**Linear Interpolation in C - Documentation**

**1. Introduction**

Linear Interpolation is a numerical method used to estimate the value of a function between two known data points. This document explains the implementation of Linear Interpolation in C, detailing its features, functionality, and how to use it.

**2. Features**

* Allows user to input polynomial equations dynamically.
* Implements **Linear Interpolation** for estimating function values.
* Customizable **error ratio** for precise approximations.
* Efficient and easy-to-use approach for interpolation problems.

**3. Installation & Usage**

**3.1 Cloning the Repository**

To get the project, use the following command:

git clone https://github.com/1230505039/Linear-Interpolation.git

cd Linear-Interpolation

**3.2 Compiling the Program**

Compile the C file using GCC:

gcc -o lineerInterpolation lineerInterpolation.c -lm

**3.3 Running the Program**

Execute the compiled program:

./lineerInterpolation

**3.4 User Input Prompts**

Once the program starts, the user will be prompted to enter:

* The number of terms in the polynomial (based on its degree).
* Coefficients of the polynomial.
* Two data points (**x1, y1**) and (**x2, y2**) for interpolation.
* Error ratio (optional).

**4. Methodology**

**4.1 How Linear Interpolation Works**

1. Given two points **(x1, y1)** and **(x2, y2)**, the program finds the equation of the line passing through these points.
2. It uses the interpolation formula:

y=y1+(x−x1)(y2−y1)(x2−x1)y = y1 + \frac{(x - x1) (y2 - y1)}{(x2 - x1)}

to compute the estimated function value at any given **x**.

1. The program iterates until the approximation meets the given error margin.
2. The estimated function value is displayed as output.

**4.2 Example Input/Output**

Enter the number of terms (Based on max degree): 3

Enter coefficient for x^2: 1

Enter coefficient for x^1: -3

Enter coefficient for x^0: 2

Enter first point (x1): 1

Enter second point (x2): 3

Estimated function value at x=2: 1.500000

**5. Code Explanation**

**5.1 Key Functions**

**findRootLine()**

This function calculates the root of the secant line formed by two points:

double findRootLine(double x1, double x2, double y1, double y2) {

double slope = (y2 - y1) / (x2 - x1);

double fixedNumber = y1 - slope \* x1;

double root = (0 - fixedNumber) / slope;

return root;

}

**findPointsOnFunc()**

This function evaluates the function value for a given **x**:

double findPointsOnFunc(double x, double equation[], int size) {

double y = 0;

for (int i = 0; i < size; i++) {

y += pow(x, size - i - 1) \* equation[i];

}

return y;

}

**Main Function**

The main function handles user input, calls the interpolation function, and displays the result.

int main(){

system("cls");

int numberOfTerms;

printf("Enter the number of terms (Based on max degree): ");

scanf("%d", &numberOfTerms);

double equation[numberOfTerms];

for (int i = 0; i < numberOfTerms; i++) {

printf("Enter %d. term: ", i + 1);

scanf("%lf", &equation[i]);

}

double bottomLimit;

double topLimit;

printf("Enter first point: ");

scanf("%lf", &bottomLimit);

printf("Enter second point: ");

scanf("%lf", &topLimit);

double myErrorRatio;

printf("Enter error ratio: ");

scanf("%lf", &myErrorRatio);

double errorRatio;

double candidateRoot;

do {

double y1 = findPointsOnFunc(bottomLimit, equation, numberOfTerms);

double y2 = findPointsOnFunc(topLimit, equation, numberOfTerms);

candidateRoot = findRootLine(bottomLimit, topLimit, y1, y2);

if(findPointsOnFunc(candidateRoot, equation, numberOfTerms) < 0) {

bottomLimit = candidateRoot;

} else {

topLimit = candidateRoot;

}

double error1 = candidateRoot - bottomLimit;

double error2 = topLimit - candidateRoot;

errorRatio = error1 > error2 ? error1 : error2;

} while (errorRatio > myErrorRatio);

printf("%lf is root of equation", candidateRoot);

return 0;

}

**6. Why Use This?**

* Ideal for **numerical methods** learners & students.
* Helps understand **interpolation techniques**.
* Demonstrates **function evaluation and error handling** in C.
* Useful for estimating missing data values in datasets.

**7. Contributing**

Feel free to fork this repo and submit pull requests! Suggestions and improvements are welcome. 🚀

**Author:** Umutcan Oğuz

**GitHub Repository:** [Linear-Interpolation](https://github.com/1230505039/Linear-Interpolation)