Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

**Лабораторна робота №5**

**Тема:** **«**Контейнерні класи. Стандартна бібліотека шаблонів (STL) в С++**»**

|  |
| --- |
| Виконав: ст. гр. КБ-23 |
| Захарченко А.О. |
| Перевірив: викладач  Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

**Мета:** ознайомитись Контейнерні класи та навчитись їх програмно

реалізовувати мовою С++.

**Завдання:**

Варіант 3

Розробіть систему управління завданнями. У цій системі потрібно

зберігати та керувати списком завдань. Використовуйте контейнерні класи зі

стандартної бібліотеки шаблонів (STL) для збереження та операцій над

списком завдань.

 Додайте декілька завдань до списку, включаючи їх назву, опис та

дедлайн.

 Виведіть список завдань.

 Видаліть одне або декілька завдань зі списку за назвою.

 Виведіть оновлений список завдань.

 Виведіть список завдань з певним дедлайном.

Ви маєте розширити програму, додати додаткові методи та функціональні

можливості, які вам здаються відповідними.

**Реалізація:**

**Main.cpp**

#include <iostream>

#include "task.h"

void showMenu() {

std::cout << "\n---- Task Manager Menu ----\n";

std::cout << "1. Add a new task\n";

std::cout << "2. Remove a task by title\n";

std::cout << "3. Show all tasks\n";

std::cout << "4. Show tasks by deadline\n";

std::cout << "5. Exit\n";

std::cout << "Enter your choice: ";

}

int main() {

TaskManager manager;

int choice;

do {

showMenu();

std::cin >> choice;

std::cin.ignore();

switch (choice) {

case 1: {

std::string title, description, deadline;

std::cout << "Enter task title: ";

std::getline(std::cin, title);

std::cout << "Enter task description: ";

std::getline(std::cin, description);

std::cout << "Enter task deadline: ";

std::getline(std::cin, deadline);

manager.addTask(Task(title, description, deadline));

std::cout << "Task added successfully.\n";

break;

}

case 2: {

std::string title;

std::cout << "Enter title of task to remove: ";

std::getline(std::cin, title);

manager.removeTaskByTitle(title);

std::cout << "Task removed successfully.\n";

break;

}

case 3: {

manager.printTasks();

break;

}

case 4: {

std::string deadline;

std::cout << "Enter deadline to filter tasks (YYYY-MM-DD): ";

std::getline(std::cin, deadline);

manager.printTasksByDeadline(deadline);

break;

}

case 5: {

std::cout << "Exiting...\n";

break;

}

default:

std::cout << "Invalid choice. Please try again.\n";

}

} while (choice != 5);

return 0;

}

**Task.cpp**

#include "task.h"

void TaskManager::addTask(const Task& task) {

tasks.push\_back(task);

}

void TaskManager::removeTaskByTitle(const std::string& title) {

tasks.erase(std::remove\_if(tasks.begin(), tasks.end(),

[&title](const Task& task) { return task.getTitle() == title; }),

tasks.end());

}

void TaskManager::printTasks() const {

if (tasks.empty()) {

std::cout << "No tasks available." << std::endl;

return;

}

std::cout << "Task List:\n";

for (const auto& task : tasks) {

task.printTask();

}

}

void TaskManager::printTasksByDeadline(const std::string& deadline) const {

bool found = false;

for (const auto& task : tasks) {

if (task.getDeadline() == deadline) {

task.printTask();

found = true;

}

}

if (!found) {

std::cout << "No tasks found with the deadline " << deadline << "." << std::endl;

}

}

**Task.h**

#ifndef TASK\_MANAGER\_H

#define TASK\_MANAGER\_H

#include <string>

#include <vector>

#include <iostream>

#include <algorithm>

class Task {

private:

std::string title;

std::string description;

std::string deadline;

public:

Task(const std::string& title, const std::string& description, const std::string& deadline)

: title(title), description(description), deadline(deadline) {}

std::string getTitle() const { return title; }

std::string getDescription() const { return description; }

std::string getDeadline() const { return deadline; }

void setDescription(const std::string& desc) { description = desc; }

void setDeadline(const std::string& date) { deadline = date; }

void printTask() const {

std::cout << "Title: " << title << ", Description: " << description << ", Deadline: " << deadline << std::endl;

}

};

class TaskManager {

private:

std::vector<Task> tasks;

public:

void addTask(const Task& task);

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

void printTasks() const;

void printTasksByDeadline(const std::string& deadline) const;

};

#endif // TASK\_MANAGER\_H

**Результат:**

****

**Висновок:** в ході виконання роботи було опрацьовано поняття контейнерних класів та стандартної бібліотеки шаблонів в С++.