Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та комп’ютерних наук

Кафедра математичних проблем управління і кібернетики

Звіт

про виконання лабораторної роботи №3

«Основи Об’єктно – орієнтоване програмування мовою С++. Класи. Протокол

класу. Конструктори та деструктори»

з дисципліни

«Об’єктно-орієнтоване програмування»

Виконала: студентка 141-А+Б групи

Жданюк А.С.

Перевірив: канд.фіз.-мат. наук,

доцент Лазорик В.В.

Оцінка:

Дата захисту:

Чернівці 2024

**Тема:**Основи Об’єктно – орієнтоване програмування мовою С++. Класи. Протокол класу. Конструктори та деструктори

**Виконання лабораторної роботи**

1. Зайти в свій обліковий запис на github.com.

2. Клонувати репозиторій: <https://classroom.github.com/a/8Ty5xvkR> на робочий комп’ютер. В репозиторії знаходиться в функції main() викликається демонстраційний приклад виконання подібної лабораторної роботи, який змінюєте на власний код.

3. Розв’язати задачі.

4. В процесі написання функцій розв’язання задач лабораторної роботи періодично здійснювати синхронізацію з репозиторієм на github.com, з поясненням виконаної роботи.

5. У функції main() створити меню яке б викликало створені функції.

6. Оформити звіт про виконання лабораторної роботи.

7. Звіт відправити в для оцінювання в [https://moodle.chnu.edu.ua](https://moodle.chnu.edu.ua/)

**Мої варіанти задач**:

1.7

Створити клас типу гра в хрестики-нулики (поля: класу - масив з (3х3), стан, активний гравець). У класі визначити

o конструктор;

o функції-члени перевірки кінця гри, виведення повідомлень по стан

гри, початку гри, ходів граків, тощо;

Написати програму тестування всіх можливостей цього класу

2.7 Створити тип даних - клас вектор, який має масив з трьох елементів типу double і змінну стану. У класі визначити

o конструктор без параметрів(інінціалізує поля в нуль);

o конструктор з одним параметром типу double (інінціалізує масив значенням параметру);

o конструктор з одним параметром вказівник на тип (інінціалізує масив класу значенням елементів масиву, який передається параметр за вказівником, якщо вказівник NULL (nulptr) то встановити код помилки);

o деструктор із виведенням інформації про стан вектора;

o визначити функцію, яка присвоює масиву деяке значення (параметр за замовчуванням);

o функцію яка одержує деякий елемент з масиву;

o визначити функції друку, додавання, віднімання та векторного добутку які здійснюють ці арифметичні операції з даними цього класу;

o функцію ділення на ціле типу short(при діленні на 0 змінити стан, а ділення не виконувати);

o визначити функції порівняння: більше, менше або рівно, які повертають true або false.

У змінну стани встановлювати код помилки, діленні на 0, при передачі NULL (nulptr) в конструкторі із вказівником. Передбачити можливість підрахунку числа об'єктів даного типу.

Написати програму тестування всіх можливостей цього класу.

**Хід лабораторної роботи**

**Перше завдання**



#include <iostream>

#include <vector>

#include <string>

using namespace std;

class TicTacToe {

private:

vector<vector<char>> board;

bool activePlayer; // true for X, false for O

bool gameActive;

public:

TicTacToe() {

// Initialize the board with empty cells

board = {{' ', ' ', ' '},

{' ', ' ', ' '},

{' ', ' ', ' '}};

activePlayer = true; // X starts the game

gameActive = true;

}

void displayBoard() {

cout << "-------------" << endl;

for (int i = 0; i < 3; ++i) {

cout << "| ";

for (int j = 0; j < 3; ++j) {

cout << board[i][j] << " | ";

}

cout << endl << "-------------" << endl;

}

}

bool checkWin(char player) {

// Check rows

for (int i = 0; i < 3; ++i) {

if (board[i][0] == player && board[i][1] == player && board[i][2] == player)

return true;

}

// Check columns

for (int i = 0; i < 3; ++i) {

if (board[0][i] == player && board[1][i] == player && board[2][i] == player)

return true;

}

// Check diagonals

if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||

(board[0][2] == player && board[1][1] == player && board[2][0] == player))

return true;

return false;

}

bool checkDraw() {

// Check if the board is full

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

if (board[i][j] == ' ')

return false; // There are still empty cells

}

}

return true; // The board is full (draw)

}

void switchPlayer() {

activePlayer = !activePlayer;

}

void makeMove(int row, int col) {

if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != ' ') {

cout << "Invalid move. Try again." << endl;

return;

}

board[row][col] = (activePlayer ? 'X' : 'O');

switchPlayer();

}

void playGame() {

while (gameActive) {

displayBoard();

char currentPlayer = (activePlayer ? 'X' : 'O');

cout << "Player " << currentPlayer << ", enter your move (row column): ";

int row, col;

cin >> row >> col;

makeMove(row, col);

if (checkWin(currentPlayer)) {

displayBoard();

cout << "Player " << currentPlayer << " wins!" << endl;

gameActive = false;

} else if (checkDraw()) {

displayBoard();

cout << "It's a draw!" << endl;

gameActive = false;

}

}

}

};

int main() {

TicTacToe game;

game.playGame();

return 0;

}

**Друге завдання**



#include <iostream>

#include <cmath>

class Vector {

private:

double data[3];

int state; // 0 for OK, -1 for error

public:

static int count; // static member to count the number of objects

Vector() : state(0) {

for (int i = 0; i < 3; ++i) {

data[i] = 0.0;

}

count++;

}

Vector(double value) : state(0) {

for (int i = 0; i < 3; ++i) {

data[i] = value;

}

count++;

}

Vector(double\* arr) : state(0) {

if (arr == nullptr) {

state = -1; // Error: nullptr passed

std::cout << "Error: nullptr passed to constructor" << std::endl;

} else {

for (int i = 0; i < 3; ++i) {

data[i] = arr[i];

}

}

count++;

}

~Vector() {

std::cout << "Vector state: " << state << std::endl;

count--;

}

void assign(double value = 0.0) {

for (int i = 0; i < 3; ++i) {

data[i] = value;

}

}

double getElement(int index) const {

if (index < 0 || index >= 3) {

std::cout << "Invalid index." << std::endl;

return 0.0; // Return 0 if index is out of bounds

}

return data[index];

}

void inputVector() {

std::cout << "Enter three values for the vector: ";

for (int i = 0; i < 3; ++i) {

std::cin >> data[i];

}

}

void print() const {

std::cout << "(" << data[0] << ", " << data[1] << ", " << data[2] << ")" << std::endl;

}

Vector add(const Vector& other) const {

Vector result;

for (int i = 0; i < 3; ++i) {

result.data[i] = data[i] + other.data[i];

}

return result;

}

Vector subtract(const Vector& other) const {

Vector result;

for (int i = 0; i < 3; ++i) {

result.data[i] = data[i] - other.data[i];

}

return result;

}

double dotProduct(const Vector& other) const {

double result = 0.0;

for (int i = 0; i < 3; ++i) {

result += data[i] \* other.data[i];

}

return result;

}

Vector scalarMultiply(double scalar) const {

Vector result;

for (int i = 0; i < 3; ++i) {

result.data[i] = data[i] \* scalar;

}

return result;

}

Vector divide(short divisor) {

if (divisor == 0) {

state = -1; // Error: division by zero

std::cout << "Error: Division by zero." << std::endl;

return \*this;

}

Vector result;

for (int i = 0; i < 3; ++i) {

result.data[i] = data[i] / divisor;

}

return result;

}

bool operator>(const Vector& other) const {

double mag1 = sqrt(data[0] \* data[0] + data[1] \* data[1] + data[2] \* data[2]);

double mag2 = sqrt(other.data[0] \* other.data[0] + other.data[1] \* other.data[1] + other.data[2] \* other.data[2]);

return mag1 > mag2;

}

bool operator<(const Vector& other) const {

double mag1 = sqrt(data[0] \* data[0] + data[1] \* data[1] + data[2] \* data[2]);

double mag2 = sqrt(other.data[0] \* other.data[0] + other.data[1] \* other.data[1] + other.data[2] \* other.data[2]);

return mag1 < mag2;

}

bool operator==(const Vector& other) const {

return (data[0] == other.data[0] && data[1] == other.data[1] && data[2] == other.data[2]);

}

};

int Vector::count = 0;

int main() {

Vector v1, v2;

std::cout << "Enter values for vector v1:" << std::endl;

v1.inputVector();

std::cout << "Enter values for vector v2:" << std::endl;

v2.inputVector();

std::cout << "Vector v1: ";

v1.print();

std::cout << "Vector v2: ";

v2.print();

Vector sum = v1.add(v2);

std::cout << "Sum of v1 and v2: ";

sum.print();

Vector difference = v1.subtract(v2);

std::cout << "Difference of v1 and v2: ";

difference.print();

double dot = v1.dotProduct(v2);

std::cout << "Dot product of v1 and v2: " << dot << std::endl;

std::cout << "Number of Vector objects: " << Vector::count << std::endl;

return 0;

}