Write a program to find the grade of the student based on the given conditions.

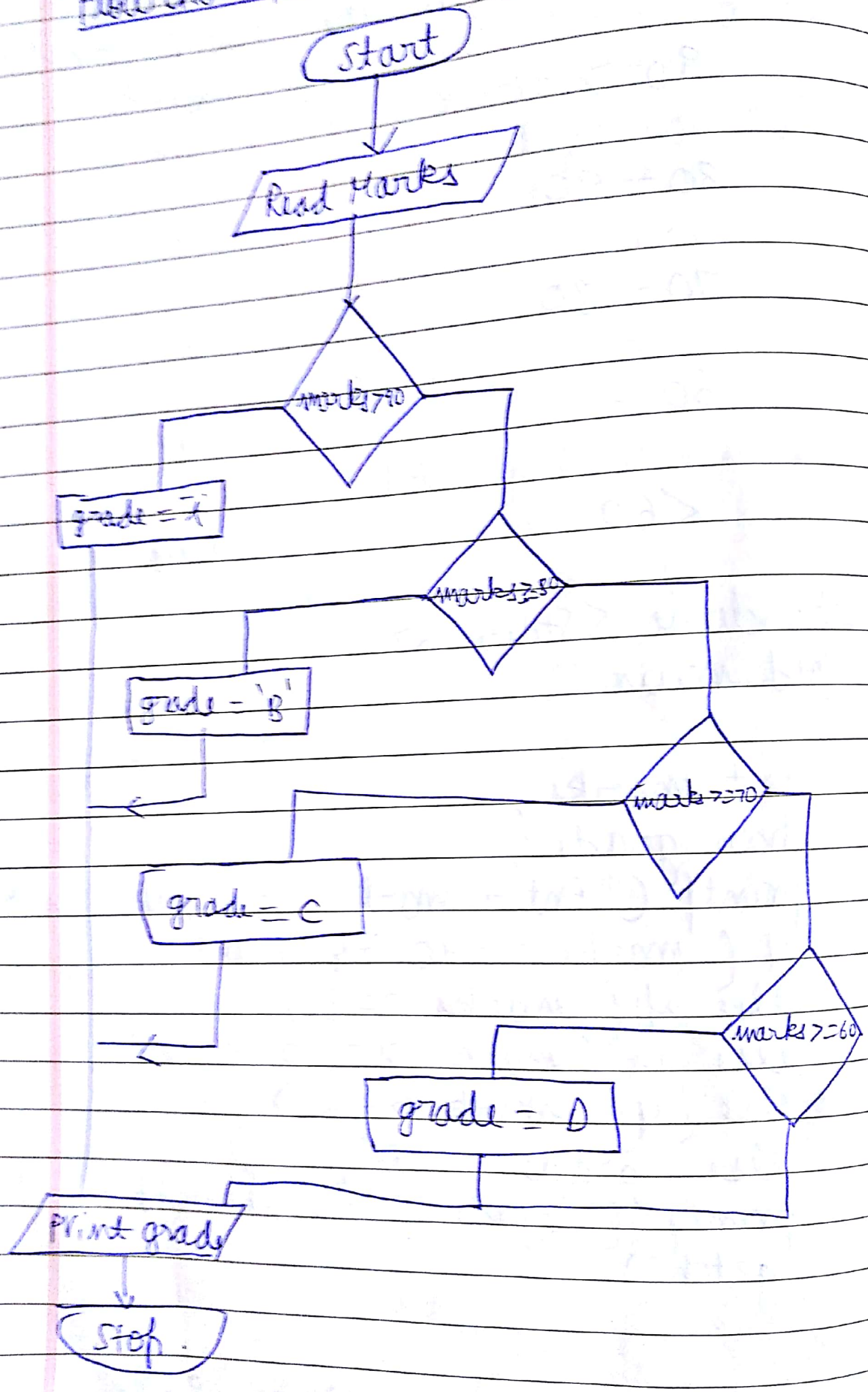
	Grade
90 > above	A
80 - 90	B
70 - 80	C
60 - 70	D
<60	Fail.

```
#include <stdio.h>
void main()
{
  int marks;
  char grade;
  printf("Enter marks"); scanf("%d",&marks);
  if (marks >=90 && marks <100) grade='A';
  else if (marks >=80)
  else if (marks >=70)
  else if (marks >=60)
  else grade = 'F';
  printf("Grade of the student = %c", grade);
  getch();
}
```

Result: Enter Marks 95.

Grade of the student = A.

Flowchart:



Write the 'C' program to check if the triangle is isosceles, scalene, equilateral taking 3 sides of triangle as input.

```
#include <stdio.h>
#include <conio.h>
Void main()
int a, b, c;
char triangle;
printf("Enter the sides"); scanf("%d%d%d",
                          &a, &b, &c);
if (a==b && b==c && c==a)
  printf("triangle is equilateral");
else if (a!=b && b!=c && c!=a)
  printf("triangle is scalene");
  else isosceles;
  getch();
}
```

→ Switch (expression)

```
{
  case constant 1: statement 1;
  case constant 2: statement 2;
    .                .
    .                .
    .                .
  default: statement x;
}
```

Multiway branching
- else if ladder.
- switch.

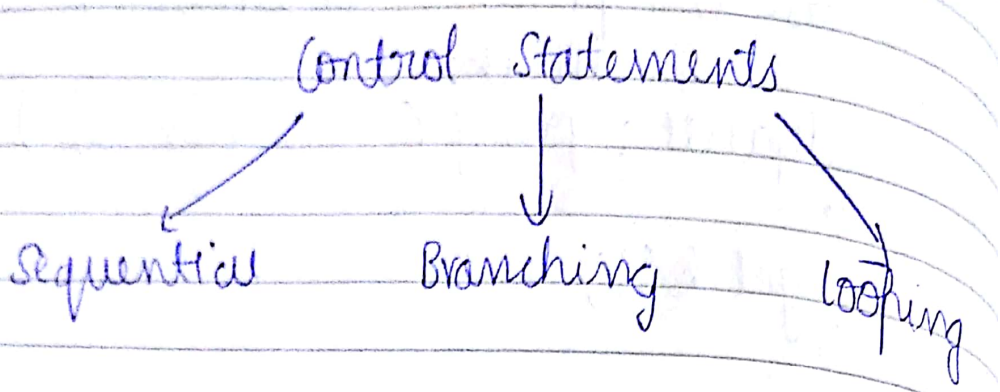
Write a program to display the month of a year taking the no as input

```
#include <stdio.h>
#include <conio.h>
Void main();
{
    int month;
    printf("Enter month (1-12):");
    scanf("%d", &month);
    switch (month)
    {
        case 1 : Printf("January");
                 break;
        case 2 : printf("February");
                 break;
        case 3 : printf("March");
                 break;
        case 4 : printf("April");
                 break;
        case 5 : printf("May");
                 break;
        case 6 : printf("June");
                 break;
        case 7 : printf("July");
                 break;
        case 8 : printf("August");
                 break;
        case 9 : printf("September");
                 break;
        case 10 : printf("October");
                  break;
```

```
case 11: printf (" November ");
         break;
case 12: printf (" December ");
         break;
default: printf (" Invalid No. ");
}
getch ();
}
```

Result:- Enter month 5
May.

looping :-



A set of statement is executed repeatedly till a condition is used to stop the loop.

→ **for loop :-**

Syntax :-

for (initialisation; test condition; increment/decrement)

{

set of statements

}.

Eg :-

```
for (a = 0; a <= 5; a = a+1)
{ printf ("%d", a); }.
```

OUTPUT :-

0 1 2 3 4 5

Eg(2)

```
for (a=2; a<=8; a=a+2)
    {
      printf("%d", a);
    }
```

OUTPUT:-

2 4 6 8

Eg(3)

```
for (a=1; a<=7; a=a+2)
    {
      Printf("%d", a);
    }
```

OUTPUT:-

1 3 5 7

Eg(4)

```
for (a=5; a>=1; a=a-1)
    {
      printf("%d", a);
    }
```

OUTPUT:-

5 4 3 2 1

Eg ⑤

```
for (a=10; a>4; a--)
{
  printf ("%d", a);
}
```

OUTPUT:-

10 9 8 7 6 5

Eg ⑥:-

```
for (a=0; a<5; a++)
{
  printf ("%d", a);
}
```

Output :- 0 1 2 3 4

Tracing	a	a < 5	o/p
Loop 1	0	T	0
loop 2	1	T	1
Loop 3	2	T	2
Loop 4	3	3<5 (T)	3
Loop 5	4	4<5 (T)	4
loop 6	5	5<5 (F)	stops

In for loop, the body of the loop is executed till the test condition is true.

while loop:

Syntax:

```
while ( test condition)
{
  body of loop
}
```

Eg①

```
a = 0;
while (a < 5)
{
  printf ("%d", a);
  a++;
}
```

Eg②

```
a = 2;
while (a <= 8)
{
  printf ("%d", a);
  a = a+2;
}
```

OUTPUT: 2 4 6 8

Eg③

```
a = 9;
while (a > 0)   or while (a >= 2)
{
  printf ("%d", a);
  a = a - 3;
}
```

OUTPUT: 9 6 3

do while loop:-

Syntax:-

```
do
{
    body of loop
}while (test condition);
```

Eg (1) - `a = 9;`

```
do
{ printf ("%d", a);
  a = a-3;
}
while (a>2);
```

OUTPUT: 9 6 3

Eg (2)

```
a = 1;
do
{ printf ("%d", a);
  a = a-3;
}while(a>2);
```

OUTPUT:- -2

In do while loop, the body of the loop is executed first and then the test condition is checked.

Atleast once the loop is executed.

Loops:

1) Pre test — for, while.
2) Post test — do while.

Write a C program to find the factorial of a Number.

$4! = 4 \times 3 \times 2 \times 1$.
$n! = 1 \times 2 \times 3 \times 4 \times \ldots \times n$.

```
Void main ()
{
  int n, fact, i;
  printf (" Enter a Number ");
  scanf ("%d", &n);
    fact = 1;
  for (i=1; i<=n; i++)
    {
      fact = fact * i;
    }
  printf ("Factorial of %d! = %d", n, fact);
    getch();
  }
```

of polynomial Simple Cm...

1) $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots \frac{1}{n}$.

2) $1 + \frac{1}{x} + \frac{1}{x^2} + \ldots + \frac{1}{x^n}$.

→. Write a C program to find the sum natural numbers.

$1 + 2 + 3 + 4 + \ldots + n$.

```
Void main ()
{
int n, sum, i;
printf (" Enter a number ");
scanf ("%d", &n);
sum = 0;
for (i=1; i<=n; i++)
  {
    sum = sum + i;
  }
printf (" Sum of Natural No's = %d", su
getch ();
}
```

①→.

```
Void main ()
{
int n, sum i;
printf (" Enter a number ");
scanf ("%d", &n);
for (i=1; i<=n; i++)
{
sum =
```

Nested loops :- One loop inside another loop is called Nested loops.

```
for ( i = 1 ; i <= 3 ; i++ )        → Outer execution
{   for ( j = 1 ; j <= 4 ; j++ )    → 4 times.
    {   printf ("%d", j);
    }
    printf ("\n");
}
```

Syntax :- (Don't know)

```
for ( intialization ; test condition ; increment/decrement )
{
    for ( intialization ; test condition ; increment/decrement )
    {
        statements ;
    }
}
```

Outer loop

Inner loop

Eg ①:

```
for ( i=1; i<=3; i++)
    for ( j=1; j<=i; j++)
    {
      printf ("%d", j);
    }
    printf ("\n");
}
```

```
1
1 2
1 2 3
```

Eg ③:

```
for ( i=1; i<=3; i++)
    for ( j=1; j<=i, j++)
    {
      printf ("*");
    }
    printf ("\n");
}
```

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

Eg ④:

```
for ( i=1; i<=4; i++)
    for ( j=1, j<=i; j++)
    {
      printf ("%d", i);
    }
    printf ("\n");
}
```

```
1
2 2
3 3 3
4 4 4 4
```

Arrays - Array is a collection of similar data items of the same data type

Declaration

```
float emp_sal [100];
int emp_id [100];
char name [40];
```

Structure:

emp-sal [0]
emp-sal [1]
emp-sal [2]
:
:
emp-sal [99]

Note: Array index will always start with.

Compile time
Run time
> Array initialization.

Compile time

```
Void main()                a[0] a[1] a[2] a[3] a[4]
{ int a[5] = { 77, 3, 25, 46, 32 };
  int sum = 0;
  for (i=0; i<5; i++)
  {
    sum = sum + a[i];
```

printf ("Sum of element of an array = %d",
getch()
}

Run Time:

```
Void main()
{
int i, N, a[100], sum = 0;
printf ("Enter the size of array");
scanf ("%d", &N);
printf ("Enter elements of an array")
for (i=0; i<N; i++)
{
scanf ("%d", &a[i]);
}
for (i=0; i<N; i++)
{
sum = sum + a[i];
}
printf ("Sum of element of an array = %d",
getch();
}
```

Write a C program to count the no of elements of an array which are divisible by 5.

```
Void main()
{
    int i, N, a[100], sum = 0, count
    printf("Enter the size of array");
    scanf("%d", &N);
    printf("Enter the elements of an array");
    for(i=0; i<N; i++)
    {
    scanf("%d", &a[i]);
    }
    for(i=0; i<N; i++)
    {
    if(a[i]%5==0) count++;
    }
    printf("Sum of element of an array=%d", sum);
```

Advantages –

Single variable name to multiple data.
Searching an element is easier.
Elements can be sorted or arranged in ascending or descending order.

Search < Linear
Binary .

```
Void main()
{
    int i, a[100], N, pos, key, found=0;
    printf("Enter the size of an array");
    scanf("%d", &N);
    printf("Enter the elements of an array");
    for(i=0; i<N; i++)
    scanf("%d", &a[i]);
```

```
printf("Enter the key element to
scanf("%d", &key);
for (i=0; i<N; i++)
{
if (a[i] == key)
{
found = 1; break; }
}
if (found == 1)
printf("Element exist in position %d
                                    i+1);
else
printf("Element does not exist");
getch();
}
```

→ Sorting -

Bubble Sort -

It is used to sort the elements by comparing the neighbouring elements and exchanging their position.

Condition: $a[j] > a[j+1]$

i=0.

| 5 | 4 | 3 | 2 | 1 |

i=0, j=0,1,

j=0 | 4 | 5 | 3 | 2 | 1 |

j=1 | 4 | 3 | 5 | 2 | 1 |

j=2 | 4 | 3 | 2 | 5 | 1 |

j=3 | 4 | 3 | 2 | 1 | 5 |

i = 1 a[j] > a[j+1].

fig①: 4 3 2 1 5.

j=0 3 4 2 1 5

j=1 3 2 4 1 5.

j=2. 3 2 1 4 5

i = 1, j = 0, 1, 2.

fig②: i = 2

3 2 1 4 5, i = 2, j = 0, 1.

j=0 2 3 1 4 5

j=1 2 1 3 4 5.

fig③: i = 3

2 1 3 4 5

j=0 1 2 3 4 5.

for (i=0; i<N-1; i++)
scanf ("%d", &a[i]);

```
Void main()
{ int i, a[100], N;
Printf("Enter the size of the an array");
scanf("%d", &N); printf("Enter the elements of an array");
for (i=0; i<n-1; i++)
  for (j=0; j<n-i-1; j++)
  {
    if (a[j] > a[j+1])
    {
      temp = a[j];
      a[j] = a[j+1];
      a[j+1] = temp;
    }
```

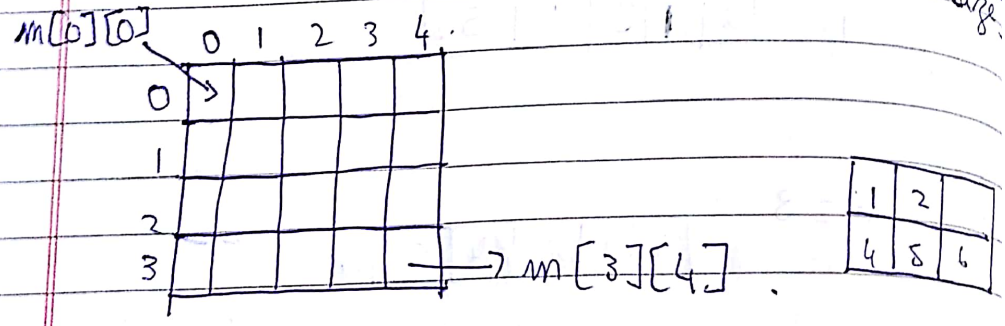
N=5

```
printf ("Sorted elements...");
for(i=0; i<N; i++)
{
  printf("%d", a[i]);
}
getch();
}
```

2D Array:-

Syntax:-

data type array name [row size][column size];




```
int a[2][3]; = {{1,2},{4,5,6}};
float emp[1][4];
int m[3][4];
```

* m[3][4] = m[3][4] * 10;

Array elements are stored in the form of matrix in different rows and columns.

Inner loop - Column
Outer loop - Row.

Eg: Write a C program to find sum of elements of 2D array.

1400, 1402

[][]
↳ 2 bytes

```
Void main()
{
  int a[50][50];   // 2500 x 2 = 5000 bytes.
  int row, col, sum=0;
  printf("Enter the row size & column size");
  scanf("%d%d", &row, &col);
  printf("Enter the elements of 2D array");
  for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  for(i=0; i<row; i++)
    for(j=0; j<col; j++)
    {
      sum = sum + a[i][j];
    }
  printf("Sum of elements of 2D array
                is %d", sum);
  getch(); for(i=0; i<row; i++)
  {
    for(j=0; j<col; j++)
    {
      printf("%d", a[i][j]);
    }
    printf("\n");
  }
  getch();
```

1) ith row
2) diagonal elements
3) Transpose.

2) Write a C program to find the sum of row of 2D array.

```
for (i=0; i<row; i++)
  for (j=0; j<col; j++)
    if (i==j)
    {
      sum = sum + a[i][j];
    }
}
```

Transpose:-

```
for (i=0; i<col; i++)
  for (j=0; j<row; j++)
  {
    printf ("%d", a[i][j]);
  }
  printf ("\n");
}
```

I/O functions:-

- formatted i/o
  - scanf
  - printf
- unformatted i/o
  - getchar()
  - putchar()
  - getch()
  - gety
  - puts

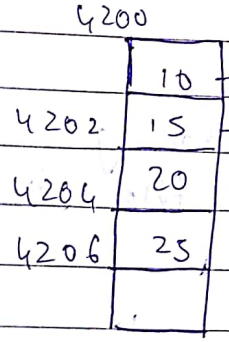
Strings :-

Collection of charactors ending with null character.

→ String is a collection of charactors ending with a null charactor.

Eg:- char name = "Gopal"; // compiler will add null charactor.

char place[6] = {'I', 'n', 'd', 'i', 'a', '\0'};



a[0] → 10

a[1] → 15

& a[1] → 4202.

Write a c program to find the length of the string

```
Void main()
{
  char str[30];
  int len, i;
  printf("Enter a string");
  gets(str); count=0; i=0;
  while(str[i]!='\0')
  {
    if((str[i]>='a'||str[i]<='x')
  } len=len+1; i++;
printf("Length of the string=%d", len);
}
```

Result :- Enter a string. Hai
Length of the string-3

Write a c program to count the no of characters 'a' present in the strings

```
Void main()
{
  char str[30];
  int count, i;
  printf("Enter 'a string");
  gets(str); count=0; i=0;
  while(str[i]!=\0)
  {
    if(str[i]=='a'||str[i]=='x')
```

}

printf("No of letters A present in : %d", count);
getch();
}

Result:- Enter a string AASHISH
No of letters A present = 2

String handling library functions -

getchar
gets
puts
putchar

Eg:-
```
#include <stdio.h>
Void main()
{
  char ch;
  puts("Enter a character");
  ch = getchar();
  putchar(ch);
}
```

getch → invisible to user
getchar → visible to user

Eg: Note :- \0 is not considered in string length function.

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  char str[10], "Hello";
  int len;
  len = strlen(str);
  printf("Length of the string = %d", len);
}
```

→ String length function is used to find the length of the string.

→ strcpy - It is used to copy one string to another string.

Syntax =

Strcpy (dest, source)

```
#include <stdio.h>
Void main()
{
  char str1[10] = "Hello", str2[10];
  strcpy (str2, str1);
```

ed in string

d", len;

find the

string

puts ("copied string is ");
puts (str2);
}

→ strcat (concatenation)

It is used to append 1 string to another string.

Eg: Syntax:

strcat (str1, str2);

```
#include <stdio.h>
#include <conio.h>
Void main()
{
    char str1[15] = "Hello";
    char str2[15] = "World";
    strcat (str1, str2);
    puts ("concatenated string");
    puts (str1);
}
```

→ strcmp (compare)

→ It is used to compare two string and
→ it returns 0 if two strings are equal
→ It returns +ve number if string1 is greater than string2
→ It returns -ve number if string1 is less than str2

Syntax

strcmp(s1, s2);
0 if equal
+ve if $s_1 \neq s_2$
-ve if $s_1 < s_2$

```
#include <stdio.h>
Void main()
{
   char str1[15] = "Hello";
   char str2[15] = "world";
if(strcmp(str1, str2) > 0)
   printf("Longest string is %s", str1);
else
   printf("Longest string is %s", str2);
}
```

→ strrev - It is used to reverse the string.

```
#include <stdio.h>
Void main()
{
   char str1[15] = "Hello"; char str2;
   printf("Reverse string = %s", strrev(str1));

   str2 = strrev(str1);
printf("Reverse string is %s", str2);
}
```

Write a c program to copy one string to another string without using library function.

```
#include <stdio.h>
Void main ()
{
  char str1[50] = "Hello";
  char str2[50];
  int i=0;
  while (str1[i] != '\0')
    {
      str2[i] = str1[i];
      i++;
    }
    str2[i] = '\0';
  Puts ("copied string is :%s", str2);
  getch();
}
```

Function :-

A function is a self-contained block of code which performs a particular task.

Syntax :-

return data type — function name (formal parameter list)

{

statements ;

}

Function has three parts :-

1) Function declaration
2) Function definition
3) Function call

```
#include <stdio.h>                          → function declaration
float area_tri ( float b, float h);
Void main()
{
  float base, height, res;
printf("Enter the base & height");
scanf("%f %f", &base, &height);

res = area_tri ( base, height);  → function call
printf("Area of triangle = %f", res);
 getch();
}
```

```
float area_tri (float b, float h)
{
    float area;
    area = 0.5 * b * b;
    return area;
}
```

function definition.

Using global variables -

```
#include <stdio.h>
Void area_tri(Void);
float b, h, area;
Void main()
{
  printf(" Enter base & height ");
  scanf("%f %f", &b, &h);
      area_tri();
  printf(" Area of triangle = %f", area);
  getch();
}
```

Funct

```
Void area_tri(Void)
{ area = 0.5 * b * h;
}
```

→ Local variables are the variables which are visible only with in the block of code in which it appears.

they are destroyed when the control goes out of the block.

→ Global variables are the variables which are declared out of the block of code.

Global variables have programs scope which means that they can be accessed everywhere in the program and they are destroyed when the program ends

Write a C program to find area of circle, area of rectangle, volume of sphere.

#

float area(

float.

```
#include <stdio.h>
#include <conio.h>
#define PI 3.14
Void area_cir (void);
void area_sph (void);
float r, area, volume;
Void main()
{
```

```
printf(" Enter the radius");
scanf("%f", &r);
    area_cir();
    &Volume_sph();
printf(" Area of circle = %f" area);
printf(" Volume of sphere = %f", vol);
getch();
}
```

→
```
Void area_cir (Void)
{
    area = 3.14 * r * r;
}
```

→
```
Void volume_sph (Void)
{
    Volume = (4/3) * 3.14 * r * r * r;
}
```

Structure

Eg:-

```
struct student
{
    int roll no;
    char name[20];
    float marks;
    long int phone;
};
```

Structure is a set of elements / variables of different datatypes.

Syntax:-

```
struct structure_name
{
    data type number 1;
    data type number 2;
    ---------
    ---------
    ---------
};
```

```
#include <stdio.h>
#include <conio.h>

Struct student
{
    int roll_no;
    char name[20];
    float marks;
    long int phone_no;
};
Void main()
{
  Struct student s1 = {34, "Priya", 75.4,
                       9430123456};
Struct student s2 = {46, "Sneha", 45.7,
                       9600999333};
Struct student s3;

  s3.roll_no = 44;
printf("Enter name");
scanf("%s", s3.name);
printf("Enter marks");
scanf("%f", &s3.marks);
  s3.phone_no = 9430111555;
printf("Student name = %s", s3.name);
printf("student marks = %f", s3.marks);
printf("student phone_no = %d", s3.phone_no);
  getch();
}
```

Write a C program to increase the salaries of employees by 5% whose Id is between whose salary is greater than 60k.

```
#include <stdio.h>
#include <conio.h>
struct emp
{
    int id;
    float salary;
    char name[20];
};
Void main()
{
    int N;
    struct emp e[100];
    printf("Enter no of employees");
    scanf("%d", &N);

    for(i=0; i<N; i++)
    {
        printf("Enter id");
        scanf("%d" &e[i].id)
        print("Enter employee salary");
        scanf("%f" &e[i].salary);
        printf("Enter name");
        scanf("%s", e[i].name);
    }
for(i=0; i<N; i++)
{
    if(e[i].Salary > 6000)
```

```
e[i].salary += 0.05 * e[i].salary;
}
printf(" Employee details ...");
printf("\n Employee id  Name  Salary");
for(i=0; i<N; i++)
{
printf("\n %d %s %f", e[i].id e[i].
                        name, e[i].salary);
}
getch();
}
```

Write a C program to create a students structure and display the student details who have scored marks < 40

```
#include <stdio.h>
#include <conio.h>
Struct students;
{
    int id;
    float marks;
    char name[20];
};
Void main()
{
    int N;
    struct students s[100];
printf(" Enter the no of students");
scanf("%d", &N);
for(i=0; i<N; i++)
{
```

```
Printf(" Enter id");
scanf("%d", &s[i].id);
Printf(" Enter the name")
scanf("%s", s[i].name);
printf(" Enter the marks"),
scanf("%f", &s[i].marks);
}
for(i=0; i<N; i++)
{
  if(s[i].marks < 40)
  { printf(" %d %s %f", s[i].rollno,
                        s[i].name, s[i].marks);
  }
}
getch();
}
```

Pointers

```
int *p;
int a[4] = {10, 11, 12, 13};
```

address

```
p = &a[0];

*p = *p + 5
```

value

Pointer is the variable that stores the address of another variable.

Syntax:-

data type * pointer Variable name;

Eg:- pointer to a int Variable:-

```
int z = 20;
int * p;   // Size of p = 2 bytes.
p = &z;
*p = *p + 10;  // z becomes 30.
```

Eg:- pointer to a float Variable:-

```
float pi = 3.14;
float * ptr;
ptr = &pi;
*ptr = 3.1469;
```

Eg:-

```
int * p;  // size of p - 2 bytes
int a[4] = {10, 11, 12, 13};
p = &a[0]; // or p = a;
*p = *p + 2;  // a[0] = 12
```

classmate
Date
Page

Pointer

* (reference operator) - It is used to link pointer to another operator.

Dreference operator is used to access the value of a variable pointed by a pointer.

Write a C program to swap 2 numbers using functions:-

Pass by-value Method :-

```
#include <stdio.h>
Void main()   Void swap (int x, int y);
{
   int a, b;
printf (" Enter 2 No's");
scanf ("%d %d", &a, &b);
   swap (a, b);
printf (" Swapped no's are %d %d", a, b);
 getch();
}
```

```
void swap(int x, int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

Pass by reference Method:-

```
#include <stdio.h>
void swap(int x, int y);
void main()
{
    int a, b;
    printf("Enter 2 No's");
    scanf("%d%d", &a, &b);
    swap(&a, &b); // actual parameters
    printf("Swapped no's are %d %d", a, b);
    getch();
}

void swap(int *x, int *y)   // formal parameter
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
```

Pass-by Value Method:-

In pass-by value method, the actual parameter values are copied to formal parameter value.

Pass-by reference Method:-

In pass-by reference method, the address of the actual parameters are passed to the formal parameter.

This method is used when the data needs to be modified in the calling function.

* Actual parameters:-

the parameters used in the function call are called actual parameters.

* The parameters used in function definition are called formal parameters.

Typedef:-

Typedef ~~is use~~ is another name used for a user defined datatype or derived datatype.

Eg:-

```
struct student
{
    int roll.no;
    char name[20];
    float marks;
};
typedef struct student stud;
void main()
{
    stud s1, s2;
    stud s3, s4;
```

Write a c program to increase the value of length and breadth declared in the main program by pass by reference method by 10.