Міністерство освіти та науки України

Інститут «Телекомунікації, радіоелектроніки та радіотехніки»

Изображение выглядит как текст, круг, Шрифт, эмблема

Автоматически созданное описание

Звіт

З лабораторної роботи №7

З дисципліни «Об`єктно-орієнтоване програмування»

На тему: «Робота з динамічними масивами»

Виконав:

Студент групи АП-22

Іщак Д.А.

Прийняв:

Алтунін С. І

Львів 2024

**Мета:** Створити, відлагодити та протестувати програму, у якій створити клас для обробки даних. Одним з членів класу є масив структур, в якому зберігаються введені дані.

**Хід роботи:**

1. продумайте, які типи змінних вам потрібно використати для виконання завдання.
2. створіть проєкт із кількома файлами, тобто крім основного файлу, наприклад main.cpp, додайте два файл, наприклад student.h та student.cpp. Для цього потрібно виправити файл ***..\.vscode\tasks.json***:

{

"version": "2.0.0", "tasks": [

{

"type": "cppbuild",

"label": "C/C++: g++.exe build active file", "command": "C:\\msys64\\ucrt64\\bin\\g++.exe", "args": [

"-fdiagnostics-color=always", "-g",

| замість відносного шляху до виконуваного файла: | вказати відносний шлях директорії проєкту до всіх файлів \*.cpp, /\*.h: |
| --- | --- |
| "${file}", | "${workspaceFolder}/\*.cpp", "${workspaceFolder}/\*.h", |

"-o",

"${fileDirname}\\${fileBasenameNoExtension}.exe"

],

"options": {

"cwd": "${fileDirname}"

},

"problemMatcher": [ "$gcc"

],

"group": {

"kind": "build", "isDefault": true

},

"detail": "compiler: C:\\msys64\\ucrt64\\bin\\g++.exe"

}

]

}

1. У заголовному файлі \*.h оголосіть клас:

class Student{

private:

std::vector<Student> students {0};

…

}

1. оголосіть відповідну структуру, наприклад student\_s\_t,перед

оголошенням класу:

struct student\_s\_t{ int id; std::string name; int course;

};

1. реалізуйте для класу конструктор та деструктор.
2. реалізуйте методи класу для введення і виведення даних.
3. методи зміни полів класу повинні перевіряти валідність заданих змінних і попереджувати користувача про некоректні значення.
4. реалізуйте методи обробки даних: додавання, вставки, вилучення, обміну, очищення (видалення).
5. можете збільшити кількість членів класу додатковими полями чи методами.
6. програма має вивести на консоль (термінал) вхідні дані та результат виконання.

**Варіант завдання:**



**Код програми:**

**main.cpp:**

**#include <iostream>**

**#include <vector>**

**#include <string>**

**// Структура для зберігання даних про автомобіль**

**struct CarInfo {**

**int id;**

**std::string brand;**

**std::string model;**

**int year;**

**// Конструктор структури**

**CarInfo(int id, const std::string& brand, const std::string& model, int year)**

**: id(id), brand(brand), model(model), year(year) {}**

**};**

**// Клас для управління списком автомобілів**

**class CarManager {**

**private:**

**std::vector<CarInfo> cars; // Динамічний масив автомобілів**

**public:**

**// Конструктор і деструктор**

**CarManager() = default;**

**~CarManager() = default;**

**// Додавання нового автомобіля**

**void addCar(int id, const std::string& brand, const std::string& model, int year) {**

**if (year < 1886 || year > 2024) {**

**std::cerr << "Некоректний рік випуску автомобіля!\n";**

**return;**

**}**

**cars.emplace\_back(id, brand, model, year);**

**std::cout << "Автомобіль додано: ID " << id << ", " << brand << " " << model << ", " << year << " рік\n";**

**}**

**// Виведення списку всіх автомобілів**

**void displayCars() const {**

**if (cars.empty()) {**

**std::cout << "Список автомобілів порожній.\n";**

**return;**

**}**

**for (const auto& car : cars) {**

**std::cout << "ID: " << car.id << ", Марка: " << car.brand**

**<< ", Модель: " << car.model << ", Рік випуску: " << car.year << "\n";**

**}**

**}**

**// Пошук автомобіля за ID**

**void findCarById(int id) const {**

**for (const auto& car : cars) {**

**if (car.id == id) {**

**std::cout << "Знайдено автомобіль: ID " << car.id << ", " << car.brand**

**<< " " << car.model << ", " << car.year << " рік\n";**

**return;**

**}**

**}**

**std::cout << "Автомобіль із ID " << id << " не знайдено.\n";**

**}**

**// Видалення автомобіля за ID**

**void deleteCarById(int id) {**

**for (auto it = cars.begin(); it != cars.end(); ++it) {**

**if (it->id == id) {**

**std::cout << "Видалено автомобіль: ID " << it->id << ", " << it->brand**

**<< " " << it->model << ", " << it->year << " рік\n";**

**cars.erase(it);**

**return;**

**}**

**}**

**std::cout << "Автомобіль із ID " << id << " не знайдено для видалення.\n";**

**}**

**};**

**// Головна функція програми**

**int main() {**

**CarManager manager;**

**// Додавання автомобілів**

**manager.addCar(1, "Toyota", "Corolla", 2020);**

**manager.addCar(2, "Honda", "Civic", 2018);**

**manager.addCar(3, "BMW", "X5", 2022);**

**// Виведення всіх автомобілів**

**std::cout << "\nСписок усіх автомобілів:\n";**

**manager.displayCars();**

**// Пошук автомобіля за ID**

**std::cout << "\nПошук автомобіля із ID 2:\n";**

**manager.findCarById(2);**

**// Видалення автомобіля**

**std::cout << "\nВидалення автомобіля із ID 2:\n";**

**manager.deleteCarById(2);**

**// Виведення списку після видалення**

**std::cout << "\nСписок автомобілів після видалення:\n";**

**manager.displayCars();**

**return 0;**

**}**

**Результати виконання роботи:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Блок-схема програми:**

**Изображение выглядит как текст, чек, документ, Шрифт

Автоматически созданное описание**

Висновок:

У цій лабораторній роботі реалізовано клас для роботи з динамічним масивом структур, використовуючи контейнер std::vector. Програма дозволяє додавати, перевіряти, виводити та очищати список вантажівок. Результатом роботи є гнучка та ефективна система для обробки даних, яка демонструє переваги використання динамічних масивів і контейнерів STL у C++.