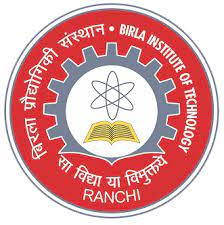
**Birla institute of technology Mesra, Ranchi**

Numerical method



Prepared by :

Robin  
ROLL NO : BTECH/15138/19

Faculty in charge

Dr. Prakash Chandra srivastava

Patna off campus

2021

**Newton Forward Method**

**Algorithm:**

1. Start

2. Read number of data (n)

3. Read data points for x and y:

For i = 0 to n-1

Read Xi and Yi,0

Next i

4. Read calculation point where derivative is required (xp)

5. Set variable flag to 0

6. Check whether given point is valid data point or not. If it is valid point then get its position at variable index

For i = 0 to n-1

If |xp - Xi| < 0.0001

index = i

flag = 1

break from loop

End If

Next i

7. If given calculation point (xp) is not in x-data then terminate the process.

If flag = 0

Print "Invalid Calculation Point"

Exit

End If

8. Generate forward difference table

For i = 1 to n-1

For j = 0 to n-1-i

Yj,i = Yj+1,i-1 - Yj,i-1

Next j

Next i

9. Calculate finite difference: h = X1 - X0

10. Set sum = 0 and sign = 1

11. Calculate sum of different terms in formula to find derivatives using Newton's forward difference formula:

For i = 1 to n-1-index

term = (Yindex, i)i / i

sum = sum + sign \* term

sign = -sign

Next i

12. Divide sum by finite difference (h) to get result first\_derivative = sum/h

13. Display value of first\_derivative

14. Stop

Code:

#include<stdio.h>

#include<conio.h>

#define MAXN 100

#define ORDER 4

main()

{

float ax[MAXN+1], ay [MAXN+1], diff[MAXN+1][ORDER+1], nr=1.0, dr=1.0,x,p,h,yp;

int n,i,j,k;

printf("\nEnter the value of n:\n");

scanf("%d",&n);

printf("\nEnter the values in form x,y:\n");

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

scanf("%f %f",&ax[i],&ay[i]);

printf("\nEnter the value of x for which the value of y is wanted: \n");

scanf("%f",&x);

h=ax[1]-ax[0];

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

diff[i][1] = ay[i+1]-ay[i];

for (j=2;j<=ORDER;j++)

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

diff[i][j] = diff[i+1][j-1] - diff[i][j-1];

i=0;

while (!(ax[i]>x))

i++;

i--;

p = (x-ax[i])/h;

yp = ay[i];

for (k=1;k<=ORDER;k++)

{

nr \*=p-k+1;

dr \*=k;

yp +=(nr/dr)\*diff[i][k];

}

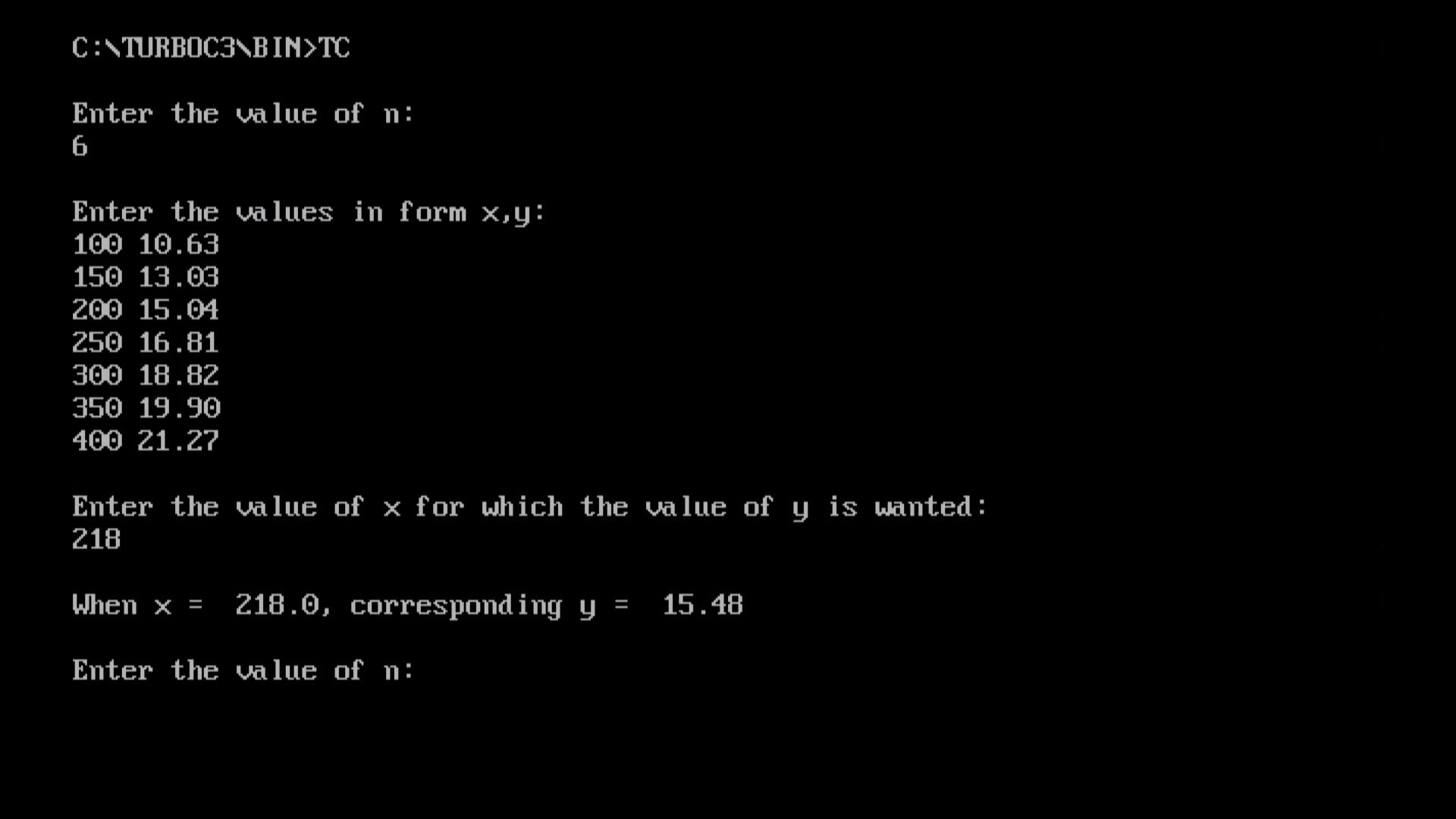
printf("\nWhen x = %6.1f, corresponding y = %6.2f\n",x,yp);

getch();

return 0;

}

**Output:**



**Newton Backward Method**

Algorithm:

1. Start

2. Read number of data (n)

3. Read data points for x and y:

For i = 0 to n-1

Read Xi and Yi,0

Next i

4. Read calculation point where derivative is required (xp)

5. Set variable flag to 0

6. Check whether given point is valid data point or not.

If it is valid point then get its position at variable index

For i = 0 to n-1

If |xp - Xi| < 0.0001

index = i

flag = 1

break from loop

End If

Next i

7. If given calculation point (xp) is not in

x-data then terminate the process.

If flag = 0

Print "Invalid Calculation Point"

Exit

End If

8. Generate backward difference table

For i = 1 to n-1

For j = n-1 to i (Step -1)

Yj,i = Yj,i-1 - Yj-1,i-1

Next j

Next i

9. Calculate finite difference: h = X1 - X0

10. Set sum = 0

11. Calculate sum of different terms in formula

to find derivatives using Newton's backward

difference formula:

For i = 1 to index

term = (Yindex, i)i / i

sum = sum + term

Next i

12. Divide sum by finite difference (h) to get result

first\_derivative = sum/h

13. Display value of first\_derivative

14. Stop

**Code:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

float x[10],y[10][10],sum,p,u,temp;

int i,n,j,k=0,f,m;

float fact(int);

clrscr();

printf("\nhow many record you will be enter: ");

scanf("%d",&n);

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

{

printf("\n\nenter the value of x%d: ",i);

scanf("%f",&x[i]);

printf("\n\nenter the value of f(x%d): ",i);

scanf("%f",&y[k][i]);

}

printf("\n\nEnter X for finding f(x): ");

scanf("%f",&p);

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

{

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

{

y[i][j]=y[i-1][j]-y[i-1][j-1];

}

}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\n x(i)\t y(i)\t y1(i) y2(i) y3(i) y4(i)");

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

{

printf("\n %.3f",x[i]);

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

{

printf(" ");

printf(" %.3f",y[j][i]);

}

printf("\n");

}

i=0;

do

{

if(x[i]<p && p<x[i+1])

k=1;

else

i++;

}while(k != 1);

f=i+1;

u=(p-x[f])/(x[f]-x[f-1]);

printf("\n\n u = %.3f ",u);

n=n-i+1;

sum=0;

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

{

temp=1;

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

{

temp = temp \* (u + j);

}

m=fact(i);

sum = sum + temp\*(y[i][f]/m);

}

printf("\n\n f(%.2f) = %f ",p,sum);

getch();

}

float fact(int a)

{

float fac = 1;

if (a == 0)

return (1);

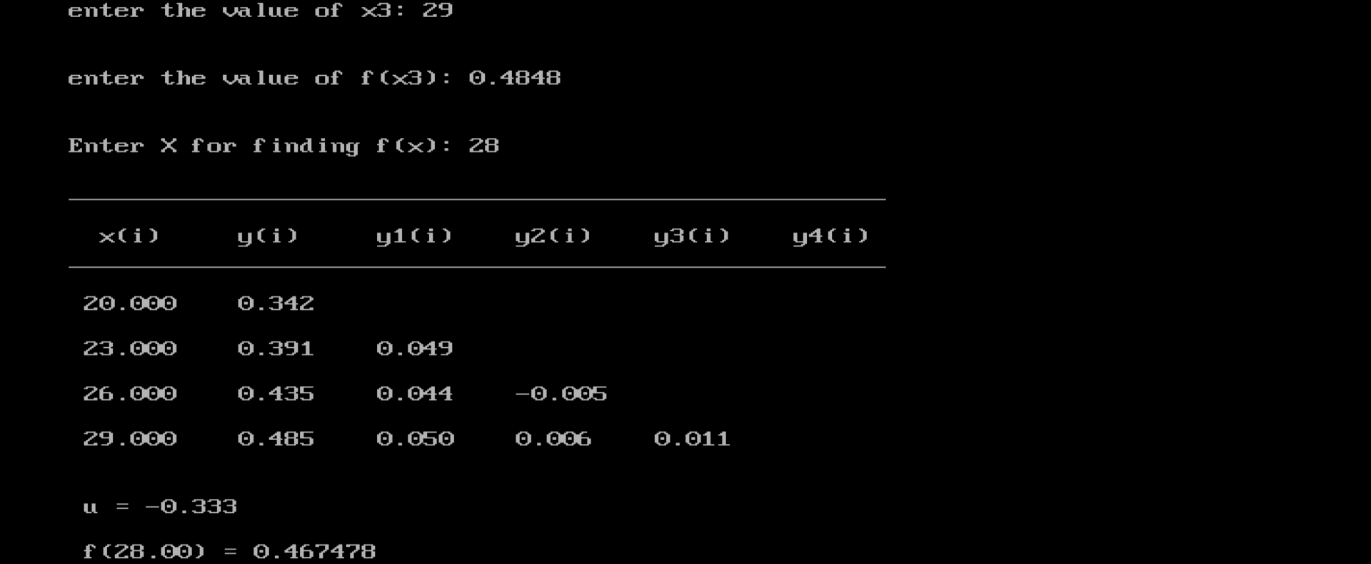
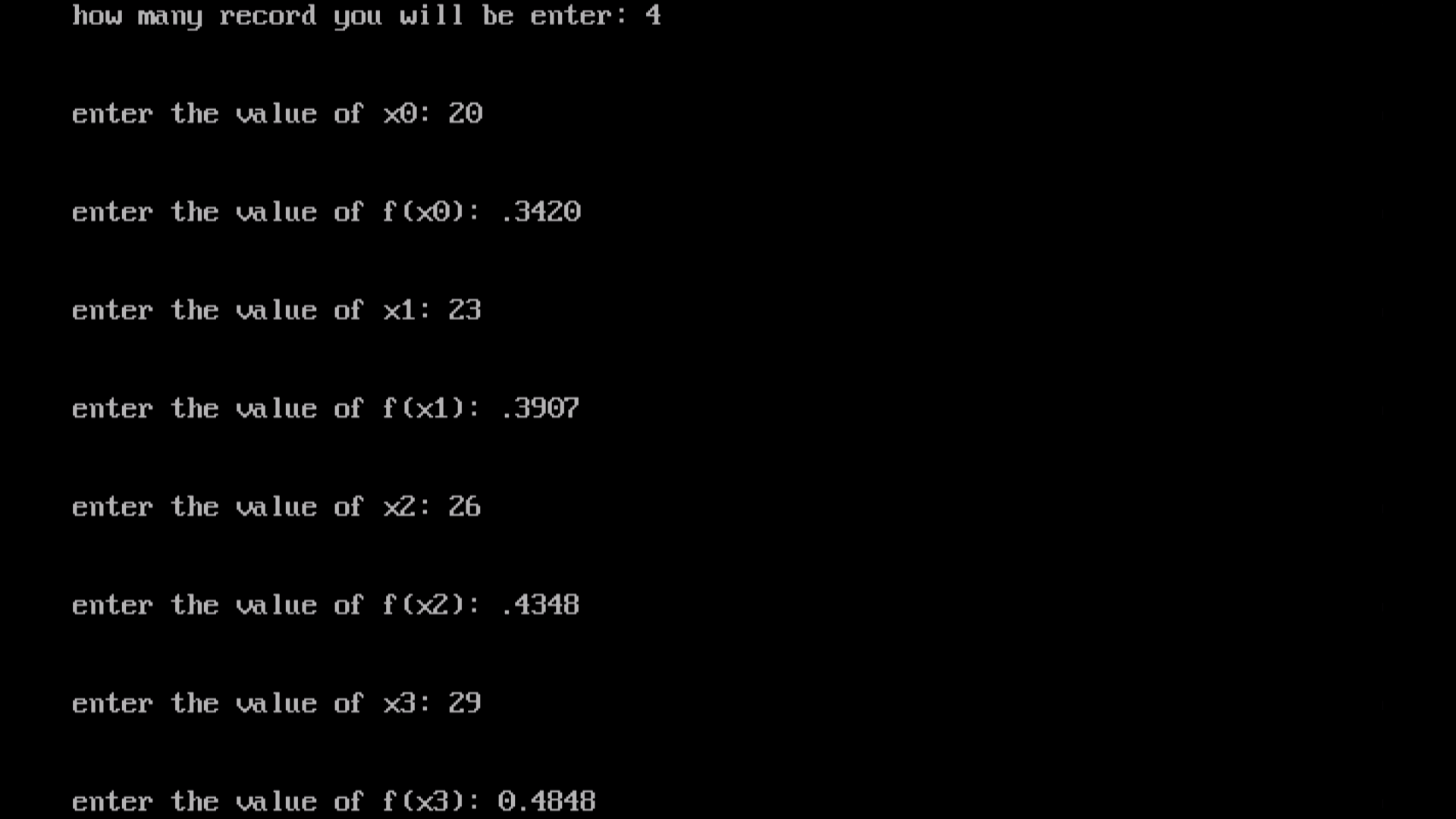
else

fac = a \* fact(a-1);

return(fac);

}

**Output:**



Exp-3 Gauss Forward Interpolation Formula

**Gauss Forward method**

**Algorithm:**

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h=ax[1]-ax[0]

Step-6.fori=0;i<n–1;i++

Step-7.diff[i][1]=ay[i+1]-ay[i]

Step-8. End Loop i

Step-9.for j=2;j<=4;j++

Step-10.fori=0;i<n–j;i++

Step-11.diff[i][j]=diff[i+1][j–1]–diff[i][j–1]

Step-12. End Loop i

Step-13. End Loop j

Step-14.i=0

Step-15. Repeat Step 16 until ax[i]<x

Step-16.i=i+1

Step-17.i=i–1;

Step-18. p=(x–ax[i])/h

Step-19. y1=p\*diff[i][1]

Step-20. y2=p\*(p–1)\*diff[i–1][2]/2

Step-21. y3=(p+1)\*p\*(p-1)\*diff[i–2][3]/6

Step-22. y4=(p+1)\*p\*(p–1)\*(p–2)\*diff[i–3][4]/24

Step-23. y=ay[i]+y1+y2+y3+y4

Step-24. Print Output x,y

Step-25.End of Program.

**Code:**

# include <stdio.h>

# include <conio.h>

# include <math.h>

# include <process.h>

# include <string.h>

void main()

{

int n;

int i,j;

float ax[10];

float ay[10];

float x;

float nr,dr;

float y=0; float h;

float p;

float diff[20][20];

float y1,y2,y3,y4;

clrscr();

printf(" Enter the number of terms - ");

scanf("%d",&n);

printf("\n Enter the value in the form of x - ");

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

{

printf(" Enter the value of x%d - ",i+1);

scanf("%f",&ax[i]);

}

printf(" Enter the value in the form of y - ");

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

{

printf("Enter the value of y%d - ",i+1);

scanf("%f",&ay[i]);

}

printf("\nEnter the value of x for - ");

printf("\nwhich you want the value of y - ");

scanf ("%f",&x);

h=ax[1]-ax[0];

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

{

diff[i][1]=ay[i+1]-ay[i];

}

for(j=2;j<=4;j++)

{

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

{

diff[i][j]=diff[i+1][j-1]-diff[i][j-1];

} }

i=0;

do {

i++;

}

while(ax[i]<x);

i--;

p=(x-ax[i])/h;

y1=p\*diff[i][1];

y2=p\*(p-1)\*diff[i-1][2]/2;

y3=(p+1)\*p\*(p-1)\*diff[i-2][3]/6;

y4=(p+1)\*p\*(p-1)\*(p-2)\*diff[i-3][4]/24;

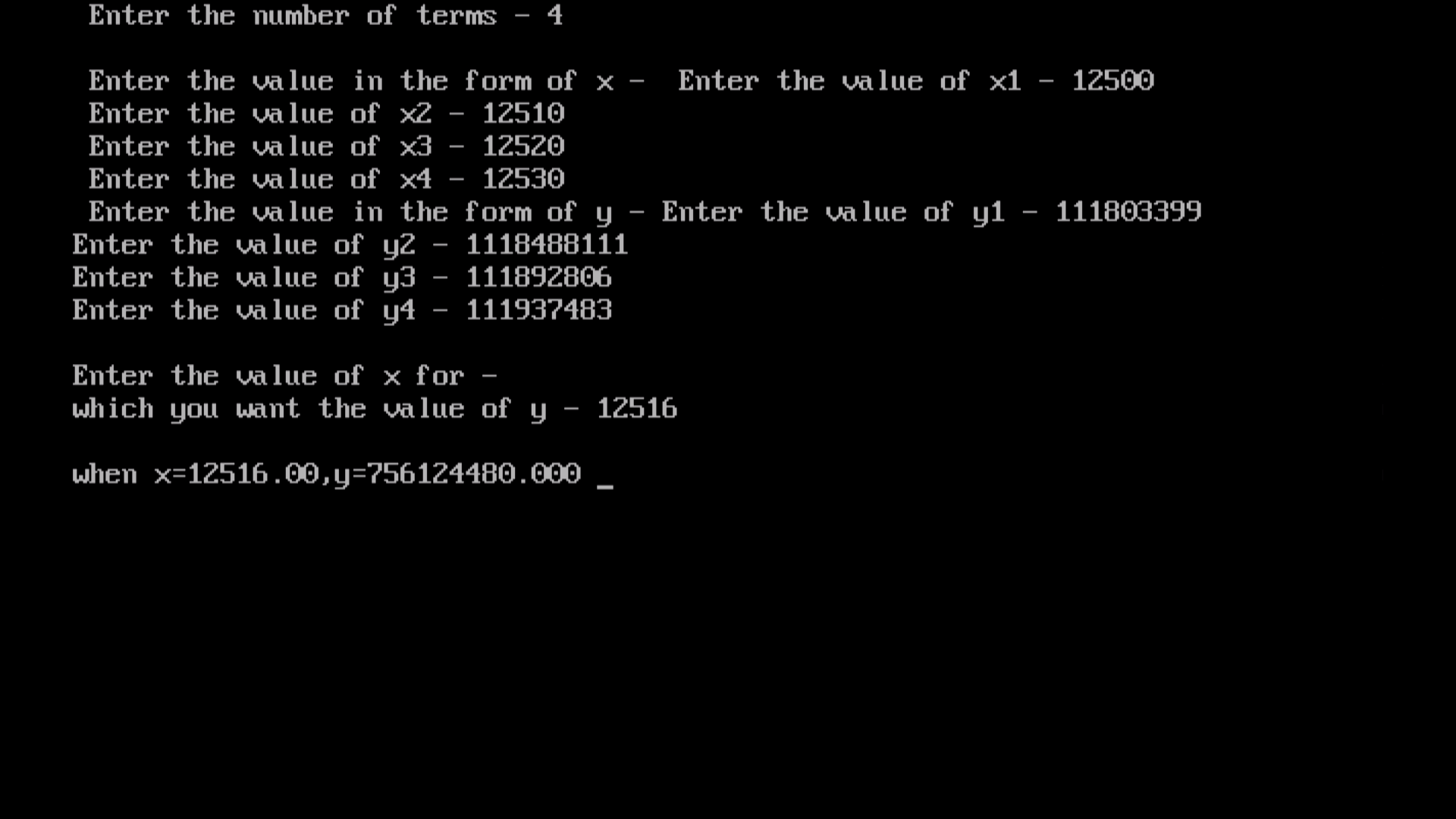
y=ay[i]+y1+y2+y3+y4;

printf("\nwhen x=%6.2f,y=%6.3f ",x,y);

getch();

}

**Output:**



**Sterling’s formula**

**Algorithm:**

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h = ax[1]-ax[0]

Step-6.fori = 1;i < n-1; i++

Step-7.diff [i][1] = ay[i + 1]-ay[i]

Step-8. End loop i

Step-9.for j = 2; j < = 4; j++

Step-10.fori = 0; i< n-j; i++

Step-11.diff[i][j] = diff[i + 1][j-1]-diff[i][j-1]

Step-12. End loop i

Step-13. End loop j

Step-14.i = 0

Step-15. Repeat until ax[i] < x

Step-16.i = i + 1

Step-17.i = i-1;

Step-18. p = (x-ax[i])/h

Step-19. y1= p\*(diff[i][1] + diff[i-1][1])/2

Step-20. y2 = p\*p\*diff[i-1][2]/2

Step-21. y3 = p\*(p\*p-1)\*(diff[i-1][3]+diff[i-2][3])/6

Step-22. y4 = p\*p\*(p\*p-1)\*diff[i-2][4]/24

Step-23. y = ay[i]+y1 + y2 + y3 + y4

Step-24. Print output

Step-25. End of program

**Code:**

#include<stdio.h>

#include<conio.h>

void main()

{

float ax[30],ay[30],h,x,y,t1=1,t2=1,u;

int n,i,j,m,k;

clrscr();

printf("enter the value of n\n");

scanf("%d",&n);

printf("\n enter length of each interval \n");

scanf("%f",&h);

printf("enetr the value of x and y \n");

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

{

scanf("%f %f",&ax[i],&ay[i]);

}

printf("enter the value of x for which value of y is wanted\n");

scanf("%f",&x);

printf("\n enter the location of x0 i.e k\n");

scanf("%d",&k);

y=ay[k];

u=(x-ax[k])/h;

m=n;

if (k<=n/2)

n=2\*k;

else

n=2\*(n-k);

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

{

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

ay[j]=ay[j+1]-ay[j];

if(i%2!=0)

{

t1=(t1\*(u-i/2))/i;

t2=(t2\*(u+i/2))/i;

}

else

{

t1=(t1\*(u+i/2))/i;

t2=(t2\*(u-i/2))/i;

}

y=y+(t1\*ay[k-(i+1)/2]+t2\*ay[k-i/2])/2;

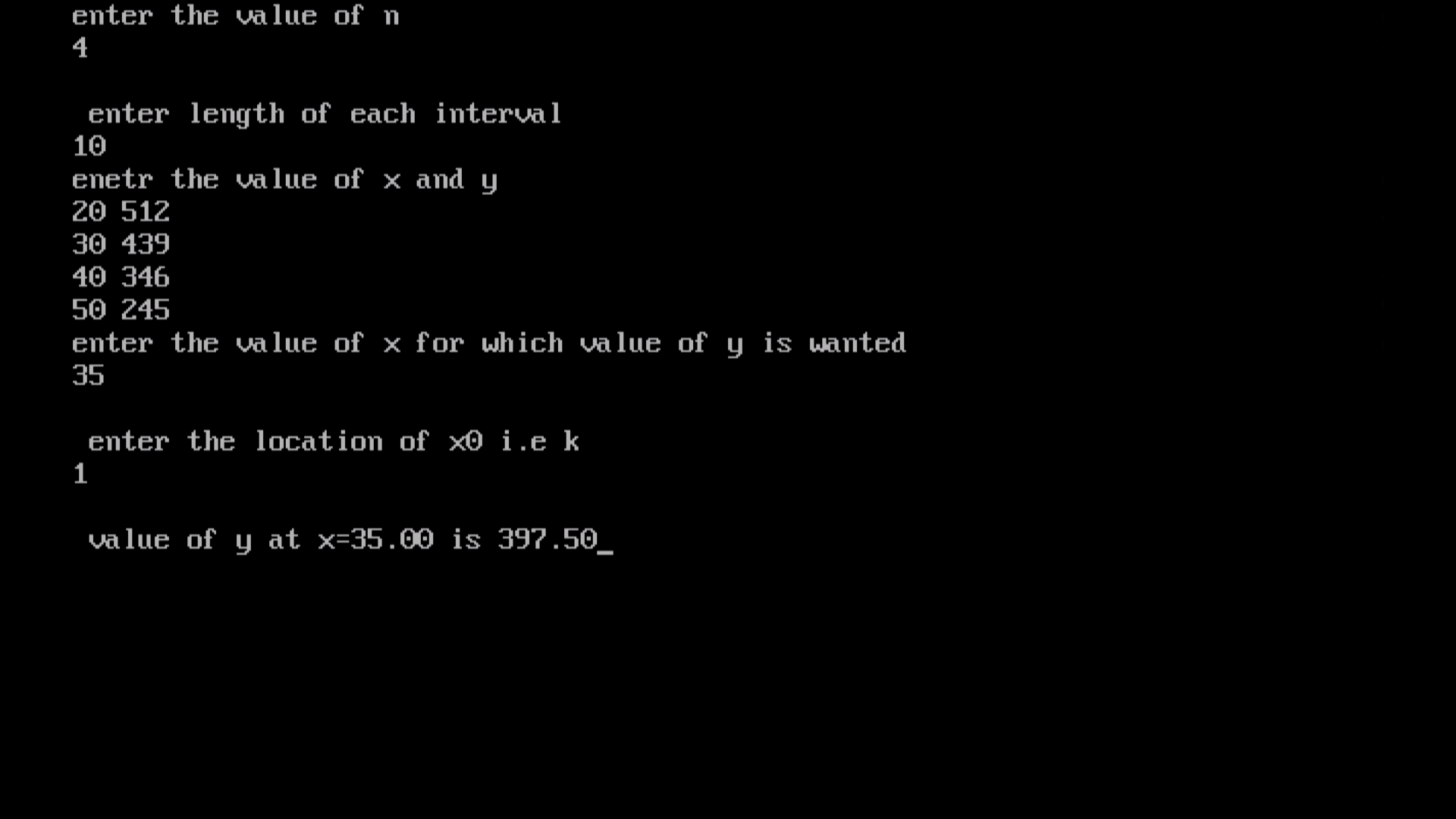
}

printf("\n value of y at x=%.2f is %.2f",x,y);

getch();

}

Output:



**Lagrange’s Interpolation Formula For Unequal Interval**

**Algorithm:**

Step-1. Start of the program

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5.fori=0; i<n; i++

Step-6.nr=1

Step-7.dr=1

Step-8.for j=0; j<n; j++

Step-9.if j !=i

a. nr=nr\*(x-ax[j])

Step-10.b.dr\*(ax[i]-ax[j])

Step-11. End Loop j

Step-12. y+=(nr/dr)∗ay[i]

Step-13. End Loop i

Step-14. Print Output x, y

Step-15. End of Program

**Code:**

#include<stdio.h>

#include<conio.h>

#define MAX 10

void main()

{

float x[MAX],y[MAX],k=0,z,nr,dr;

int i,j,m;

//clrscr();

printf("\n enter the range ");

scanf("%d",&m);

printf("\n enter the x value ");

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

scanf("%f",&x[i]);

printf("\n enter the y value ");

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

scanf("%f",&y[i]);

printf("\n enter value of x for which respective y is to be calculated ");

scanf("%f",&z);

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

{ nr=1;dr=1;

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

{

if (j!=i)

{

nr=nr\*(z-x[j]);

dr=dr\*(x[i]-x[j]);

} }

k=k+((nr/dr)\*y[i]);}

printf("\n final result=%f\n",k);

getch();

}

**Output:**

