Автор: Наймитенко С. КІТ-119а

Дата: 14 лютого 2020

# Лабораторна робота №1

КЛАСИ

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

1 ЗАВДАННЯ ДО РОБОТИ

Загальне завдання

Для предметної галузі з табл. 1.2 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.

Прикладна галузь – Компоненти програм

Базовий клас – Бібліотека, що підключається (модуль)

Опис класів

Базовий клас – C\_Library

Клас що має в собі динамічний масив базового класу та методи для роботи з ним – С\_List

Опис змінних

Int average\_number\_of\_uses – поле класу C\_Library (середня кількість використань функції)

Int year\_creating - – поле класу C\_Library (рік створення функції)

Int number\_of\_functions – поле класу C\_Library(кількість функцій в бібліотеці)

Int size – поле класу C\_List (змінна розміру масива)

C\_Library\* list – поле класу C\_Library (динамічний масив)

С\_List list\_lib - об’єкт класу С\_List

Int size – змінна розміру масиву

Int order – змінна індексу (порядкового номеру)

С\_Library new\_lib – змінна для нової бібліотеки

Опис методів

C\_Library\* Create() – метод класу C\_List (створення масиву і заповнення його данними)

C\_Library\* Add(C\_Library,int) - метод класу C\_List (додавання нового елементу в масив)

C\_Library\* Delete(int) - метод класу C\_List (видалення елемента з масиву)

Void Index\_output(const C\_Library)const - метод класу C\_List (виведення на екран по індексу)

C\_Library Index\_return(const int) - метод класу C\_List (повернення елемента по індексу)

Void Output()const; - метод класу C\_List (виведення на екран)

Опис функцій

С\_Library New\_Library() – створення одного елементу типу С\_Library та заповнення його данними з масиву

Bool Test\_Add() – тест методу додавання об’єкта

Bool Test\_Delete() – тест методу видалення об’єкта

Bool Test\_Index\_return() – тест методу повернення об’єкта по індексу

Текст програми

Файл Builder.h

#pragma once

#include "C\_List.h"

C\_Library New\_Library();

Файл C\_Library.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <string.h>

#define CRTDBG\_MAP\_ALLOC

#include <crtdbg.h>

#define DEBUG\_NEW new(\_NORMAL\_BLOCK, FILE, \_\_LINE)

#include <iostream>

#include <clocale>

class C\_Library

{

private:

int average\_number\_of\_uses;

int year\_creating;

int number\_of\_functions;

public:

void setAverage\_number\_of\_uses(const int);

void setYear\_Creating(const int);

void setNumber\_of\_function(const int);

int getAverage\_number\_of\_uses()const;

int getYear\_Creating()const;

int getNumber\_of\_function()const;

};

Файл C\_List.h

#pragma once

#include "C\_Library.h"

class C\_List

{

private:

int size;

public:

C\_Library\* list;

void setSize\_array(const int size);

int getSize\_array()const;

C\_Library\* Create();

C\_Library\*Add(C\_Library,int);

C\_Library\* Delete(const int);

void Index\_output(const C\_Library)const ;

C\_Library Index\_return(const int index) ;

void Output()const;

};

Файл Test.h

#pragma once

bool Test\_Add();

bool Test\_Delete();

bool Test\_Index\_return();

Файл Builder.cpp

#include "Builder.h"

#include "C\_Library.h"

C\_Library New\_Library() // 2

{

int a = 0;

C\_Library new\_lib;

printf("Введите данные о новой библиотеке\n");

printf("среднее количество использований библиотеки\n");

scanf("%i", &a);

new\_lib.setAverage\_number\_of\_uses(a);

printf("Год создания библиотеки\n");

scanf("%i", &a);

new\_lib.setYear\_Creating(a);

printf("Количество функций в библиотеке\n");

scanf("%i", &a);

new\_lib.setNumber\_of\_function(a);

return new\_lib;

}

Файл C\_Library.cpp

#include "C\_Library.h"

void C\_Library::setAverage\_number\_of\_uses(const int average\_number\_of\_uses)

{

this->average\_number\_of\_uses = average\_number\_of\_uses;

}

void C\_Library::setYear\_Creating(const int year\_creating)

{

this->year\_creating = year\_creating;

}

void C\_Library::setNumber\_of\_function(const int number\_of\_functions)

{

this->number\_of\_functions = number\_of\_functions;

}

int C\_Library::getAverage\_number\_of\_uses()const

{

return this->average\_number\_of\_uses;

}

int C\_Library::getYear\_Creating()const

{

return this->year\_creating;

}

int C\_Library::getNumber\_of\_function()const

{

return this->number\_of\_functions;

}

Файл C\_List.cpp

#include "C\_Library.h"

#include "C\_List.h"

#include "Builder.h"

void C\_List::setSize\_array(int size)

{

this->size = size;

}

int C\_List::getSize\_array()const

{

return this->size;

}

C\_Library\* C\_List::Create() // 1

{

int a = getSize\_array();

list = new C\_Library[a];

printf("Введите данные об библиотеках. Их всего %i\n", size);

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

list[i] = New\_Library();

return list;

}

C\_Library\* C\_List::Add(C\_Library new\_lib,int order) // 3

{

int size = getSize\_array();

size++;

setSize\_array(size);

C\_Library\* new\_list = new C\_Library[size];

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

{

if (i != order - 1)

{

new\_list[i] = list[j];

j++;

}

else

new\_list[i] = new\_lib;

}

delete[] list;

return new\_list;

}

C\_Library\* C\_List::Delete(int order) // 4

{

int size = getSize\_array();

size--;

setSize\_array(size);

C\_Library\* new\_list = new C\_Library[size];

printf("Елемент который вы удалили\n");

Index\_output(list[order-1]);

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

new\_list[i] = list[i];

for (int i = order - 1; i < size; i++)

new\_list[i] = list[i + 1];

delete[] list;

return new\_list;

}

C\_Library C\_List::Index\_return(const int index)

{

return list[index];

}

void C\_List::Index\_output(const C\_Library lib)const // 5

{

int a;

a = lib.getAverage\_number\_of\_uses();

printf("%-23i \t\t", a);

a = lib.getYear\_Creating();

printf("%-4i \t\t", a);

a = lib.getNumber\_of\_function();

printf("%i\n", a);

}

void C\_List::Output()const // 6

{

printf("Вивод на экран\n");

printf("Среднее количество использований\tГод создания\tКоличество функций в библиотеке\n");

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

Index\_output(list[i]);

}

Файл Test.cpp

#include "C\_Library.h"

#include "Test.h"

int main()

{

setlocale(LC\_ALL, "Rus");

bool result;

result = Test\_Add();

if (result)

printf("Test\_Add пройден успешно!\n");

else

printf("Test\_Add не пройден!\n");

result = Test\_Delete();

if (result)

printf("Test\_Delete пройден успешно!\n");

else

printf("Test\_Delete не пройден!\n");

result = Test\_Index\_return();

if (result)

printf("Test\_Index\_return пройден успешно!\n");

else

printf("Test\_Index\_return не пройден!\n");

return 0;

}

Файл Test\_method.cpp

#include "C\_Library.h"

#include "C\_List.h"

#include "Test.h"

bool Test\_Add()

{

C\_List list;

list.setSize\_array(1);

list.list = new C\_Library[1];

C\_Library new\_lib;

list.Add(new\_lib,2);

int size = list.getSize\_array();

if (size == 2)

return true;

else

return false;

}

bool Test\_Delete()

{

C\_List list;

list.setSize\_array(2);

list.list = new C\_Library[2];

C\_Library new\_lib;

list.Delete(2);

int size = list.getSize\_array();

if (size == 1)

return true;

else

return false;

}

bool Test\_Index\_return()

{

C\_List list;

list.setSize\_array(2);

list.list = new C\_Library[2];

list.list[0].setAverage\_number\_of\_uses(1);

list.list[0].setNumber\_of\_function(2);

list.list[0].setYear\_Creating(3);

list.list[1].setAverage\_number\_of\_uses(4);

list.list[1].setNumber\_of\_function(5);

list.list[1].setYear\_Creating(6);

C\_Library return\_lib = list.Index\_return(2-1);

int count = 0 ,value;

value = return\_lib.getAverage\_number\_of\_uses();

if ( value == 4)

count++;

value = return\_lib.getNumber\_of\_function();

if (value == 5)

count++;

value = return\_lib.getYear\_Creating();

if (value == 6)

count++;

if (count == 3)

return true;

else

return false;

}

Файл Main.cpp

#include "C\_Library.h"

#include "C\_List.h"

#include "Builder.h"

int main()

{

setlocale(LC\_ALL, "Rus");

C\_List list\_lib;

printf("Cоздание масива\nВведите размерность масва\n");

int size;

scanf("%i", &size);

list\_lib.setSize\_array(size);

list\_lib.list = list\_lib.Create();

list\_lib.Output();

C\_Library new\_lib = New\_Library();

int order = 0;

printf("Введите порядковый номер который вы хотите присвоить новому елементу\n");

scanf("%i", &order);

printf("Добавления нового елемента\n");

list\_lib.list = list\_lib.Add(new\_lib,order);

list\_lib.Output();

printf("Удаления елемента\n");

printf("Введите номер елемента которого вы хотите удалить\n");

scanf("%i", &order);

printf("Элемент который вы удалили\n");

list\_lib.list = list\_lib.Delete(order);

list\_lib.Output();

printf("Введите индекс елемента данные которого вы хотите получить \n");

scanf("%i", &order);

C\_Library index\_lib = list\_lib.Index\_return(a - 1);

printf("Вывод полученого элемента на екран\n");

list\_lib.Index\_output(index\_lib);

delete[] list\_lib.list;

if (\_CrtDumpMemoryLeaks())

printf("\n\nЕсть утечка памяти\n");

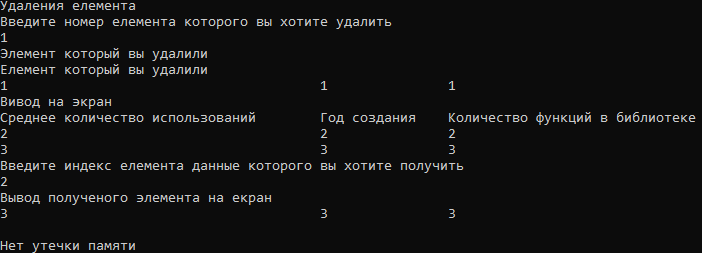
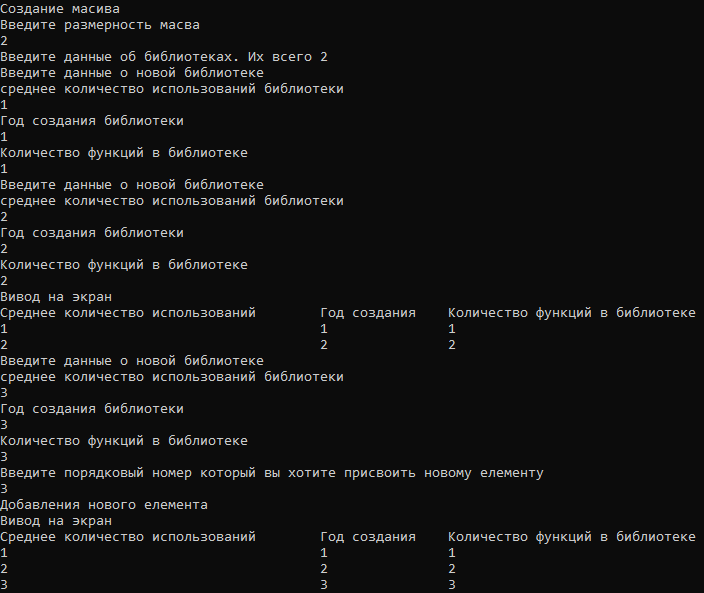
else

printf("\nНет утечки памяти\n");

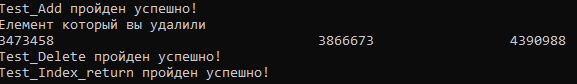
return 0;

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ



Результат роботи основоного завдання



Результат тестування методів

(Add,Delete,Index\_return)

ВИСНОВКИ

Отримав базові знання про класи. Дослідив механізм інкапсуляції.