Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ

ОТЧЕТ

по лабораторной работе № 1

Название: Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций

Дисциплина: Вычислительные алгоритмы

Студент	ИУ7И - 46Б		Андрич К.
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			В.М. Градов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2021

Цель работы

Получение навыков построения алгоритма интерполяции таблично заданных функций полиномами Ньютона и Эрмита.

Исходные данные

1. Таблица функции и её производных

x	y	y'
0.00	1.000000	--1.000000
0.15	0.838771	-1.14944
0.30	0.655336	-1.29552
0.45	0.450447	-1.43497
0.60	0.225336	-1.56464
0.75	-0.018310	-1.68164
0.90	-0.278390	-1.78333
1.05	-0.552430	-1.86742

2. Степень аппроксимирующего полинома - n
3. Значение аргумента, для которого выполняется интерполяция

Описание алгоритма

1. Алгоритм Ньютона
    a. Вводятся n и x. Потом строится компоновка из n + 1 узлов
    b. Потом строим таблицу которая содержит разделенние разности
    c. Затем строим полином пользуя верхную строку таблици

2. Алгоритм Эрмита
    a. Вводятся n и x. Потом строится компоновка из n + 1 узлов. У нас есть условие что кратност узлов не может быть больше двух
    b. Потом строим таблицу которая содержит разделенние разности. Есть условие что $y(x_0, x_0) = y_0'$
    c. Затем строим полином пользуя верхную строку таблици

3. Алгоритм для нахождения корня пользуя метод обратной интерполяции
    a. Проверим если корень существует
    b. Поменяем места столбцам x и y
    c. Строим полином Ньютона, с тем что n равно 0

Код программы

В программе есть 5 файлов: 2 заголовочных файла (functions.h и errors.h) и 3 файла кода в СИ (main.c, in_out.c и methods.c)

Файл: errors.h

```
#ifndef ERRORS_H
#define ERRORS_H

#define OK 0
#define ERR_IO 1
#define ERR_ARGS 2
#define ERR_FILE 3
#define NO_ROOT 4


#endif //ERRORS_H
```

Файл: functions.h

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <stdio.h>
#include <stdlib.h>

#define N 8

typedef struct
{
    float x;
    float y;
    float dy;
}data_t;

int read_obj(FILE *f, data_t *object);
int read_file(FILE *f, data_t *arr);
int input_exponent(int *n);
int chosen_x(float *n, data_t *arr);
void print_obj(const data_t object);
void print_table(data_t *arr);
int comparator(const void *q, const void *p);
void check_beg_end(int *b, int *e, int n);
int get_cur_pos(data_t *arr, float x);
float newton(data_t *arr, float x, int n);
float hermit(data_t *arr, float x, int n);
int check_for_root(data_t *arr);
void swap_columns(data_t *arr);

#endif //FUNCTIONS_H
```

Файл: in_out.c

```c
#include "functions.h"
#include "errors.h"

int read_obj(FILE *f, data_t *object)
{
    if (fscanf(f, "%f", &object->x) != 1)
        return ERR_IO;
    if (fscanf(f, "%f", &object->y) != 1)
        return ERR_IO;
    if (fscanf(f, "%f", &object->dy) != 1)
        return ERR_IO;
    return OK;
}

int read_file(FILE *f, data_t *arr)
{
    data_t cur;
    int k = 0;
    int rc = OK;
    while (k < N)
    {
        rc = read_obj(f, &cur);
        if (rc == OK)
        {
            arr[k] = cur;
            k++;
        }
        else
            break;
    }
    return rc;
}

int input_exponent(int *n)
{
    printf("\nСтепень плинома: ");
    if (scanf("%d", n) != 1)
        return ERR_IO;
    if (*n < 0 || *n > 4)
        return ERR_IO;
    return OK;
}

int chosen_x(float *n, data_t *arr)
{
    printf("\nВыбарный аргумент для интерполации: ");
    if (scanf("%f", n) != 1)
        return ERR_IO;
```

```
    if (*n < arr[0].x || *n > arr[7].x)
        return ERR_IO;
    return OK;
}

void print_obj(const data_t object)
{
    printf("| %10f ", object.x);
    printf("| %9f ", object.y);
    printf("| %10f | \n", object.dy);
}

void print_table(data_t *arr)
{
    printf("|     x      |     y     |     y'     | \n");
    for(long i = 0; i < N; i++)
        print_obj(arr[i]);
}
```

Файл: methods.c

```
#include "functions.h"
#include "errors.h"

int comparator(const void *q, const void *p)
{
    const data_t *a = q;
    const data_t *b = p;
    return a->x - b->x;
}

void check_beg_end(int *b, int *e, int n)
{
    if (n % 2 == 0)
        *b += 1;
    if (*b < 0)
    {
        *b += 1;
        *e += 1;
    }
}

int get_cur_pos(data_t *arr, float x)
{
    int i = 0;
    while (i < N)
    {
        if (arr[i].x > x)
```

```
            break;
        i++;
    }
    return i;
}

float newton(data_t *arr, float x, int n)
{
    float table[N - 2][N - 2], rem, result;
    int beginning, end;
    int i, k, j = 0;
    qsort(arr, N, sizeof(data_t), comparator);
    i = get_cur_pos(arr, x);
    beginning =  i - ((n + 1) / 2) - 1;
    end =  i + ((n + 1) / 2) - 1;
    check_beg_end(&beginning, &end, n + 1);
    for (i = beginning; i <= end; i++, j++)
        table[j][0] = arr[i].y;
    for (i = 1; i <= n; i++)
        for (j = 0, k = beginning; k < (end - i + 1); j++, k++)
            table[j][i] = (table[j][i - 1] - table[j + 1][i - 1]) /
(arr[k].x - arr[k + i].x);
    result = table[0][0];
    rem =  x - arr[beginning].x;
    beginning++;
    for (i = 1; i <= n; i++, beginning++)
    {
        result += rem * table[0][i];
        rem = rem * (x - arr[beginning].x);
    }
    return result;
}

float hermit(data_t *arr, float x, int n)
{
    float table[N - 2][N - 2], rem, result;
    int beginning, end;
    int i, j;
    i = get_cur_pos(arr, x);
    beginning = i - (n + 1) / 4 - 1;
    end = i + (n + 1) / 4 - 1;
    check_beg_end(&beginning, &end, n + 1);
    for (i = 0, j = beginning; i <= n; i++)
    {
        table[i][0] = arr[j].x;
```

```
        if (i < n)
        {
            i++;
            table[i][0] = arr[j].x;
            j++;
        }
    }
    for (i = 0, j = beginning; i < n; i++)
    {
        if (table[i][0] != table[i + 1][0])
            table[i][1] = (arr[j - 1].y - arr[j].y) / (arr[j - 1].x -
arr[j].x);
        else
        {
            table[i][1] = arr[j].dy;
            j++;
        }
    }
    for (i = 2; i < n + 1; i++)
        for (int j = 0; j < n - i + 1; j++)
            table[j][i] = (table[j][i - 1] - table[j + 1][i - 1]) /
(table[j][0] - table[j + i][0]);
    result = arr[beginning].y;
    rem = x - table[0][0];
    for (i = 1; i < n + 1; i++)
    {
        result += rem * table[0][i];
        rem = rem * (rem - table[i][0]);
    }
    return result;
}

int check_for_root(data_t *arr)
{
    int i, rc = OK;
    for (i = 0; i < N; i++)
        if ((arr[i].y * arr[i + 1].y) < 0)
            break;
    if (i == N)
        rc = NO_ROOT;
    return rc;
}

void swap_columns(data_t *arr)
{
```

```
    float buffer;
    for (int i = 0; i < N; i++)
    {
        buffer = arr[i].x;
        arr[i].x = arr[i].y;
        arr[i].y = buffer;
    }
}
```

Файл: main.c

```
#include "functions.h"
#include "errors.h"

int main(int argc, char *argv[])
{
    int rc = OK;
    FILE *f;
    data_t data[N];
    int n;
    float x, new, her, root;
    if (argc < 2)
    {
        printf("Недостаточно количество аргументов\n");
        rc = ERR_ARGS;
    }
    else
    {
        f = fopen(argv[1], "r+");
        if (f == NULL)
        {
            printf("Не возможно октрыть файл\n");
            return ERR_FILE;
        }
        else
        {
            rc = read_file(f, data);
            if (!rc)
            {
                print_table(data);
                rc = input_exponent(&n);
                if (!rc)
                {
                    rc = chosen_x(&x, data);
                    if (!rc)
                    {
                        new = newton(data, x, n);
                        her = hermit(data, x, n);
                        printf("\nПолином Ньютона: %f \n", new);
                        printf("Полином Эрмита: %f \n", her);
```

```
                        rc = check_for_root(data);
                        if (!rc)
                        {
                            swap_columns(data);
                            root = newton(data, 0, n);
                            printf("Корень: %f \n",root);
                        }
                        else
                            printf("Не возможно найти корень\n");
                    }
                    else
                        printf("Неправильный ввод\n");
                }
                else
                    printf("Неправильный ввод\n");
            }
            else
                printf("Невозможно прочитать файл\n");
            fclose(f);
        }
    }
    return rc;
}
```

Результаты работы

Степень	полином Ньютона	полином Эпмита	корень
1	0.337892	0.342684	0.738729
2	0.340419	0.355156	0.739046
3	0.340314	0.35624	0.739079
4	0.340324	0.372102	0.739088

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/Algorithm/LR1$ ./app.exe table.txt
|      x      |      y      |      y'     |
|   0.000000  |  1.000000   |  -1.000000  |
|   0.150000  |  0.838771   |  -1.149440  |
|   0.300000  |  0.655336   |  -1.295520  |
|   0.450000  |  0.450447   |  -1.434970  |
|   0.600000  |  0.225336   |  -1.564640  |
|   0.750000  | -0.018310   |  -1.681640  |
|   0.900000  | -0.278390   |  -1.783330  |
|   1.050000  | -0.552430   |  -1.867420  |

Степень плинома: 1

Выбарный аргумент для интерполации: 0.525

Полином Ньютона: 0.337892
Полином Эрмита: 0.342684
Корень: 0.738729
```

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/Algorithm/LR1$ ./app.exe table.txt
|      x      |      y      |      y'      |
|   0.000000  |  1.000000   |  -1.000000   |
|   0.150000  |  0.838771   |  -1.149440   |
|   0.300000  |  0.655336   |  -1.295520   |
|   0.450000  |  0.450447   |  -1.434970   |
|   0.600000  |  0.225336   |  -1.564640   |
|   0.750000  | -0.018310   |  -1.681640   |
|   0.900000  | -0.278390   |  -1.783330   |
|   1.050000  | -0.552430   |  -1.867420   |

Степень плинома: 2

Выбарный аргумент для интерполации: 0.525

Полином Ньютона: 0.340419
Полином Эрмита: 0.355156
Корень: 0.739046
```

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/Algorithm/LR1$ ./app.exe table.txt
|      x      |      y      |      y'      |
|   0.000000  |  1.000000   |  -1.000000   |
|   0.150000  |  0.838771   |  -1.149440   |
|   0.300000  |  0.655336   |  -1.295520   |
|   0.450000  |  0.450447   |  -1.434970   |
|   0.600000  |  0.225336   |  -1.564640   |
|   0.750000  | -0.018310   |  -1.681640   |
|   0.900000  | -0.278390   |  -1.783330   |
|   1.050000  | -0.552430   |  -1.867420   |

Степень плинома: 3

Выбарный аргумент для интерполации: 0.525

Полином Ньютона: 0.340314
Полином Эрмита: 0.356624
Корень: 0.739079
```

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/Algorithm/LR1$ ./app.exe table.txt
|      x      |      y      |      y'      |
|   0.000000  |  1.000000   |  -1.000000   |
|   0.150000  |  0.838771   |  -1.149440   |
|   0.300000  |  0.655336   |  -1.295520   |
|   0.450000  |  0.450447   |  -1.434970   |
|   0.600000  |  0.225336   |  -1.564640   |
|   0.750000  | -0.018310   |  -1.681640   |
|   0.900000  | -0.278390   |  -1.783330   |
|   1.050000  | -0.552430   |  -1.867420   |

Степень плинома: 4

Выбарный аргумент для интерполации: 0.525

Полином Ньютона: 0.340324
Полином Эрмита: 0.372102
Корень: 0.739088
```

Вопросы при защите лабораторной работы

1. *Будет ли работать программа при степени полинома n=0?*

```
katarina@LAPTOP-I1VEUM2H:/mnt/c/Users/katar/Desktop/MSTU Bauman/Algorithm/LR1$ ./app.exe table.txt
|     x      |     y      |     y'     |
|   0.000000 |  1.000000 |  -1.000000 |
|   0.150000 |  0.838771 |  -1.149440 |
|   0.300000 |  0.655336 |  -1.295520 |
|   0.450000 |  0.450447 |  -1.434970 |
|   0.600000 |  0.225336 |  -1.564640 |
|   0.750000 | -0.018310 |  -1.681640 |
|   0.900000 | -0.278390 |  -1.783330 |
|   1.050000 | -0.552430 |  -1.867420 |

Степень плинома: 0

Выбарный аргумент для интерполации: 0.525

Полином Ньютона: 0.450447
Полином Эрмита: 0.450447
Корень: 0.750000
```

   Видно что программа работает. Результатом стало ближайшее значение функции.

2. *Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?*
   Практически можем оценить погрешность интерполяции так что оценим первый отброшенный член. Для этих целей теоретическую оценку сложно применить потому что нам нужно знать производные функции, но они неизвестны.

3. *Если в двух точках заданы значения функции и ее первых производных, то полином какой минимальной степени может быть построен на этих точках?*
   Полином Ньютона: 0 и 1 степень
   Полином Эрмита: 0, 1, 2 и 3 степень
   ⇒ минимальная допущена степень полинома - 0

4. *В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?*
   Информация об упорядоченности аргумента функции существенна в месте построения компоновки узлов. Из них строится таблица разделенных разностей. Если аргументы упорядочены, тогда легко можем найти соседние аргументы к нашему аргументу. Но, если они не упорядочены, тогда надо пробегать по таблице несколько раз.

5. *Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?*
   Выравнивающими переменными являются переменные $\eta = \eta(i)$, $\xi = \xi(k)$, с помощью которых можно добиться, чтобы график быстро меняющейся функции в новых переменных $\eta(\xi)$ был близок к прямой, по крайней мере, в некоторых областях. В этом случае интерполяция выполняется по переменным (η, ξ), а затем $yi = y(\eta i)$ находится обратной интерполяцией. Однако преобразования $\eta(i)$, $\xi(k)$ должны быть достаточно простыми, например, логарифмическими). Также должны убедиться, что обратное преобразование и (η) не сложное.