МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет

імені Олеся Гончара

Факультет прикладної математики

Кафедра комп'ютерних технологій

**ЗВІТ**

**З ЛАБОРАТОРНОЇ РОБОТИ № 2**

«***Використання класів в мові С++:***

***інкапсуляція, наслідування та перевантаження операцій"»***

з дисципліни «Об’єктно-орієнтоване програмування»

Виконав:

Студент групи ПА-22-2

Мирошниченко Олександр Ілліч

м. Дніпро

2024 р.

**Зміст**

Оглавление

[Постановка задачі 3](#_Toc175680695)

[Опис розв’язку 4](#_Toc175680696)

[Вихідний текст програми розв’язку задачі 6](#_Toc175680697)

[Опис інтерфейсу програми (керівництво користувача) 16](#_Toc175680698)

[Висновок 19](#_Toc175680699)

# Постановка задачі

Розробити об’єктно-орієнтовану бібліотеку для роботи зі структурами даних за

однією з нижченаведених тем у відповідності з нижченаведеними вимогами.

Властивості та методи для класів розробити у відповідності з відомими

визначеннями відповідних структур даних. Скласти тести для перевірки

працездатності бібліотеки. Скласти програму, що демонструє можливості

розробленої бібліотеки.

Загальні вимоги

В незалежності від індивідуального варіанта повинні бути реалізовані наступні

можливості:

1. Реалізація методів ініціалізації (конструктор за замовчуванням та

конструктор з параметрами), копіювання (конструктор копіювання),

індексації (перевантаження []), присвоювання (перевантаження =),

візуалізації, збереження (на диск) та відновлення, діалогового керування,

"розумного доступу" (перевантаження ->), а також псевдо змінних

(забезпечення можливості виду: f(x)=const).

2. Перевантаження (спільне використання) потокового введення/виведення

(введення з файл, виведення в файл).

3. Створення та використання файла бібліотеки (\*.lib, або \*.obj).

4. Повторне використання класів без їх перекомпіляції (“ReUse”, тобто

наслідування).

5. Результати виконання лабораторної роботи повинні бути викладені у

вигляді звіту.

В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому

режимі (за допомогою меню) виконувати наступні операції:

1. Створювати об’єкти класу (у довільній кількості).

2. Вводити/виводити об’єкти на диск.

3. Візуалізувати об’єкти.

4. Виконувати операції над об’єктами.

Тема індивідуального завдання

13. Множини (додавання елемента до множини, видалення елемента з множини,

перевірка наявності елемента в множині, об’єднання множин, перетин множин).

# Опис розв’язку

Ця програма реалізує клас для роботи з великими числами, зокрема з базовими арифметичними операціями та збереженням/відновленням значень у файлах. Вона складається з трьох файлів:

main.cpp - головний файл програми, який відповідає за взаємодію з користувачем.

longArithmetics\_2.cpp - реалізація методів класу LongArithmetic.

longArithmetics\_2.h - заголовочний файл, який містить визначення класу LongArithmetic.

Клас LongArithmetic

Клас LongArithmetic призначений для представлення та виконання операцій над великими числами, представленими у вигляді вектора типу double.

Основні компоненти класу:

Конструктори:

LongArithmetic() - конструктор за замовчуванням.

LongArithmetic(double value) - конструктор, що ініціалізує об'єкт з одним значенням.

LongArithmetic(const LongArithmetic& other) - копіюючий конструктор.

LongArithmetic num1(2000);

У цьому прикладі створюється об'єкт num1 з початковим значенням 2000.

Деструктор:

~LongArithmetic() - деструктор, який звільняє ресурси.

Оператори:

Перевантажені оператори дозволяють виконувати арифметичні операції над об'єктами класу:

Додавання: operator+

Віднімання: operator-

Множення: operator\*

Ділення: operator/

LongArithmetic num5 = num1 + num2;

Тут виконується додавання двох об'єктів num1 та num2, результат зберігається у num5.

Збереження та відновлення:

Метод saveToFile(const std::string& filename) зберігає значення об'єкта у файл.

Метод restoreFromFile(const std::string& filename) відновлює значення об'єкта з файлу.

num1.saveToFile("number.txt");

Цей код зберігає значення об'єкта num1 у файл number.txt.

Діалогове управління:

Методи input() та output() забезпечують введення та виведення значень об'єкта через консоль.

Взаємодія з користувачем (main.cpp)

Основна функція програми (main()) реалізує діалог з користувачем через консоль. Вона дозволяє створювати об'єкти класу LongArithmetic, зберігати їх у файл, відновлювати з файлу, а також виконувати арифметичні операції між об'єктами.

Основні функції:

createObjects(std::vector<LongArithmetic>& objects) - створює новий об'єкт на основі введеного користувачем значення та додає його до списку об'єктів.

saveToFile(const std::vector<LongArithmetic>& objects) - зберігає всі об'єкти зі списку у файл.

restoreFromFile(std::vector<LongArithmetic>& objects) - відновлює всі об'єкти з файлу та додає їх до списку.

performOperations(const std::vector<LongArithmetic>& objects) - виконує арифметичні операції між об'єктами зі списку.

# Вихідний текст програми розв’язку задачі

longArithmetics\_2.cpp:

#ifndef LONGARITHMETICS\_2\_H

#define LONGARITHMETICS\_2\_H

#include <iostream>

#include <vector>

#include <fstream>

class LongArithmetic {

public:

// Конструктори

LongArithmetic();

LongArithmetic(double value);

LongArithmetic(const LongArithmetic& other);

// Деструктор

~LongArithmetic();

// Оператори

LongArithmetic& operator=(const LongArithmetic& other);

double& operator[](size\_t index);

const double& operator[](size\_t index) const;

LongArithmetic operator+(const LongArithmetic& other) const;

LongArithmetic operator-(const LongArithmetic& other) const;

LongArithmetic operator\*(const LongArithmetic& other) const;

LongArithmetic operator/(const LongArithmetic& other) const;

// Перевантаження потокового вводу/виводу

friend std::ostream& operator<<(std::ostream& os, const LongArithmetic& la);

friend std::istream& operator>>(std::istream& is, LongArithmetic& la);

// Збереження та відновлення

void saveToFile(const std::string& filename) const;

void restoreFromFile(const std::string& filename);

// Діалогове управління

void input();

void output() const;

// "Розумний доступ"

double\* operator->();

const double\* operator->() const;

// Псевдо змінні

double f(double x) const;

private:

std::vector<double> values;

void ensureCapacity(size\_t index);

};

#endif // LONGARITHMETIC\_H

main.cpp:

#include <iostream>

#include <limits>

#include <vector>

#include "longArithmetics\_2.h"

// Function to clear the input stream

void clearInputStream() {

std::cin.clear(); // Clears error flags

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Ignores the rest of the line

}

// Function to create objects of LongArithmetic

void createObjects(std::vector<LongArithmetic>& objects) {

double value;

std::cout << "Enter a value for the new object: ";

std::cin >> value;

if (std::cin.fail()) {

clearInputStream();

std::cerr << "Input error. Please try again." << std::endl;

return;

}

objects.push\_back(LongArithmetic(value));

std::cout << "Object created." << std::endl;

}

// Function to save objects to a file

void saveToFile(const std::vector<LongArithmetic>& objects) {

std::string filename;

std::cout << "Enter the filename for saving: ";

std::cin >> filename;

std::ofstream ofs(filename, std::ios::app); // Open in append mode

if (ofs.is\_open()) {

for (const auto& obj : objects) {

ofs << obj << std::endl; // Write all objects

}

std::cout << "Objects saved to file." << std::endl;

} else {

std::cerr << "Failed to open file for saving." << std::endl;

}

}

// Function to restore objects from a file

void restoreFromFile(std::vector<LongArithmetic>& objects) {

std::string filename;

std::cout << "Enter the filename for restoration: ";

std::cin >> filename;

std::ifstream ifs(filename);

if (ifs.is\_open()) {

objects.clear(); // Clear old objects

double value;

while (ifs >> value) { // Read all values from file

objects.push\_back(LongArithmetic(value));

}

std::cout << "Objects restored from file." << std::endl;

} else {

std::cerr << "Failed to open file for restoration." << std::endl;

}

}

// Function to visualize objects

void visualizeObjects(const std::vector<LongArithmetic>& objects) {

std::cout << "Objects:" << std::endl;

for (const auto& obj : objects) {

std::cout << obj << std::endl;

}

}

// Function to perform arithmetic operations on objects

void performOperations(const std::vector<LongArithmetic>& objects) {

if (objects.size() < 2) {

std::cerr << "At least two objects are required to perform operations." << std::endl;

return;

}

std::cout << "Operations with objects:" << std::endl;

std::cout << "1. Addition" << std::endl;

std::cout << "2. Subtraction" << std::endl;

std::cout << "3. Multiplication" << std::endl;

std::cout << "4. Division" << std::endl;

std::cout << "Select an operation: ";

int choice;

std::cin >> choice;

if (std::cin.fail()) {

clearInputStream();

std::cerr << "Input error. Please try again." << std::endl;

return;

}

size\_t index1, index2;

std::cout << "Enter the indices of two objects (starting from 0): ";

std::cin >> index1 >> index2;

if (std::cin.fail() || index1 >= objects.size() || index2 >= objects.size()) {

clearInputStream();

std::cerr << "Invalid indices. Please try again." << std::endl;

return;

}

try {

LongArithmetic result;

switch (choice) {

case 1:

result = objects[index1] + objects[index2];

std::cout << "Addition result: " << result << std::endl;

break;

case 2:

result = objects[index1] - objects[index2];

std::cout << "Subtraction result: " << result << std::endl;

break;

case 3:

result = objects[index1] \* objects[index2];

std::cout << "Multiplication result: " << result << std::endl;

break;

case 4:

result = objects[index1] / objects[index2];

std::cout << "Division result: " << result << std::endl;

break;

default:

std::cerr << "Invalid choice." << std::endl;

break;

}

} catch (const std::exception& e) {

std::cerr << e.what() << std::endl;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

std::vector<LongArithmetic> objects;

int choice;

bool running = true;

while (running) {

std::cout << " Menu " << std::endl;

std::cout << "1. Create object(s) " << std::endl;

std::cout << "2. Save/Load objects to/from disk " << std::endl;

std::cout << "3. Visualize objects " << std::endl;

std::cout << "4. Perform operations on objects " << std::endl;

std::cout << "5. Exit " << std::endl;

std::cout << "-----------------------------------" << std::endl;

std::cout << "Select an option: ";

std::cin >> choice;

if (std::cin.fail()) {

clearInputStream();

std::cerr << "Input error. Please try again." << std::endl;

continue;

}

switch (choice) {

case 1:

createObjects(objects);

break;

case 2:

saveToFile(objects);

break;

case 3:

visualizeObjects(objects);

break;

case 4:

performOperations(objects);

break;

case 5:

running = false;

std::cout << "Exiting the program." << std::endl;

break;

default:

std::cerr << "Invalid choice. Please try again." << std::endl;

break;

}

}

return 0;

}

longArithmetics\_2.h:

#ifndef LONGARITHMETICS\_2\_H

#define LONGARITHMETICS\_2\_H

#include <iostream>

#include <vector>

#include <fstream>

class LongArithmetic {

public:

// Конструктори

LongArithmetic();

LongArithmetic(double value);

LongArithmetic(const LongArithmetic& other);

// Деструктор

~LongArithmetic();

// Оператори

LongArithmetic& operator=(const LongArithmetic& other);

double& operator[](size\_t index);

const double& operator[](size\_t index) const;

LongArithmetic operator+(const LongArithmetic& other) const;

LongArithmetic operator-(const LongArithmetic& other) const;

LongArithmetic operator\*(const LongArithmetic& other) const;

LongArithmetic operator/(const LongArithmetic& other) const;

// Перевантаження потокового вводу/виводу

friend std::ostream& operator<<(std::ostream& os, const LongArithmetic& la);

friend std::istream& operator>>(std::istream& is, LongArithmetic& la);

// Збереження та відновлення

void saveToFile(const std::string& filename) const;

void restoreFromFile(const std::string& filename);

// Діалогове управління

void input();

void output() const;

// "Розумний доступ"

double\* operator->();

const double\* operator->() const;

// Псевдо змінні

double f(double x) const;

private:

std::vector<double> values;

void ensureCapacity(size\_t index);

};

#endif // LONGARITHMETIC\_H

# Опис інтерфейсу програми (керівництво користувача)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описаниеЯк тільки програма буде запущена, вона виведе користувачеві меню, за допомогою якого він може обирати дії, які хоче виконати з об'єктами великої арифметики.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание1.Створити об'єкт/об'єкти

Якщо обрати цей пункт, програма попросить користувача ввести значення для створення нового об'єкта класу LongArithmetic. Після введення числа програма створить об'єкт і додасть його до списку існуючих об'єктів. Якщо під час введення значення сталася помилка, програма повідомить про це і попросить ввести значення знову.

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание2. Збереження об'єктів у файл

Якщо обрати цей пункт, програма попросить ввести ім'я файлу, в який потрібно зберегти поточні об'єкти. Після введення імені файлу програма збереже всі створені об'єкти в цей файл. Якщо виникнуть проблеми з відкриттям файлу, програма повідомить про це.

3. Відновлення об'єктів з файлу

Якщо обрати цей пункт, програма запитає ім'я файлу, з якого потрібно відновити об'єкти. Після введення імені файлу програма зчитає значення з файлу і створить на їх основі об'єкти, додавши їх до списку. Якщо виникнуть проблеми з відкриттям файлу, програма повідомить про це.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание4. Візуалізація об'єктів

Якщо обрати цей пункт, програма виведе на екран всі створені на даний момент об'єкти класу LongArithmetic. Це дозволяє користувачу переглянути поточний стан

об'єктів.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание5. Виконати операції над об'єктами

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеЯкщо обрати цей пункт, програма запропонує виконати одну з арифметичних операцій (додавання, віднімання, множення або ділення) між двома об'єктами класу LongArithmetic. Користувач повинен вибрати тип операції, а також вказати індекси двох об'єктів, між якими необхідно виконати операцію. Результат операції буде виведено на екран. Якщо було введено некоректні індекси або сталася інша помилка, програма повідомить про це.

6. Вихід

Якщо обрати цей пункт, програма завершить свою роботу і закриється.

# Висновок

Виконуючи цю лабораторну роботу, я дізналася, як реалізувати та

використовувати основні концепції об'єктно-орієнтованого програмування на

прикладі створення та маніпулювання множинами в мові C++. Я навчилася

створювати класи та конструктори, включаючи конструктор копіювання, і

використовувати перевантаження операторів для зручного доступу до елементів

об'єктів.

Окрім цього, я створила методи для додавання, видалення та перевірки

наявності елементів у множині. Важливим аспектом було використання дружніх

функцій для доступу до приватних членів класу, що допомогло краще

організувати код. Також я реалізувала методи для об'єднання та перетину

множин, що дозволило ефективно комбінувати та аналізувати множини.

Ця лабораторна робота допомогла мені краще зрозуміти основи роботи з

класами та об'єктами в C++, а також практично застосувати знання з об'єктноорієнтованого програмування для вирішення конкретних задач.