ЗВІТ

Виконав:

Заночкин Є., КІТ-119а

Дата: 23 лютого 2020 р.

Лабораторна робота №1

Класи та специфікатори доступу. Інкапсуляція. Константи.

Мета роботи. отримати базові знання про класи. Дослідити механізм інкапсуляції.

1.Завдання до роботи

Індивідуальне завдання 6.

Прикладна галузь: Самостійні роботи студентів.

Базовий клас: Розрахунково-графічне завдання.

Для предметної галузі розробити два класи:

- клас, що відображає сутність «базового класу». При цьому, в даному класі повинно бути мінімум три числових поля (бажано, щоб одне з цих полів було унікальним ідентифікатором об’єкта);
- клас, що має у собі динамічний масив об’єктів базового класу та має в собі методи додавання, видалення елемента, отримання елемента по індексу (або ідентифікатору), вивід усіх елементів на екран.

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: Task

Клас, що має в собі масив базового класу та методи для роботи з ним: List

2.2 Опис змінних

int mark – поле класу Task - оцінка за роботу.

int count_of_done_exercises – поле класу Task - кількість виконаних завдань.

int student_index – поле класу Task - індекс студента.

int list_size – поле класу List - розмір масиву елементів класу Task.

Task* stud – поле класу Task - масив елементів класу Task.

List list – об’єкт класу List.

2.3 Опис методів

void setList_size(int) – запис даних у змінну розміру масиву елементів класу Task (метод класу List).

int getList_size() const – отримання даних змінної розміру масиву елементів класу Task (метод класу List).

void Create_list() – створення масиву елементів і заповнення даними (метод класу List).

void Print_all() const – виведення даних елементів у консоль (метод класу List).

void Print_one_student(int) const – виведення даних одного елементу у консоль (метод класу List).

void Add_student(Task) – додавання нового елементу в масив (метод класу List).

void Delete_student(int) – видалення елемента з масиву (метод класу List).

void Free_memory() – звільнення динамічного масиву (метод класу List).

Task Get_student_ID(int)const – отримання даних елемента по індексу (метод класу List).

2.4 Опис функцій

void Menu() – функція меню.

void Test_Add_student(List&, int) – тест функції додавання об’єкта до масиву об’єктів.

void Test_Delete_student(List&, int) – тест функції видалення об’єкта з масиву об’єктів.

3. Текст програми

Task.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#define __CRTDBG_MAP_ALLOC
#include <crtdbg.h>
```

```
#define DEBUG_NEW new(_NORMAL_BLOCK, __FILE__, __LINE__)
#define new DEBUG_NEW

#include <cstdio>
#include <iostream>
using namespace std;

class Task
{
private:
  int mark;
  int count_of_done_exercises;
  int student_index;

public:
  int getMark() const;
  void setMark(int);

  int getCount_of_done_exercises() const;
  void setCount_of_done_exercises(int);

  int getStudent_index() const;
  void setStudent_index(int);
};

void Menu();
```

List.h

```
#pragma once
#include "Task.h"

class List
{
private:
  int list_size;

public:
  Task* stud;

  int getList_size() const;
  void setList_size(int);
  void Add_student(Task);
  void Delete_student(int);
```

```
  void Print_all() const;
  void Print_one_student(int) const;
  void Create_list();
  void Free_memory();
  void Get_student_ID(int)const;
};

Task Create_student();
Task Create_student2();
```

Task.cpp

```
#include "Task.h"
#include "List.h"

int Task::getMark() const { return mark; }
void Task::setMark(int mark1) { Task::mark = mark1; }

int Task::getCount_of_done_exercises() const { return count_of_done_exercises; }
void Task::setCount_of_done_exercises(int count_of_done_exercises1) { Task::count_of_done_exercises =
count_of_done_exercises1; }

int Task::getStudent_index() const { return student_index; }
void Task::setStudent_index(int exercises1) { Task::student_index = exercises1; }

Task Create_student()
{
  Task stud;
  stud.setMark(5);
  stud.setCount_of_done_exercises(5);
  return stud;
}
Task Create_student2()
{
  Task stud;
  stud.setMark(2);
  stud.setCount_of_done_exercises(2);
  return stud;
}
```

List.cpp

```
#include "List.h"
#include "Task.h"
```

```
int List::getList_size() const { return list_size; }
void List::setList_size(int size) { list_size = size; }

void List::Add_student(const Task task)
{
  setList_size(getList_size() + 1);
  Task* newstud = new Task[list_size];
  for (int i = 0; list_size > i; i++)
        newstud[i] = stud[i];
  newstud[list_size - 1] = task;
  newstud[list_size - 1].setStudent_index(stud[list_size - 2].getStudent_index() + 1);
  delete[] stud;
  stud = new Task[list_size];
  for (int i = 0; list_size > i; i++)
        stud[i] = newstud[i];
  delete[] newstud;
}
void List::Create_list()
{
  stud = new Task[list_size];
  for (int i = 0; list_size > i; i++)
  {
        stud[i] = Create_student();
        stud[i].setStudent_index(i + 1);
  }
}
void List::Delete_student(int c)
{
  setList_size(getList_size() - 1);
  Task* newstud = new Task[list_size];
  int i = 0;
  for (; i < getList_size(); i++)
  {
        if (stud[i].getStudent_index() == c)
              break;
        newstud[i] = stud[i];
  }
  for (; i < getList_size(); i++)
        newstud[i] = stud[i + 1];
  delete[] stud;
  stud = new Task[list_size];
  for (int i = 0; i < getList_size(); i++)
```

```
            stud[i] = newstud[i];
  delete[] newstud;
}
void List::Print_one_student(int number) const
{
  cout << "Index\t";
  cout << "Mark\t";
  cout << "Exercises" << endl;
  printf("%-10d", stud[number].getStudent_index());
  printf("%-10d", stud[number].getMark());
  printf("%d", stud[number].getCount_of_done_exercises());
}
void List::Print_all() const
{
  cout << "Index\t";
  cout << "Mark\t";
  cout << "Exercises";
  for (int i = 0; List::getList_size() > i; i++)
  {
            cout << endl;
            printf("%-10d", stud[i].getStudent_index());
            printf("%-10d", stud[i].getMark());
            printf("%d", stud[i].getCount_of_done_exercises());
  }
}
void List::Free_memory()
{
  delete[] stud;
}

void List::Get_student_ID(int id) const
{
  for (int i = 0; i < list_size; i++)
            if (stud[i].getStudent_index() == id)
            {
                        Print_one_student(i);
                        return;
            }
  cout << "Wrong ID" << endl;
}
```

Menu.cpp

```
#include "Task.h"
```

```
#include "List.h"

void Menu()
{
  List list;
  int c = 0;
  int count_of_students = 2;
  int menu_number = 1;
  int delete_number;
  list.setList_size(count_of_students);
  list.Create_list();
  while (menu_number)
  {
        menu_number = 0;
        cout << endl << "Menu:" << endl;
        cout << "1.Add a new student" << endl;
        cout << "2.Delete one student" << endl;
        cout << "3.Show all student" << endl;
        cout << "4.Show student via his index" << endl;
        cout << "5.End program" << endl;
        cin >> menu_number;
        switch (menu_number)
        {
        case 1:
              list.Add_student(Create_student2());
              break;
        case 2:
              cout << "Enter a index of student who you want to delete:" << endl;
              cin >> delete_number;
              if (delete_number < 1) {
                    cout << "Wrong student index" << endl;
                    break;
              }
              for (int i = 0; list.getList_size() > i; i++)
              if (delete_number == list.stud[i].getStudent_index())
              {
                    list.Delete_student(delete_number);
                    break;
              }
              break;
        case 3:
              list.Print_all();
```

```
                break;
        case 4:
                cout << "Enter a index of student:";
                cin >> c;
                list.Get_student_ID(c);
                break;
        case 5:
                list.Free_memory();
                return;
        default:
                cout << "You have chosen the wrong number of the menu";
                break;
        }
  }
}
```

main.cpp

```
#include "Task.h"

int main()
{
  Menu();
  if (_CrtDumpMemoryLeaks())
        cout << endl << "WARNING! Memory leak" << endl;
  else
        cout << endl << "There is no memory leak" << endl;
  return 0;
}
```

Test.cpp

```
#include "List.h"
#include "Task.h"

int Test_Add_student(List&, int);
int Test_Delete_student(List&, int);

int main()
{
  List test;
  int count = 0;
  test.setList_size(2);
  test.Create_list();
```

```
  count = Test_Add_student(test, count);
  count = Test_Delete_student(test, count);

  if (count == 2)
        cout << "All tests are successful" << endl;
  else
        cout << "Not all tests are successful" << endl;

  test.Free_memory();
  if (_CrtDumpMemoryLeaks())
        cout << endl << "WARNING! Memory leak" << endl;
  else
        cout << endl << "There is no memory leak" << endl;
  return 0;
}

int Test_Add_student(List& test, int count)
{
  test.Add_student(Create_student2());
  if (test.getList_size() == 3)
  {
        cout << "Test: Add_student - successful" << endl;
        count++;
  }
  else
        cout << "Test: Add_student - unsuccessful" << endl;
  return count;
}

int Test_Delete_student(List& test, int count)
{
  test.Delete_student(test.getList_size());
  if (test.getList_size() == 2)
  {
        cout << "Test: Delete_student - successful" << endl;
        count++;
  }
  else
        cout << "Test: Delete_student - unsuccessful" << endl;
  return count;
}
```

4. Результати роботи програми

```
Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
3
Index   Mark    Exercises
1         5         5
2         5         5
Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
1

Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
3
Index   Mark    Exercises
1         5         5
2         5         5
3         2         2
Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
2
Enter a index of student who you want to delete:
2

Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
3
Index   Mark    Exercises
1         5         5
3         2         2
Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
4
Enter a index of student:3
Index   Mark    Exercises
3         2         2
Menu:
1.Add a new student
2.Delete one student
3.Show all student
4.Show student via his index
5.End program
5

There is no memory leak
Для продолжения нажмите любую клавишу . . .
```

```
C:\Windows\system32\cmd.exe
Test: Add_student - successful
Test: Delete_student - successful
All tests are successful

There is no memory leak
Для продолжения нажмите любую клавишу . . .
```

5. Висновки

При виконанні лабораторної роботи набуто навички роботи з класами та їх специфікаторами доступу, інкапсуляцією, константами.
Програма протестована, виконується без помилок, витоків пам’яті немає.