Міністерство освіти і науки України

КПІ ім. Ігоря Сікорського

Кафедра ІПІ

**ЗВІТ**

з виконання лабораторної роботи № 3

з кредитного модуля

“Основи програмування-2. Методології програмування”

**Варіант № 5**

Виконав:

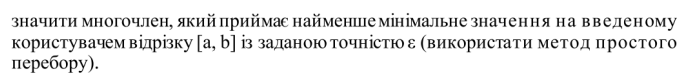
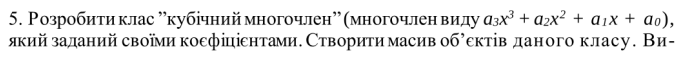
Студент 1-го курсу

Групи ІП-22 ФІОТ

Гринько Олександр Максимович

Київ 2023

**Постановка задачі**



**Вимоги до програми**

Изображение выглядит как текст

Автоматически созданное описание

**Код програми**

**main.cpp**

#include "Functions.h"

void main() {

float a = getLimitA();

float b = getLimitB();

float e = getE(a, b);

float m = getAmmountOfcubicPolynomials();

cubicPolynomial\* array = new cubicPolynomial[m];

array = makeArray(m);

output(array, a, b, e, m);

counting(array, m, a, b, e);

}

**Functions.h**

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

using namespace std;

int getAmmountOfcubicPolynomials();

float getLimitA();

float getLimitB();

float getE(float, float);

void check\_int(int\*);

void check\_float(float\*);

class cubicPolynomial {

float a0;

float a1;

float a2;

float a3;

float a;

float b;

public:

float getA0();

void setA0(float aa0);

float getA1();

void setA1(float aa1);

float getA2();

void setA2(float aa2);

float getA3();

void setA3(float aa3);

float getA();

void setA(float aa);

float getB();

void setB(float bb);

};

cubicPolynomial\* makeArray(int);

void output(cubicPolynomial\*, float, float, float, int);

void counting(cubicPolynomial\*, int, float, float, float);

**Functions.cpp**

#include "Functions.h"

float cubicPolynomial::getA0() {

return a0;

}

float cubicPolynomial::getA1() {

return a1;

}

float cubicPolynomial::getA2() {

return a2;

}

float cubicPolynomial::getA3() {

return a3;

}

float cubicPolynomial::getA() {

return a;

}

float cubicPolynomial::getB() {

return b;

}

void cubicPolynomial::setA0(float aa0) {

a0 = aa0;

}

void cubicPolynomial::setA1(float aa1) {

a1 = aa1;

}

void cubicPolynomial::setA2(float aa2) {

a2 = aa2;

}

void cubicPolynomial::setA3(float aa3) {

a3 = aa3;

}

void cubicPolynomial::setA(float aa) {

a = aa;

}

void cubicPolynomial::setB(float bb) {

b = bb;

}

void check\_int(int\* variable) {

char sym;

bool er;

do {

er = false;

cin >> \*variable;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Incorrect input, try again" << endl;

er = true;

}

else {

cin.get(sym);

if (sym != '\n') {

cout << "Incorrect input, try again" << endl;

er = true;

}

}

} while (er);

}

void check\_float(float\* variable) {

char sym;

bool er;

do {

er = false;

cin >> \*variable;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Incorrect input, try again" << endl;

er = true;

}

else {

cin.get(sym);

if (sym != '\n') {

cout << "Incorrect input, try again" << endl;

er = true;

}

}

} while (er);

}

int getAmmountOfcubicPolynomials() {

int m;

cout << "Enter the ammount of cubic polynomial: " << endl;

check\_int(&m);

return m;

}

float getLimitA() {

float a;

cout << "Enter a: " << endl;

check\_float(&a);

return a;

}

float getLimitB() {

float b;

cout << "Enter b: " << endl;

check\_float(&b);

return b;

}

float getE(float a, float b) {

float e;

bool f = true;

do {

cout << "Enter e: " << endl;

f = true;

check\_float(&e);

if (b > a && e < 0) {

f = false;

cout << "e must be more than 0" << endl;

}

else if (b < a && e > 0) {

f = false;

cout << "e must be more less 0" << endl;

}

else if (e == 0) {

f = false;

cout << "e can't be 0" << endl;

}

} while (f == false);

return e;

}

cubicPolynomial\* makeArray(int m) {

cubicPolynomial\* objectsArray = new cubicPolynomial[m];

cout << "\na3 \* x^3 + a2 \* x^2 + a1 \* x + a0\n\n\n";

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

cout << "Enter the " << i+1 << " cubic polynomial : " << endl;

float a0, a1, a2, a3;

cout << "\tEnter a0: ";

check\_float(&a0);

objectsArray[i].setA0(a0);

cout << "\tEnter a1: ";

check\_float(&a1);

objectsArray[i].setA1(a1);

cout << "\tEnter a2: ";

check\_float(&a2);

objectsArray[i].setA2(a2);

cout << "\tEnter a3: ";

check\_float(&a3);

objectsArray[i].setA3(a3);

}

return objectsArray;

}

void output(cubicPolynomial\* array, float a, float b, float e, int m) {

cout << "\n\na: " << a << "\tb: " << b << "\te: " << e << "\n\n";

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

cout << "The " << i + 1 << " cubic polynomial a0: " << setprecision(4) << array[i].getA0() << "\t\ta1: "<< setprecision(4) << array[i].getA1() << "\t\ta2: " << setprecision(4) << array[i].getA2() << "\t\ta3: " << setprecision(4) << array[i].getA3() << endl;

}

}

void counting(cubicPolynomial\* array, int m, float a, float b, float e) {

float y;

float min = powf(a, 3) \* array[0].getA3() + powf(a, 2) \* array[0].getA2() + a \* array[0].getA1() + array[0].getA0();

float final\_x = a;

int index = 0;

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

cout << "\n";

if (e > 0)

for (float x = a; x <= b; x += e) {

y = powf(x, 3) \* array[i].getA3() + powf(x, 2) \* array[i].getA2() + x \* array[i].getA1() + array[i].getA0();

cout << "The " << i + 1 << " cubic polynomial x: " << setprecision(4) << x << "\t\ty: " << setprecision(4) << y << "\t\ta0: " << setprecision(4) << array[i].getA0() << "\t\ta1: " << setprecision(4) << array[i].getA1() << "\t\ta2: " << setprecision(4) << array[i].getA2() << "\t\ta3: " << setprecision(4) << array[i].getA3() << endl;

if (y < min) {

min = y;

final\_x = x;

index = i;

}

}

else

for (float x = a; x >= b; x += e) {

y = powf(x, 3) \* array[i].getA3() + powf(x, 2) \* array[i].getA2() + x \* array[i].getA1() + array[i].getA0();

cout << "The " << i + 1 << " cubic polynomial x: " << setprecision(4) << x << "\t\ty: " << setprecision(4) << y << "\t\ta0: " << setprecision(4) << array[i].getA0() << "\t\ta1: " << setprecision(4) << array[i].getA1() << "\t\ta2: " << setprecision(4) << array[i].getA2() << "\t\ta3: " << setprecision(4) << array[i].getA3() << endl;

if (y < min) {

min = y;

final\_x = x;

index = i;

}

}

}

cout << '\n' << array[index].getA3() << " \* " << final\_x << "^3 + " << array[index].getA2() << " \* " << final\_x << "^2 + " << array[index].getA1() << " \* " << final\_x << " + " << array[index].getA0() << " = " << min;

cout << "\n\n\nThe minimal y = " << setprecision(4) << min;

}

**Результат роботи**

Изображение выглядит как стол

Автоматически созданное описание

**Гіт хаб: https://github.com/esk4nz/OP\_labs**