РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. ПАТРИСА ЛУМУМБЫ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Вычислительные методы

Студент:

Мажитов Магомед Асхабович

Группа:

НКНбд-01-21

МОСКВА

2023 г.

Цель:

Написать программу для расчета полинома Ньютона.

Теоретическая справка:

1. Построить равномерное разбиение отрезка [a, b] на N = 10 частей точками a = x_0, x_1, …, x_N = b.
2. Рассчитать значения функции f(x) в узлах интерполяции: $y_0 = f(x_0)$, $y_1 = f(x_1)$, …, $y_N = f(x_N)$.
3. Построить интерполяционный полином Ньютона $P_N(x) = A_0 + A_1(x - x_0) + A_2(x - x_0)(x - x_1) + \dots + A_i(x - x_0)\dots(x - x_i) + \dots + A_N(x - x_0)\dots(x - x_{N-1})$ согласно значениям из п.1, 2, также посчитать погрешность интерполяции в точке x_j.
4. Построить равномерное разбиение отрезка [a, b] из задания на M = 3N частей точками a = $\overline{x_0}$, $\overline{x_1}$, …, $\overline{x_M}$ = b.
5. Посчитать значения исходной функции f(x) из задания и построенного в п.3 полинома Ньютона $P_N(x)$ в точках $\overline{x_0}$, $\overline{x_1}$, …, $\overline{x_M}$, полученных в п.4, также посчитать погрешность интерполяции в точке $\overline{x_j}$.
6. Подобрать такое значение N, при котором максимальная погрешность меньше 0,1

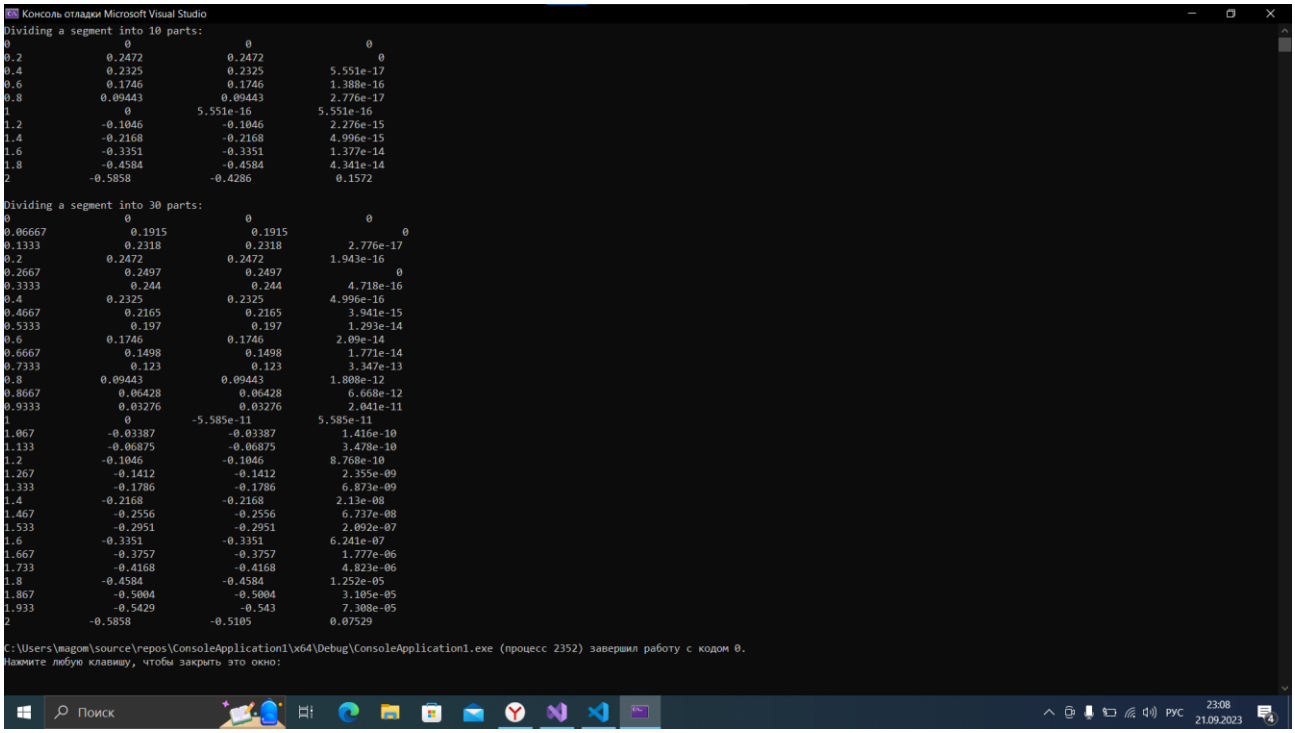
Листинг:

```
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

//sqrt(x) - x, a = 0, b = 2.

double* split(double a, double b, int n){//функция для разбиения отрезка на N равных отрезков
    auto* s = new double [n];//создаем массив для записи точек
    s[0] = a;//начальная и конечная точка константы
    double step = (b - a)/n; //считаем шаг
    for(int i = 1; i <= n; i++){ //проходим по циклу и к предыдущей точке прибавляем шаг
        s[i] = s[i-1] + step;
    }
    return s;//возвращаем массив из узлов интерполяций
}

double* function(double* x, int n){//функция для подсчета функции в узлах интерполяции
```

```
    auto* f = new double [n]; //массив для записи функций
    for(int i = 0; i <= n; i++){//проходим по циклу и считаем функции в узлах
интерполяции
        f[i] = sqrt(x[i]) - x[i];
    }
    return f; //возвращаем массив из функций
}

double newtonPolinomial(const double *x, const double *y, int n, double
xx){//функция для подсчета полинома в точке
    double omg, newt = y[0], a;
    for(int i = 1; i < n; i++) { //проходим по циклу и считаем полином
        a = 0.0;//задаем начальное значение a
        for(int j = 0; j <= i; j++) {//проходим по циклу и считаем a
            omg = 1.0;//задаем начальное значение омеги
            for(int k = 0; k <= i; k++) { //проходим по циклу и считаем значение
омеги
                if (k != j) {
                    omg *= (x[j] - x[k]);//умножаем предыдущее значение омеги с
разницей между точками разбиения
                }
            }
            a +=  (y[j] / omg);//складываем предыдущее значение a на частное j-
ого элемента функции и значение омеги
        }
        for (int j = 0; j < i; j++) {//проходим по циклу и считаем новые
значения a для полинома
            a *= (xx - x[j]);
        }
        newt += a;//складываем значение a с предыдущим значением полинома
    }
    return newt;//возвращаем значение полинома
}

double* newtonPolinomialAll(double* x, double* y, int n){//функция для подсчета
полинома в каждой точке
    auto* newton = new double [n]; //создаем массив для записи полинома в каждой
точке
    for(int i = 0; i <= n; i++){//проходим по циклу и в каждой точке считаем
значения полинома и записываем в массив
        newton[i] = newtonPolinomial(x, y, n, x[i]);
    }
    return newton;//возвращаем массив
}

void print(double* pol, int n, double* x, double* y){//функция для красивого
вывода
    cout << "Dividing a segment into " << n << " parts: " << endl;
    for(int i = 0; i <= n; i++){
        cout << setprecision(4) << x[i] << setw(20);
        cout << y[i] << setw(20);
        cout << pol[i] << setw(20);
        cout << abs(y[i] - pol[i]) << setprecision(20) << endl;
    }
}

int main() {
    int n = 10, m = 3*n;
    double a = 0, b= 2;
    double *x = split(a, b, n);
    double *y = function(x, n);
    double *x1 = split(a, b, m);
    double *y1 = function(x1, m);
    double* ne = newtonPolinomialAll(x, y, n);
```

```
    print(ne, n, x, y);
    cout << endl;
    double* ne1 = newtonPolinomialAll(x1, y1, m);
    print(ne1, m, x1, y1);
    return 0;
}
```

Работа программы:



При N=10 погрешность интерполяции ровна 0, таким образом подходит условию пункта №6.

Вывод:

В ходе работы я реализовал на языке C++ полином Ньютона.