UNIT 3 C PROGRAMS

LINEAR INTERPOLATION ALGORITHM AND CODE IN C

Interpolation is the process of estimation of an unknown data by analyzing the given reference data. For the given data, (say 'y' at various 'x' in tabulated form), the 'y' value corresponding to 'x' values can be found by interpolation.

Linear Interpolation Algorithm:

1. Start
2. Input two values of x1 & x2 between which unknown value of 'x' lies
3. Input f1 = f(x1) and f2 = f(x2)
4. Input the value of 'x' for which unknown function ' f ' you want to find.
5. Compute: f = f1 + ((f2 - f1) * (x - x1) / (x2 - x1))
6. Output the value of 'f ' as a result.
7. Stop

Below is a source code in **C program for Linear Interpolation** to find the square root of 2.5 from the given data for square root of various integers.

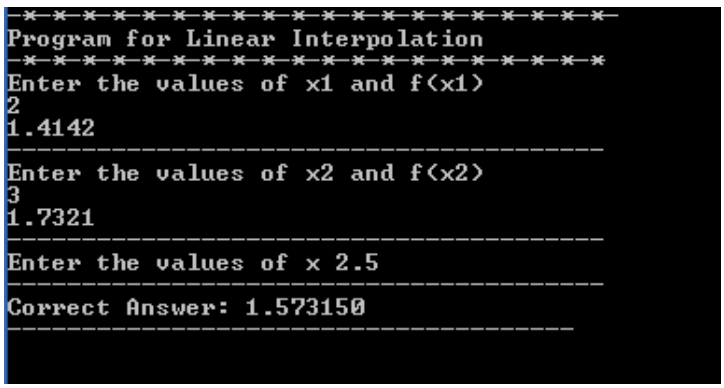
x	1	2	3	4	5
f(x)	1	1.4142	1.7321	2	2.2361

Source Code for Linear Interpolation in C:

```
1. /*Linear Interpolation*/
2. #include<stdio.h>
3.
4. #include<conio.h>
5.
6. void main()
7.
8. {
9.
10.     float x2,x1,x,f1,f2,f;
```

```
11.
12.     puts("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-");
13.
14.     puts("Program for Linear Interpolation");
15.
16.     puts("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*");
17.
18.     printf("Enter the values of x1 and f(x1)\n");
19.
20.     scanf("%f %f",&x1,&f1);
21.
22.     puts("----------------------------------------");
23.
24.     printf("Enter the values of x2 and f(x2)\n");
25.
26.     scanf("%f %f",&x2,&f2);
27.
28.     puts("----------------------------------------");
29.
30.     printf("Enter the values of x ");
31.
32.     scanf("%f",&x);
33.
34.     puts("----------------------------------------");
35.
36.     f=f1+((f2-f1)*(x-x1)/(x2-x1));
37.
38.     printf("Correct Answer:\t%f",f);
39.
40.     puts("\n----------------------------------------");
41.
42.     getch();
```

Input/Output:



LAGRANGE INTERPOLATION ALGORITHM, FLOWCHART AND CODE IN C

In case of equally spaced 'x' values, a number of interpolation methods are available such as the *Newton's forward and backward interpolation, Gauss's forward and backward interpolation, Bessel's formula, Laplace-Everett's formula* etc. But, all these methods fail when the spacing of 'x' is unequal. In such case, ***Lagrange interpolation*** is one of the best options.

Lagrange interpolation Algorithm:

1. Start
2. Enter the total number of given data.
3. Enter all the values of x and corresponding y in one dimensional arrays
4. Input the value of 'x' at which the unknown value of 'y' is to be interpolated.
5. Compute: prod=prod*(X-x[j])/(x[i]-x[j]) in user defined function "Li", where "prod" corresponds to Lagrange interpolation function $l_i(x)$.
6. Now compute Lagrange interpolation formula for polynomials: sum=sum+Li(i,n,x,X)*y[i] . in user defined function "Pn".
7. Output the unknown value of 'y' corresponding to 'x' as a result.
8. Stop

The source code given below in C program for Lagrange interpolation is for interpolating data in which augments are unequally spaced or in cases where it is not possible to fit the curve of given data.

Source Code for Lagrange interpolation in C:

```
/***********************************
******LAGRANGE INTERPOLATION*******
***********************************/
#include<stdio.h>

/*Function to evaluate Li(x)*/

double Li(int i, int n, double x[n+1], double X)
{
int j;
double prod=1;
for(j=0;j<=n;j++){
```

```
10.      if(j!=i)
11.      prod=prod*(X-x[j])/(x[i]-x[j]);
12.      }
13.      return prod;
14.      }
15.
16.      /*Function to evaluate Pn(x) where Pn is the Lagrange
   interpolating polynomial of degree n*/
17.
18.      double Pn(int n, double x[n+1], double y[n+1], double X){
19.      double sum=0;
20.      int i;
21.      for(i=0;i<=n;i++){
22.      sum=sum+Li(i,n,x,X)*y[i];
23.      }
24.      return sum;
25.      }
26.      main(){
27.      int i,n;  //n is the degree
28.      printf("*********************************\n******LAGRANGE
   INTERPOLATION*******\n*********************************\n\n");
29.      printf("Enter the number of data-points:\n");
30.      scanf("%d",&n);  //no. of data-points is n+1
31.      n=n-1;
32.      //Arrays to store the (n+1) x and y data-points of size n+1
33.      double x[n+1];
34.      double y[n+1];
35.      printf("Enter the x data-points:\n");
36.      for(i=0;i<n+1;i++){
37.      scanf("%lf",&x[i]);
38.      }
39.
40.      printf("Enter the y data-points:\n");
41.      for(i=0;i<n+1;i++){
42.      scanf("%lf",&y[i]);
43.      }
44.
45.      double X;  //value of x for which interpolated value is
   required
46.      printf("Enter the value of x for which you want the
   interpolated value of y(x):\n");
47.      scanf("%lf",&X);
48.      printf("The interpolated value is %lf",Pn(n,x,y,X));
49.      }
```

Input/Output:

```
**********************************
******LAGRANGE INTERPOLATION*******
**********************************

Enter the number of data-points:
3
Enter the x data-points:
2       3       4
Enter the y data-points:
1.4142  1.7321  2
Enter the value of x for which you want the interpolated value of y(x):
2.5
The interpolated value is 1.579400
Process returned 0 (0x0)   execution time : 31.602 s
Press any key to continue.
```

Fitting Linear Curve using Least Square Method ALGORITHM, FLOWCHART AND CODE IN C

Algorithm :

Step 1: Start

Step 2: Enter the value for total number of data, n.

Step 3: Input the values of x and corresponding y. In the program, x and y are defined as array. Therefore, x and y are input using for loop.

Step 4: Calculates the sum of x, y, xy, x^2 etc.

Step 5: Calculate a and b using following formula:

$$b= (n\sum xy - \sum y \sum x)/(n\sum x2-(\sum x)2)$$

$$a=(\sum y - b\sum x)/n$$

Step 6: Output the value of a & b

Step 7: Output the required linear equation in the form y=a+bx

Step 8: Stop

Below is a source code in **C program for Fitting Linear equation y=a+bx using Least Square Method** for the given data.

Velocity(m/s)	10	20	30	40	50	60	70	80

Force (N)	24	68	378	552	608	1218	831	1452

Source Code for Fitting Linear cuve using Least Square Method in C:

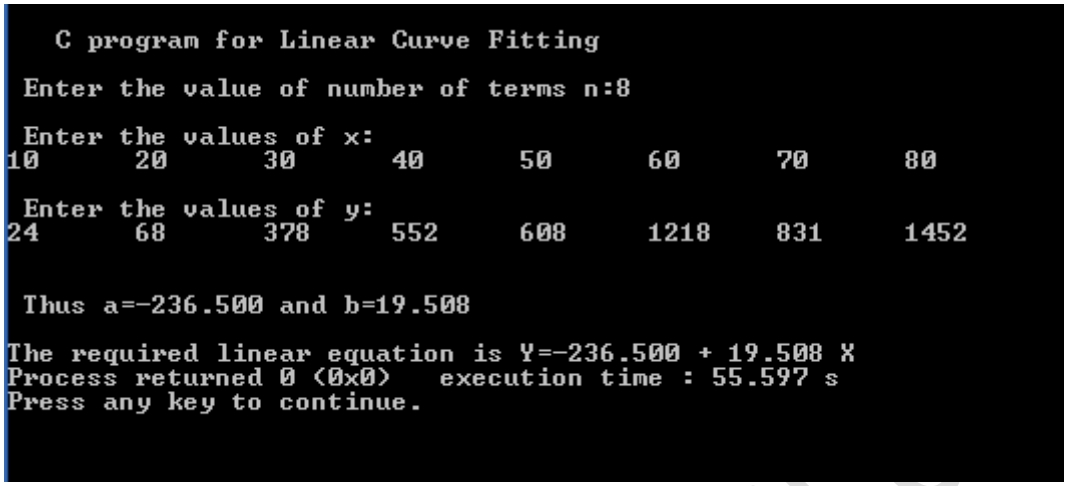
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{

    int n,i,x[20],y[20],sumx=0,sumy=0,sumxy=0,sumx2=0;
    float a,b;
    printf("\n   C program for Linear Curve Fitting \n ");
        printf("\n Enter the value of number of terms n:");
        scanf("%d",&n);
        printf("\n Enter the values of x:\n");
        for(i=0;i<=n-1;i++)
        {
            scanf(" %d",&x[i]);

        }
        printf("\n Enter the values of y:\n");
        for(i=0;i<=n-1;i++)
        {
            scanf("%d",&y[i]);
        }
        for(i=0;i<=n-1;i++)
        {
            sumx=sumx +x[i];
            sumx2=sumx2 +x[i]*x[i];
            sumy=sumy +y[i];
            sumxy=sumxy +x[i]*y[i];

        }
        a=((sumx2*sumy -sumx*sumxy)*1.0/(n*sumx2-sumx*sumx)*1.0);
        b=((n*sumxy-sumx*sumy)*1.0/(n*sumx2-sumx*sumx)*1.0);
        printf("\n\n Thus a=%3.3f and b=%3.3f",a,b);
        printf("\n\nThe required linear equation is Y=%3.3f +
  %3.3f X",a,b);
        return(0);
    }
```

Input/Output:



Fitting Exponential or transcendental equation Curve using Least Square Method ALGORITHM, FLOWCHART AND CODE IN C

The working principle of curve fitting C program as exponential equation is also similar to linear but this program first converts exponential equation into linear equation by taking log on both sides

Algorithm :

Step 1: Start

Step 2: Enter the value for total number of data, n.

Step 3: Input the values of x and corresponding y. In the program, x and y are defined as array. Therefore, x and y are input using for loop.

Step 4: Calculates the sum of x, y, logy, xlogy & x2.

Step 5: Calculate a and b for $y = ae^{(bx)}$ which gives $\ln y = bx + \ln a$ using following formula:

$$b = (n\sum x \ln y - \sum \ln y \sum x)/(n\sum x2 - (\sum x)2)$$

$$\log a = R = (\sum \ln y - b\sum x)/n$$

$$a = e^R$$

Step 6: Output the value of a & b

Step 7: Output the required linear equation in the form $y=ae^{bx}$

Step 8: Stop

Below is a source code in **C program for Fitting exponential equation $y=ae^{bx}$ using Least Square Method** for the given data.

x	1	2	3	4
y	1.65	2.70	4.50	7.35

Source Code for Fitting exponential equation using Least Square Method in C:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{

    int n,i;
    float Y[20],sumx=0,sumy=0,sumxy=0,sumx2=0,x[20],y[20];
    float a,b,A;
         printf("\n  C program for Exponential Curve fitting\n");
         printf("\n Enter the value of number of terms n:");
         scanf("%d",&n);
         printf("\n Enter the values of x:\n");
         for(i=0;i<=n-1;i++)
         {
             scanf("%f",&x[i]);

         }
         printf("\n Enter the values of y:\n");
         for(i=0;i<=n-1;i++)
         {
             scanf("%f",&y[i]);
         }
         for(i=0;i<=n-1;i++)
         {
             Y[i]=log(y[i]);
         }
         for(i=0;i<=n-1;i++)
         {
             sumx=sumx +x[i];
             sumx2=sumx2 +x[i]*x[i];
```

```
32.             sumy=sumy +Y[i];
33.             sumxy=sumxy +x[i]*Y[i];
34.
35.         }
36.         A=((sumx2*sumy -sumx*sumxy)*1.0/(n*sumx2-sumx*sumx)*1.0);
37.         b=((n*sumxy-sumx*sumy)*1.0/(n*sumx2-sumx*sumx)*1.0);
38.         a=exp(A);
39.         printf("\n\n Thus a=%3.3f and b=%3.3f",a,b);
40.         printf("\n\n The best fit for the curve is Y=
   %4.3fe^%4.3fX",a,b);
41.         return(0);
42.     }
```

Input/Output:

```
 C program for Exponential Curve fitting

 Enter the value of number of terms n:4

 Enter the values of x:
1       2       3       4

 Enter the values of y:
1.65    2.70    4.50    7.35


 Thus a=1.000 and b=0.499

 The best fit for the curve is Y= 1.000e^0.499X
Process returned 0 (0x0)   execution time : 44.635 s
Press any key to continue.
```

THE END