МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Факультет Комп’ютерних наук та програмної інженерії

Кафедра Інформатики та інтелектуальної власності

ЗВІТ

До лабораторной роботи №5 з дисципліни

«Об’єкто-орієнтоване програмування»

Студент Бородай Д.А

Викладач Івашко А.В.

Харків 2022

**Лабораторна робота №5**

**Тема: «Створення та використання бібліотек DLL»**

**Мета роботи:** навчитись створювати та використовувати власної бібліотеки динамічного компонування засобами середовища Visual Studio.

**Завдання:**

1. Для лабораторної роботи №1 приховати реалізований функціонал власного варіанту до бібліотеки func.dll. Перед виконанням модифікації проекту треба обов’язково створити його копію. Перевірити роботу проекту

2. Для лабораторної роботи №2 приховати реалізацію перевантаження операторів власного варіанта до бібліотеки oper.dll. Перед виконанням модифікації проекту треба обов’язково створити його копію. Перевірити роботу проекту.

3. Для лабораторної роботи №3 приховати реалізацію методів кожного класу за власним варіантом у відповідній dll-бібліотеці, а функціонал керування усією системою класів – у бібліотеці comtrol.dll Перед виконанням модифікації проекту треба обов’язково створити його копію. Перевірити роботу проекту.

Зміст

[1 Аналіз до предметної області 4](#_Toc119660108)

[2 Виконання роботи 5](#_Toc119660109)

[2.1 Інструкція по створенню бібліотеки 5](#_Toc119660110)

[2.2 Створення бібліотеки func для першої лабораторної 12](#_Toc119660111)

[2.3 Створення бібліотеки oper для другої лабораторної роботи 12](#_Toc119660112)

[2.4 Створення бібліотеки класів для третьої лабораторної 12](#_Toc119660113)

[Висновок 13](#_Toc119660114)

[Додаток А 15](#_Toc119660115)

[Додаток Б 17](#_Toc119660116)

[Додаток В 27](#_Toc119660117)

[Додаток Д 29](#_Toc119660118)

## 1 Аналіз до предметної області

У Windows бібліотека динамічного компонування (DLL) є виконуваним файлом, який виступає як загальна бібліотека функцій і ресурсів. Динамічне компонування - це можливість операційної системи. Вона дає змогу виконуваному файлу викликати функції або використовувати ресурси, що зберігаються в окремому файлі. Ці функції і ресурси можна компілювати і розгортати окремо від виконуваних файлів, які їх використовують.

Бібліотека DLL не є окремим виконуваним файлом. Бібліотеки DLL виконуються в контексті додатків, які їх викликають. Операційна система завантажує бібліотеку DLL в область пам'яті програми. Це робиться або під час завантаження програми (неявне компонування), або за запитом під час виконання (явне компонування). Бібліотеки DLL також спрощують спільне використання функцій і ресурсів різними виконуваними файлами. Кілька додатків можуть здійснювати одночасний доступ до вмісту однієї копії бібліотеки DLL у пам'яті.

У разі статичного компонування весь код об'єктів копіюється зі статичної бібліотеки в виконувані файли, що їх використовують, під час складання. Під час динамічного компонування включаються тільки ті відомості, які дають змогу Windows знайти і завантажити бібліотеку DLL, що містить елемент даних або функцію, під час виконання. Під час створення бібліотеки DLL також створюється бібліотека імпорту, що містить цю інформацію. Під час складання виконуваного файлу, який викликає бібліотеку DLL, компоновщик використовує експортовані символи в бібліотеці імпорту, щоб зберегти ці відомості для завантажувача Windows. Коли завантажувач завантажує бібліотеку DLL, її зіставляють з областю пам'яті програми. Для виконання операцій ініціалізації, необхідних бібліотеці DLL, викликається спеціальна функція DllMain з бібліотеки DLL (якщо вона є).

## 2 Виконання роботи

### 2.1 Інструкція по створенню бібліотеки

У меню виберіть Файл>Створити>Проект, щоб відкрити діалогове вікно Створення проекту.

У верхній частині діалогового вікна задайте мову C++, задайте для платформи Windows і задайте для Project тип "Бібліотека".

У відфільтрованому списку типів проектів клацніть Бібліотека динамічної компонування (DLL) , а потім натисніть кнопку Далі.

На сторінці Налаштувати новий проект введіть бажану назву проекту у полі Ім'я проекту. Прийміть за промовчанням Розташування та Ім'я рішення. Для параметра Рішення задайте Створити нове рішення. Зніміть прапорець Розмістити рішення та проект в одному каталозі, якщо його встановлено.

Натисніть кнопку Створити, щоб створити проект.

Після створення рішення створений проект разом з вихідними файлами відобразиться у вікні браузера рішень у Visual Studio.

Додавання файлу заголовка до бібліотеки DLL

Щоб створити файл заголовка для функцій, клацніть Проект>Додати новий елемент.

У діалоговому вікні Додати новий елемент у лівій області клацніть Visual C++ . У центральній області виберіть Заголовний файл (.h) . Вкажіть назва.h як ім'я файлу заголовка.

Натисніть кнопку , щоб створити порожній файл заголовка, який відображається в новому вікні редактора.

Замініть весь вміст цього файлу заголовка наступним кодом:

#pragma once

#ifdef MATHLIBRARY\_EXPORTS

#define MATHLIBRARY\_API \_\_declspec(dllexport)

#else

#define MATHLIBRARY\_API \_\_declspec(dllimport)

#endif

extern "C++" MATHLIBRARY\_API прототип 1;

extern "C++" MATHLIBRARY\_API прототип 2;

MATHLIBRARY\_EXPORTS При визначенні MATHLIBRARY\_API макрос макрос задає модифікатор \_\_declspec(dllexport) для оголошень функцій. Цей модифікатор наказує компілятору та компонувальнику експортувати функцію або змінну з бібліотеки DLL для використання іншими програмами. Якщо значення MATHLIBRARY\_EXPORTS не визначено, наприклад, якщо файл заголовка включений клієнтською програмою, MATHLIBRARY\_API застосовує \_\_declspec(dllimport) модифікатор до оголошень. Цей модифікатор оптимізує імпорт функції або змінної програми.

Додавання реалізації до бібліотеки DLL

У браузері рішень клацніть правою кнопкою миші вузол Файли рішення та виберіть Додати > Новий елемент. Створіть новий CPP-файл з ім'ям MathLibrary.cpp, аналогічно додавання нового файлу заголовка на попередньому кроці.

У вікні редактора виберіть вкладку MathLibrary.cpp, якщо вона вже відкрита. Якщо ні, то в браузері рішень двічі клацніть файл MathLibrary.cpp у папці Вихідні файли проекту MathLibrary.

У редакторі замініть вміст файлу MathLibrary.cpp наступним кодом:

#include "pch.h" // стандартний файл

#include "MathLibrary.h" //створений заготовочний файл

Реалізація1()

{

…

}

Реалізація2()

{

…

}

…

Реалізаціяn()

{

…

}

Щоб переконатися, що все працює, скомпілюйте бібліотеку динамічного компонування. Щоб виконати компіляцію, послідовно виберіть Складання>Зібрати рішення. Бібліотека DLL та пов'язані вихідні дані компілятора розміщуються у папці з ім'ям Debug безпосередньо під папкою рішення. При створенні складання випуску вихідні дані розміщуються в папці з ім'ям Release. Результат має виглядати так.

1>------ Build started: Project: MathLibrary, Configuration: Debug Win32 ------

1>pch.cpp

1>dllmain.cpp

1>MathLibrary.cpp

1>Generating Code...

1> Creating library C:\Users\username\Source\Repos\MathLibrary\Debug\MathLibrary.lib and object C:\Users\username\Source\Repos\MathLibrary\Debug\MathLibrary.exp

1>MathLibrary.vcxproj -> C:\Users\username\Source\Repos\MathLibrary\Debug\MathLibrary.dll

========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Створення клієнтської програми у Visual Studio

У рядку меню виберіть FileNew>>Project щоб відкрити діалогове вікно "Створення проекту".

У верхній частині діалогового вікна задайте для параметра Мова значення C++, для параметра Платформа значення Windows, а для Типу проекту — Консоль.

У відфільтрованому списку типів проектів клацніть Консольний додаток, а потім натисніть кнопку Далі.

На сторінці Налаштувати новий проект введіть MathClient у полі Ім'я проекту. Прийміть за промовчанням Розташування та Ім'я рішення. Для параметра

Рішення задайте Створити нове рішення. Зніміть прапорець Розмістити рішення та проект в одному каталозі, якщо його встановлено.

Натисніть кнопку Створити, щоб створити проект клієнта.

Створюється мінімальний проект консольної програми. Ім'я головного вихідного файлу співпадатиме з раніше введеним ім'ям проекту. У цьому прикладі використовується ім'я назва.cpp. Ви можете створити проект, але він ще не використовує бібліотеку DLL.

Потім, щоб викликати функції у вихідному коді, ваш проект має містити файл назва.h. Цей файл заголовка можна скопіювати до проекту клієнтської програми, а потім додати його до проекту як існуючого елемента. Цей спосіб підходить для сторонніх бібліотек. Однак, якщо ви працюєте з кодом для бібліотеки DLL і клієнта одночасно, файли заголовків можуть виявитися несинхронізованими. Щоб уникнути цієї проблеми, задайте шлях Додаткові каталоги файлів, що включаються в проекті, щоб додати шлях до вихідного заголовка.

Додавання заголовка бібліотеки DLL у шлях включення

Клацніть правою кнопкою миші вузол MathClient у браузері рішень, щоб відкрити діалогове вікно Сторінки властивостей.

У розкривному списку Конфігурація виберіть Усі конфігурації, якщо він ще не вибраний.

В області зліва виберіть пункт Властивості конфігурації >C/C++ >Загальні.

На панелі властивостей клацніть елемент керування, що розкривається, поруч із полем введення параметра Додаткові каталоги файлів, а потім клацніть Правка.

Двічі клацніть у верхній панелі діалогового вікна Додаткові каталоги файлів, що включаються, щоб увімкнути елемент керування "Поле введення". Або клацніть піктограму папки, щоб створити новий запис.

В елементі керування "Поле введення" вкажіть шлях до розташування файлу заголовка назва.h. Щоб перейти до потрібної папки, можна вибрати елемент керування з трьома крапками ( ... ).

Також можна ввести відносний шлях від вихідних файлів клієнта до папки, що містить файли заголовків бібліотеки DLL. Якщо ви дотримувалися інструкцій з розміщення клієнтського проекту в окремому рішенні, відмінному від бібліотеки DLL, відносний шлях має виглядати так:

..\..\назва\назва

Якщо бібліотеки DLL та клієнтські проекти знаходяться в одному рішенні, відносний шлях може виглядати так:

..\назва

Якщо бібліотеки DLL та проекти клієнта знаходяться в інших папках, змініть відносний шлях для відповідності. Або скористайтеся елементом керування багато крапка для пошуку папки.

Після введення шляху до файлу заголовка в діалоговому вікні Додаткові каталоги файлів натисніть кнопку ОК. У діалоговому вікні "Сторінки властивостей" натисніть кнопку OK, щоб зберегти зміни.

Тепер можна додати файл назва.h і використовувати функції, які він оголошує у вашій клієнтській програмі. Замініть вміст файлу MathClient.cpp за допомогою наступного коду:

#include <iostream>

#include "назва.h"

int main()

{

…

}

Цей код можна скомпілювати, але не скомпілювати. Якщо ви створюєте клієнтську програму, у списку помилок з'явиться кілька помилок LNK2019. Це пов'язано з тим, що у проекті відсутні деякі відомості: Ви не вказали, що проект поки що залежить від бібліотеки назва.lib. І ви не вказали компонувальнику, як знайти файл назва.lib.

Щоб вирішити цю проблему, можна скопіювати файл бібліотеки безпосередньо в проект клієнтської програми. Компонувальник зможе знайти та використовувати його автоматично. Однак якщо бібліотека та клієнтська програма знаходяться на стадії розробки, це може призвести до змін в одній копії, які не відображатимуться в іншій. Щоб уникнути цієї проблеми, можна встановити властивість Додаткові залежності, щоб повідомити систему збирання про те, що проект залежить від MathLibrary.lib. Також можна задати

шлях Додаткові каталоги бібліотек у проекті, включивши шлях до вихідної бібліотеки при компонуванні.

Додавання бібліотеки імпорту DLL до проекту

Клацніть правою кнопкою миші вузол MathClient у браузері рішень і виберіть Властивості, щоб відкрити діалогове вікно Сторінки властивостей.

У розкривному списку Конфігурація виберіть Усі конфігурації, якщо він ще не вибраний. Це гарантує, що будь-які зміни властивостей застосовуються до складання налагодження та випуску.

У області зліва виберіть пункт Властивості конфігурації > Компонувальник > Введення. На панелі властивостей клацніть елемент керування, що розкривається, поруч із полем введення параметра Додаткові залежності, а потім клацніть Правка.

У діалоговому вікні Додаткові залежності додайте MathLibrary.lib до списку у верхньому елементі керування "Поле введення".

Натисніть кнопку OK, щоб повернутися до діалогового вікна "Сторінки властивостей".

У області зліва виберіть пункт Властивості конфігурації > Компонувальник > Загальні. На панелі властивостей клацніть елемент керування, що розкривається, поруч із полем введення параметра Додаткові каталоги бібліотек, а потім клацніть Правка.

Двічі клацніть у верхній панелі діалогового вікна Додаткові каталоги бібліотек, щоб увімкнути елемент керування "Поле введення". В елементі керування "Поле введення" вкажіть шлях до розташування файлу назва.lib. За замовчуванням він знаходиться у папці з іменем Debug безпосередньо у папці DLL. Під час створення збірки випуску файл міститься в папці з ім'ям Release. Можна використовувати макрос $(IntDir), щоб компонувальник міг знайти бібліотеку DLL незалежно від типу створюваної збірки. Якщо ви дотримувалися інструкцій з розміщення клієнтського проекту в окремому рішенні, відмінному від проекту DLL, відносний шлях має виглядати так:

..\..\назва\$(IntDir)

Якщо бібліотеки DLL та проекти клієнта знаходяться в інших місцях, змініть відносний шлях для відповідності.

Як тільки ви ввели шлях до файлу бібліотеки, у діалоговому вікні Додаткові каталоги бібліотек натисніть кнопку ОК, щоб повернутися до діалогового вікна Сторінки властивостей. Натисніть кнопку ОК, щоб зберегти зміни властивостей.

Ваша клієнтська програма тепер можна компілювати і компонувати, але в ній, як і раніше, немає всього необхідного для запуску. Коли операційна система завантажує вашу програму, вона шукає бібліотеку DLL назва. Якщо вона не може знайти бібліотеку DLL у певних системних каталогах, у дорозі середовища чи локальному каталозі програми, завантаження завершується збоєм. Залежно від операційної системи ви побачите повідомлення про помилку такого вигляду:

Щоб уникнути цієї проблеми, можна скопіювати бібліотеку DLL у каталог, в якому знаходиться файл клієнта, що виконується, в процесі складання. Можна додати подію після складання до вашого проекту, щоб додати команду, яка копіює бібліотеку DLL у вихідний каталог вашого складання. Вказана команда копіює бібліотеку DLL тільки в тому випадку, якщо вона відсутня або була змінена. Він використовує макроси для копіювання розташування налагодження або випуску на основі конфігурації збірки.

Копіювання бібліотеки DLL у подію після збирання

Клацніть правою кнопкою миші вузол назва у браузері рішень і виберіть Властивості, щоб відкрити діалогове вікно Сторінки властивостей.

У розкривному списку Конфігурація виберіть Усі конфігурації, якщо він ще не вибраний.

У області зліва виберіть пункт Властивості конфігурації >Події збирання >Подія після збирання.

В області властивостей клацніть елемент керування "Поле введення" у полі Командний рядок. Якщо ви дотримуєтеся інструкцій з розміщення клієнтського проекту в окремому рішенні, відмінному від проекту DLL, введіть наступну команду:

xcopy /y /d "..\..\MathLibrary\$(IntDir)MathLibrary.dll" "$(OutDir)"

Якщо бібліотеки DLL та проекти клієнтів знаходяться в інших каталогах, змініть відносний шлях до бібліотеки DLL для відповідності.

Натисніть кнопку OK, щоб зберегти зміни у властивостях проекту.

Тепер у вашому клієнтському додатку є все, що потрібно для збирання та запуску. Зберіть програму, клацнувши команду Складання>Зібрати рішення в меню. Вікно Виведення у Visual Studio повинно мати приблизно такий вигляд залежно від версії Visual Studio, що використовується:

1>------ Build started: Project: MathClient, Configuration: Debug Win32 ------

1>MathClient.cpp

1>MathClient.vcxproj -> C:\Users\username\Source\Repos\MathClient\Debug\MathClient.exe

1>1 File(s) copied

========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

### 2.2 Створення бібліотеки func для першої лабораторної

Згідно інструкції було створено бібліотеку з відповідним найменуванням та протестовано. Повний код цієї бібліотеки наведено у додатку А та Б.

### 2.3 Створення бібліотеки oper для другої лабораторної роботи

Згідно інструкції було створено бібліотеку з відповідним найменуванням та протестовано. Повний код цієї бібліотеки наведено у додатку В та Д.

### 2.4 Створення бібліотеки класів для третьої лабораторної

Згідно інструкції було чотири бібліотеки з відповідним найменуванням та протестовано. Повний код цієї бібліотеки наведено у додатку А1, Б1, В1, Д1.

## Висновок

Під час виконання лабораторної роботи було засвоєно на практиці створення власних динамічних бібліотек та їх використання.

Хоча і бібліотеки DLL, і додатки є виконуваними модулями, вони відрізняються деякими особливостями. Найбільш очевидна відмінність полягає в тому, що бібліотеку DLL не можна запустити. З погляду системи, між додатками і бібліотеками DLL є дві істотні відмінності.

1) У системі може одночасно виконуватися кілька екземплярів програми. Примірник бібліотеки DLL може бути тільки один.

2) Додаток може завантажуватися як процес. Йому можуть належати такі компоненти, як стек, потоки виконання, глобальна пам'ять, дескриптори файлів і черга повідомлень. У бібліотеки DLL таких компонентів бути не може.

Динамічне компонування коду і ресурсів має деякі переваги над статичним.

Динамічне компонування економить пам'ять і скорочує підкачування. Багато процесів можуть використовувати бібліотеку DLL спільно, одночасно звертаючись до однієї доступної тільки для читання копії її частин у пам'яті. На відміну від цього, кожен додаток, створений за допомогою бібліотеки статичного компонування, має повну копію коду бібліотеки, яку система Windows повинна завантажити в пам'ять.

Динамічне компонування економить місце на диску і пропускну здатність. Кілька додатків можуть спільно використовувати одну копію бібліотеки DLL на диску. На відміну від цього, кожен додаток, створений за допомогою бібліотеки статичного компонування, має код бібліотеки, пов'язаний з виконуваним образом. Це збільшує займане на диску місце і використовувану для передачі даних пропускну здатність.

Обслуговування, застосування виправлень для системи безпеки та оновлення можуть бути простішими. Якщо додатки використовують загальні функції в бібліотеці DLL, можна реалізувати виправлення помилок і розгорнути оновлення для неї. Під час оновлення бібліотек DLL додатки, що їх використовують, не потрібно перекомпілювати або повторно компонувати. Вони можуть використовувати нові бібліотеки DLL відразу після їх розгортання. На відміну від цього, при внесенні виправлень до коду статично скомпонованого об'єкта необхідно повторно скомпонувати і розгорнути кожну програму, що використовує його.

За допомогою бібліотек DLL можна надавати післяпродажну підтримку. Наприклад, бібліотеку DLL драйвера дисплея можна змінити так, щоб вона підтримувала дисплей, який не був доступний на момент надання програми.

За допомогою явного компонування можна виявляти і завантажувати бібліотеки DLL під час виконання. Наприклад, це можуть бути розширення додатка, які додають нові функціональні можливості без повторного складання та розгортання додатка.

Динамічне компонування спрощує підтримку додатків, написаних різними мовами програмування. Програми, написані різними мовами програмування, можуть викликати одну й ту саму функцію в бібліотеці DLL за умови дотримання угоди про її виклик. Програми та функція в бібліотеці DLL повинні відповідати таким вимогам до сумісності: очікуваний функцією порядок передавання аргументів у стек; виконання очищення стека функцією або додатком; передавання аргументів у регістрах.

Динамічне компонування забезпечує механізм для розширення класів бібліотеки Microsoft Foundation Classes (MFC). На основі наявних класів MFC можна створювати похідні класи та поміщати їх до бібліотеки розширення DLL, використовуваної додатками MFC.

Динамічне компонування спрощує створення міжнародних версій програми. Бібліотеки DLL - це зручний спосіб надання ресурсів для конкретних мовних стандартів, завдяки чому значно спрощується створення міжнародних версій програми. Замість надання безлічі локалізованих версій застосунку можна помістити рядки та зображення для кожної мови в окрему бібліотеку DLL ресурсів. Потім додаток може завантажувати ресурси для потрібного мовного стандарту під час виконання.

Можливим недоліком використання бібліотек DLL є те, що додатки не є автономними. Вони вимагають наявності окремого модуля DLL, який має перевірятися в процесі встановлення.

## Додаток А

func.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#ifdef FUNC\_EXPORTS

#define FUNC\_API \_\_declspec(dllexport)

#else

#define FUNC\_API \_\_declspec(dllimport)

#endif

#define MAXSIZE 300

#include <iostream>

#include <string>

#include <iomanip>

#include <ctime>

using namespace std;

class Abiturient {

string FullName;

string address;

short int grade;

public:

Abiturient\* next, \* prev;

//Abiturient();

// setter

void setterAll(const string& FullName, const string& address, const short int& grade);

//getters

string getterFullname() const; string getterAddress() const; short int getterGrade() const;

void show();

//~Abiturient() {};

friend void Sort(Abiturient\*\* head, short int set);

};

class Main

{

private:

int size;

public:

Main() noexcept : size(0) {};

int GetSize() noexcept { return size; };

void CreateList(short int count, Abiturient\*\* head, Abiturient\* tail);

void PrintList(Abiturient\* head);

void Insert(Abiturient\*\* head, Abiturient\*\* tail, int pos);

void DeleteInList(Abiturient\*\* Dhead, Abiturient\*\* Dtail, int pos);

bool DcheckNULL(Abiturient\* head);

void FreeList(Abiturient\*\* head);

void table\_header();

// main tasks

void unsatisfactory(Abiturient\*\* head);

void byGrade(Abiturient\*\* head, short int setted);

void hightesGradeandHalf(Abiturient\*\* head);

// working method

void main();

//~Main() {};

};

extern "C++" FUNC\_API void complite();

## Додаток Б

func.срр

#include "pch.h"

#include "func.h"

string dataNames[32] = { "Ethan", "Josh", "Sam", "Tom", "Bruse", "Michael", "Diego", "Adrian", "Brian", "Carl", "Caleb", "Charles", "Daniel", "Derek", "Elliot", "Kuel",

"Sophia", "Amelia", "Ella", "Chloe", "Penelope", "Emma", "Ava", "Isabella", "Elizabath", "Ginna", "Layla", "Zoey", "Olivia", "Sarah", "Rachel", "Emily" };

string dataSecondNames[50] = { "Smith", "Johnson", "Williams", "Jones", "Brown","Davis","Miller", "Wilson","Moore","Taylor", "Anderson","Thomas","Jackson", "White","Harris","Martin", "Thompson","Garcia","Martinez", "Robinson","Clark","Rodriguez", "Lewis","Lee","Walker", "Hall","Allen","Young", "Hernandez","King","Wright", "Lopez","Hill","Scott", "Green","Adams","Baker",

"Gonzalez", "Nelson","Carter","Mitchell", "Perez","Roberts","Turner", "Phillips","Campbell","Parker", "Evans","Edwards","Collins" };

// main class

void Main::main()

{

short int count, button, checker;

Abiturient\* head = NULL, \* tail = NULL;

while (true) {

cout << "Work with data - 1 \nClear data - 4 \nExit - 0 \n-->"; cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

}

switch (button)

{

case 1:

while (button != 0) {

system("cls");

if (head == NULL) {

cout << "How long you list could be? \n---> "; cin >> count;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

}

CreateList(count, &head, tail);

system("cls");

}

table\_header();

PrintList(head);

cout << "Find students with unsatisfactory grades - 1\n" <<

"Find students by grades - 2\n" <<

"Find students whith highest grades and almost satisfactory - 3\n" <<

"Delete one data - 4\n" <<

"Add new data - 5\n" <<

"Sort - 6\n" <<

"Back - 0\n" <<

"-->";

cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

system("pause");

}

switch (button)

{

case 1:

system("cls");

unsatisfactory(&head);

system("pause");

break;

case 2:

system("cls");

cout << "Input grade limit \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

byGrade(&head, checker);

system("pause");

break;

case 3:

system("cls");

hightesGradeandHalf(&head);

system("pause");

break;

case 4:

system("cls");

cout << "Input number data \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

DeleteInList(&head, &tail, checker);

break;

case 5:

system("cls");

cout << "Input number data \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

Insert(&head, &tail, checker);

break;

case 6:

system("cls");

cout << "Sort by name - 1 \nSort by grades - 2 \nBack - 0 \n-->"; cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

switch (button)

{

case 1:

Sort(&head, button);

table\_header();

PrintList(head);

system("pause");

break;

case 2:

Sort(&head, button);

table\_header();

PrintList(head);

system("pause");

break;

case 0:

system("cls");

break;

default:

break;

}

break;

case 0:

system("cls");

break;

default:

break;

}

}

break;

/\*case 2:

system("cls");

SaveInFileList(head);

break;

case 3:

system("cls");

LoadFromFileList(&head, &tail);

break;\*/

case 4:

system("cls");

if (DcheckNULL(head) != 0)

FreeList(&head);

break;

case 0:

system("cls");

FreeList(&head);

exit(0);

break;

default:

break;

}

}

}

//Abiturient::Abiturient()

//{

// this->FullName = "User invalid";

// this->address = "None";

// this->grade = -1;

//

//}

//set methods

void Abiturient::setterAll(const string& FullName, const string& address, const short int& grade)

{

this->FullName = FullName;

this->address = address;

this->grade = grade;

}

// get methods

string Abiturient::getterFullname() const { return this->FullName; }

string Abiturient::getterAddress() const { return this->address; }

short int Abiturient::getterGrade() const { return this->grade; }

// show method

void Abiturient::show()

{

cout << "|" << setw(22) << this->FullName << setw(4) << "|" << setw(32) << this->address << setw(4) << "|" << setw(4) << this->grade << setw(4) << "|";

cout << "\n ---------------------------------------------------------------------\n";

}

///////////////////////////////////////////////////////////////////////

///////////////////// Main tasks ///////////////////////////////////

/////////////////////////////////////////////////////////////////////

void Main::unsatisfactory(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* unsatisfactoryGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() < 60) {

unsatisfactoryGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with unsatisfactory grades: " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

unsatisfactoryGrade[i].show();

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] unsatisfactoryGrade; unsatisfactoryGrade = NULL;

}

void Main::byGrade(Abiturient\*\* head, short int setted)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > setted) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with grades higher " << setted << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show();

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

void Main::hightesGradeandHalf(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > 89 or (current->getterGrade() > 34 and current->getterGrade() < 60)) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\tStudents " << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show();

}

if (count != 1) {

short int choose = -1;

Abiturient\* chosen = new Abiturient[count];

short int flag = 0;

short index = 0;

current = \*head;

while (choose != 0 and flag != count) {

cout << "Choose the studenet by number (0-stop) --> "; cin >> choose;

if (choose > count) {

cout << "\tFalse input, try again!\n";

continue;

}

if (choose != 0) { flag++; };

for (size\_t i = 0; i < choose; i++) {

if (i == choose - 1) {

chosen[index] = byGrade[choose - 1];

index++;

}

current = current->next;

}

}

if (flag != 0) {

table\_header();

for (size\_t i = 0; i < flag; i++) {

chosen[i].show();

}

}

else

cout << "\tStudents haven`t been chosen!\n";

delete[] chosen; chosen = NULL;

}

else {

cout << "You shouldn`t to choose the one student!\n";

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

void Main::table\_header()

{

cout << "\n ---------------------------------------------------------------------\n";

cout << "|" << setw(17) << "Full Name" << setw(9) << "|" << setw(23) << "Email" << setw(13) << "|" << setw(6) << "Grade" << setw(2) << "|";

cout << "\n ---------------------------------------------------------------------\n";

}

///////////////////////////////////////////////////////////////////////

///////////////////// Doubly Node ///////////////////////////////////

/////////////////////////////////////////////////////////////////////

void Main::CreateList(short int count, Abiturient\*\* head, Abiturient\* tail)

{

string FullName, buffer, address; short int grade;

if (count > 0) {

(\*head) = new Abiturient;

FullName = dataNames[rand()%32];

address += FullName;

FullName += " ";

buffer = dataSecondNames[rand() % 32];

FullName += buffer;

address += buffer += "@gmail.com";

grade = rand() % 101;

(\*head)->setterAll(FullName, address, grade);

(\*head)->prev = tail;

(\*head)->next = NULL;

CreateList(count - 1, &((\*head)->next), (\*head));

}

else (\*head) = NULL;

size = count;

}

void Main::PrintList(Abiturient\* head) {

if (head != NULL) {

head->show();

PrintList(head->next);

}

else cout << endl << endl;

}

void Main::Insert(Abiturient\*\* head, Abiturient\*\* tail, int pos) {

Abiturient\* temp = new Abiturient;

if (pos == 0) {

temp->next = \*head;

(\*head)->prev = temp;

\*head = temp;

temp->prev = NULL;

}

else {

Abiturient\* current = \*head;

for (size\_t i = 0; i < pos - 1; i++)

current = current->next;

if (current->next == NULL) {

current->next = temp;

temp->prev = \*tail;

temp->next = NULL;

\*tail = temp;

}

else {

temp->next = current->next;

current->next = temp;

temp->next->prev = temp;

temp->prev = current;

}

}

string FullName, buffer, address; short int grade;

FullName = dataNames[rand() % 32];

address += FullName;

FullName += " ";

buffer = dataSecondNames[rand() % 32];

FullName += buffer;

address += buffer += "@gmail.com";

grade = rand() % 101;

temp->setterAll(FullName, address, grade);

}

void Main::DeleteInList(Abiturient\*\* head, Abiturient\*\* tail, int pos) {

Abiturient\* current = \*head;

if (pos == 0) {

\*head = (\*head)->next;

(\*head)->prev = NULL;

delete current;

}

else {

for (size\_t i = 0; i < pos - 1; i++)

current = current->next;

Abiturient\* temp = current->next;

if (temp->next == NULL) {

current->next = NULL;

\*tail = current;

}

else {

current->next = temp->next;

temp->next->prev = current;

}

delete temp;

}

}

bool Main::DcheckNULL(Abiturient\* head) { // cheking on empty

if (head == NULL) {

cout << "List is empty." << endl;

return 0;

}

return 1;

}

void Main::FreeList(Abiturient\*\* head) { // clean list

Abiturient\* current = \*head;

while (current) {

\*head = (\*head)->next;

free(current);

current = \*head;

}

}

void Sort(Abiturient\*\* head, short int set) {

Abiturient\* left = \*head;

Abiturient\* right = (\*head)->next;

Abiturient\* temp = new Abiturient;

// preset switch

switch (set) {

case 1: // sort by name

while (left->next) {

while (right) {

if ((left->FullName.compare(right->FullName) > 0)) {

temp->FullName = left->FullName;

left->FullName = right->FullName;

right->FullName = temp->FullName;

}

right = right->next;

}

left = left->next;

right = left->next;

}

break;

case 2: // sort by grades

while (left->next) {

while (right) {

if (right->grade > left->grade) {

temp->grade = right->grade;

right->grade = right->grade;

left->grade = temp->grade;

}

right = right->next;

}

left = left->next;

right = left->next;

}

break;

default:

//system("cls");

break;

}

}

void complite()

{

srand(time(NULL)); Main main; main.main();

}

## Додаток В

oper.h

#pragma once

#include "pch.h"

#ifdef OPER\_EXPORTS

#define OPER\_API \_\_declspec(dllexport)

#else

#define OPER\_API \_\_declspec(dllimport)

#endif

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

extern "C++" OPER\_API void complite();

class Fraction

{

float denominator;

float numerator;

public:

Fraction(const float denominator = rand() % 50 + 1, const float numerator = rand() % 50 + 1) {

if (numerator != 0)

(this->denominator = denominator) / (this->numerator = numerator);

else

cout << "\tYou whant to brake the math rules? -\_-\n";

};

// stream operators

friend ostream& operator << (ostream& out, const Fraction& ob) { out << ob.denominator << " / " << ob.numerator << " = " << setprecision(10) << ob.denominator / ob.numerator; return out; };

friend istream& operator >> (istream& in, Fraction& ob) { cout << "Denominator --> "; in >> ob.denominator; cout << "Numerator --> "; in >> ob.numerator; return in; };

// ar operation

friend Fraction& operator + (const Fraction& ob1, const Fraction& ob2) noexcept;

friend Fraction& operator - (const Fraction& ob1, const Fraction& ob2) noexcept;

friend Fraction& operator / (const Fraction& ob1, const Fraction& ob2) noexcept;

friend Fraction& operator \* (const Fraction& ob1, const Fraction& ob2) noexcept;

Fraction& operator = (const Fraction& ob) noexcept;

//compering operations

bool operator == (const Fraction& ob) noexcept { return (denominator / numerator == ob.denominator / ob.numerator); }

bool operator != (const Fraction& ob) noexcept { return !(denominator / numerator == ob.denominator / ob.numerator); }

bool operator > (const Fraction& ob) noexcept { return (denominator / numerator > ob.denominator / ob.numerator); };

bool operator < (const Fraction& ob) noexcept { return (denominator / numerator < ob.denominator / ob.numerator); };

bool operator >= (const Fraction& ob) noexcept { return (denominator / numerator >= ob.denominator / ob.numerator); };

bool operator <= (const Fraction& ob) noexcept { return (denominator / numerator <= ob.denominator / ob.numerator); };

// tt function

void task(Fraction arr[], int& size) const;

float operator[](const int i) noexcept { return denominator / numerator; };

void print(Fraction arr[], int size) const;

};

class Main

{

public:

void main();

};

## Додаток Д

oper.cpp

#include "pch.h"

#include "Oper.h"

void complite() { Main main; main.main(); }

void Main::main()

{

srand(time(NULL));

int size = 10;

Fraction arr[10], newarr[10], result;

int button, fchoose, schoose;

cout << "First array: \n";

arr->print(arr, size);

cout << endl;

cout << "Second array: \n";

newarr->print(newarr, size);

cout << endl;

while (true) {

cout << "1 - Print and clear\n" <<

"2 - Operator (+)\n" <<

"3 - Operator (-)\n" <<

"4 - Operator (/)\n" <<

"5 - Operator (\*)\n" <<

"6 - Operator (=)\n" <<

"7 - Comparison Operators\n" <<

"8 - Add previous to every second element of array\n" <<

"0 - Exit\n" <<

"--> "; cin >> button;

switch (button)

{

case 1:

system("cls");

cout << "First array: \n";

arr->print(arr, size);

cout << endl;

cout << "Second array: \n";

newarr->print(newarr, size);

cout << endl;

break;

case 2:

cout << "\nChoose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) {

cout << "Invalid input\n\n"; break;

}

result = arr[fchoose - 1] + newarr[schoose - 1];

cout << "\nResult: \n" << arr[fchoose - 1] << "\n + \n" << newarr[schoose - 1] << endl << " ======= " << endl << result << endl << endl;

break;

case 3:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) {

cout << "Invalid input\n\n"; break;

}

result = arr[fchoose - 1] - newarr[schoose - 1];

cout << "\nResult: \n" << arr[fchoose - 1] << "\n - \n" << newarr[schoose - 1] << endl << " ======= " << endl << result << endl << endl;

break;

case 4:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) {

cout << "Invalid input\n\n"; break;

}

result = arr[fchoose - 1] / newarr[schoose - 1];

cout << "\nResult: \n" << arr[fchoose - 1] << "\n ------------------- \n" << newarr[schoose - 1] << endl << " ======= " << endl << result << endl << endl;

break;

case 5:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) {

cout << "Invalid input\n\n"; break;

}

result = arr[fchoose - 1] \* newarr[schoose - 1];

cout << "\nResult: \n" << arr[fchoose - 1] << "\n \* \n" << newarr[schoose - 1] << endl << " ======= " << endl << result << endl << endl;

break;

case 6:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) {

cout << "Invalid input\n\n"; break;

}

arr[fchoose - 1] = newarr[schoose - 1];

break;

case 7:

system("cls");

cout << "First array: \n";

arr->print(arr, size);

cout << endl;

cout << "Second array: \n";

newarr->print(newarr, size);

cout << endl;

while (button != 0) {

cout << "Compere two items of arrays: \n"

<< "1 - Print and clear\n"

<< "2 - by == \n"

<< "3 - by != \n"

<< "4 - by > \n"

<< "5 - by < \n"

<< "6 - by >= \n"

<< "7 - by <= \n"

<< "0 - Back \n"

<< "--> "; cin >> button;

switch (button)

{

case 1:

system("cls");

cout << "First array: \n";

arr->print(arr, size);

cout << endl;

cout << "Second array: \n";

newarr->print(newarr, size);

cout << endl;

break;

case 2:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "Your chosen comparison \"=\" \n";

if (arr[fchoose - 1] == newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 3:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "\nYour chosen comparison \"!=\" \n";

if (arr[fchoose - 1] != newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 4:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "\nYour chosen comparison \">\" \n";

if (arr[fchoose - 1] > newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 5:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "\nYour chosen comparison \"<\" \n";

if (arr[fchoose - 1] < newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 6:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "\nYour chosen comparison \">=\" \n";

if (arr[fchoose - 1] >= newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 7:

cout << "Choose one lemenet from first array and than from second \n\nFirst array`s element -->"; cin >> fchoose;

cout << "\nSecond array`s element-->"; cin >> schoose;

if (fchoose - 1 > size or fchoose - 1 < 0 or schoose - 1 > size or schoose - 1 < 0) { cout << "\nInvalid input!\n\n"; break; }

cout << "\nYour chosen comparison \"<=\" \n";

if (arr[fchoose - 1] <= newarr[schoose - 1])

cout << "\n\t True \t\n\n";

else

cout << "\n\t False \t\n\n";

break;

case 0:

system("cls");

break;

default:

break;

}

}

break;

case 8:

cout << "\nSelect the array to which this operation will be applied \n1 - First array \n2 - Second array \n--> "; cin >> button;

if (button != 1 and button != 2) { cout << "\nInvalid input!\n\n"; break; }

if (button == 1)

arr->task(arr, size);

else if (button == 2)

newarr->task(newarr, size);

cout << "First array: \n";

arr->print(arr, size);

cout << endl;

cout << "Second array: \n";

newarr->print(newarr, size);

cout << endl;

break;

case 0:

system("cls");

exit(0);

break;

default:

break;

}

}

}

void Fraction::print(Fraction arr[], int size) const

{

for (size\_t i = 0; i < size; i++)

cout << i + 1 << ") " << arr[i] << endl;

}

void Fraction::task(Fraction arr[], int& size) const

{

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

if ((i + 1) % 2 == 0 and i - 1 >= 0)

arr[i] = arr[i] + arr[i - 1];

}

Fraction& operator+(const Fraction& ob1, const Fraction& ob2) noexcept

{

Fraction result((ob1.denominator \* ob2.numerator) + (ob2.denominator \* ob1.numerator), ob1.numerator \* ob2.numerator);

return result;

}

Fraction& operator-(const Fraction& ob1, const Fraction& ob2) noexcept

{

Fraction result((ob1.denominator \* ob2.numerator) - (ob2.denominator \* ob1.numerator), ob1.numerator \* ob2.numerator);

return result;

}

Fraction& operator/(const Fraction& ob1, const Fraction& ob2) noexcept

{

Fraction result(ob2.numerator \* ob1.denominator, ob2.denominator \* ob1.numerator);

return result;

}

Fraction& operator\*(const Fraction& ob1, const Fraction& ob2) noexcept

{

Fraction result(ob1.denominator \* ob2.denominator, ob1.numerator \* ob2.numerator);

return result;

}

Fraction& Fraction::operator=(const Fraction& ob) noexcept

{

denominator = ob.denominator; numerator = ob.numerator;

return \*this;

}