Автор: Наймитенко С. КІТ-119а

Дата: 23 лютого 2020

# **Лабораторна робота №2**

**ПЕРЕВАНТАЖЕННЯ МЕТОДІВ**

Мета роботи: отримати базові знання про класи, конструктори та деструктори. Дослідити механізм створення та видалення об’єктів.

**ЗАВДАННЯ ДО РОБОТИ**

Загальне завдання

Поширити попередню лабораторну роботу таким чином:

1) в базовому класі необхідно додати:

- мінімум одне поле типу char\*;

- конструктор за замовчуванням, копіювання та конструктор з аргументами;

- деструктор;

2) у клас-список потрібно додати метод обходу масиву для виконання індивідуального завдання.

Приклад сигнатури такого методу:

CPhone& findCheapestPhone(float diagonal);

У наведеному прикладі реалізоване завдання пошуку самого дешевого телефону з заданою діагоналлю (повертається один телефон).

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**

const char\* dynamically – поле класу C\_Library (чи є бібліотека динамічною)

Int ID – поле класу C\_Library (ідентифікатор елемену)

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**

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

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

void Delete(const int) - метод класу C\_List (видалення елемента з масиву)

void Index\_output(const C\_Library,const int)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)

float Difference(); - метод класу C\_List (визначення різниці у скільки разів обсяг бібліотек ,що дінамічно підключаються, ніж загальний обсяг бібліотек )

**Опис функцій**

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

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

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

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

Bool Test\_Difference(C\_List&)– тест методу різниці динамічних бібліотек і загальної кількості

**Текст програми**

Файл Builder.h

#pragma once

#include "C\_List.h"

C\_Library New\_Lib(int);

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>

#include <ctime>

class C\_Library

{

private:

const char\* dynamically;

int ID;

int year\_creating;

int number\_of\_functions;

public:

void setDynamically(const char\*);

void setID(const int);

void setYear\_Creating(const int);

void setNumber\_of\_function(const int);

const char\* getDynamically()const;

int getID()const;

int getYear\_Creating()const;

int getNumber\_of\_function()const;

C\_Library();

C\_Library(string,const int,const int,const int );

C\_Library(const C\_Library& );

~C\_Library();

};

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;

void Create();

void Add(C\_Library&, const int);

void Delete(const int);

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

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

void Output()const;

float Difference();

};

Файл Test.h

#pragma once

bool Test\_Add(C\_List&);

bool Test\_Delete(C\_List&);

bool Test\_Index\_return(C\_List&);

bool Test\_Difference(C\_List&);

Файл Builder.cpp

#include "Builder.h"

#include "C\_Library.h"

C\_Library New\_Lib(int i) // 2

{

int a = 0;

C\_Library new\_l;

int x;

srand(static\_cast<unsigned int>(time(0)));

if (i % 2 == 0)

x = rand() % 1;

else

x = 2;

if (x==0)

{

static char dynamically[] = "no";

C\_Library new\_lib(dynamically, 20+i, 1998, 40);

return new\_lib;

}

else if (x==1)

{

static char dynamically[] = "yes";

C\_Library new\_lib(dynamically, 46+i, 2000, 20);

return new\_lib;

}

else if (x==2)

{

C\_Library new\_lib(new\_l);

int id = new\_lib.getID();

new\_lib.setID(id + i);

return new\_lib;

}

}

Файл C\_Library.cpp

#include "C\_Library.h"

#include "C\_Library.h"

void C\_Library::setDynamically(const char\* dynamically)

{

this->dynamically = dynamically;

}

void C\_Library::setID(const int ID)

{

this->ID = ID;

}

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;

}

const char\* C\_Library::getDynamically()const

{

return dynamically;

}

int C\_Library::getID()const

{

return this->ID;

}

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

{

return this->year\_creating;

}

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

{

return this->number\_of\_functions;

}

C\_Library::C\_Library() :dynamically("yes"), ID(10), year\_creating(1999), number\_of\_functions(50)

{

}

C\_Library::C\_Library(char\* dynamically, const int ID, const int year\_creating, const int number\_of\_functions) : dynamically(dynamically), ID(ID), year\_creating(year\_creating), number\_of\_functions(year\_creating)

{

}

C\_Library::C\_Library(const C\_Library& lib) : dynamically(lib.dynamically), ID(lib.ID), year\_creating(lib.year\_creating), number\_of\_functions(lib.year\_creating)

{

}

C\_Library::~C\_Library()

{

}

Файл C\_List.cpp

#include "C\_Library.h"

#include "C\_List.h"

#include "Builder.h"

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

{

this->size = size;

}

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

{

return this->size;

}

void C\_List::Create() // 1

{

printf("Происходит заполнеие масива данными\n");

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

list[i] = New\_Lib(i+1);

}

void C\_List::Add(C\_Library& lib, const int order) // 3

{

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] = lib;

}

delete[] list;

list = new C\_Library[size];

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

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

delete[] new\_list;

}

void C\_List::Delete(const int order) // 4

{

size--;

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

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

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

list = new C\_Library[size];

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

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

delete[] new\_list;

}

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

{

int a;

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

{

a = list[i].getID();

if (a == index)

{

a = i;

i = size;

}

}

return list[a];

}

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

{

int a;

printf("%-5i", i);

a = lib.getID();

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

a = lib.getYear\_Creating();

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

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

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

const char\* b;

b = lib.getDynamically();

printf("%s", b );

printf("\n");

}

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

{

printf("Вивод на экран всех библиотек\n");

printf(" ID библиотеки\tГод создания\tКоличество функций в библиотеке\tДинамически подключается?\n");

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

Index\_output(list[i],i+1);

}

float C\_List::Difference()

{

int count = 0;

const char\* y;

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

{

y = list[i].getDynamically();

if (strcmp("yes", y))

count++;

}

float dif = (float) size/(float) count;

printf("В %3.3f раз количество библиотек, которые динамически подключаются, меньше чем общее количество библиотек\n",dif);

return dif;

}

Файл Test.cpp

#include "C\_Library.h"

#include "C\_List.h"

#include "Test.h"

int main()

{

setlocale(LC\_ALL, "Rus");

bool result;

C\_List list;

list.list = new C\_Library[3];

list.setSize\_array(3);

result = Test\_Add(list);

if (result)

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

else

printf("Test\_Add не пройден!\n");

result = Test\_Delete(list);

if (result)

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

else

printf("Test\_Delete не пройден!\n");

result = Test\_Index\_return(list);

if (result)

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

else

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

result = Test\_Difference(list);

if (result)

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

else

printf("Test\_Difference 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)

{

C\_Library new\_lib;

list.Add(new\_lib, 4);

int size = list.getSize\_array();

if (size == 4)

return true;

else

return false;

}

bool Test\_Delete(C\_List& list)

{

list.Delete(1);

int size = list.getSize\_array();

if (size == 3)

return true;

else

return false;

}

bool Test\_Index\_return(C\_List& list)

{

static const char diff[] = "yes";

C\_Library new\_el(diff, 20, 2001, 29);

list.list[0] = new\_el;

C\_Library return\_lib = list.Index\_return(20);

int count = 1, value = 0;

const char\* y = return\_lib.getDynamically();

if (strcmp("yes", y))

count++;

value = return\_lib.getID();

if (value == 20)

count++;

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

if (value == 2001)

count++;

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

if (value == 29)

count++;

if (count == 4)

return true;

else

return false;

}

bool Test\_Difference(C\_List& list)

{

static char dynamically[] = "no";

C\_Library new\_lib1(dynamically, 20, 1998, 40);

list.list[1] = new\_lib1;

C\_Library new\_lib2(new\_lib1);

list.list[2] = new\_lib2;

float diff = list.Difference();

if (diff == 1.5)

return true;

else

return false;

}

Файл Main.cpp

#include "C\_Library.h"

#include "C\_List.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 = new C\_Library[size];

list\_lib.Create();

list\_lib.Output();

int order = 0;

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

scanf("%i", &order);

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

printf("Добавления нового елемента\n");

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.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(order);

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

list\_lib.Index\_output(index\_lib,1);

float diff = list\_lib.Difference();

delete[] list\_lib.list;

if (\_CrtDumpMemoryLeaks())

printf("\n\nЕсть утечка памяти\n");

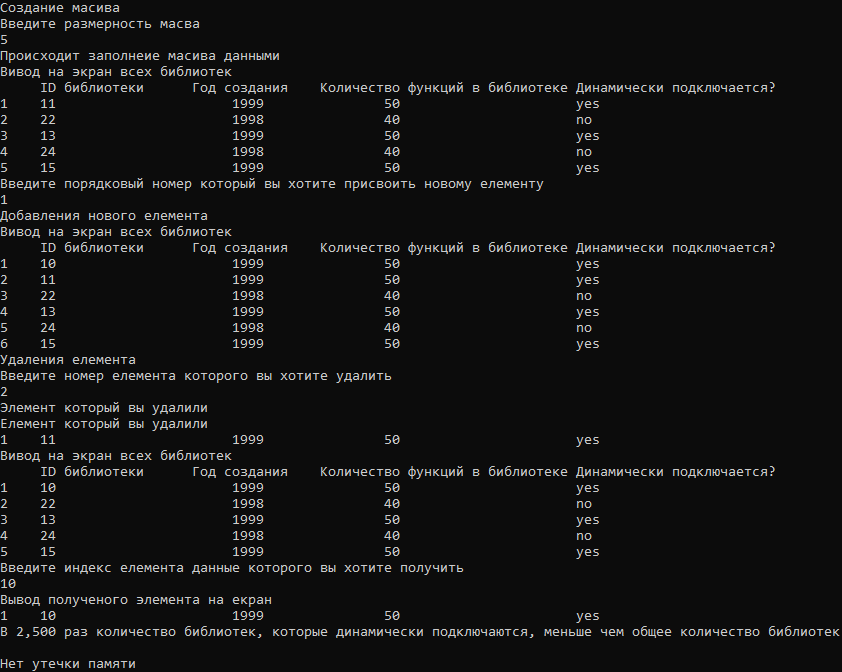
else

printf("\nНет утечки памяти\n");

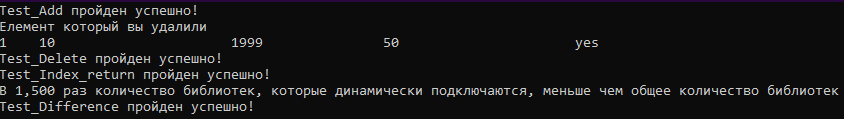
return 0;

}

**РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ**



Результат роботи основоного завдання

 Результат тестування методів

(Add,Delete,Index\_return,Difference)

ВИСНОВКИ

Отримав базові знання про класи, конструктори та деструктори. Дослідив механізм створення та видалення об’єктів..