  }  }  } | main()  {  int i,j,char x='A';  for(i=1;i<=5;i++,printf("/n")  {  for(j=1;j<=6-i;j++)  printf("\* ");  ("%d ",i);  ("%d ",j);  ("%c ",x);  }  } |
| main()  {  int i,j;  char x='A';  for(i=1;i<=5;i++,printf("\n"))  {  for(j=1;j<=6-i;j++)  printf("\* ");  ("%d",i);  ("%d",j);  ("%c",x);  }  } | main()  {  int i,j;  for(i=1;i<=5;i++,printf("\n"))  {  for(j=1;j<=i;j++)  printf("\*");  for(j=1;j<=6-i;j++)  printf(" ");  for(j=1;j<=6-i;j++)  printf(" ");  for(j=1;j<=i;j++)  printf("\*");  }  } |

**Write a program to print pyramid by using stars**

main()

{

int i,j;

for(i=1;i<=5;i++; printf(“\n”))

{

for(j=1;j<=6-i;j++)

printf(“ “);

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

printf(“ \* ”);

}

for(i=1;i<=5;i++;printf(“\n”)

{

for(j=1;j<=5;j++)

printf(“\* ”);

}

}

**Output:**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

**Example:**

main()

{

int i,j;

for(i=1; i<=5; i++; printf(“\n”))

{

for(j=1; j<=6-i; j++)

printf(“ ”);

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

printf(“\* “);

}

for(i=1; i<=5; i++, printf(“\n”))

{

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

printf(“ ”);

for (j=1; j<=6-i; j++)

printf(“\* “);

}

}

**Output:**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Write a program to print pascal triangle:**

main()

{

int i,j,b,n;

printf(“enter n”);

scanf(“%d”,&n);

for(i=0; i<=n; i++,printf(“\n”))

{

for(j=0; j<=n+1-i; j++)

printf(“ “);

for(j=0; j<=i; j++)

{

if(j = =0)

b=1;

else

b=b\*(i-j+1)/j;

printf(“%d ”,b);

}

}

}

**Output:**

**Enter a number 4**

1

1. 1

1 2 1

1 3 3 1

**Example:**

main()

{

int i,j;

for(i=1; i<=5; i++,printf(“\n”))

{

for(j=1; j<=6-i; j++)

printf(“\*”);

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

printf (“ “);

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

printf (“ “);

for (j=1; j<=6-i; j++)

printf (“ \*”);

}

for(i=1; i<=5; i++, printf(“\n”))

{

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

printf(“ \*”);

for(j=1; j<=6-i; j++)

printf(“ “);

for(j=1; j<=6-i; j++)

printf(“ “);

for(j=1; j<=-i; j++)

printf(“ \*”);

}

}

**Series programs**

**(\*) 1+x/1!+x2/2!+...........=ex**

main( )

{

double x,pro,fa,sum;

int i,j,k,n;

printf("enter n value");

scanf("%lf",&n);

printf("enter n value");

scanf("%d",&n);

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

{

for(pro=1;fa=1;k=1;k<=j;k++)

{

pro=pro\*x;

fa=fa\*x;

}

sum=sum+pro/fa;

}

printf("%lf",Sum);

}

|  |
| --- |
| (\*) 1+x2/2!+x4/4!+x6/6!+………+x2n/2n!  for(sum=1,i=1,j=2;i<n;i++,j+=2) |
| (\*) 1+x/1!+x3/3!+x5/5!+---------+x2n-1/(2n-1)!  For(sum=1,i=1,j=1;i<n;i++,j+=2) |
| (\*) x/1!+x3/3!+x5/5!+------+x2n-1/(2n-1)!  For(sum=0,i=1,j=1;i<=n;i++,j+=2) |

**(\*) 1-x/1!+x2/2!-x3/3!+---------**

main( )

{

double x,pro,fa,sum;

int i,j,k,n;

printf("enter n value");

scanf("%lf",&n);

printf("enter n value");

scanf("%d",&n);

for(sum=1,i=1,j=1;i<n;i++,j++)

{

for(pro=1,fa=1,k=1;k<=j;k++)

{

pro=pro\*n;

fa=fa\*k;

}

if(i%2!=0)

{

pro=-pro;

}

sum=sum+pro/fa;

}

printf(“%lf”,sum);

}

|  |
| --- |
| **(\*) 1-x2/2!+x4/4!-x6/6!+x8/8!----------+x2n/2n!**  for(sum=1;i=1;j=2;i<n;i++;j+=2)  {  for(pro=1;fa=1;k=1;k<=j;k++)  {  pro=pro\*x;  fa=fa\*k;  }  if(i%2!=0)  {  pro=-pro;  }  sum=sum+pro/fa;  } |
| **(\*) 1-x/1!+x3/3!-x5/5!+x7/7!-------------+x2n-1/(2n-1)!**  for(sum=1;i=1;j=1;i<n;i++;j+=2)  {  for(pro=1;fa=1;k=1;k<=j;k++)  {  pro=pro\*n;  fa=fa\*k;  }  if(i%2!=0)  {  pro=-pro;  }  sum=sum+pro/fa;  } |
| **(\*) x-x3/3!+x5/5!-x7/7!+---------+x(2n-1)!/(2n-1)!**  for(sum=0;i=1;j=1;i<=n;i++;j+=2)  {  for(pro=1;fa=1;k=1;k<0;k++)  {  pro=pro\*n;  fa=fa\*k;  }  if(i%2!=0)  {  pro=-pro;  }  sum=sum+pro/fa;  } |

**5.Jump Control Statements**

**a.break**

**b.exit( )**

**c.continue**

**d.goto**

**e.return**

**(a)break:**

We use this statement either in loop or switch.Whenever this statement is executed control comes out of the loop or switch.

|  |  |
| --- | --- |
| main( )  {  int i;  i=1;  while(i<=10)  {  if(i==5)  {  break;  }  printf(“%d”,i);  i++;  }  printf(“smile costs nothing”);  }  **Output:1 2 3 4 5 smile costs nothing** | main( )  {  int i,j;  i=1;  while(i<=5)  {  j=1;  while(j<=5)  {  if(j==3)  {  break;  }  printf(“%d d%”,i,j);  j++;  }  i++;  }  printf(“smile cost nothing”);  }  **Output:11 12 13 14 15 21 22 23 24 25 41 42 43 44 45 51 52 53 54 55 smile cost nothing** |

**(b) exit():**

We use this statement anywhere in the program.Whenever this statement is executed no other statement in the program is executed and the control goes to end of the program.

|  |  |
| --- | --- |
| (\*) # include<Stdlib.h>  main()  {  int i;  i=1;  while(i<=10)  {  if(i==5)  {  exit(1);  }  printf(“%d”,i);  i++;  }  printf(“smile cost nothing”);  }  **Output:1 2 3 4 5** | (\*) # include<Stdlib.h>  main()  {  int i,j;  i=1;  while(i<=5)  {  j=1;  while(j<=5)  {  if(j==3)  {  exit(1);  }  printf(“%d %d”,i,j);  j++;  }  i++;  }  printf(“smile costs nothing”);  }  **Output**:**11 12 13 14 15 21 22 23 24 25** |

**(c) Continue:**

We can write this statement only in loop.Whenever this is executed control goes back to loop.

|  |  |
| --- | --- |
| main( )  {  int i;  for(i=1;i<=10;i++)  {  if(i==5|| i==6)  {  continue;  }  printf(“%d”,i);  }  }  **Output:123478910** | main( )  {  int i;  for(i=65;i<=122;i++)  {  if(i>90 && i<97)  {  continue;  }  printf(“%c”,i);  }  }  **Output: ABC……..Z abc…………z** |

**(d) goto:**

We use this statement anywhere in the program.There are two types of goto statements.

**goto label;**

This is called unconditional.

**if(con)**

**{**

**goto label;**

**}**

This is conditional.

|  |  |
| --- | --- |
| main( )  {  int i,n,fa=1;  printf(“enter n”);  scanf(“%d”,&n);  i=1;  tech:  fa=fa\*1;  i++;  if(i<=n)  {  goto tech;  }  printf(“%d”,fa);  } | main( )  {  int n,m,s=0;  printf(“enter n”);  scanf(“%d”,&n);  tech:  m=n%10;  s=s\*m;  n=n/10;  if(n!=10)  {  goto tech;  }  printf(“%d”,s);  } |
| main()  {  int n,m,s=0;  printf(“enter n”);  scanf(“%d”,&n);  tech:  m=n%10;  s=s+m;  n=n/10;  if(n!=0)  {  goto tech;  }  printf(“%d”,s);  } | main()  {  int i,n;  printf(“enter n”);  scanf(“%d”,&n);  i=1;  tech:  printf(“%d”,i);  i++;  if(i<=n)  {  goto tech;  }  } |
| main()  {  int i,n,s=0;  printf(“enter f”);  scanf(“%d”,&n);  i=1;  tech:  s=s+i;  i++;  if(i<=n)  goto tech;  printf(“%d”,s);  } | main( )  {  int n,m,s=0,n1;  printf(“enter n”);  scanf(“%d”,&n);  n1=n;  tech:  m=n%10;  s=s\*10+m;  n=n/10;  if(n!=0)  goto tech:  if(n1==s)  printf(“pal”);  else  printf(“not pal”);  } |
| main()  {  int n,m,s=0,n1;  printf(“enter n”);  scanf(“%d”,&n);  n1=n;  tech:  m=n%10;  s=s+m\*m\*m;  n=n/10;  if(n!=0)  goto tech;  if(n1==s)  printf(“Amstrong”);  else  prinf(“Not Amstrong”);  } | main()  {  int a,b;  printf(“enter a,b”);  scanf(“%d %d”,&a,&b);  sar:  if(a>b)  a=a-b;  else  b=b-a;  if(a!=b)  goto sar;  printf(“%d”,a);  } |
| main()  {  int a,b,c,d,lcm;  printf(“enter a,b”);  scanf(“%d %d”,&a,&b);  c=a;  d=b;  xxx:  if(a>b)  a=a-b;  else  b=b-a;  if(a!=b)  goto xxx;  lcm=c\*d/a;  printf(“%d”,lcm);  } | main()  {  int i,n;  printf(“enter n”);  scanf(“%d”,&n);  xxx:  printf(“%d %d=%d\n”,n,i,n\*i);  i++;  if(i<=n)  goto xxx;  } |
| main()  {  int i,min=0,max=1,n;  printf(“enter n”);  scanf(“%d”,&n);  i=1;  tech:  printf(“%d”,min);  sum=min+max;  max=sum;  min=max;  if(i<=n)  goto tech;  } | main()  {  int i,n,c=0;  printf(“enter n”);  scanf(“%d”,&n);  i=1;  xxx:  if(n%i==0)  c++;  goto xxx;  if(c==2)  printf(“prime”);  else  printf(“not prime”);  } |

**Arrays**

* An array is a collection of similar data types.
* Array elements are always stored in continous memory locations.
* If n is size of an array its subscript starts from ‘0’ and ends with ‘n-1’.
* An array is also called subscripted variable.
* Before using an array its size and type must be declared.

**Declaration of an array:**

datatype arrayname[size] =values

**\*Initializing into an array:**

|  |  |
| --- | --- |
| main()  {  int a[5]={1,2,3,4,5};  printf(“%d %d %d %d %d”,a[0],a[1],a[2],a[3],a[4]);  }  **Output: 1 2 3 4 5** | main()  {  int a[5]={1,2,3};  printf(“%d%d%d%d%d”,a[0],a[1],a[2],a[3],a[4]);  }  **Output: 1 2 3 0 0** |
| main()  {  int a[]={1,2,3};  printf(“%d %d %d”,a[0],a[1],a[2]);  }  **Output:1 2 3** | main()  {  int a[5]={1,2,3,4,5,6}; **//incorrect**  } |

**Assignment:**

main()

{

int a[5];

a[0]=1;

a[1]=2;

a[2]=3;

a[3]=4;

a[4]=5;

printf("%d%d%d%d%d",a[0],a[1],a[2],a[3],a[4]);

}

**Reading values using scanf:**

main()

{

int a[5];

printf("enter 5 values");

scanf("%d%d%d%d%d",&a[0],&a[1],&a[2],&a[3],&a[4]);

printf("%d-%d",a[0]-a[4]);

}

main()

{

int a[5];

printf("%d\t %d\t %d\t %d\t %d",sizeof(a[0]),sizeof(a[1]),sizeof(a[2]),sizeof(a[3]),sizeof(a[4]));

printf("\n%d",,sizeof(a));

printf("\n%u\t %u\t %u\t %u\t %u",&a[0],&a[1],&a[2],&a[3],&a[4]);

}

**Output: 2 2 2 2 2**

**10**

**2000 2002 2004 2006 2008**

**Differences between ordinary variables and arrays**

|  |  |
| --- | --- |
| **Ordinary variables** | **Arrays** |
| \* Ordinary variables are not stored in continous memory locations. | Arrays elements are always stored in continous memory locations |
| Ordinary variables do not have subscripts | Array elements have subscript. |
| We can copy two ordinary variables directly. | We cannot copy two arrays directly. |
| We can compare two ordinary directly | We cannot compare two arrays directly. |
| We can perform any arithmetic operation on two ordinary variables directly | We cannot perform any arithmatic operation on two arrays directly. |
| Memory allocation is difficult. | Memory allocation is easy |
| Ordinary variables are not as flexible as arrays. | Arrays are more flexible than ordinary variables. |

|  |  |
| --- | --- |
| **Write a program to print array elements:**  main()  {  int a[50],i,n;  printf("enter how many elements you have");  scanf("%d”,&n);  printf("enter %d elements",n);  for(i=0;i<n;i++)  scanf("%d",&a[i]);  for(i=0;i<n;i++)  scanf("%d",a[i]);  } | **Write a program to find even or odd in given n numbers**  main()  {  int a[20],i,n;  printf("enter hmeyh");  scanf("%d",&n);  printf("enter %d elements",n);  for(i=0;i<n;i++)  scanf("%d",&a[i]);  for(i=0;i<n;i++)  {  if(a[i]%2==0)  {  printf("%d is even\n",a[i]);  }  else  {  printf("%d is odd \n",a[i]);  }  }  } |
| **Write a program to find how many even and odd numbers and their sum individually in given 'n' numbers**  main()  {  int a[20],i,n,ec=0,oc=0,es=0,os=0;  printf("enter hmeyh");  scanf("%d",&n);  printf("enter %d elements",n);  for(i=0;i<n;i++)  scanf("%d",&a[i]);  for(i=0;i<n;i++)  {  if(a[i]%2==0)  {  ec++;  es=es+a[i];  }  else  {  oc++;  os=os+a[i];  }  }  printf("ec=%d,oc=%d,es=%d,os=%d,ec,oc,es,os");  } | **Write a program to find average of n subject marks**  main()  {  int s[20],i,n,sum=0;  printf("enter hmsyh");  scanf("%d,&n");  printf("enter %d subject marks",n);  for(i=0;i<n;i++)  sum=sum+s[i];  avg=(float)sum/n;  printf("avg=%f",avg);  } |
| **Write a program to find the smallest and biggest number in given n numbers**  main()  {  int a[20],i,n,big,small;  printf("enter hmeyh");  scanf("%d",&n);  printf("enter %d elements",n);  for(i=0;i<n;i++)  scanf("%d",&a[i]);  big=a[0];  small=a[0];  for(i=1;i<n;i++)  {  if(a[i]>big)  {  big=a[i];  }  if(a[i]<small)  {  small=a[i];  }  }  printf("big=%d,small=%d,big,small");  } | **Write a program to n number of multiplication tables:**  main()  {  int a[20],i,n,j;  printf("enter hmeyh");  scanf("%d",&n);  printf("enter %d elements",n);  for(i=0;i<n;i++)  scanf("%d",&a[i]);  for(i=0;i<=n;i++)  {  for(j=1;j<=10;j++)  {  printf("%d\*%d=%d\n",a[i],j,a[i]\*j);  }  }  } |

**Strings**

* A string is a collection of characters

main()

{

char a[4];

a[0]=’s’;

a[1]=’r’;

a[2]=’e’;

a[3]=’e’;

printf("%c%c%c%c",a[0],a[1],a[2],a[3]);

}

**Output: sree**

|  |  |
| --- | --- |
| **Reading characters from keyboard:**  main()  {  char a[4];  printf("enter 4 characters");  scanf("%c%c%c%c", &a[0],&a[1],&a[2],&a[3]);  printf("%c%c%c%c", a[0],a[1],a[2],a[3]);  } | **Intializing characters into an array:**  main()  {  char a[4]={'s','r','e','e'};  printf("%c%c%c%c", a[0],a[1],a[2],a[3]);  } |

**Initializing a string into array:**

|  |  |  |
| --- | --- | --- |
| main()  {  char a[12]="techgenius";  printf("%s",a);  }  **Output: techgenius** | main()  {  char a[10]="vissu123";  printf("%s",a);  }  **Output: vissu123** | main()  {  char a[10]="sree2.5";  printf("%s",a);  }  **Output: sree2.5** |

**Note:**If you want to store n characters string we need n+1 memory locations.

A string always terminates with ‘\0’ (null) characters.

|  |  |
| --- | --- |
| main()  {  char a[10]="sreedhar";  printf("%d",sizeof(a));  }  **Output: 10** | **Program: reading and printing a string**  main()  {  char a[10];  printf("enter a string");  scanf("%s",a);  printf("%s",a);  } |
| **Counting number of characters in a given string (or) string length**  main()  {  char a[10];  int i;  printf("enter a string");  scanf("%s",a);  i=0;  while(a[i]!='\0')  {  i++;  }  printf("%d",i);  } | **Copying one string to another. (Or) string copy**  main()  {  char a[20],b[20];  int i;  printf("enter a string");  scanf("%s",a);  for(i=0;a[i]!=’\0’;i++)  {  b[i]=a[i];  }  b[i]='\0';  printf("%s",b);  } |
| **Reverse a given string in an array (or) string reverse**  main()  {  char a[20],b[20];  int i,j;  printf("enter a string");  scanf("%s",a);  i=0;  while(a[i]!='\0')  {  i++;  }  for(i=i-1,j=0;j>=0;i--,j++)  {  b[j]=a[i];  }  b[j]='\0';  printf("%s",b);  } | **Comparing two strings: (or) string comparison**  main()  {  char a[40],b[20];  int i;  printf("enter two strings");  scanf("%s%s",a,b);  i=0;  while(a[i]=b[i] && a[i]!='\0' && b[i]!='\0')  {  i++;  }  if(a[i]!=b[i])  printf("not equal");  else  printf("equal");  } |

**Concatenation of two strings: (or) string concatination**

|  |  |
| --- | --- |
| **First method:**  main()  {  char a[20],b[20],c[40];  int i,j,k;  printf("enter a string");  scanf("%s%s",a,b);  for(i=0;a[i]!='\0';i++)  c[i]=a[i];  for(j=0;b[j]!='\0';j++)  c[i+j]=b[j];  c[i+j]='\0';  printf("%s",c);  } | **Second method:**  main()  {  char a[40],b[20];  int i,j;  printf("enter two strings");  scanf("%s%s",a,b);  i=0;  while(a[i]!='\0')  {  i++;  }  for(j=0;b[j]!='\0';i++,j++)  {  a[i]=b[j];  }  a[i]='\0';  printf("%s",a);  } |

|  |  |
| --- | --- |
| **To check whether the given string is palindrome or not**  main()  {  char a[20],b[20];  int i,j;  printf("enter a string");  scanf("%s",a);  i=0;  while(a[i]!='\0')  {  i++;  }  for(i=i-1,j=0;i>=0;i--,j++)  {  b[j]=a[i];  }  b[j]='\0';  i=0;  while(a[i]==b[i] && a[i]!='\0' && b[i]!='\0')  {  i++;  }  if(a[i]!=b[i])  printf("not pal");  else  printf("pal");  } | **Insert a sub string into main string:**  main()  {  char a[20],b[20],c[40];  int i,j,k,m;  printf("enter a string");  scanf("%s",a);  printf("enter sub string");  scanf("%s",b);  printf("enter position to insert string");  scanf("%d",&m);  for(i=0,k=0;i<=m-2;i++,k++)  {  c[k]=a[i];  }  for(j=0;b[j]!='\0';j++,k++)  {  c[k]=b[j];  }  while(a[i]!='\0')  {  c[k]=a[i];  i++,k++;  }  c[k]='\0';  printf("%s",c);  } |

**Deleting n characters from a given string:**

|  |  |
| --- | --- |
| **First method:**  main()  {  char a[20],b[20];  int i,j,m,n;  printf("enter a string");  scanf("%s",a);  printf("enter no.of chars to delete");  scanf("%d",&n);  printf("enter starting position to delete");  scanf("%d",&m);  for(i=0,j=0;i<=m-2;i++,j++)  {  b[j]=a[i];  }  for(i=m+n-1;a[i]!='\0';j++)  {  b[j]=a[i];  }  b[j]='\0';  printf("%s",b);  } | **Second method:**  main()  {  char a[20];  int i,m,n;  printf("enter a string");  scanf("%s",a);  printf("enter no.of characters to delete");  scanf("%d",&n);  printf("enter position");  scanf("%d",&m);  for(i=0;i<=m-2;i++)  a[i]=a[i];  for(i=m;a[i]!='\0';i++)  a[i]=a[i+n];  printf("%s",a);  } |

**Inserting a substring into main string:**

main()

{

int a[20],b[20];

int i,j,m,n;

printf("enter a string");

scanf("%s",a);

printf("enter a substring to insert");

scanf("%s",b);

printf("enter starting position to insert");

scanf("%d",&m);

j=0;

while(b[j]!=’\0’)

{

j++;

}

n=j;

for(i=0;a[i]!=’\0’;i++);

while(i>=m-1)

{

a[i+n]=a[i];

i--;

}

for(i=i+1,j=0;b[j]!=’\0’;i++,j++)

a[i]=b[j];

printf(“%s”,a);

}

**Differences between formatted and unformatted string reading function:**

|  |  |
| --- | --- |
| **Formatted** | **Unformatted** |
| scanf is a formatted string reading function | gets is an unformatted string reading function. |
| scanf gets terminated when either space or enter key is encountered. | gets is terminated only for enter key. |
| We can read any no.of strings by only one statement. | We can read only one string. |
| scanf prototype declaration is available in <stdio.h> | gets prototype declaration is available in <conio.h> |

|  |  |
| --- | --- |
| main()  {  char a[20];  printf("enter a string");  gets(a);  printf("%s",a);  } | main()  {  char a[20];  puts("enter a string");  gets(a);  puts(a);  } |

**Differences between formatted and unformatted output functions:**

|  |  |
| --- | --- |
| **Formatted** | **Unformatted** |
| printf is a formatted output function. | puts is a unformatted output function. |
| We can print any no.of strings by only one statement. | we can print only one string by one statement |
| printf prototype declaration is available in <stdio.h> | puts prototype declaration is available in <conio.h> |

**String Functions:**

we should start the program with #include <string.h>\*.strlen()

**strlen()**

**strrev()**

**strcpy()**

**strncpy()**

**strcat()**

**strncat()**

**strcmp()**

**stricmp()**

**strcmpi()**

|  |  |
| --- | --- |
| **String Length:**  #include<string.h>  main()  {  int n;  char a[20];  puts("enter a string");  gets(n);  n=strlen(a);  puts(n);  } | #include<string.h>  main()  {  int n;  n=strlen("sree");  printf("%d",n);  } |
| **String Reverse:**  #include<string.h>  main()  {  char a[20];  puts("enter a string");  gets(a);  strrev(a);  puts(a);  } | **String Concatenation:**  #include<string.h>  main()  {  char a[40],b[20];  puts("enter two strings");  gets(a);  gets(b);  strcat(a,b);  puts(a);  } |
| #include<string.h>  main()  {  char a[20];  strcat(a,"sree");  strcat(a,"dhar");  puts(a);  } | **String Copy:**  #include<string.h>  main()  {  char a[20],b[20];  puts("enter a string");  gets(a);  strcpy(b,a);  puts(b);  } |
| #include<string.h>  main()  {  char a[20]="sree";  char b[20], c[20];  strcpy(b,"Techgenius");  strcpy(c,a);  puts(b);  puts(c);  } |  |

**Strcmp:**

It compares two strings.If both are equal it gives '0', otherwise it gives difference between their ASCII codes or nonzero values.

|  |  |
| --- | --- |
| **Comparing two strings:**  #include<string.h>  main()  {  char a[20],b[20];  puts("enter two strings");  gets(a);  gets(b);  n=strcmp(a,b);  if(n==0)  printf("equal");  else  printf("not equal");  } | **Given string is palindrome or not**  #include<string.h>  main()  {  char a[20],b[20];  int n;  puts("enter a string");  gets(a);  strcpy(b,a);  strrev(b);  n=strcmp(a,b);  if(n==0)  printf("Pal");  else  printf("not pal");  } |

**Character handling functions:**

#include<ctype.h> is Starting statement:

**isalpha()**

**isdigit()**

**isalnum()**

**islower()**

**isupper()**

**ispunct()**

**isspace()**

**isaascii()**

**tolower()**

**toupper()**

**toaascii()**

**example:**

|  |  |
| --- | --- |
| #include<ctype.h>  main()  {  char ch;  printf("enter a char");  scanf("%c",&ch);  if(isalpha(ch))  printf("alpha");  else if(isdigit(ch))  printf("digit");  else  printf("special char");  } | **write a program to convert lower case letter to upper case and upper case letter to lower case**  #include<ctype.h>  main()  {  char ch;  printf("enter a char");  scanf("%c",&ch);  if(isalpha(ch))  {  if(islower(ch))  ch=toupper(ch));  else  ch=tolower(ch));  }  printf("%c",ch);  } |

**counting no.of characters, vowels, consonants, digits,tabs, special characters, words in a given string.**

main()

{

char a[50];

int i,nch=0,nv=0,nc=0,nd=0,nt=0,nsch=0,nw=1;

printf("enter a string");

gets(a);

for(i=0;a[i]!='\0';i++)

{

if(isalpha(a[i]))

{

a[i]=tolower(a[i]);

if(isalpha(a[i]))

{

a[i]=tolower(a[i]);

if(a[i]='a'||a[i]='e'||a[i]='i'||a[i]='o'||a[i]='u')

{

nch++;

nv++;

}

else

{

nch++;

nc++;

}

}

else if(isdigit a[i]))

{

nch++;

nd++;

}

else if(a[i]=='\t')

{

nch++;

nsch++;

nt++;

nw++;

}

else

{

nch++;

nsch++;

}

printf("nch=%d,nv=%d,nc=%d,nd=%d,nsch=%d,nw=%d",nch,nv,nc,nd,nsch,nw);

}

}

}

**Double Dimensional Array:**

main()

{

int a[3][3];

printf("%d\t%d\t%d",sizeof(a[0]),sizeof(a[1]),sizeof(a[2]));

printf("%d",sizeof(a));

}

**Output: 6 6 6**

**18**

**Initializing values into double dimensional array:**

main()

{

int a[3][3]={{1,2,3},{4,5,6},{7,8,9}}; // int a[3][3]={1,2,3,4,5,6,7,8,9};

a[0][0]=1;

a[0][1]=2;

a[0][2]=3;

a[1][0]=4;

a[1][1]=5;

a[1][2]=6;

a[2][0]=7;

a[2][1]=8;

a[2][2]=9;

printf("enter a value");

scanf("%d %d %d %d %d %d %d %d %d”, &a[0][0],&a[0][1],&a[0][2],&a[1][0],&a[1][1],&a[1][2],&a[2][0],&a[2][1],&a[2][2]);

printf("%d %d %d %d %d %d %d %d %d”,a[0][0],a[0][1],a[0][2],a[1][0],a[1][1],a[1][2],a[2][0],a[2][1],a[2][2]);

}

**Note:in double dimensional arrays column size is compulsory:**

|  |  |  |  |
| --- | --- | --- | --- |
| main()  {  int a[][3]={1,2,3,4,5,6,7};  printf("%d",sizeof(a));  } | main()  {  int a[2][1]={1,2,3,4}; X  } | | main()  {  int a[][]={1,2,3,4}; X  } |
| **Write a program to print the values in matrix format:**  main()  {  int a[5][5],i,j,R,C;  printf("Enter Rows & Columns of matrix");  scanf("%d%d",&R,&C);  printf("enter %d\*%d mat A",R,C);  for(i=0;i<R;i++)  for(j=0;j<C;j++)  scanf("%d",&a[i][j]);  for(i=0;i<R;i++),printf("\n"))  for(j=0;j<C;j++)  printf("%d",a[i][j]);  } | | **Transpose of a given matrix:**  main()  {  int a[5][5],b[5][5],R,C,i,j;  printf("enter R & C of mat A");  scanf("%d%d",&R,&C);  printf("enter %d\*%d matrix A",R,C);  for(i=0;i<R;i++)  for(j=0;j<C;j++)  scanf("%d",&a[i][j]);  for(i=0;i<R;i++)  for(j=0;j<C;j++)  b[j][i]=a[i][j];  for(i=0;i<C;i++,printf("\n")  for(j=0;j<R;j++)  printf("%d",b[i][j]);  } | |
| **Addition of Matrices:**  main()  {  int a[5][5],b[5][5],c[5][5],R1,C1,R2,C2,i,j;  printf("enter rows & columns of Mat A");  scanf("%d%d",&R1,&C1);  if(R1==R2 && C1==C2)  {  printf("enter %d \* %d mat A",R1,C1);  for(i=0;i<R1;i++)  for(j=0;j<C1;j++)  scanf("%d",&a[i][j]);  printf("enter %d \* %d mat B",R2,C2);  for(i=0;i<R2;i++)  for(j=0;j<C2;j++)  scanf("%d",&b[i][j]);  for(i=0;i<R1;i++)  for(j=0;j<C1;j++)  c[i][j]=a[i][j]+b[i][j];  for(i=0;i<R1;i++;printf("\n"))  for(j=0;j<C1;j++)  printf("%d",c[i][j]);  }  else  {  printf("not possible");  }  } | | **Multiplication of Two Matrices:**  main()  {  int a[5][5],b[5][5],c[5][5],R1,C1,R2,C2,i,j,k;  printf("Enter R and C of mat A");  scanf("%d%d",&R1,&C1);  printf("Enter R and C of mat B");  scanf("%d%d",&R2,&C2);  if(C1==R2)  {  printf("enter %d\*%d mat A",R1,C1);  for(i=0;i<R1;i++)  for(j=0;j<C1;j++)  scanf("%d",&a[i][j]);  printf("enter %d\*%d mat B",R2,C2);  for(i=0;i<R2;i++)  for(j=0;j<C2;j++)  for(c[i][j]=0,k=0;k<R2;k++)  c[i][j]=c[i][j]+a[i][k]\*b[k][j];  for(i=0;i<R1;i++,printf("\n"))  for(j=0;j<C2;j++)  printf("%d",c[i][j]);  }  else  {  printf("not possible");  }  } | |
| **symmetric or not:**  #include<stdlib.h>  main()  {  int a[5][5],b[5][5],r,c,i,j;  printf("enter r&c of mat A");  scanf("%d%d",&r,&c);  if(r==c)  {  printf("enter %d\*%d mat A",r,c);  for(i=0;i<r;i++)  for(j=0;j<c;i++)  scanf("%d",&a[i][j]);  for(i=0;i<r;i++)  for(j=0;j<c;j++)  b[j][i]=a[i][j];  for(i=0;i<r;i++)  for(j=0;j<c;j++)  {  if(a[i][j]!=b[i][j])  {  printf("not stmmetric");  exit(1);  }  }  printf("symmetric");  }  else  {  printf("not possible");  }  } | | **Diagonal elements, lower triangle matrix, upper triangle matrix printing**  main()  {  int a[5][5],r,c,i,j,sum=0;  printf("enter r&c of mat A");  scanf("%d%d",&r,&c);  if(r==c)  {  printf("enter %d\*%d",mat A",r,c);  for(i=0;i<r;i++)  for(j=0;j<c;j++)  {  if(i>j)  {  printf("%d ",a[i][j]); //sum=sum+a[i][j])  }  else  {  printf(“ “);  }  }  //printf("%d",sum);  }  else  {  printf("not possible");  }  } | |

**Taking number of string and printing using double dimensional array:**

|  |  |
| --- | --- |
| main()  {  char a[5][20]={"sree",sllt","hai","hello","boy"}  printf(%s%s%s%s%s",a[0],a[1],a[2],a[3],a[4]);  } | main()  {  char a{10][20];  int i,n;  printf("enter how many strings you have");  scanf("%d",&n);  printf("enter %d strings",n);  for(i=0;i<n;i++)  gets(a[i]);  for(i=0;i<n;i++,printf("\n"))  puts(a[i]);  } |

**MULTIDIMENSIONAL ARRAYS:**

main()

{

int a[3][3][3];

}

**FUNCTIONS:**

**Functions introduction:**

* A function in c language is a block of code that performs a specific task. It has a name and it is reusable i.e. it can be executed from as many different parts in a c program as required. It also optionally returns a value to the calling program
* So function in a C program has some properties discussed below.
* Every function has a unique name. This name is used to call function from “main()” function. A function can be called from within another function.
* A function is independent and it can perform its task without intervention from or interfering with other parts of the program.
* A function performs a specific task. A task is a distinct job that your program must perform as a part of its overall operation, such as adding two or more integer, sorting an array into numerical order, or calculating a cube root etc.
* A function returns a value to the calling program. This is optional and depends upon the task your function is going to accomplish. Suppose you want to just show few lines through function then it is not necessary to return a value. But if you are calculating area of rectangle and wanted to use result somewhere in program then you have to send back (return) value to the calling function.
* . A function can call other functions & also itself
* Function can call other function.
* Function can call itself , which is called as “recursive” function.
* Recursive functions are also useful in order to write system functions
* C language is collection of various library functions. If you have written a program in C then it is evident that you have used C’s inbuilt functions.printf,scanf,clrscr etc are C’s inbuilt functions. You cannot imagine a C program without function.

A general form of a C function looks like this:

**<return type> FunctionName (Type Argument1,Type Argument2, Type Argument3……)**  
**{**  
**Statement1;**  
**Statement2;**  
**Statement3;**  
**}**

**An example of function.**

int sum (int x, int y)  
{  
 int result;  
 result = x + y;  
 return (result);  
}

**example1:**

main()

{

bike();

}

bike()

{

body

}

paris()

{

printf("i am in paris\n");

}

**Example2:**

main()

{

printf("i am in main\n");

japan();

printf(" i am in main\n");

singapore();

printf("i am back to main\n");

paris();

printf("i am back to main\n");

}

singapore()

{

printf("i am in singapore\n");

}

japan()

{

printf("i am in japan\n");

}

**Example3:**

main()

{

printf(" i am in main\n");

japan();

printf("i am back to main\n");

}

japan()

{

printf("i am in japan\n");

paris()

printf("i am back to japan\n");

}

paris()

{

printf("i am in paris\n");

usa();

printf("i am back to paris\n");

}

usa()

{

printf("i am in usa\n");

}

* We may have any no.of functions.
* We may not define functions as the sequence called in the main function.
* We can define functions alone main also due the program exhecution starts from main only.
* A function can be called any no. of times.
* Any function can be called by any other function,even main function can be called by anyother

function.

* A function can be called by itself,is called recursion.

**Functions are divided into 2 type:**

(1)system defined functions

(2)user defined functions

**(1)system defined functions:**If the defination of the function is given by system itself is called sdf.

**(2)user defined functions:**If the defination of the function is given by user is called udf.

**NOTE:**

main is a user derfined function called by the compiler.

Because main function definition will be written by the user

**Types of user defined functions:**

without argument without return.

with argument without return.

without argument with return.

with argument with return.

**STORAGE CLASSES:**

(i)Auto

(ii)register

(iii)Static

(iv)Extern

**(i)Auto:**

storage location-------------------ram

Default initiala value-------------garbage

Scope---------------------------------local to block( { } )

Life time-----------------------------control with in the block

main()

{

int a=7;

{

int a=6;

{

int a=5;

printf("%d",a);

}

printf("%d",a);

}

printf("%d",a);

}

**(ii)Register:**

storage location---------------cpu register

Default initisl value-----------garbage

Scope-----------------------------local to block

Life time-------------------------control with in thw block

* we should apply bregister storage class on **ints and chars** only.
* we should not apply register storage class on long,float,double and long double if at all we apply

then the compiler will not show any error but it converts directly into auto and allots memory in

ram.

* eventhrough we apply on ints and chars we may not tell that those are stored in cpu registers.
* register storage class is used on frequently used loop counters.
* we should not apply address operator in register storage class.

**(iii)Static:**

storage location----------------ram

Default initial value------------0

Scope------------------------------local to block

life time----------------------------end of the program

main()

{

incr();

incr();

incr();

}

incr()

{

auto int i=1;

printf("%d",i);

i++;

}

**Output: 1 1 1**

incr()

{

static int i=1;

printf("%d",i)

i++;

} \*/

**Output: 1 2 3**

**(iv)Extern:**

storage location-------------------ram

default initial value---------------zero

scope---------------------------------global

life time------------------------------end of the program

int a=5,b=6,c;

main()

{

add();

}

add()

{

c=a+b;

printf("%d",c);

}

1. **WITHOUT ARGUMENT WITHPUT RETURN:**

|  |  |
| --- | --- |
| **Adding two numbers by using without argument and without return**  void add(void);  main()  {  add();  }  void add(void)  {  int a,b,c;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a+b;  printf("%d",c);  } | **Printing n natural numbers**  void num(void);  main()  {  num();  }  void num(void)  {  int i,n;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  printf("%d",i);  } |
| **Factorial of a given number**  void sum(void);  main()  {  sum();  }  void sum(void)  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  for(i=1;i<n;i++)  s=s+i;  printf("%d",s);  } | **Prime number**  void prime(void);  main()  {  prime();  }  void prime(void)  {  int i,n,c=0;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  if(n%i==0)  {  c++;  }  if(c==2)  printf("prime");  else  printf("not prime");  } |
| **Palindrome or not:**  void pal(void);  main()  {  pal();  }  void pal(void)  {  int n,n1,m,s=0;  printf("enter n");  scanf("%d",&n);  n1=n;  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  if (n1==s)  printf("pal");  else  printf(“not palindrome”);  } | **Reverse a given number:**  void rev(void);  main()  {  sum();  }  void sum(void)  {  int n,m.s=0;  printf("enter n");  scanf("%d",&n);  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  printf("%d",s);  } |
| **Armstrong or not:**  void ams(void);  main()  {  ams();  }  void ams(void)  {  int n,n1,m,s=0;  printf("enter n");  scanf("%d",&n);  n1=n;  while(n!=0)  {  m=n%10;  s=s+m\*m\*m;  n=n\10;  }  if(n1==s)  printf("amstrong");  else  printf("not amstrong");  } | **Gcd of two numbers:**  void gcd(void);  main()  {  gcd();  }  void gcd(void)  {  int a,b;  printf("enter a,b");  scanf("%d%d",&a,&b);  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  printf("%d",a);  } |
| **Lcm of two numbers**  void lcm(void);  main()  {  lcm();  }  void lcm(void)  {  int a,b,c,d,lcm;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a;  d=b;  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  lcm=c\*d/a;  printf("%d",lcm);  } | **Multiplication table printing:**  void table(void);  main()  {  table();  }  void table(void)  {  int i,n;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=10;i++)  printf("%d%d=%d\n",n,i,n\*i);  }  **Sum of the digits of a given number:**  void sum(void);  main()  {  sum();  }  void sum(void)  {  int n,m,s=0;  printf("enter n");  scanf("%d"&n);  while(n!=0)  {  m=n%10;  s=s+m;  n=n/10;  }  printf("%d",s);  } |

**(II)WITH ARGUMENT WITHOUT RETURN:**

|  |  |
| --- | --- |
| **Adding two numbers by using with argument without return**  #include <stdio.h>  void add(int a,int b);  void main(void)  {  int a,b;  printf("enter a,b);  scanf("%d%d",&a,&b);  add(a,b);  }  void add(int a,int b)  {  int c;  c=a+b;  printf("%d",c);  } | **N Natural numbers printing**  #include<stdio.h>  void num(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  num(n);  }  void num( int n)  {  int i;  for(i=1;i<=n;i++)  printf("%d",i);  } |
| **Sum of n natural numbers**  #include<stdio.h>  void sum(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  sum(n);  }  void sum(int n);  {  int i,s=0;  for(i=1;i<=n;i++)  s=s+i;  printf("%d",s);  } | **Sum of the digits of a given number:**  #include<stdio.h>  void sum(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  sum(n);  }  void sum(int n);  {  int n,s=0;  while(n!=0)  {  m=n%10;  s=s+m;  n=n/10;  }  printf("%d",s);  } |
| **Reverse a given number:**  #include<stdio.h>  void rev(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  rev(n);  }  void rev(int n)  {  int m,s=0;  while(n!=0)  {  m=n%10;  s=s\*10+m;  m=n/10;  }  printf("%d",s);  } | **Palindrom or not:**  #include<stdio.h>  void pal(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  pal(n);  }  void pal(int n)  {  int n1,s=0,m;  n1=n;  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  if(n1==s)  printf("pal");  else  printf("not pal");  } |
| **Armstrong number:**  #include<stdio.h>  void ams(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  ams(n);  }  void ams(void)  {  int n1,m,s=0;  n1=n;  while(n!=0)  {  m=n%10;  s=s+m\*m\*m;  n=n/10;  }  if(n1==s)  printf("amstrong");  else  printf("not armstrong");  } | **Multiplication table printing:**  #include<stdio.h>  void table(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  table(n);  }  void table(int n)  {  int i;  for(i=1;i<=10;i++)  printf("%d\*%d=%d\n",i,n,n\*i);  } |
| #include<stdio.h>  void fab(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  fab(n);  }  void fab(int n)  {  int i,min=0,max=1,sum;  for(i=1;i<=n;i++)  {  printf("%d",min);  sum=min+max;  min=max;  max=sum;  }  } | **Prime number or not:**  #include<stdio.h>  void prime(int n);  void main(void)  {  int n;  printf("enter n");  scanf("%d",&n);  prime(n);  }  void prime(int n)  {  int i,c=0;  for(i=1;i<=n;i++)  if(n%i==0)  {  c++;  }  if(c==2)  printf("prime");  else  printf("not prime");  } |
| **Gcd of two numbers:**  #include<stdio.h>  void gcd(int a,int b);  void main(void)  {  int a,b;  printf("enter a,b");  scanf("%d%d",&a,&b);  gcd(a,b);  }  void gcd(inta,int b)  {  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  } | **Lcm of two numbers:**  #include<stdio.h>  void lcm(int a, int b);  void main(void)  {  int a,b;  printf("enter a,b");  scanf("%d%d",&a,&b);  lcm(a,b);  }  void lcm(int a,int b)  {  int c,d,lcm;  c=a;d=b;  while (a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  lcm=c\*d/a;  printf("%d",lcm);  } |

**(III)WITHOUT ARGUMENT WITH RETURN:**

|  |  |
| --- | --- |
| int add (int a,int b);  void main()  {  int a,b,c;  printf("enter a,b");  scanf("%d%d',&a,&b);  c=add(a,b);  printf("%d",c);  }  int add(int a, int b)  {  int c;  c=a+b;  return c;  } | int rev(int n);  void main()  {  int n,s=0;  printf("enter n");  scanf("%d",&n);  s=rev(n);  print("%d",s);  }  int rev(int n)  {  int m,s=0;  while(n!=0)  {  m=n%10  s=s\*10+m;  n=n/10;  }  return s;  } |
| int fact (int n);  void main()  {  int n,fa;  printf("enter n");  scanf("%d",&n);  fa=fact(n);  printf("%d",fa);  }  int fact(int n)  {  int i,fa=1;  for(i=1;i<=n;i++)  fa=fa\*i;  return fa;  } | int gcd(int a,int b);  void main()  {  inta,b,g;  printf("enter a,b");  scanf("%d%d",&a,&b);  g=gcd(a,b);  print("%d",g);  }  int gcd(int a,int b)  {  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  return a;  } |
| int sum(int n);  void main()  {  int n,s;  printf("enter n");  scanf("%d",&n);  s=sum(n);  printf("%d",s);  }  int sum(int n)  {  int i,s=0;  for(i=1;i<=n;i++)  s=s+i;  return s;  } | int sum(int n);  void main()  {  int n,s;  printf("enter n");  scanf("%d",&n);  s=sum(n);  pirntf("%d",s);  }  int sum(int n)  {  int m,s=0;  while(n!=0)  {  m=n%10;  s=s+m;  n=n/10;  }  return s;  } |

**(IV)WITHOUT ARGUMENT WITH RETURN:**

|  |  |
| --- | --- |
| int add(void);  main()  {  int c;  c =add();  printf("%d",c);  }  int add(void)  {  int a,b,c;  printf("enter a,b");  scanf("%d%d",&a,&b);  c=a+b;  return c;  } | int fact(void)  main()  {  int fa;  fa=fact();  printf("%d",fa);  }  int fact(void)  {  int i,n,fa=1;  printf("enter n");  scanf("%d",&n);  for(i=1;i<n;i++)  fa=fa\*i;  return fa;  } |
| int rev(void);  main()  {  int s;  s=rev();  printf("%d",s);  }  int rev(void)  {  int n,m,s=0;  printf("enter n");  scanf("%d",&n);  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  return s;  } | int gcd(void);  main()  {  int g;  g=gcd();  printf("%d",g);  }  int gcd(void)  {  int a,b;  printf("enter a,b");  scanf("%d",&s,&b);  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  return a;  } |
| int sum(void);  main()  {  int s;  s=sum();  printf(""%d",s);  }  int sum(void)  {  int n,m,s=0;  printf("enter n");  while(n!=0)  {  m=n%10;  s=s+m;  n=n/10;  }  return s;  } | int num(void)  main()  {  int i;  i=num();  printf("%d",i);  }  int num(void)  {  int n,i;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  return i;  } |
| int sum (void);  main()  {  int s;  s=sum();  printf("%d",s);  }  int sum(void)  {  int i,n,s=0;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  s=s+i;  return s;  } | int pal(int n);  void main()  {  int n,s,n1;  printf("enter n");  scanf("%d",&n);ni=n;  s=pal(n);  if(ni==s)  printf("pal");  else  printf("not pal");  }  int pal(int n)  {  int m,s=0;  while(n!=0)  {  m=n%10;  s=s\*10+m;  n=n/10;  }  return s;  } |
| int ams(int n);  void main()  {  int n,n1,s;  printf("enter n");  scanf("%d",&n);  n1=n;  s=ams(n);  if(n1==s)  printf("amstrong");  else  printf("not amstrong");  }  int ams(int n)  {  while(n!=0)  {  m=n%10;  s=s+m\*m\*m;  n=n/10;  }  return s:  } | int LCM(int a,int b);  void main()  {  int a,b,lcm;  printf("enter a,b");  scanf("%d%d",&a,&b);  lcm=LCM(a,b);  printf("%d",lcm);  }  int LCM(int a,int b)  {  int c,d,lcm;  c=a;  d=b;  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  lcm=c\*d/a;  return lcm;  } |

**AREA OF RECTANGLE:**

|  |  |
| --- | --- |
| **i)without argument without return:-**  void area(void);  main()  {  area();  }  void area(void)  {  float l,b,a;  printf("enter l,b");  scanf("%f%f",&l,&b);  a=l\*b;  printf("%f",a);  } | **ii)with argument without return:-**  void area(float l;float b);  main()  {  float l,b;  printf("enter l,b");  scanf("%f%f",&l,&b);  area();  }  void area(float l,float b);  {  float a;  a=l\*b;  printf("%f",a);  } |
| **iii)with argument with return:-**  float area(float l,float b);  main()  {  float l,b,a;  printf("enter l,b");  scanf("%f%f",&l,&b);  a=area(l,b);  printf("%f",a);  }  float area(float l,float b)  {  float a;  a=l\*b;  return a;  } | **iv) without argument with return:-**  float area(void);  main()  {  float a;  a=area();  printf("%f",a);  }  float area(void)  {  float l,b,a;  printf("enter l,b");  scanf("%f%f",&l,&b);  a=l\*b;  return a;  } |

**Write AREA OF TRIANGLE program with functions:-**

|  |  |
| --- | --- |
| **i)without argument without return:-**  void area(void);  main()  {  area();  }  void area(void)  {  float b,h,a;  printf("enter b,h");  scanf("%f%f",&b,&h);  a=0.5\*b\*h;  printf("%f",a);  } | **ii)with argument without return:-**  void area(float b,float h);  main()  {  float b,h;  printf("enter b,h");  scanf("%f%f",&b,&h);  area(b,h);  }  void area(float b,float h)  {  float a;  a=0.5\*b\*h;  printf("%f",a);  } |
| **iii)with argument with return:-**  float area(float b,float h);  main()  {  float b,h,a;  printf("enter b,h");  scanf(%f%f",&b,&h);  a=area(b,h);  printf("%f",a);  }  float area(float b,float h)  {  float a;  a=0.5\*b\*h;  return a;  } | **iv)without argument with return:-**  float area(void);  main()  {  float a;  a=area();  printf("%f",a);  }  float area(void)  {  float b,h,a;  printf("enter b,h");  scanf("%f%f",&b,&h);  a=0.5\*b\*h;  return a;  } |

**Arguments:-**

* The sending arguments are said to be actual arguments or actual parameters.
* The receiving arguments are said to be formal arguments or formal parameters.
* we can send any no. of arguments,but the no. of sending arguments must be equal to no. of receiving arguments.
* An argument may be any expression it may be a function call also.

**Example:-**

void fun(int a,int b,int c,int d,float e,char f);

main()

{

int a=5,b=6;

fun(a , a+b , a\*b+2 , 25 , 5.2 , 'A');

}

void fun(int a,int b,int c,int d,float e,char f)

{

printf("%d%d%d%d%f%c",a,b,c,d,e,f);

}

**Example:-**

int add(int a,int b);

main()

{

int a=5,b=6,c=8,d;

d=add(add(a,b),add(b,c));

printf("%d",d);

}

int add(int a,int b)

{

int c;

c=a\*b;

return c;

}

**Return:-**

whenever this statement is executed control immediately goes back to calling function.

A return statement can be written in three ways.

i)return exp;

ii)return (exp);

iii)return;

The default return value of return statement is integer only.

The return value may be any expression.

The max no. of values can a return statement return is only one.

return (a+b)

return (a\*b)

return (5)

return (5\*2)

We can use any no. of return statementsin a given programme but only one return statement is

executed.

**Write a program to change lower case letter to upper case vice versa**

char change(char x);

main()

{

char x,y;

printf("enter an alphabet");

scanf("%c",&x);

y=change(x);

printf("%c",y);

}

char change(char x)

{

if(islower(x))

return toupper(x);

else

return tolower(x);

}

**Recursion:-**

A function which is called by itself is called recurssion.

|  |  |
| --- | --- |
| **Gce of two numbers with recursion :**  int gcd(intm a,int b);  main()  {  int a,b,g;  printf("enter a,b");  scanf("%d%d",&a,&b);  g=gcd(a,b);  printf("%d",g);  }  int gcd(int a,int b)  {  if(a==b)  return a;  elseif(a>b)  gcd(a-b,b);  else  gcd(a,b-a);  } | **Factorial of a given number:**  int fact(int n);  main()  {  int n,fa;  printf("enter n");  scanf("%d",&n);  fa=fact(n);  printf("%d",fa);  }  int fact(int n)  {  if(n==0;n==1)  return 1;  else  return n\*fact(n-1);  }  **Output:**4\*3\*2\*1 |
| **Febinocci series with recursion:**  int feb(int n);  main()  {  int i,n,f;  printf("enter n");  scanf("%d",&n);  for(i=1;i<=n;i++)  {  f=feb(i);  printf("%d",f);  }  }  int feb(int n)  {  if(n==1)  return 0;  elseif(n==2)  return 1;  else  return feb(n-1)+feb(n-2);  } | **Write a program to find ncr value:**  int fact(int n);  main()  {  int n,r,ncr;  printf("enter n,r");  scanf("%d%d",&n,&r);  ncr=fact(n)/(fact(n-r)\*fact(r));  printf("5d",ncr);  }  int fact(int n)  {  int i,fa=1;  for(i=1;i<=n;i++)  {  fa=fa\*i;  }  return fa;  } |

**what is a function?why do we use functions?(or) what are the advantages of functions?**

A function is a block of code that performs particular task.

**syntax:-** return type function name(type arg1,type arg....)

{

body

}

**Advantages of functions:-**

* program code is reduced.
* Repeatative works can be done very easily.
* Debugging(error detection) of the program is very easy.
* Complex problem is divided into small parts called modules or functions.
* The length of the source program can be reduced by using functions at appropriate places.
* We can divide c program in smaller modules.
* We can call module whenever require. e.g suppose we have written calculator program then we can write 4 modules (i.e add,sub,multiply,divide)
* Modular programming make c program more readable.
* Modules once created , can be re-used in other programs.
* Every C program starts from main function.
* Every function is  called directly or indirectly through main.
* Program development become easy
* Frequently used functions can be put together in the customized library
* We can put frequently used functions in our custom header file.
* After creating header file we can re use header file. We can include header file in other program
* It is easier to understand the Program topic
* We can get overall idea of the project just by reviewing function names.
* Program development made easy
* Program testing becomes easy Code sharing becomes possible
* Code re-usability increases
* Increases program readability
* Function facilitates procedural abstraction
* Functions facilitate the factoring of code

**Sending an array to a function:**

|  |  |
| --- | --- |
| void for(int b[]);  main()  {  int a[5]={1,2,3,4,5};  fun(a); printf("%d%d%d%d%d",a[0],a[1],a[2],a[3],a[4]];  }  void fun(int b[])  {  b[0]=b[0]+1;  b[1]=b[1]+1;  b[2]=b[2]+1;  b[3]=b[3]+1;  b[4]=b[4]+1;  } | void copy(int x[],int y[],int n);  main()  {  int a[20],b[20],i,n;  printf("enter n");  scanf("%d",&n);  printf("enter %d elements",n);  for(i=0;i<=n;i++)  scanf(%d",a[i]);  copy(a,b,n);  printf("%d",b[i]);  }  void copy (int n[],int y[],int n);  {  int i;  for(i=0;i<=n;i++)  y[i]=x[i];  } |
| void copy(char x[],char y[]);  main()  {  char a[20],b[20];  puts("enter a string");  gets(a);  copy(a,b);  puts(b);  }  void copy(char x[],char y[])  {  int i;  for(i=0;x[i]!='\0';i++)  {  y[i]=x[i];  }  y[i]='\0';  } |  |

**Sending double dimensional array to a function:**

void add(int x[5][5],int y[5][5],int z[5][5],int r,int c);

main()

{

int a[5][5],b[5][5],c[5][5],r1,c1,R2,c2,i,j;

printf("enter R and C of mat A");

scanf("%d%d",&R1,&C1);

printf("enter R and C of mat B");

scanf("%d%d",&R2,&c2);

if(R1==R2&&C1==C2)

{

printf("enter %d\*%d mat A",R1,C1);

for(i=0;i<R1;i++)

for(j=0;j<C1:j++)

scanf("%d",a[i][j]);

printf("enter %d\*%d mat B',R2,C2);

for(i=0;i<R;i++)

for(j=0;j<C!;j++)

scanf("%d",b[i][j]);

add(a,b,c,R1,C1);

for(i=0:i<R1;i++)

for(j=0;j<c1:j++)

printf("%d",c[i][j]);

}

else

{

printf(“not possible”);

}

}

void add(int x[5][5],int y[5][5],int z[5][5],int R,int c)

{

int i,j;

for(i=0;i<R;i++)

for(j=0;j<c;j++)

z[i][j]=x[i]pj]+y[i][j];

}

**Structures**

A structure is a collection of dissimilar data types.

**Declaration of a structure**

A structure can be declared as 3ways.declaration of structure is not creation of memory.

**Declaration1:**

struct details:-

{

data type var1;

data type var2;

. . .

. . .

. . .

data type varn;

};

struct details p;

p is variable or object.

**Declaration2:**

struct details

{

data type var 1;

data type var 2;

. . .

. . .

. . .

data type var n;

}p;-> variable or object

**Declaration3:**

struct

{

data type var 1;

data type var 2;

. . .

. . .

. . .

data type var n;

}p;-> variable or object

**Example:**

main()

{

struct details

{

char x;

int y;

float z;

};

struct details p={19,30,58};

|  |
| --- |
| Initializing values |

|  |  |
| --- | --- |
| struct details p;  p.x='A';  p.y='B';  p.z=58; | Assigning values |
| struct details  printf("enter a char,int,float");  scanf("%c%d%f",&.p.x,&p.y,&p.z); | Reading values from scanf |

printf("%c%d%f",p.x,p.y,p.z);

}

**Example:**

main()

{

struct details

{

char a name[20];

int age;

float height:

};

struct details p={"saranya",17,51};

strcpy(p.name,"saranya");

p.age=17;

p.height=51;

printf(enter name,age,height");

scanf("%s5df",p.name,&p.age&p.height);

printf("%s%d%f",.p.name,p.age,p.height);

}

**Copying one object data into another object**

main ()

{

struct details

{

char name[20];

int age;

float height;

}

struct details p={"saranya',17,51},q,r;

strcpy(q.name,p.name);

q.height=p.height

q.age=p.age;

r=p;

printf("%s%d%f%s%d%f%s%D%f",pname,p.age,p.height,q.name,r.name,r.age,r.height);

}

**Compare two sructures:-**

main()

{

struct details

{

char name[20];

int age;

float height;

};

struct details p={"saranya',17,51},q;

q=p;

if(strcmp(p.name,q.name)==0 && p.age=q.age && p.height==q.height)

printf("equal");

else

printf("not equal");

}

**Array of structure:-**

main()

{

struct details

{

char name[20];

int age;

float height;

};

struct details p[10];

int i,n;

printf("enter number of students");

scanf("%d",&n);

printf("enter %d students name,age,height",n);

for(i=0;i<n;i++)

scanf("%s%d%f",&p[i].name,&p[i].age,&p[i].height);

for(i=0;i<n;i++)

printf("%s%d%f",p[i].name,p[i].age,p[i].height);

}

**Nested structure:-**

A structure with in a structure is called nested structure.

**Example1: (storing and printing name age and date of birth)**

main()

{

struct DB

{

int d;

int m;

int y;

};

struct details

{

char name[20];

int age;

struct DB db;

};

struct details p;

printf("enter name,age,db,mb,yb");

scanf("%s%d%d%d%d",&p.name,&p.age,&p.db.d,&p.db.m,&p.db.y);

printf("%s%d%d%d%d",p.name,p.age,p.db.d,p.db.m,p.db.y);

}

**Example2:(storing and printing name and date of birth)**

main()

{

struct name

{

char first[10];

char middle[10];

char last[10];

};

struct db

{

int date;

int month;

int year;

}

struct a

{

struct name p;

struct db q;

};

struct a r;

printf("enter first middle last names");

scanf("%s%s%s",r.p.first,r.p.middle,r.p.last);

printf("enter date,month, year of birth");

scanf("%d%d%d",&r.q.date,&r.q.month,&r.q.year);

printf("name:%s\\_%s\\_%s\\_\n",r.p.first,r.p.middle,r.p.last);

printf("date of birth:%d\%d\%d",r.q.date,r.q.month,r.q.year);

}

**Example:(storing and printing name,age,height and date of birth)**

main()

{

struct name

{

char first[10];

char middle[10];

char last[10];

int age;

float height;

};

struct db

{

int date;

int month;

int year;

};

struct data

{

struct name p;

struct db q;

};

struct data r;

printf("enter name,age,height");

scanf("%s%s%s%d%f",r.p.first,r.p.middle,r.p.last,&r.p.age,&r.p.height);

printf("enter date,month,month,year of birth");

scanf("%d%d%d",&r.q.date,&r.q.month,&r.q.year);

printf("name:%s\\_%s\\_%s\n age:%d\n height:%g",r.p.first,r.p.middle,r.p.last,r.p.age,r.p.height);

printf("\n date of birth:%d/%d/%d",r.q.date,r.q.month,r.q.year);

}

Differences between structures and arrays:

Structures: Arrays:

|  |
| --- |
| structures are collection of different arrays are collection of similar  data types data types  structures are accessed by dot(.) arrays are elements are accessed by  operator array operator i.e.,subscript  Declaration of structure is not Declaration of array is allocation  allocation of memory of memory  structures can be copied directly arrays cannot be copied directly  Two structures can be compared directly Two arrays cannot be compared directly  main() main()  { {  struct details data type array name[size];  { .  data type var1; .  data type var2; .  . . }  . .  . .  data type varn;  };  struct details.p;  .  .  .  } |

**similarities between structures and arrays:**

* arrays and structures come under secondary data types
* memory allocation is continue in both arrays and structures
* We cannot apply any arithmetic operators directly on arrays and as well as structures

**Sending a structure or passing a structure to a fuction:**

* A structure can be passed to a function by three ways:

1)Sending individual values

2)Sending an entire structure

3)By called by address

**1)sending individual values:**

void fun(char name[],int age,float height);

main()

{

struct details

{

char name[20];

int age;

float height;

};

struct details p={"saranya",17,5.1};

fun(p.name,p.age,p.height);

printf("%s%d%f",p.name,p.age,p.height);

}

void fun(char name[],int age,float height)

{

strepy(name,"chaitu");

age=19;

height=5.2;

}

**Output:**

The output above programme is chaitu 175.1 i.e., if you send a structure to

a function the effected values in the function will not effect in the main

function

**2)Sending an entire structure:**

struct details

{

char name[20];

int age;

float height;

};

void fun(struct details p);

main()

{

struct details p={"saranya",17,5.1};

fun(p);

printf("%s%d%f",p.name,p.age,p.height);

}

void fun(struct details p)

{

strcpy(p.name,"chaitu");

p,age=19;

p.height=5.2;

}

**Output:**

The output above programme is chaitu 175.1 i.e., if you send a structure to

a function the effected values in the function will not effect in the main

function

**3)By called by address:**

struct details

{

char name[20];

int age

float height;

};

void fun(struct details\*p);

main()

{

struct details p={"saranya",17,5.1};

fun(&p);

printf("%s%d%f",p.name,p.age,p.height);

}

void fun(struct details \*p)

{

strcpy(p->name,"chaitu");

p->age=19;

p->height=5.2;

}

**Output:**

The output above programme is chaitu 175.1 i.e., if you send a structure to

a function the effected values in the function will not effect in the main

function

**UNIONS**

* Union is collection of dissimilar data types
* Union declaration is not creation of memory

main()

{

union details

{

char x;

int y;

float z;

long u;

}

union details p;

printf("%d",sizeof(p));

}

**Output: 4**

**Differences between structures and unions:**

|  |
| --- |
| **Structures:** **Unions:**    struct details union details  { {  char x; char x;  int y; int y;  float z; float z;  }; };  Different memory locations same memory location is  are allocated for different shared by different i.e.,  members members  Different memory locations same memory location is  are reffered different reffered by different  names names  Overlaping of data does Overlaping of data take  not take place place  No savage of memory savage of memory |

**SIMILARITIES:**

1)Both are collection of dissimilar data types

2)Both are secondary data types

3)Declaration of structure is same as declaration of union

4)Accessing of structure and union members is by dot(.) operator only

5)Everything is same except memory allocation

**Example:**

main()

{

union details

{

char a[2];

int x;

};

union details p;

p.x=512;

printf("%d%d%d",p.a[0],p.a[1],p.x);

}

|  |  |
| --- | --- |
| **union of structures:**  main()  {  struct xyz;  {  char a[2];  int x;  };  struct mno;  {  char b[2];  int y;  };  union pqr  {  struct xyz p;  struct mno q;  };  union pqr s;  s.p.a[0]=0;  s.p.b[1]=2;  s.p.x=512;  printf("%d%d%d%d%d%d",s.p.a[0],s.p.a[1],s.p.x,s.p.b[0],s.p.b[1],s.p.y);  } | **Structure of unions:**  main()  {  union xyz  {  char a[2];  int x;  };  union mno  {  char b[2]  int y;  };  struct pqr  {  union xyz p;  union mno q;  }  struct pqr s;  s.p.a[0]=0;  s.p.a[1]=2;  s.p.y=512;  printf("%d%d%d%d%d%d",s.p.a[0],s.p.a[1],s.q.b[0],s.q.b[1],s.p.x,s.q.y);  } |

**BITFIELDS:**

* **T**aking individual bits is called bit fields.We can use this only in strucyures.
* We should not apply address operator on bitfields.
* We should not apply pointers concept on bitfields.
* We should not apply sizeof() operators on bit fields. Because sizeof operator gives the size of the

variable in bytes only

* For signed datas we should take minimum of 2 bits.

**Syntax:**

struct name

{

data type var1:no of bits;

data type var2:no of bits;

. . .

. . .

. . .

data type varn:no of bits;

};

**example**

main()

{

struct details

{

unsigned int gen:1;

unsigned int ms:1:

unsigned int age:6;

};

struct details p;

p.gen=1;

p.ms=0;

p.age=20;

printf("%d%d%d",p.gen,p.ms,p.age);

}

**Output:1 0 20**

**POINTERS:**

**Pointer is special member which holds address of the another variable**

Pointers are widely used in programming; they are used to refer to memory location of another variable without using variable identifier itself. They are mainly used in linked lists and call by reference functions.

Declaring pointers can be very confusing and difficult at times (working with structures and pointer to pointers). To declare pointer variable we need to use \* operator (indirection/dereferencing operator) before the variable identifier and after data type. Pointer can only point to variable of the same data type.

|  |  |
| --- | --- |
| main()  {  int a=6,\*p;  p=&a;  printf("%d",\*p);  }  **Output: 6** | main()  {  int a=5,b=6,c,\*p,\*q,\*r;  p=&a;  q=&b;  r=&c;  \*r=\*p+\*q;  printf("%d",\*r);  }  **Output: 11** |
| **Factorial of a given number with using pointer:**  main()  {  int i,n,fa=1,\*p,\*q;  p=&n;  q=&fa;  printf("enter n");  scanf("%d",p);  for(i=1;i<=\*p;i++)  {  \*q = \*q \* i;  }  printf("%d",\*q);  } | **Reverse a given number with using pointers:**  main()  {  int n,m,s=0,\*p;  p=&n;  printf("enter n");  scanf("%d",p);  while(\*p!=0)  {  m=\*p%10;  s=s\*10+m;  \*p=\*p/10;  }  printf("%d",s);  } |

**Gcd of two numbers with pointers:**

main()

{

int a,b,\*p,\*q;

p=&a;

q=&b;

printf("enter a,b");

scanf("%d%d",p,q);

while(\*p!=\*q)

{

if(\*p > \*q)

\*p=\*p-\*q;

else

\*q=\*q-\*p;

}

printf("%d",\*p);

}

**Note:**

Any pointer takes 2 bytes of memory,but character pointer gets incrimented by 1 byte,

integer pointer gets incrimented by 2 bytes and float by 4 bytes……………etc

**example:**

main()

{

int a=5,\*p;

float b=2.5,\*q;

char c='A',\*r;

p=&a;

q=&b;

r=&c;

printf("%d\t%d\t%d",sizeof(p),sizeof(q),sizeof(r));

printf("\n%u\t%u\t%u",p,q,r);

p++;

q++;

r++;

printf("\n%u\t%u\t%u",p,q,r);

}

**Output:2 2 2**

**65524 65520 65519**

**65526 65524 65520**

**Note:**

When we give like a[2] compiler will convert like \*(a+2)

Here “a” means an array and also starting address of an array

So we may write a[2] as \*(a+2) or \*(2+a) or 2[a]

main()

{

int a[5]={1,2,3,4,5},\*p;

p=a;

printf("%d",a[2]);

printf("%d",\*(a+2));

printf("%d",\*(2+a));

printf("%d",2[a]);

printf("%d",p[2]);

printf("%d",\*(p+2));

printf("%d",\*(2+p));

printf("%d",2[p]);

}

**Output: 3 3 3 3 3 3 3 3**

**Write a program to print the number of even numbers,their sum and number of odd numbers ,their sum with pointers**

main()

{

int a[20],i,n,ec=0,oc=0,ec=0,os=6,\*p;

p=a;

printf("enter n");

scanf("%d",&n);

printf("enter %d elements",n);

for(i=0;i<n;i++)

scanf("%d",p+i);

for(i=0;i<n;i++)

{

if(\*(p+i)%2==0)

{

ec++;

es=es+ \*(p+i);

}

else

{

oc++;

os=os+\*(p+i);

}

printf("ec=%d,oc=%d,es=%d,os=%d",ec,oc,es,os);

}

}

**Write a program to print the number of subjects marks average:**

main()

{

int s[20],i,n,sum=0,\*p;

float avg;

p=s;

printf("enter how many subjects you have");

scanf("%d",&n);

printf("enter %d subject marks",n);

for(i=0;i<n;i++)

{

scanf("%d",p+i);

sum=sum+ \*(p+i);

}

avg=(float)sum/n;

printf("avg=%f",avg);

}

**Write a program to print biggest and smallest numbers in given array with pointers**

main()

{

int a[20],i,n,big,small,\*p;

printf(enter hmeyh");

scanf("%d",&n);

printf("enter %d elements",n);

for(i=0;i<n;i++)

{

scanf("%d",p+i);

big=\*(p+0);

small=\*p;//small=\*(p+0)

}

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

{

if(\*(p+i)>big)

big=\*(p+i);

if(\*(p+i)<small)

small=\*(p+i);

}

printf("big=%d,small=%d",big, small);

}

**String copy with pointers:**

main()

{

int i;

char a[20],\*p,\*q;

p=a;

q=b;

puts("enter a string");

gets(p);

for(i=0;\*(p+i)!='\0';i++)

{

\*(q+i)=\*(p+i);

}

\*(q+i)='\0';

puts(q);

}

**Note:**

* According to double dimensional arrays we are giving simply like a[1][2] then compile will convert this one as \*(\*(a+1)+2)
* When we are going to point double dimensional array with pointer we have to say coloumn size to pointer
* Lets imagin double dimensional array is a[4][3] here coloumn size is 3 so lets take pointer as (\*p)[3]

main()

{

int a[5],b[5][5],c[5][5],R1,c1,R2,c2,(\*p)[5],(\*q)[5],(\*r)[5];

p=a;

q=b;

r=c;

printf("enter R and C of mat A");

printf("enter R and C of mat B");

scanf("%d%d",&R2,&c2);

if(R1=R2 && c1=c2)

{

printf("enter %d\*%d mat A",R1,c1);

for(i=0;i<R1;i++)

for(j=0;j<c1;j++)

scanf("%d",\*(p+i)+j);

printf("enter %d\*%d mat B",R2,c2);

for(i=0;i<R2;i++)

for(j=0;j<c2;j++)

scanf("%d",\*(q+i)+j);

for(i=0;i<R1;i++)

for(j=0;j<c1;j++)

\*(\*(r+i)+j)=\*(\*(p+i)+j)+\*(\*(q+i)+j);

for(i=0;i<R1;i++,printf(“\n”))

for(j=0;j<c1;j++)

printf("%d",\*(\*(r+i)+j));

}

else

{

printf("not possible");

}

}

**Called by value and called by address:**

**called by value:**sending just values is called called by value

void swap(int a,int b);

main()

{

int a,b=10;

printf("values before swap a=%d,b=%d",a,b);

swap(a,b);

printf("values after swap a=%d,b=%d",a,b);

}

void swap(int a,int b)

{

int c;

c=a;

a=b;

b=c;

}

**Output:**

The output of above programme is

values before swap a=5,b=10

values after swap a=5,b=10.

i.e.,by using called by value we can't effect more than one value.

**Called by address (or) called by reference:**sending address to a function is called called by address

void swap(int \*a,int \*b);

main()

{

int a=5,b=10;

printf("values before swap a=%d,b=%d",a,b);

swap(&a,&b);

printf("values after swap a=%d,b=%d",a,b);

}

void swap(int \*a,int \*b)

{

int c;

c=\*a;

\*a=\*b;

\*b=c;

}

**Output:**

The output of above programme is

values before swap a=5,b=10

values after swap a=10,b=5

i.e.,by using called by value we can effect any no of values.

**Write a program print area ,perimeter of a rectangle with called by address:**

void area(float l,float b,float \*a,float \*p);

main()

{

float l,b,a,p;

printf("enter l,b");

scanf("%f%f",&l,&b);

area(l,b,&a,&p);

printf("%f%f",a,p);

}

void area(float l,float b,float \*a,float \*p);

{

\*a=l\*b;

\*p=2\*(l+b);

}

**Write a program that string copy with called by address:**

void copy (int \* n, int \* y, int n);

main ()

{

int a[20], b[20],i, n;

printf ("enter how many elements you have");

scanf("%d", &n);

printf ("enter %d ele", n)

for (i =0, i,<n; i+ +)

scanf("%d, &a[i]);

copy (a, b, n);

for (i =0; i<n; i+ +)

printf ("%d", b[i]);

}

void copy (int \*x, int \*y, int n);

{

int i;

for (i =0; i<n; i+ +)

\*(y+i) =\*(x+i);

}

**Write a program that copy a string :**

void copy (char \*x, char \*y);

main ()

{

char a[20], b[20];

puts ("enter a string");

gets(a);

copy(a,b);

puts(b);

}

void copy (char \*x, char \*y)

{

for (i =0; \*(x+i)! = '10'; i+ +)

{

\*(y+i)= \*(x+i);

}

\*(y+i) = '\0';

}

**Multiplication of two matrices with called by address (Or) it is the example of sending a double dimensional array to a function**

void add (int (\*x)[5], int (\*y)[5], int (\*r)[5], int R1,int C1);

main ()

{

int i, j, a[5][5], b[5][5], c[5][5],R1,c1,R2,c2;

printf("enter R & C of mat A");

scanf ("%d%d",&R1,&C1);

printf ("enter R & C of mat B");

scanf ("%d%d", &R2,&C2);

if(R1 == R2 && c1 == c2)

{

printf ("enter %d\*%d mat A",R1 C1);

for(i=0,i<R1;i++)

for(j=0; j<c1;j++)

scanf("%d",&a[i][j]);

printf("enter %d\*%d matB",R2,C2);

for(i=0; i<R2;i++)

for(j=0; j<c2;j++)

scanf("%d",&b[i][j]);

add(a,b,c,R1,C1);

for(i=0;i<R1;i++,printf("1n"))

for(j=0;j<C1;j++)

printf("%d",c[i][j]);

}

else

{

printf("not possible");

}

}

void add (int (\*x)[5] , int (\*y)[5],int (\*z)[5],int R1,int C1)

{

int i,j;

for(i=0; i<R1;i++)

for(j=0; j<C1;j++)

\*(\*(z+i)+j) = \*(\*(x+i)+j) + \*(\*(y+i)+j);

}

**Circumference and area of a circle with called by address:-**

void area(float r,float\*a,float\*c);

main ()

{

float r,a,c;

printf("enter r");

scnf("f",& r);

area(r,&a,&c);

printf("%f%f",a,c);

}

void area(float r,float \*a,float \*c);

{

\*a=3.14\*r\*r;

\*c=2\*3.14\*r;

}

**Function returning a pointer:-**

int\* add(int a, int b);

main()

{

int a=5,b=6,\*c;

c=add(a,b);

printf("%d",\*c);

}

int\* add(int a, int b)

{

int c;

c=a+b;

return &c;

}

**pointer to a function:-**

void display(void);

main()

{

void (\*p)(void);

p=display;

p();

}

void display(void)

{

printf("good");

}

**Example:(printing address of the main)**

main()

{

printf("%u”,main);

}

**pointer to a function with argument with return**

int add(int a,int b);

main()

{

int(\*p)(inta,intb);

int a,b,c;

printf("enter a,b");

scanf ("%d%d",&a,&b);

p=add;

c=p(a,b);

printf("%d",c);

}

int add(int a,int b)

{

int c;

c=a+b;

return c;

}

**We may write function(with argument and with return) in following**

|  |
| --- |
| int add (int a,int b); function[int,int] |
| int add (int \*a,int \*b); function[int address int] |
| int \* add(int a,int b); function[int,int address] |
| int \* add(int \*a,int \*b); function[int address,int address] |
| int (\*add)(int a,int b); pointer(int,int) |
| int (\*add)(int \*a,int \*b); pointer(int address,int) |
| int \* (\*add)(int a,int b); pointer(int , int ) |
| int \* (\*add)(int \*a,int \*b); pointer(int \* , int \*) |

**displaying students details by using pointers:**

**when we are accessing structure variables by using pointers we have to use arrow(->)operator**

main()

{

struct details

{

char name[20];

int age;

float height;

};

struct details p={“saranya”,17,5.1},\*q;

printf("%s%d%f",p.name,p.age,p.height);

printf("%s%d%f",q->name,q->age,q->height);

}

**Pointer to a printer:-**

main()

{

int a=5,\*p,\*\*q,\*\*\*r;

p=&a;

q=&p;

r=&q;

printf("%d",\*p);

printf("%d",\*\*q);

printf("%d",\*\*\*r);

}

**Output: 5 5 5**

**Dynamic memory allocation and de allocation:-**

The memory which is allocated and deallocated during run time of the program is called dynamic memory allocation and deallocation.

various functions for dynamic memory allocation and allocation.

1.malloc()

2.calloc()

3.realloc()

4.free()

**1.malloc():-**

**syntax:-** **(type\*)malloc(size in bytes)**

* malloc allocates memory size in bytes.Allocated memory cells are always in continous memory location.Allocated memory cells have intial garbage values.
* If malloc is successful to create memory it returns starting address of the allocated memory
* memory space is allocated,and the return address or pointer is void data type pointers,so it needs type costing.
* If malloc is not successful to allocate memory it returns NULL pointer.

**Write a program to find average of n elements**

main()

{

int i,n,\*p,sum=0;

float avg;

printf("enter nsyn");

scanf("%d",&n);

p=(int\*)malloc(n\*2);

if(p==NULL)

{

printf("memory is not allocated");

exit(1);

}

printf(enter %d elements",n);

for(i=0,i<n;i++)

scanf("%d", p+i);

for(i=0;i<n;i++)

sum=sum+\*(p+i);

free(p);

avg=(float)sum1n;

printf("%d",avg);

}

**Program:Write a program to print number of even numbers and their sum, number of odd numbers and their sum from n numbers with dynamic memory allocation**

main()

{

int i,n,ec=0,oc=0,es=0,os=0,\*p;

printf("enter how many elements you have");

scanf("%d",&n);

p=(int\*)malloc(n\*sizeof(int));

if(p==NULL)

{

printf("memory is not created");

exit(1);

}

printf("enter %d elements",n);

for(i=0;i<n;i++)

scanf(“%d”,p+i);

for(i=0;i<n;i++)

{

if(\*(p+i)%2==0)

{

ec++;

es=es+\*(p+i);

}

else

{

oc++;

os=os+\*(p+i);

}

}

free(p);

printf("ec=%d,oc=%d,es=%d,os=%d",ec,oc,es,os);

}

**Program:write a program to print biggest and smallest numbers from n numbers with dynamic memory allocation**

main()

{

int i,n,big,small,\*p;

printf("enternmeyn");

scanf("%d",&n);

p=(int \*)malloc(n\*sizeof(int));

if(p==NULL)

{

printf("memory is not created"));

exit(1);

}

printf("enter %d elements",n);

for(i=0;i<n;i++)

scanf("%d",p+i);

big=\*p;

small=\*p;

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

{

if(\*(p+i)>big)

big=\*(p+i);

if(\*(p+i)<small)

small=\*(p+i);

}

free(p);

printf("big=%d,small=%d",big,small);

}

**Progam: String copy with dynamic memory allocation:**

main()

{

char \*p,\*q;

int i,n;

printf("enter how many charecters do you want to enter");

scanf("%d",&n);

p=(char\*)malloc(n\*sizeof(char)+1);

q=(char\*)malloc(n\*sizeof(char)+1);

if(p==NULL || q==NULL)

{

printf("memory is not created);

exit(1);

}

printf("enter a string");

gets(p);

for(i=0;\*(p+i)!='\0';i++)

{

\*(q+i)=\*(p+i);

}

\*(q+i)='\0';

puts(q);

free(p);

free(q);

}

**2.Calloc:-**

**syntx:-(type\*)calloc(n,size in bites)**

* calloc allocates memory n rows of size in bytes.
* Allocated cells are always in continous memory location.
* Allocated cells have intial zeros.
* If calloc is succesful to allocate memory it returns starting address of allocated memory space. i.e void pointer.
* If calloc is not successful to allocate memory it returns null pointer.

**3.Realloc:-**

**syntax:- realloc(old pointer,new size)**

It is used to reallocate memory.

It reallocates memory of new size and transfers old data into newly created memory space.

main()

{

char \*p,\*q;

p=(char \*)malloc(8);

p="tech";

q=realloc(p,15);

strcat(q,"genius")

puts(q);

free(q);

}

**4.Free:-**

It is used to deallocate memory which is allocate by dynamic memory allocation.

**Pointers Advantages:-**

* Fast exhicution time.
* By called by address we can effect any no.of values.
* Savage of memory during string handling
* By dynamic memory allocation we allocate and deallocate during run time of the program.

**Disadvantages:-**

* Wastage of memory.
* Confusion.
* Data is not secure.

**Similarities of malloc and calloc:-**

* Both needs type costing.
* Allocated cells are always in continous memory location.
* If malloc and calloc are successful to allocate memory it returns starting address of allocated memory
* space and the return address or pointer is void data type.
* If malloc and calloc are not successful to allocate memory they returns null pointer. Both are the functions for dynamic memory allocation and deallocation.

**Differences between malloc and calloc:-**

|  |  |
| --- | --- |
| **malloc** | **Calloc** |
| syntax: (type\*)malloc(size in bytes) | syntax: (type\*)calloc(n,size in bytes) |
| Allocated memory cells have intial garbage values. | \* Allocated memory cells have intial zeros. |
| malloc gives n bytes of size | Calloc gives n rows of size in bytes. |
| It is mostly used in array. | It is mostly used in records such as structures |

**Airthmetic operations on pointers:-**

**possible operations:-**

* A pointer can be added with a constant.
* A pointer can substract a constant.
* Two pointers can be compared.
* Two pointers can be substracted.

**Not possible operations:-**

* A pointer cannot be multiplied with a constant.
* A pointer cannot be divided with a constant.
* We cannot add two pointers.
* Two pointers cannot be divided.
* Two pointers cannot be multiplied.
* Two pointers cannot be substracted.

main()

{

char \*p="saranya";

puts(p);

}

**Note:**We can say that from the program pointers can save memory

**Taking number of strings by using double dimensional array:**

main()

{

char a[5][20]={"sree","siit","hello","hai","bye"};

printf("%s%s%s%s%s",a[0],a[1],a[2],a[3],a[4]);

}

**Taking number of strings by using array of pointer:**

main()

{

char \*p[5]={"sree","siitt","hello","hai","bye"};

puts("%s%s%s%s%s",p[0],p[1],p[2],p[3],p[4];

}

**Difference b/w (\*p)[n] and \*p[n]:-**

**(\*p)[n]**  **\*p[n]**

\* Pointer to an array. \* Array of pointers.

\* one pointer is created. \* n pointers are created.

\* Size of pointer is 2 bytes \* Size of array is  **n\*2**  bytes.

**Function returning a pointer:-**

int \* add(int a,int b);

**pointer to a function:-**

int (\*add) (int a,int b);

**formatting:**

printf("formal string",arg1,arg2,.....);

\*format specifics

\*escape sequence

\*ordinary or normal

%i

%c printf("%f",234.56) 234.560000

%lf

%Lf printf("%g",234.56) 234.56

%d

%x printf("%e",234.56) 2.3456e+02

%0

%x %e->2.3456E+02

|  |  |
| --- | --- |
| main()  {  int a=25,b=12345, c=3;  printf("in a:%5d",a);  printf("in b:%5d",b);  printf("in c:%5d",c);  }  **output:-**in a:...25  in b:12345  in c:....3 | main()  {  int a=25,b=1234,c=-3;  printf("in a:%+08d",a);  printf("in b:%+08d",b);  printf("in c:%+08d",c);  }  **output:-**in a:+0000023  in b:+0012345  in c:-0000003 |
| main()  {  int a=25,b=12345,c=-3;  printf("in a:%08d",a);  printf("in b:%08d",b);  printf("in c:%08d",c);  }  **output:-**in a: 0000023  in b:0012345  in c:0000003 | main()  {  int a=25;  printf("in a:%#d",a);  printf("in a:%#0",a);  printf("in a:%#x",a);  }  **output:-** in a:25  in b:031  in a:0x19 |
| main()  {  int a=25;  printf("in %d",a);25  printf("in %#d",a);25  printf("in %+d",a);+25  printf("in %5d",a);...25  printf("in %05d",a);00025  } | main()  {  char s1[]="name",s2[]="qualification',s3[]="age";  char r1[]="saranya",r2[]="B.tech",r3[]="17";  printf("in%-13s:%s",s1,r1);  printf("in%-13s:%s",s2,r2);  printf("in%-13s:%s",s3,r3);  }  **Output:-** name :Saranya  qualification :b tech  age :17 |
| main( )  {  int a=25,b=3346,c=12345;  int v=5;  printf(“%\*d”, v, a );  printf(“%\*d”, v, b);  printf(“%\*d”, v, c );  }  **Output:-** 25  3346  12345 | main( )  {  char s1[ ]=”name”,s2[ ]=”qualification”,s3[ ]=”age”;  char r1[ ]= ”Saranya”,r2[ ]=”btech”,r3[ ]=”17”;  v=25;  print f(“\n%-\*s:%s”,v,s1,r1);  print f(“\n%-\*s:%s”,v,s2,r2);  print f(“\n%-\*s:%s”,v,s3,r3);  print f(“\n%-\*s:%s”, v, ”date of birth of student”,”17-12-1991”);  }  **Output:-**  name : Saranya  qualification: b tech  age :17  date of birth of student:17-12-1991 |
| main( )  {  int a=25;  float b=25.34;  printf(“\n a:%5d”,a );  printf(“\n b:%5.2f”, b);  }  **Output:-**  a: 25  b: 25.34 | main( )  {  int a, b, v ;  printf*(“enter* a, b” );  v=scanf(“%d % d”, &a, &b );  printf(“a=%d b=%d v=%d”, a, b, v);  }  **Output:-**  enter a, b : 25 50  25 50 2  enter a, b: 25a 50  25 Garbage 1  enter a ,b: a25 50  Garbage Garbage D |
| main( )  {  int v, n, m ,s ;  printf*(“enter* time” );  v=scanf(“%d: %d: %d”, &n, &m, &s );  printf(“entered time is %d: %d: %d”, n, m, s);  printf(“invalid input item :%d”, v);  }  **output:-**  Enter time  11:30:27  Entered time is 11:30:27 | main( )  {  int s, d, p ;  print f(“enter pin code :”);  scanf(“%1d%2d%3d”,&s,&d,&p);  printf(“state code: %d”,s);  printf(“district code : %d”,d);  printf(“p. o code: %d”,p);  }  **output:-**  Enter pin code : 500079 |
| main( )  {  int a=012;  print f(‘%#d”, a);  print f(%#10”,a);  print f(%#x”, a);  }  **Output-**  10  012  0x0 | main( )  {  int a=0x12;  print f(%#d\n”, a);  print f(%#0\n”, a);  print f(%#x\n”, a);  }  **Output-**  18  022  0x12 |
| main( )  {  print f(“%#d\n”,0XAB);  print f(“%#0\n”,0XAB);  }  **Output-**  171  0253  main( )  {  print f(“%d”, sizeof (2.5));  print f(“%d”, sizeof(2.5f));  print f(“%d”, sizeof (5l));  } | **NOTE-**  5-int  5.2-double  5.2f-float  5l-long  ‘a’-character  “good”-string |

**FILES**

**fopen (“filename” ,”mode”)**

**Mode:-**read mode, write mode, append mode

* fopen( ) is used to open a FILE. If fopen( ) is successful to open a file it returns the starting address of the opened file **i.e.** file pointer
* If fopen( ) is not successful to open a file it returns **NULL** pointer

**“w”(write mode):-**

It is used to write Data into a file. It creates new file if file does not exist. If file already exists it opens and rewrites

i.e. we loose our previous data

**“r”(Read mode):-**

It is used to read data from a file. If file exists it opens otherwise it cannot create a new file.

**“a”(append mode):-**

It is used to append data to a file. If file exists it opens but it cannot create a new file.

**getc(ptr):-**

It is used to get a character from a file .It takes a file ptr as argument

**putc(ch, p):-**

It is used to put a character into a file. It takes a character as well as file pointer as arguments

**getchar():**

it is used to read characters from keybord

**putchar(ch,p):**

it is used to write a character on output

**Writng data into a file**

main( )

{

FILE\*p;

char ch;

p=fopen(“good .c”, ”w”);

if (p = =NULL)

{

printf(“file is not created”);

exit(1);

}

printf(“enter data”);

while(1)

{

Ch=getchar ( );

If ( ch = =EOF)

{

break;

}

putc( ch, p );

}

fclose (p);

}

**Writing and reading a file:-**

#include<stdio. h>

#include<stdlib. h>

main( )

{

FILE\*p;

char ch;

p=fopen( “bad .c”, ”w”)

If(p = =NULL)

{

printf(“file is not created”);

exit(1);

}

printf(“enter data”);

while((ch = getchar( ) ) !=EOF)

{

putc (ch, p );

}

fclose(p);

p=fopen(“bad .c”, r ”);

if( p = =NULL)

{

printf(“file does not exist”);

exit(1);

}

while(ch =getc(p))!=EOF)

{

putchar (ch);

}

fclose(p);

}

**WAP to append data to a file. Display file before and after appending data**

#include<stdio.h>

#include<stdlib.h>

main( )

{

FILE\*p;

char ch;

p=fopen( “bad .c”, ”r”)

if(p = =NULL)

{

printf(“file does not exist”);

exit(1);

}

while((ch = getc(p ) ) !=EOF)

{

putchar (ch, p );

}

fclose(p);

p=fopen(“bad .c”, a ”);

if( p = =NULL)

{

printf(“file does not exist”);

exit(1);

}

printf(“enter data”);

while((ch = getchar( ) ) !=EOF)

{

putc (ch, p );

}

fclose(p);

p=fopen(“bad .c”, r ”);

if( p = =NULL)

{

print f(“file does not exist”);

exit(1);

}

while(ch =getc(p))!=EOF)

{

putchar (ch);

}

fclose(p);

}

**WAP to copy a file:-**

#include<stdio.h>

#include<stdlib.h>

main ( )

{

FILE\*sp,\*tp;

char ch;

sp=fopen(“bad. c”, ”r” );

if(p = =NULL)

{

print f(“source file does not exist”);

exit(1);

}

tp =fopen(“good. c”, ”w ”);

if(tp = =NULL)

{

print f(“target file does not exist”);

exit(1);

}

while((ch=getc(sp))!=EOF)

{

putc(ch, tp );

}

fclose(sp);

fclose(tp );

tp=fopen(“good . c”, ”r”);

if(tp = =NULL)

{

printf(“file does not exist”);

exit(1);

}

while((ch=getc(tp)!=EOF)

{

putchar(ch);

}

fclose(tp);

}

**NOTE:-**

**fclose( ptr):-**

It is used to close a file .It takes file pointer as argument

**getw(ptr):-**

It is used to get an integer from a file. It takes a file ptr as argument.

**putw(ptr):-**

It is used to put an integer into a file. It takes an integer as well as a file ptr as argument

**fgets(ptr):-**

It is used to get a string from file. It takes file ptr as arguments

**fputs(ptr):-**

It is used to put a string into a file . It takes file ptr as arguments

**feof(ptr):-**

It is used to check whether end of file is reached or not. It takes file ptr as argument. If eof is reached it returns ‘1’ (true value) otherwise it returns false value ‘0’.

**ferror (ptr):-** It is used to identify whether an error is encountered or not. If error is encountered it returns true value i.e. other than zero. If other wise it returns false value i.e. zero. It takes a file ptr as argument.

**rewind(ptr):-**

It is used to get a file ptr to the beginning of the file. It takes a file ptr as an argument

**ftell(ptr):-** It is used to know information about the ptr position from beginning of the file(distance of ptr).It takes a file ptr as argument. It returns long value

#include<stdio.h>

#include<stdlib.h>

main( )

{

FILE\*p;

p=fopen(“good. c”, ”r”);

if(p = = NULL)

{

printf(“ file does not exist”);

exit(1);

}

while(!(feof(p))||while(feof (p) = =0)||while(feof (p)!=1)

{

ch=get c(p);

putchar(ch);

}

fclose(p);

}

**fseek( ptr, off set value ,mode):-**

It is used to move file ptr from one position to another position, where mode represents 0-beginning,1-current,2-ending.

fseek(p,ml,0):- move m+1 bytes forward from beginning of file

fseek(p,ml,1):- move m bytes forward from current position of cursor

fseek(p,-ml,1):- move m bytes backward from current position of cursor

fseek(p,-ml,2):- move m bytes backward from end of file

fseek returns a non zero the value if it is successful to move ptr otherwise it returns ‘-1’

SYNTAX:-

fprintf(ptr,”string”,var1,var2,…..)

fscanf(ptr,”formats”,&var1,&var2,….)

**write a program to store student details(name 3 subjects marks and average of 3 subjects marks) into a file , and read the same data from file print on output**

#include<stdio.h>

main( )

{

FILE\*p;

char name[20];

int s1,s2,s3;

float avg ;

p=fopen(“marks .c”, ”w” );

if(p = =NULL)

{

fprintf (stdout, ”file is not created”);

exit(1);

}

fprintf (stdout, ”enter name,3 sub marks”);

while (fscanf (stdin,”%s%d%d%d”,&name,&s1,&s2,&s3)!=EOF)

{

avg =(s1+s2+s3)/3.0;

fprintf (p,”%s\t %d\t %d\t %d\t%f”,name,s1,s2,s3,avg);

}

fclose (p);

p=fopen (“marks. c”, ”r”);

if(p = =NULL)

{

fprintf (stdout, ”file does not exist”);

exit(1);

}

while(fscanf (p,”%s%d%d%d%f”,&name,&s1,&s2,&s3,&avg)!=EOF)

{

fprintf (stdout,”%s\t %d\t %d\ t %d\t%f\n”,name,s1,s2,s3,avg);

}

fclose (p);

}

**PREPROCESSOR DIRECTIVES:-**

* # Indicates preprocessor directive
* A preprocessor directive should not end with semicolon(;)
* No two preprocessor directives are written in one line
* Various preprocessor directives:-

#include #define

#if def #ifndef

#if #else

#e l if #end if

#un def #line

#error #pragma

**i.)#include:-**

#include <xxx .h>,it searches in include directory for header file

#include ”xxx . h” ,it searches in all directories in TC (source directory)

**ii.)#define:-**

#define is a macro, it substitute the code

**Conditional complication:-**

#ifdef , #if , #elif , #ifndef , #else , #endif

**iii) #ifdef;-**

**SYNTAX-**

#ifdef label

s-1;

s-2;

s-3;

#endif;

If labels is defined then only the three statements (s1,s2,s3) are complied otherwise treats as comment lines i.e. they are not complied

**iv.) #ifndef:-**

**SYNTAX-**

#ifndef label

s-1;

s-2;

s-3;

#endif;

If label is not defined then only the 3 statements (s1,s2,s3) are complied otherwise treats as comment lines i.e. they are not complied

v.) #if:-

SYNTAX-

#if constant expression

s-1;

s-2;

s-3;

#end if;

If constant expression is true s1,s2,s3 are complied otherwise treats as comment lines i.e. they are not complied

**vi.) #else:-**

**SYNTAX-**

#if constant expression

s-1;

s-2;

s-3;

#else

s-4;

s-5;

s-6;

#end if;

If constant expression is true s1,s2,s3 are complied otherwise s4,s5,s6 are complied

I.E. #if constant expression1

s-1;

s-2;

s-3;

#else if con exp2

s-4;

s-5;

s-6;

#else if con exp3

s-7;

s-8;

s-9;

#else

s-10;

s-11;

s-12;

#endif;

**vii.) #elif:-**

**SYNTAX-**

# if constant expression1

s-1;

s-2;

s-3;

#elif constant exp2

s-4;

s-5;

s-6

#elif con exp3

s-7;

s-8;

s-9;

#else

s-10;

s-11;

s-12;

#endif;

**viii.) #endif:-**

every conditional complications ends with #endif

**ix.) #undef:-**

it undefines defined macro

**x.) #error:-**

it defines error messages we can give any error statement that we want

main( )

{

int a=5,b=2,c;

# ifndef add

#error “add is not defined”

#end if

#ifdef and

c=a +b;

print f(“%d”, c );

#endif

}