Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №7

**КЛАСИ. ПОЛІМОРФІЗМ. АБСТРАКТНІ КЛАСИ**

**Мета:** Отримати знання про парадигму ООП – програмування. Навчитися застосовувати отримані знання на практиці.

1 ЗАВДАННЯ ДО РОБОТИ

**Загальне завдання.** Модернізувати попередню роботу шляхом:  
 - базовий клас зробити абстрактним;  
 - розроблені класи-списки поєднуються до одного таким чином, щоб він міг працювати як з базовим класом, так і з його спадкоємцями.

2 РОЗРОБКА АЛГОРИТМІВ ФУНКЦІЙ

У ході роботи змінено клас-список Works. Тепер він працює як з базовим класом, так і з класами спадкоємцями.

Клас-список **Works**  
 **private поля:**

class Node – інкапсульований клас-вузол класу-списку  
int size – кількість вузлів  
Node \*head – покажчик на перший вузол  
int sumOfHWs – загальна кількість домашніх робіт  
int sumOfWorks – загальна кількість самостійних робіт

Клас-вузол **Node  
 private поля:**

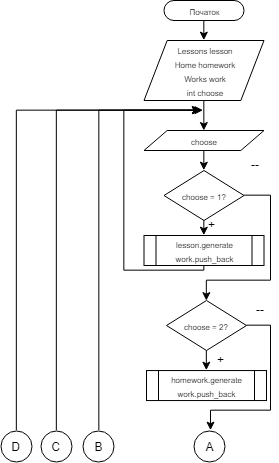
Lessons \*lesson – покажчик на перший елемент масиву об’єктів класу Lessons  
Node \*next – покажчик на наступний вузол

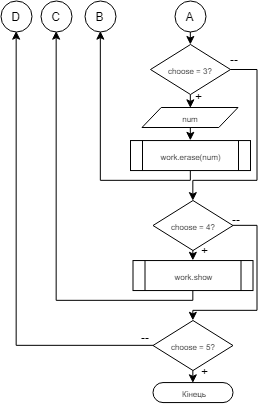
Методи:  
 **void push\_back ( Lessons \*lesson )**   
 Призначення: запис готового об’єкта базового класу або класу-спадкоємця у список.  
 Опис роботи: функція перевіряє кількість вже записаних об’єктів, якщо записів немає, то вона створює новий вузол і записує у нього об’єкт. Якщо записи є, то функція переходить у кінець списку, створює новий вузол та записує у нього новий об’єкт.  
 Схема алгоритму подана на рис. 2

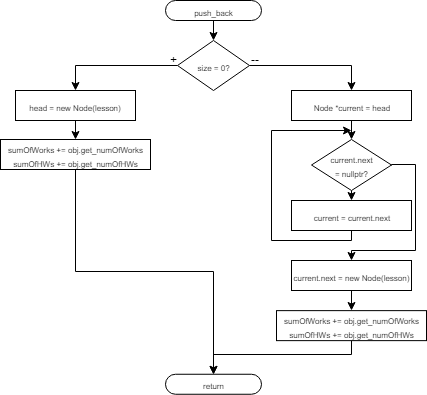
**void erase ( int pos )** Призначення: стирання зі списку об’єкта за заданою позицією.  
 Опис роботи: функція перевіряє кількість вузлів у списку. Якщо вузлів немає, виводить помилку: «Видалення об’єкта неможливе». А якщо є – функція, використовуючи цикл та алгебру покажчиків, перехотить до вузла, що є попереднім для того, який видаляється. Створюється покажчик на вузол, який потрібно видалити. Покажчик на наступний елемент попереднього вузла переноситься на наступний вузол. Вузол за індексом видаляється.  
 Схема алгоритму подана на рис. 3

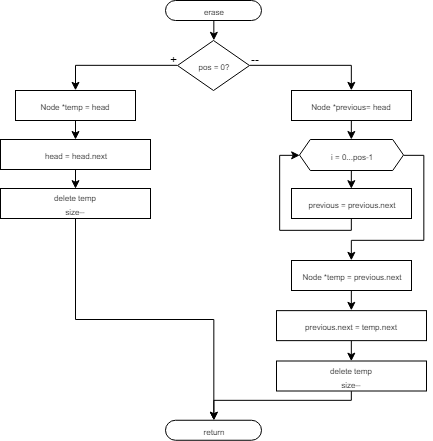
**void show ( )** Призначення: виведення інформації про усі об’єкти, що записані у список.  
 Опис роботи: функція перевіряє кількість вузлів у списку. Якщо вузлів немає, виводить помилку: «Виведення об’єктів неможливе». А якщо є – функція, цикл та алгебру покажчиків, виводить на екран інформацію про усі об’єкти списку.  
 Схема алгоритму подана на рис. 4

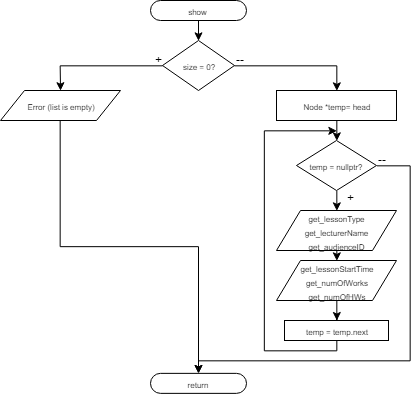
**Схеми алгоритмів функцій**



  
Рисунок 1 – Схема алгоритму функції main

  
Рисунок 2 – Схема алгоритму функції push\_back

  
Рисунок 3 – Схема алгоритму функції erase

  
Рисунок 4 – Схема алгоритму функції show

3 ТЕКСТ ПРОГРАМИ

**Текст файлу main.cpp**

#include <iostream>

#include "Lessons.h"

#include "Works.h"

#include "Home.h"

#include "Func.h"

int main()

{

Lessons lesson;

Home homework;

Works work;

int choose;

do {

cout << endl;

MenuBar();

cin >> choose;

if (choose < 1 || choose > 5)

{

cout << "\nError (choose 1-5).\n";

continue;

}

switch (choose)

{

case 1:

lesson.generate();

work.push\_back(&lesson);

break;

case 2:

homework.generate();

work.push\_back(&homework);

break;

case 3:

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

{

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

break;

}

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

int num;

while (true)

{

cin >> num;

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

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

else break;

}

work.erase(num-1);

cout << endl;

break;

case 4:

work.show();

break;

case 5:

cout << "\nStudent performs " << work.get\_works() << " tasks per semester.";

cout << "\nStudent performs " << work.get\_homeworks() << " hometasks per semester.";

cout << "\nStudent performs " << work.get\_works() + work.get\_homeworks() << " tasks and hometasks per semester.";

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

break;

}

} while (choose != 5);

system("PAUSE");

return 0;

}

**Текст файлу Works.h**

#pragma once

#include "Lessons.h"

#include "Home.h"

class Works

{

public:

Works();

~Works();

//Functions

void push\_back(Lessons\* lesson);

void pop\_front();

void erase(int pos);

void show();

void clear();

//Accesssors

inline const int& get\_works() const { return this->sumOfWorks; };

inline const int& get\_homeworks() const { return this->sumOfHWs; };

inline const int& get\_size() const { return this->size; };

private:

int size;

class Node

{

public:

Lessons \*lesson;

Node \*next;

Node(Lessons \*l = nullptr, Node \*n = nullptr);

};

Node \*head;

int sumOfHWs;

int sumOfWorks;

};

**Текст файлу Works.cpp**

#include <iostream>

#include "Works.h"

void Works::push\_back(Lessons\* lesson)

{

if (!size)

{

head = new Node(lesson);

sumOfWorks += lesson->get\_numOfWorks();

if (((Home \*)lesson)->get\_numOfHWs() > 0)

sumOfHWs += ((Home \*)lesson)->get\_numOfHWs();

}

else

{

Node \*current = this->head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = new Node(lesson);

sumOfWorks += lesson->get\_numOfWorks();

if (((Home \*)lesson)->get\_numOfHWs() > 0)

sumOfHWs += ((Home \*)lesson)->get\_numOfHWs();

}

this->size++;

}

void Works::erase(int pos)

{

if (!pos)

pop\_front();

else

{

Node \*previous = head;

for(int i = 0; i < pos-1; i++)

previous = previous->next;

Node \*temp = previous->next;

previous->next = temp->next;

delete temp;

temp = nullptr;

size--;

}

}

void Works::show()

{

if (this->head == nullptr)

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

else

{

Node \*temp = head;

int i = 0;

while (temp != nullptr)

{

cout << "\nInfo about lesson #" << ++i << ": " << endl;

cout << "Subject: " << temp->lesson->get\_lessonType() << endl;

cout << "Lecturer: " << temp->lesson->get\_lecturerName() << endl;

cout << "Audience ID: " << temp->lesson->get\_audienceID() << endl;

cout << "Lesson start time: " << temp->lesson->get\_lessonStartTime() << endl;

cout << "The number of works student wrote for the semester: " << temp->lesson->get\_numOfWorks() << endl;

if (((Home \*)temp->lesson)->get\_numOfHWs() > 0)

cout << "The number of hometasks student wrote per semester: " << ((Home \*)temp->lesson)->get\_numOfHWs() << endl;

temp = temp->next;

}

}

}

Works::Node::Node(Lessons \*l, Node \*n)

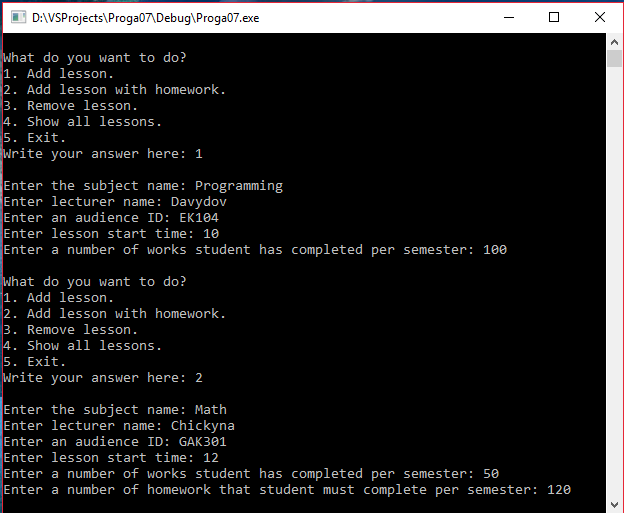
{

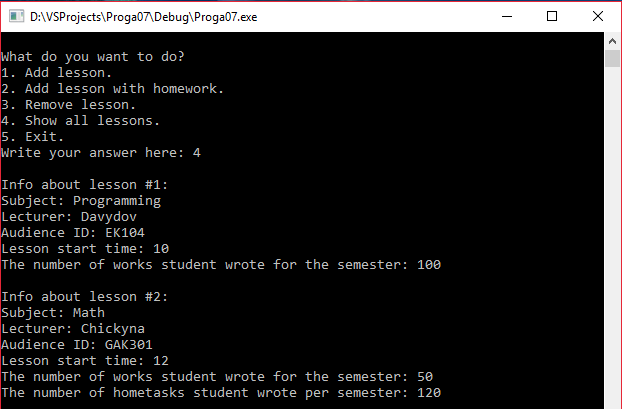
this->lesson = l;

this->next = n;

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

  
Рисунок 5 – Результат роботи програми (частина 1)

  
Рисунок 6 – Результат роботи програми (частина 2)

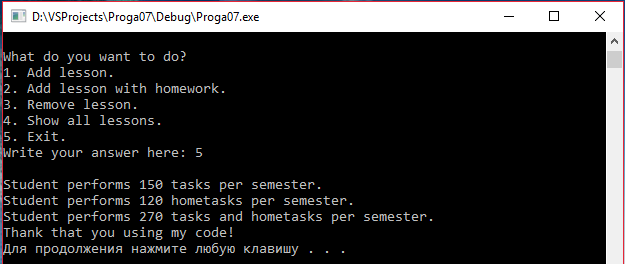


Рисунок 7 – Результат роботи програми (частина 3)

ВИСНОВКИ

При виконанні лабораторної роботи було перероблено попередню роботу (Спадкування) таким чином:  
 - на остові двох класів-списків було створено новий, який може керувати як елементами базового класу, так і елементами класів спадкоємців.  
 Досліджено механізм поліморфізму.