Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №1

КЛАСИ ТА СПЕЦИФІКАТОРИ ДОСТУПУ. ІНКАПСУЛЯЦІЯ. КОНСТАНТИ

**Мета:** Отримати базові знання про класи. Дослідити механізм інкапсуляції.

1 ЗАВДАННЯ ДО РОБОТИ

**Основне завдання:** для предметної галузі розробити два класи:  
 ‒ клас, що відображає сутність «базового класу». При цьому, в даному класі повинно бути мінімум три числових поля.

‒ клас, що має в собі динамічний масив об’єктів базового класу та має в собі методи додавання, видалення елементу, отримання елементу по індексу (або ідентифікатору), вивід усіх елементів на екран.

**Індивідульне завдання.**

Прикладна галузь: роботи студентів. Базовий клас: аудиторні заняття.

2 РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

**2.1. Опис змінних**

Lessons\* lesson – покажчик на масив об’єктів типу Lessons (список занять)

int subAmount – змінна, що зберігає розмір масиву об’єктів \*lesson  
Works work – об’єкт типу Works

**Базовий клас Lessons  
Приватні поля:**

string audienceID – унікальний ідентифікатор аудиторії  
string lecturerName – фамілія лектора  
string lessonType – назва заняття  
int lessonStartTime – час початку заняття  
int studAmount – кількість студентів в аудиторії

**Публічні поля:**

Lessons \*next – показник на наступний елемент списку  
void set\_audienceID(const string index) – сеттер поля audienceID  
void set\_lecturerName(const string name) – сеттер поля lecturerName  
void set\_lessonType(const string type) – сеттер поля lessonType  
void set\_lessonStartTime(const int time) – сеттер поля lessonStartTime  
void set\_studAmount(const int number) – сеттер поля studAmount  
const string get\_audienceID() const – геттер поля audienceID  
const string get\_lecturerName() const – геттер поля lecturerName  
const string get\_lessonType() const – геттер поля lessonType  
const int get\_lessonStartTime() const – геттер поля lessonStartTime  
const int get\_studAmount() const – геттер поля studAmount  
void generateLesson() – метод, призначений для створення об’єктів

**Прикладна галузь – клас Works  
 Приватні поля:**

Lessons\* Head – показник на голову списку

**Публічні поля:**

void addWork(Lessons& lesson) – метод додавання елементів до списку  
void removeWork(int pos) – метод видалення елементів зі списку  
Lessons\* getWork(int index) – метод отримання елемента за індексом  
void showAll() – метод, що виводить усі елементи списку на екран

**2.2. Опис функцій**

**int main**  
 Призначення: головна функція.  
 Схема алгоритму функції подана на рис. 1.  
 Опис роботи:  
 1. Створення динамічного масиву об’єктів.  
 2. Виклик меню вибору дій.  
 3. В залежності від вибору, виклик відповідного методу, або завершення програми.  
 4. Очищення пам’яті виділеної під динамічний масив.

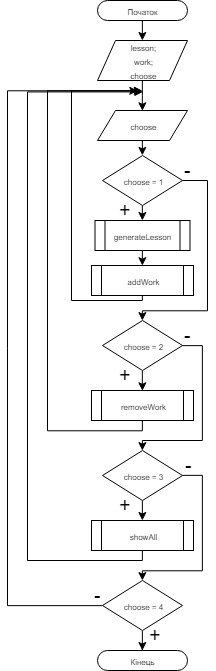
**void Lessons::generateLesson ( )** – метод класу LessonsПризначення: створення об’єкта.  
 Схема алгоритму подана на рис. 2.  
 Опис роботи: введення даних з клавіатури.

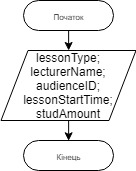
**void Works::addWork (Lessons** \*& **lesson)** – метод класу WorksПризначення: додавання об’єкта у список.  
 Схема алгоритму подана на рис. 3.  
 Опис роботи: якщо раніше у списку нічого не було, то ми створюємо список і заносимо у нього готовий об’єкт, а якщо він не пустий, то просто додаємо ще один готовий об’єкт, який передається.

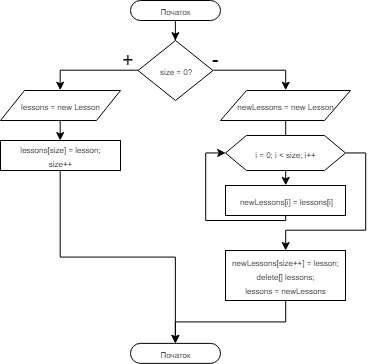
**void Works::removeWork (int pos)** – метод класу Works  
 Призначення: видалення об’єкта зі списку.  
 Схема алгоритму подана на рис. 4.  
 Опис роботи: створення списку, на 1 меньшого від попереднього, перенесення значень попереднього у новий, ігноруючи те значення, яке видаляємо.

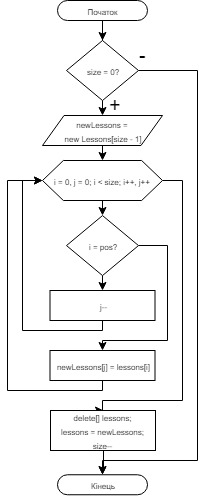
**void Works::showAll ( )** – метод класу Works  
 Призначення: виведення даних на екран.  
 Схема алгоритму подана на рис. 5.  
 Опис роботи: використовуючи цикл, функція виводить дані списку на екран.

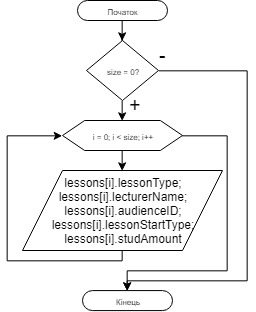
**2.3. Схеми алгоритмів функцій**

  
Рисунок 1 – Схема алгоритму функції main

  
Рисунок 2 – Схема алгоритму функції Lessons::generateLesson

  
Рисунок 3 – Схема алгоритму функції Works::addWork

  
Рисунок 4 – Схема алгоритму функції Works::removeWork

  
Рисунок 5 – Схема алгоритму функції Works::showAll

3 ТЕКСТ ПРОГРАМИ

**Текст файлу main.cpp**

#include <iostream>

#include "Lessons.h"

#include "Works.h"

#include "Func.h"

int main()

{

Lessons lesson;

Works work;

int choose;

do {

cout << endl;

MenuBar();

cin >> choose;

if (choose < 1 || choose > 4)

{

cout << "\nError (choose 1-4).\n";

continue;

}

switch (choose)

{

case 1:

lesson.generateLesson();

work.addWork(lesson);

break;

case 2:

cout << "Enter a psition of lesson you want to deleting: ";

int num;

while (true)

{

cin >> num;

if ((work.get\_size() == 0))

break;

if ((num - 1) >= work.get\_size() || (num - 1) < 0)

cout << "Error (pos is higher than qty of subs), try again: ";

else break;

}

work.removeWork(num-1);

cout << endl;

break;

case 3:

work.showAll();

break;

case 4:

work.endOfProgramm();

cout << "\nThank that you using my code!\n";

break;

}

} while (choose != 4);

system("PAUSE");

return 0;

}

**Текст файлу Lessons.h**

#include <string>

using namespace std;

#pragma once

class Lessons

{

public:

void set\_audienceID(const string index);

void set\_lecturerName(const string name);

void set\_lessonType(const string type);

void set\_studAmount(const int number);

void set\_lessonStartTime(const int time);

const string get\_audienceID() const;

const string get\_lecturerName() const;

const string get\_lessonType() const;

const int get\_studAmount() const;

const int get\_lessonStartTime() const;

void generateLesson();

private:

string audienceID;

string lecturerName;

string lessonType;

int studAmount;

int lessonStartTime;

};

**Текст файлу Lessons.cpp**

#include <iostream>

#include "Lessons.h"

void Lessons::set\_audienceID(const string index)

{

this->audienceID = index;

}

void Lessons::set\_lecturerName(const string name)

{

this->lecturerName = name;

}

void Lessons::set\_lessonType(const string type)

{

this->lessonType = type;

}

void Lessons::set\_studAmount(const int number)

{

this->studAmount = number;

}

void Lessons::set\_lessonStartTime(const int time)

{

this->lessonStartTime = time;

}

const string Lessons::get\_audienceID() const

{

return audienceID;

}

const string Lessons::get\_lecturerName() const {

return lecturerName;

}

const string Lessons::get\_lessonType() const {

return lessonType;

}

const int Lessons::get\_studAmount() const {

return studAmount;

}

const int Lessons::get\_lessonStartTime() const {

return lessonStartTime;

}

void Lessons::generateLesson()

{

cout << "\nEnter the subject name: ";

cin >> lessonType;

cout << "Enter lecturer name: ";

cin >> lecturerName;

cout << "Enter an audience ID: ";

cin >> audienceID;

cout << "Enter lesson start time: ";

cin >> lessonStartTime;

cout << "Enter a number of students in audience: ";

cin >> studAmount;

}

**Текст файлу Works.h**

#pragma once

#include "Lessons.h"

class Works : public Lessons

{

public:

void addWork(Lessons& lesson);

void removeWork(int pos);

void showAll();

void endOfProgramm();

const int get\_size() const;

private:

Lessons\* lessons;

int size = 0;

};

**Текст файлу Works.cpp**

#include <iostream>

#include "Works.h"

void Works::addWork(Lessons& lesson)

{

if (size == 0) {

lessons = new Lessons[size + 1];

lessons[size] = lesson;

size++;

}

else {

Lessons \*newLessons = new Lessons[size + 1];

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

newLessons[i] = lessons[i];

}

newLessons[size] = lesson;

delete[]lessons;

lessons = newLessons;

size++;

}

}

void Works::removeWork(int pos)

{

if (size == 0)

cout << "\nError (list is empty)\n";

else {

Lessons \*newLessons = new Lessons[size - 1];

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

if (i == pos) {

j--;

continue;

}

newLessons[j] = lessons[i];

}

delete[](lessons);

lessons = newLessons;

size--;

}

}

void Works::showAll()

{

if (size == 0)

cout << "\nError (the list is empty)\n";

else {

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

{

printf("\nInfo about lesson #%d: \n", i+1);

cout << "Subject: " << lessons[i].get\_lessonType() << endl;

cout << "Lecturer: " << lessons[i].get\_lecturerName() << endl;

cout << "Audience ID: " << lessons[i].get\_audienceID() << endl;

cout << "Lesson start time: " << lessons[i].get\_lessonStartTime() << endl;

cout << "Number of students in audience: " << lessons[i].get\_studAmount() << endl;

}

}

}

void Works::endOfProgramm()

{

delete[]lessons;

lessons = nullptr;

}

const int Works::get\_size() const {

return size;

}

**Текст файлу Func.h**

#pragma once

void MenuBar();

**Текст файлу Func.cpp**

#include <iostream>

using namespace std;

void MenuBar()

{

cout << "What do you want to do?\n";

cout << "1. Add lesson.\n";

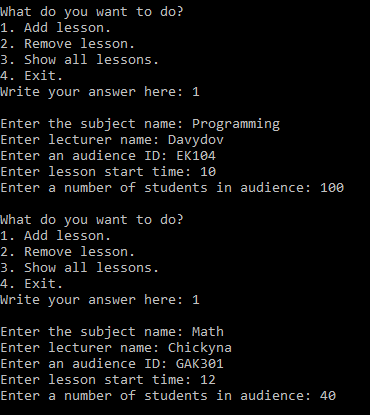
cout << "2. Remove lesson.\n";

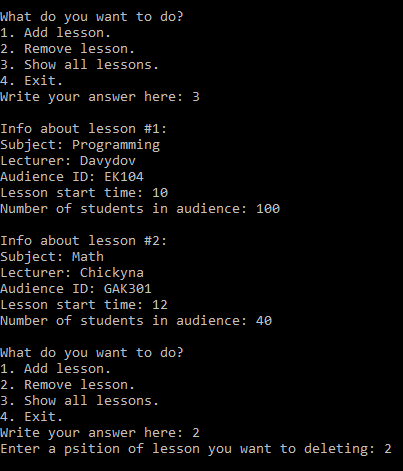
cout << "3. Show all lessons.\n";

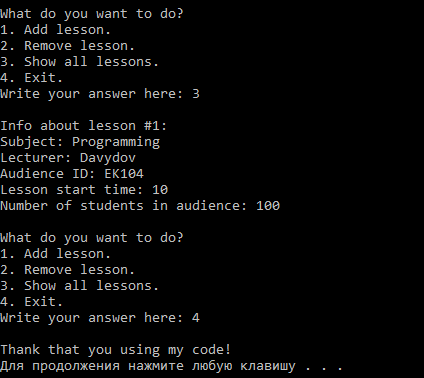
cout << "4. Exit.\n";

cout << "Write your answer here: ";

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ  
  
  
Рисунок 6 – Результат роботи програми (частина 1)

  
Рисунок 7 – Результат роботи програми (частина 2)

  
Рисунок 8 – Результат роботи програми (частина 3)

ВИСНОВКИ

При виконанні лабораторної роботи я навчився створювати класи, також дослідив механізм інкапсуляції, який полягає у приховуванні деяких характеристик класу від користувача.