Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №2

з дисципліни «Об’єктно-орієнтоване програмування»

**Тема:** «Конструктори і деструктори»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Гребенюк Д.О. |
| Перевірив: асистент  Козірова Н.Л. |
|  |

Кропивницький

2024

**Мета:** Ознайомитись з основними поняттями конструктор і деструктор в ООП та навчитись їх програмно реалізовувати мовою С++.

**Варіант 4**

**Завдання 1**

1. Реалізуйте конструктор за замовчуванням, конструктор з параметрами та копіюючий конструктор для вашого класу з лабораторної роботи 1.

• Конструктор за замовчуванням має встановлювати значення полів за замовчуванням.

• Конструктор з параметрами має приймати значення для кожного поля.

• Копіюючий конструктор має копіювати значення полів з іншого об'єкта класу.

2. Реалізуйте деструктор для класу. Деструктор має виводити повідомлення про знищення об'єкта.

3. У функції main створіть об'єкт за допомогою конструктора за замовчуванням та виведіть значення його полів.

4. Створіть новий об'єкт за допомогою конструктора з параметрами та встановіть значення для полів. Виведіть значення полів цього об'єкта.

5. Створіть ще один об'єкт і скопіюйте значення полів з першого об'єкта за допомогою копіюючого конструктора. Виведіть значення полів цього об'єкта.

6. Завершіть функцію main, що призведе до виходу з області видимості створених об'єктів і виклику їх деструкторів. Переконайтесь, що повідомлення про знищення об'єктів виводяться.

**Завдання 2**

Розробіть клас «Многочлен» – Polynom ступеня 𝑛. Клас повинен містити конструктори. Реалізуйте методи для обчислення значення многочлена, додавання, віднімання та множення многочленів. Створіть масив об'єктів класу Polynom. Передайте його у функцію, яка обчислює суму многочленів масиву та повертає результуючий многочлен, який виводиться на екран у головній програмі.

**Завдання 1**

*Лістинг employee.h*

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

class Employee

{

public:

Employee();

Employee(std::string name, int id, int salary);

Employee(const Employee& employee);

~Employee();

std::string getName();

void setName(std::string newName);

int getId();

void setId(int newId);

int getSalary();

void setSalary(int newSalary);

private:

std::string name;

int id;

int salary;

};

#endif // EMPLOYEE\_H

#endif // EMPLOYEE\_H

*Лістинг employee.cpp*

#include <iostream>

#include "employee.h"

Employee::Employee() {

name = "None";

id = -1;

salary = 0;

}

Employee::Employee(std::string name, int id, int salary)

{

this->name = name;

this->id = id;

this->salary = salary;

}

Employee::Employee(const Employee &employee)

{

this->name = employee.name;

this->id = employee.id;

this->salary = employee.salary;

}

Employee::~Employee()

{

std::cout << "Employee object has been destroyed." << std::endl;

}

std::string Employee::getName()

{

return name;

}

void Employee::setName(std::string newName)

{

name = newName;

}

int Employee::getId()

{

return id;

}

void Employee::setId(int newId)

{

id = newId;

}

int Employee::getSalary()

{

return salary;

}

void Employee::setSalary(int newSalary)

{

salary = newSalary;

}

*Лістинг main.cpp*

#include <iostream>

#include "employee.h"

void printEmployeeData(Employee &employee)

{

std::cout << "Id: " << employee.getId() << std::endl;

std::cout << "Name: " << employee.getName() << std::endl;

std::cout << "Salary: " << employee.getSalary() << std::endl;

}

int main()

{

std::cout << "========== Default constructor ==========" << std::endl;

Employee emptyEmployee;

printEmployeeData(emptyEmployee);

std::cout << "========== Constructor with parameters ==========" << std::endl;

Employee bob("Bob", 1, 10000000);

printEmployeeData(bob);

std::cout << "========== Constructor with parameters ==========" << std::endl;

Employee copiedEmployee(emptyEmployee);

printEmployeeData(copiedEmployee);

std::cout << std::endl;

return 0;

}

*Результат виконання*



**Завдання 2**

*Лістинг polynom.h*

#ifndef POLYNOM\_H

#define POLYNOM\_H

#include <string>

#include <vector>

struct Uninom {

int coefficient;

int degree;

};

class Polynom

{

public:

Polynom(int degree, const std::vector<Uninom> &members);

void printPolynom();

Polynom add(const Polynom &other);

Polynom subtract(const Polynom &other);

Polynom multiply(const Polynom &other);

private:

int degree;

std::vector<Uninom> members;

Polynom calculate(const Polynom &other, std::string calcOperator);

};

#endif // POLYNOM\_H

*Лістинг polynom.cpp*

#include "polynom.h"

#include <iostream>

Polynom::Polynom(int degree, const std::vector<Uninom> &members) {

this->degree = degree;

this->members = members;

}

void Polynom::printPolynom()

{

for (int index = 0; index < (int)this->members.size(); index++) {

Uninom member = this->members.at(index);

if (member.coefficient > 0 && index > 0)

std::cout << "+";

std::cout << member.coefficient;

if (member.degree > 0)

std::cout << "x^" << member.degree;

}

std::cout << std::endl;

}

Polynom Polynom::add(const Polynom &other) {

return calculate(other, "+");

}

Polynom Polynom::subtract(const Polynom &other)

{

return calculate(other, "-");

}

Polynom Polynom::multiply(const Polynom &other)

{

std::vector<Uninom> newPolynomMembers;

int firstPolynomSize = this->members.size();

int secondPolynomSize = other.members.size();

for (int firstIndex = 0; firstIndex < firstPolynomSize; firstIndex++) {

Uninom firstMember = this->members.at(firstIndex);

for (int secondIndex = 0; secondIndex < secondPolynomSize; secondIndex++) {

Uninom secondMember = other.members.at(secondIndex);

Uninom result = {firstMember.coefficient \* secondMember.coefficient, firstMember.degree + secondMember.degree};

newPolynomMembers.push\_back(result);

}

}

return Polynom(this->degree + other.degree, newPolynomMembers);

}

Polynom Polynom::calculate(const Polynom &other, std::string calcOperator)

{

std::vector<Uninom> newPolynomMembers;

int firstPolynomSize = this->members.size();

int secondPolynomSize = other.members.size();

int highestPolynomDegree = (other.degree > this->degree) ? other.degree : this->degree;

for (int index = 0; index < highestPolynomDegree + 1; index++) {

int firstMember = 0,

secondMember = 0,

result = 0;

if (index < firstPolynomSize)

firstMember = this->members.at(index).coefficient;

if (index < secondPolynomSize)

secondMember = other.members.at(index).coefficient;

if (calcOperator == "+")

result = firstMember + secondMember;

else if (calcOperator == "-")

result = firstMember - secondMember;

newPolynomMembers.push\_back({result, index});

}

return Polynom(highestPolynomDegree, newPolynomMembers);

}

*Лістинг main.cpp*

#include <iostream>

#include <vector>

#include "polynom.h"

Polynom sumPolynoms(const std::vector<Polynom> &polynoms) {

if (polynoms.empty()) {

return Polynom(0, {});

}

Polynom sum = polynoms[0];

for (int index = 1; index < (int)polynoms.size(); index++) {

sum = sum.add(polynoms[index]);

}

return sum;

}

int main()

{

std::cout << "========== Initialising polynomials ==========" << std::endl;

std::vector<Uninom> members1 = { {2, 0}, {-3, 1}, {5, 2}, {4, 3} };

Polynom polynom1(3, members1);

std::cout << "Polynomial 1: ";

polynom1.printPolynom();

std::vector<Uninom> members2 = { {5, 0}, {7, 1}, {-6, 2} };

Polynom polynom2(2, members2);

std::cout << "Polynomial 2: ";

polynom2.printPolynom();

std::vector<Uninom> members3 = { {5, 0}, {2, 1} };

Polynom polynom3(1, members3);

std::cout << "Polynomial 3: ";

polynom3.printPolynom();

std::cout << "========== Result of adding polynomials ==========" << std::endl;

Polynom newPolynom = polynom1.add(polynom2);

newPolynom.printPolynom();

std::cout << "========== Result of subtracting polynomials ==========" << std::endl;

newPolynom = polynom1.subtract(polynom2);

newPolynom.printPolynom();

std::cout << "========== Result of multiplying polynomials ==========" << std::endl;

newPolynom = polynom1.multiply(polynom2);

newPolynom.printPolynom();

std::cout << "========== Result of adding polynomials of an array ==========" << std::endl;

std::vector<Polynom> polynoms = { polynom1, polynom2, polynom3 };

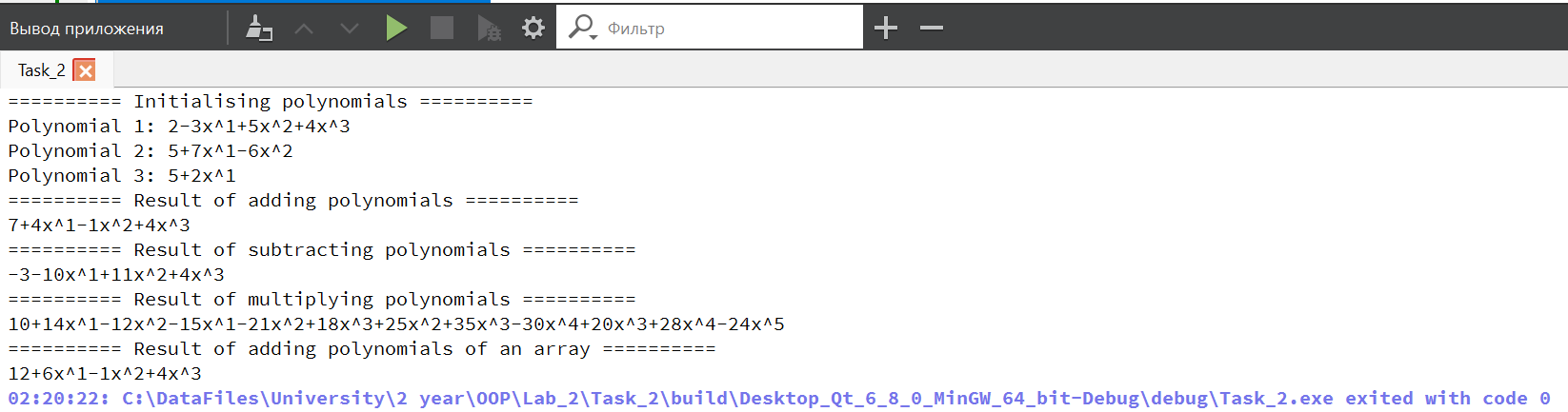
Polynom result = sumPolynoms(polynoms);

result.printPolynom();

return 0;

}

*Результат виконання*



**Висновок:**

У ході виконання лабораторної роботи я ознайомився з поняттями конструкторів і деструкторів в об’єктно-орієнтованому програмуванні на мові C++. Я реалізував конструктори за замовчуванням, з параметрами, а також копіюючий конструктор для класу Employee. Окрім цього, я написав деструктор, який виводить повідомлення про знищення об'єкта. У функції main я створив об'єкти, перевірив їх роботу та переконався, що деструктори викликаються коректно.

Також я реалізував клас «Многочлен» із методами для виконання операцій додавання, віднімання та множення многочленів. На прикладі кількох об'єктів я продемонстрував коректність виконання цих операцій і використання масиву об'єктів для підсумування многочленів.

Загалом, робота допомогла мені закріпити знання з теми конструкторів і деструкторів та застосувати їх на практиці при роботі з класами і многочленами.