МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 2

з навчальної дисципліни

# “ ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ ”

Конструктори і деструктори

ВИКОНАВ

студент академічної групи КБ-23

Сергій Козирь

ПЕРЕВІРИВ

Викладач

Козірова Н.Л.

Кропивницький – 2024

Лабораторна робота №2

Конструктори і деструктори

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

Варіант 1

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

Варіант 1 Розробіть клас «Дріб» – Fraction у вигляді пари m, n. Клас повинен містити конструктори. Реалізуйте методи для додавання, віднімання, множення та ділення дробів. Створіть масив об'єктів і передайте його у функцію, яка змінює кожен елемент масиву з парним індексом шляхом додавання наступного за ним елемента масиву.

**Завдання 1**

**main.cpp**

#include <iostream>

#include "Book.h"

int main() {

setlocale(LC\_ALL, "ukr");

// 1. Створення об'єкта за допомогою конструктора за замовчуванням

Book defaultBook;

std::cout << "Назва книги за замовчуванням: " << defaultBook.getTitle() << std::endl;

std::cout << "Автор книги за замовчуванням: " << defaultBook.getAuthor() << std::endl;

std::cout << "Рік видання за замовчуванням: " << defaultBook.getYear() << std::endl;

// 2. Створення об'єкта за допомогою конструктора з параметрами

Book paramBook("1984", "George Orwell", 1949);

std::cout << "Назва книги: " << paramBook.getTitle() << std::endl;

std::cout << "Автор книги: " << paramBook.getAuthor() << std::endl;

std::cout << "Рік видання: " << paramBook.getYear() << std::endl;

// 3. Створення об'єкта за допомогою копіюючого конструктора

Book copyBook(paramBook);

std::cout << "Назва скопійованої книги: " << copyBook.getTitle() << std::endl;

std::cout << "Автор скопійованої книги: " << copyBook.getAuthor() << std::endl;

std::cout << "Рік видання скопійованої книги: " << copyBook.getYear() << std::endl;

// Завершення програми, автоматичне викликання деструкторів

return 0;

}

**Book.cpp**

#include "Book.h"

// Конструктор за замовчуванням

Book::Book() : title(""), author(""), year(0) {

std::cout << "Конструктор за замовчуванням викликано\n";

}

// Конструктор з параметрами

Book::Book(std::string t, std::string a, int y) : title(t), author(a), year(y) {

std::cout << "Конструктор з параметрами викликано\n";

}

// Копіюючий конструктор

Book::Book(const Book& other) : title(other.title), author(other.author), year(other.year) {

std::cout << "Копіюючий конструктор викликано\n";

}

// Деструктор

Book::~Book() {

std::cout << "Деструктор викликано для книги: " << title << "\n";

}

// Методи доступу

void Book::setTitle(const std::string& t) {

title = t;

}

std::string Book::getTitle() const {

return title;

}

void Book::setAuthor(const std::string& a) {

author = a;

}

std::string Book::getAuthor() const {

return author;

}

void Book::setYear(int y) {

year = y;

}

int Book::getYear() const {

return year;

}

**Book.h**

#ifndef BOOK\_H

#define BOOK\_H

#include <string>

#include <iostream>

class Book {

private:

std::string title;

std::string author;

int year;

public:

// Конструкторы

Book(); // Конструктор по умолчанию

Book(std::string title, std::string author, int year); // Конструктор с параметрами

Book(const Book& other); // Копирующий конструктор

// Деструктор

~Book();

// Методы доступа

void setTitle(const std::string& title);

std::string getTitle() const;

void setAuthor(const std::string& author);

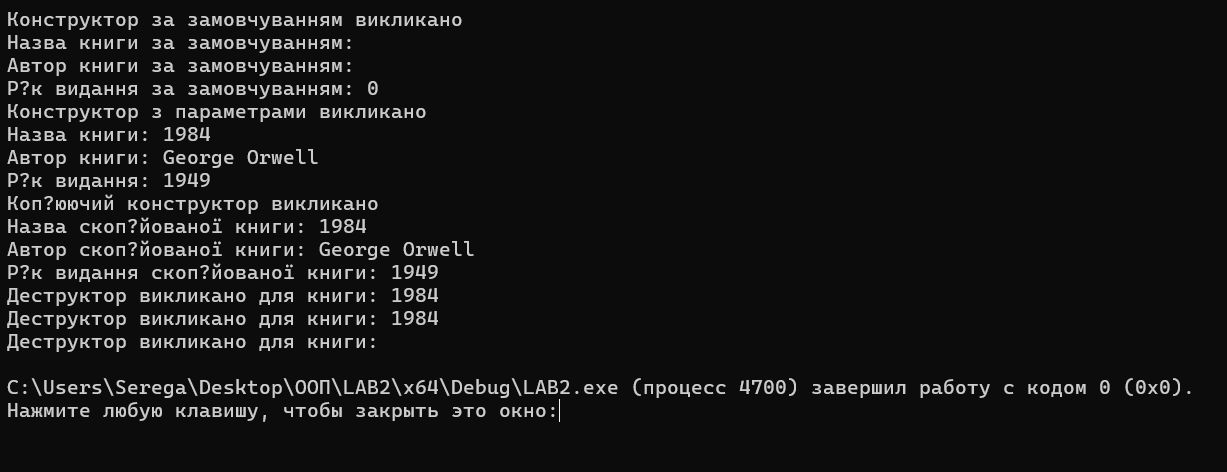
std::string getAuthor() const;

void setYear(int year);

int getYear() const;

};

#endif // Закрываем директиву #ifndef



**Завдання 2**

**main.cpp**

#include <iostream>

#include "Fraction.h"

// Функция для изменения массива дробей

void modifyArray(Fraction arr[], int size) {

for (int i = 0; i < size - 1; i += 2) {

arr[i] = arr[i].add(arr[i + 1]);

}

}

int main() {

setlocale(LC\_ALL, "ukr");

// Создаем массив объектов Fraction

const int size = 4;

Fraction fractions[size] = { Fraction(1, 2), Fraction(2, 3), Fraction(3, 4), Fraction(4, 5) };

std::cout << "Исходный массив дробей:\n";

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

fractions[i].print();

std::cout << " ";

}

std::cout << "\n";

// Изменяем массив дробей

modifyArray(fractions, size);

std::cout << "Массив после изменения:\n";

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

fractions[i].print();

std::cout << " ";

}

std::cout << "\n";

return 0;

}

**Fraction.cpp**

#include "Fraction.h"

#include <iostream>

// Реализация функции gcd (алгоритм Евклида)

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

// Конструктор по умолчанию

Fraction::Fraction() : m(0), n(1) {}

// Конструктор с параметрами

Fraction::Fraction(int m, int n) : m(m), n(n) {

if (n == 0) {

std::cerr << "Ошибка: знаменатель не может быть нулевым.\n";

exit(1);

}

reduce(); // Сокращаем дробь при создании

}

// Внутренний метод для сокращения дроби

void Fraction::reduce() {

int divisor = gcd(m, n);

m /= divisor;

n /= divisor;

if (n < 0) { // Убедимся, что знаменатель всегда положителен

m = -m;

n = -n;

}

}

// Метод сложения дробей

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

int num = m \* other.n + other.m \* n;

int denom = n \* other.n;

return Fraction(num, denom);

}

// Метод для вывода дроби

void Fraction::print() const {

std::cout << m << "/" << n << std::endl;

}

**Fraction.h**

#ifndef FRACTION\_H

#define FRACTION\_H

#include <iostream>

class Fraction {

private:

int m; // Числитель

int n; // Знаменатель

// Внутренний метод для сокращения дроби

void reduce();

public:

// Конструкторы

Fraction(); // Конструктор по умолчанию

Fraction(int m, int n); // Конструктор с параметрами

// Методы для арифметических операций

Fraction add(const Fraction& other) const;

Fraction subtract(const Fraction& other) const;

Fraction multiply(const Fraction& other) const;

Fraction divide(const Fraction& other) const;

// Методы для вывода и доступа к полям

void print() const;

int getNumerator() const { return m; }

int getDenominator() const { return n; }

};

#endif

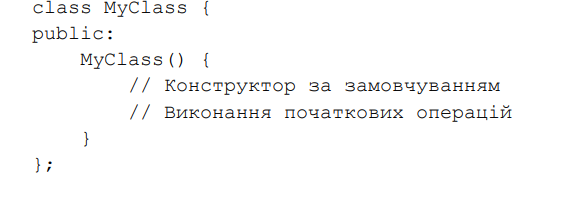


**Конструктор** є спеціальним методом в класі, який викликається автоматично при створенні нового об'єкта цього класу. Його основна функція полягає в ініціалізації об'єкта, тобто встановленні його початкового стану.

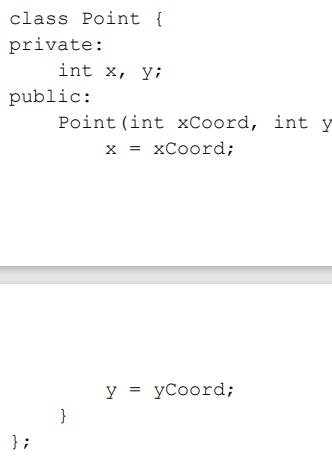
• Конструктор за замовчуванням (default constructor):

Це конструктор без параметрів, який викликається при створенні об'єкта без передачі яких-небудь значень. Він може ініціалізувати члени даних об'єкта значеннями за замовчуванням.

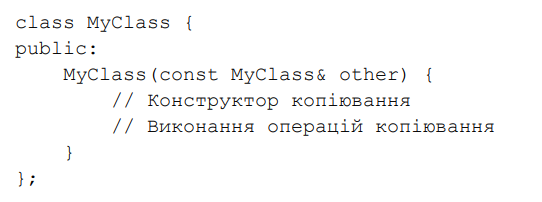
Наприклад:

 • Параметризований конструктор (parameterized constructor): Це конструктор, який приймає параметри і використовує їх для ініціалізації членів даних об'єкта.

Наприклад:



• Конструктор копіювання (copy constructor): Це конструктор, який приймає об'єкт того ж класу як параметр і створює новий об'єкт, який є копією цього переданого об'єкта. Конструктор копіювання використовується, коли потрібно зробити копію об'єкта для подальшої роботи з ним, а не просто посилання на нього. Наприклад:

Конструктори дозволяють забезпечити коректну ініціалізацію об'єктів і можуть бути потужним інструментом у процесі програмування на C++.

**Деструктор** (~) є спеціальним методом в класі, який викликається автоматично під час звільнення пам'яті, знищення об'єкта або виходу з блоку, в якому об'єкт був створений. Він використовується для виконання ресурсоозберігаючих операцій або очищення пам'яті, які можуть бути пов'язані з об'єктом.