Автор: Наймитенко С. КІТ-119а

Дата: 02 квітня 2020

# **Лабораторна робота №7**

**ПОЛІМОРФІЗМ**

Мета роботи: отримати знання про парадигму ООП – поліморфізм; навчитися застосовувати отримані знання на практиці.

1 ЗАВДАННЯ ДО РОБОТИ

Модернізувати попередню лабораторну роботу шляхом: - додавання ще одного класу-спадкоємця до базового класу. Поля обрати самостійно; - базовий клас зробити абстрактним. Додати абстрактні поля; - розроблені класи-списки поєднуються до одного класу таким чином, щоб він міг працювати як з базовим класом, так і з його спадкоємцями. При цьому серед полів класу-списку повинен бути лише один масив, що містить усі типи класів ієрархії. Оновити методи, що працюють з цим масивом.

**Опис класів**

Абстрактний базовий клас – C\_Library

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

Клас для зображення композиції – С\_Function

Клас для зображення агрегації – C\_Language

Клас наслідник базового класу C\_Library – С\_Expansion

Клас наслідник базового класу C\_Library – Сapabilities

**Опис змінних**

String expansion\_file – поле класу С\_Expansion(розширення файлу бібліотеки)

String language – поле класу C\_Language (назва мови програмування)

String function – поле класу С\_Function (назва функції з бібліотеки)

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)

Int Read\_file(string) - метод класу C\_List (зчитування з файлу)

void Count\_line(string) – метод класу C\_List (підрахунок кількості рядків у файлі)

C\_Library Distribution(string) - метод класу C\_List (розподіл , із рядка в об’єкт)

stringstream Str\_return(C\_Library&)const - метод класу C\_List (створення рядку змінних з об’єкта)

void Str\_output(stringstream&,int)const - метод класу C\_List (виведення рядка на екран)

float Difference() - метод класу C\_List (підрахуванн різнці між кількістю динамічно підключених бібліотек від іх загальної кількості)

int Write\_file(string) - метод класу C\_List (запис до файлу)

void Sort(func) - метод класу C\_List (сортування масиву елементів)

void If\_lib\_connected() - метод класу C\_List (виведення на екран данних про підключену бібліотеку (в якій мові програмування використовується, яку функцію має))

void What\_functions\_are(int) - метод класу C\_List ( шо виводить функцію з бібліотке на екран)

**Опис функцій**

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\_Differense(C\_List&) – тест методу що визначає відношення між загальною кількістю бібліотек та бібліотек що динамічно підлючаються.

Bool Test\_Sort(C\_List&)) – тест методу сортування

**Текст програми**

Файл С\_Language.h

#pragma once

#include "C\_Library.h"

using std::string;

class C\_Language /\*\*< Оголошення класу C\_Language\*/

{

private:

string language;/\*\*< поле класу C\_Language - мова програмування\*/

public:

void setLanguage\_programming(string);

string Which\_language\_programming();

C\_Language(); /\*\*< Оголошення конструктора по замовчуваннням\*/

C\_Language(string);/\*\*< Оголошення конструктора з параметрами\*/

C\_Language(const C\_Language&);/\*\*< Оголошення конструктора копіювання\*/

~C\_Language();/\*\*< Оголошення деструктора\*/

};

Файл C\_Library.h

.04.05

\*/

#pragma once

#define CRTDBG\_MAP\_ALLOC

#include <crtdbg.h>

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

#include <iostream>

#include <clocale>

#include <ctime>

#include <iomanip>

#include <fstream>

#include <sstream>

#include <regex>

#include <Windows.h>

#include <stddef.h>

#include <cstddef>

#include "C\_Language.h"

using std::cin;

using std::cout;

using std::string;

using std::endl;

using std::setw;

using std::setprecision;

using std::ifstream;

using std::ofstream;

using std::getline;

using std::stringstream;

using std::getline;

using std::istringstream;

using std::regex;

using std::regex\_match;

using std::regex\_search;

typedef bool (func)(int, int);

class C\_Library /\*\*< Оголошення базового класу C\_Library\*/

{

protected:

string dynamically; /\*\*< поле класу C\_Library - чи динамічна функція\*/

string name; /\*\*< поле класу C\_Library - ім'я бібліоткеи\*/

int ID; /\*\*< поле класу C\_Library - идентифікатор бібліотеки\*/

int year\_creating; /\*\*< поле класу C\_Library - рік створення\*/

int number\_of\_functions; /\*\*< поле класу C\_Library - кількість функцій\*/

class C\_Function /\*\*< Оголошення класу C\_Function\*/

{

private:

string function; /\*\*< поле класу C\_Function - одна з функцій в цій бібліотеці\*/

public:

void setFunction(string);

string What\_function\_is\_in\_this\_library();

C\_Function();/\*\*< Оголошення конструктора по замовчуваннням\*/

C\_Function(string);/\*\*< Оголошення конструктора з параметрами\*/

C\_Function(const C\_Function&);/\*\*< Оголошення конструктора копіювання\*/

~C\_Function();/\*\*< Оголошення деструктора\*/

};

C\_Function function;/\*\*< поле класу C\_Library - обьект класу C\_Function\*/

C\_Language lang;/\*\*< поле класу C\_Library - обьект класу C\_Language\*/

public:

virtual void setDynamically(string) =0;

virtual void setName(string) = 0;

virtual void setID(const int) = 0;

virtual void setYear\_Creating(const int) = 0;

virtual void setNumber\_of\_function(const int) = 0;

virtual void setLanguage\_programming(string) = 0;

virtual void setFunction(string) = 0;

virtual string getDynamically()const = 0;

virtual string getName()const = 0;

virtual int getID()const = 0;

virtual int getYear\_Creating()const = 0;

virtual int getNumber\_of\_function()const = 0;

virtual string getFunction() = 0;

virtual string getLanguage\_programming() = 0;

C\_Library();/\*\*< Оголошення конструктора по замовчуваннням\*/

C\_Library(string, string, const int, const int, const int, C\_Function, C\_Language);/\*\*< Оголошення конструктора з параметрами\*/

C\_Library(const C\_Library&);/\*\*< Оголошення конструктора копіювання\*/

~C\_Library();/\*\*< Оголошення деструктора\*/

};

class C\_Expansion : public C\_Library/\*\*< Оголошення класа спадкоемця C\_Expansion\*/

{

private:

string expansion\_file;/\*\*< поле класу C\_Expansion - розширення файлу\*/

public:

void setExpansion\_file(string);

string getExpansion\_file();

void setDynamically(string)override;

void setName(string)override;

void setID(const int) override;

void setYear\_Creating(const int)override;

void setNumber\_of\_function(const int)override;

void setLanguage\_programming(string)override;

void setFunction(string)override;

string getDynamically()const override;

string getName()const override;

int getID()const override;

int getYear\_Creating()const override;

int getNumber\_of\_function()const override;

string getFunction()override;

string getLanguage\_programming()override;

C\_Expansion();/\*\*< Оголошення конструктора по замовчуваннням\*/

C\_Expansion(string, string, string, const int, const int, const int, string, string);/\*\*< Оголошення конструктора з параметрами\*/

C\_Expansion(const C\_Expansion&);/\*\*< Оголошення конструктора копіювання\*/

~C\_Expansion();/\*\*< Оголошення деструктора\*/

};

class Capabilities : public C\_Library/\*\*< Оголошення класа спадкоемця Capabilities \*/

{

private:

string capabilities;/\*\*< поле класу Capabilities - одна з можливостей функції\*/

C\_Expansion obj;/\*\*< поле класу Capabilities - обьект класу C\_Expansion\*/

public:

C\_Expansion getExpansion();

void setExpansion(C\_Expansion);

string getExpansion\_file();

void setExpansion\_file(string);

void setCapabilities(string);

string getCapabilities();

void setDynamically(string)override;

void setName(string)override;

void setID(const int) override;

void setYear\_Creating(const int)override;

void setNumber\_of\_function(const int)override;

void setLanguage\_programming(string)override;

void setFunction(string)override;

string getDynamically()const override;

string getName()const override;

int getID()const override;

int getYear\_Creating()const override;

int getNumber\_of\_function()const override;

string getFunction()override;

string getLanguage\_programming()override;

Capabilities();/\*\*< Оголошення конструктора по замовчуваннням\*/

Capabilities(string, string, string, const int, const int, const int, string, string ,C\_Expansion);/\*\*< Оголошення конструктора з параметрами\*/

Capabilities(const Capabilities&);/\*\*< Оголошення конструктора копіювання\*/

~Capabilities();/\*\*< Оголошення деструктора\*/

};

Файл C\_List.h

#include "C\_Library.h"

class C\_List /\*\*< Оголошення класу списку\*/

{

private:

float size; /\*\*< поле класу C\_List - розмір масиву\*/

public:

Capabilities\* list; /\*\*< оголошення масиву класу Capabilities\*/

void setSize\_array(const int size);

int getSize\_array()const;

int Read\_file(string, string);/\*\*< Оголошення методу зчитування інформації з файлу\*/

int Count\_line(string);/\*\*< Оголошення методу визначення розміру масиву\*/

void Distribution(string,string,int);/\*\*< Оголошення методу створення обьекту із інформації зчитаної з файлу\*/

void Create();/\*\*< Оголошення методу заповнення масиву данними\*/

void Add(Capabilities, const int);/\*\*< Оголошення методу додавання нового елементу в масив\*/

void Delete(const int);/\*\*< Оголошення методу видалення елементу з масиву\*/

stringstream Str\_return(Capabilities)const;/\*\*< Оголошення методу зчитування інформації з обьекту та перетворення її в потік\*/

void Str\_output(stringstream&, int)const;/\*\*< Оголошення методу виведення на екран одного єлементу\*/

Capabilities& Index\_return(const int index);/\*\*< Оголошення методу що повертає обьект визначивши його по дентифікатору\*/

void Output()const;/\*\*< Оголошення методу виведення всього масиву на екран\*/

void Sort(func);/\*\*< Оголошення методу сортування масиву \*/

float Difference();/\*\*< Оголошення методу що визначає відношення кількості бібліотек що динамічно підключаються до загальної кількості елементів\*/

void Info\_about\_lib();/\*\*< Оголошення методу виведення на екран інформації про бібліотеки\*/

int Write\_file(string);/\*\*< Оголошення методу зчитування в файл\*/

void Check();/\*\*< Оголошення методу виведення на екран обьектів в ім'ї яких 2 слова\*/

~C\_List();/\*\*< Оголошення деструктора\*/

};

Файл Test.h

#pragma once

bool Test\_Add(C\_List&);/\*\*< Оголошення функції що перевіряє метод Add\*/

bool Test\_Delete(C\_List&);/\*\*< Оголошення функції що перевіряє метод Delete\*/

bool Test\_Index\_return(C\_List&);/\*\*< Оголошення функції що перевіряє метод Index\_return\*/

bool Test\_Difference(C\_List&);/\*\*< Оголошення функції що перевіряє метод Difference\*/

bool Test\_Sort(C\_List&);/\*\*< Оголошення функції що перевіряє метод Sort\*/

Файл С\_Language.cpp

#include"C\_Library.h"/\*\*< підключення файлу C\_Library.h\*/

void C\_Language::setLanguage\_programming(string name)

{

language = name;

}

string C\_Language::Which\_language\_programming()

{

return language;

}

C\_Language::C\_Language() :language("Default")/\*\*< Реалізація конструктора по замовчуванням\*/

{

}

C\_Language::C\_Language(string language) : language(language)/\*\*< Реалізація конструктора з параметрами\*/

{

}

C\_Language::C\_Language(const C\_Language& lang) : language(lang.language)/\*\*< Реалізація конструктора копіювання\*/

{

}

C\_Language::~C\_Language()/\*\*< Реалізація деструктора \*/

{

}

Файл C\_Library.cpp

#include "C\_Library.h"/\*\*< підключення файлу C\_Library.h\*/

void C\_Library::C\_Function::setFunction(string name)

{

this->function = name;

}

string C\_Library::C\_Function::What\_function\_is\_in\_this\_library()

{

return function;

}

void C\_Library::setDynamically(string dynamically)

{

this->dynamically = dynamically;

}

void C\_Library::setName(string name)

{

this->name = name;

}

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;

}

void C\_Library::setLanguage\_programming(string lang)

{

this->lang.setLanguage\_programming(lang);

}

void C\_Library::setFunction(string name)

{

function.setFunction(name);

}

string C\_Library::getDynamically()const

{

return dynamically;

}

string C\_Library::getName()const

{

return name;

}

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;

}

string C\_Library::getFunction()

{

return function.What\_function\_is\_in\_this\_library();

}

string C\_Library::getLanguage\_programming()

{

return lang.Which\_language\_programming();

}

C\_Library::C\_Function::C\_Function() : function("Default") /\*\*< Реалізація конструктора по замовчуванням\*/

{

}

C\_Library::C\_Function::C\_Function(string function) : function(function) /\*\*< Реалізація конструктора з параметрами\*/

{

}

C\_Library::C\_Function::C\_Function(const C\_Function& func) : function(func.function)/\*\*< Реалізація конструктора копіювання\*/

{

}

C\_Library::C\_Function::~C\_Function()/\*\*< Реалізація деструктора \*/

{

}

C\_Library::C\_Library() :dynamically("yes"), name("Default"), ID(10), year\_creating(1999), number\_of\_functions(50), function("default"), lang("default")

{

}

C\_Library::C\_Library(string dynamically, string name, const int ID, const int year\_creating, const int number\_of\_functions, C\_Function func, C\_Language lang) : dynamically(dynamically), name(name), ID(ID), year\_creating(year\_creating), number\_of\_functions(number\_of\_functions), function(func), lang(lang)/\*\*< Реалізація конструктора з параметрами\*/

{

}

C\_Library::C\_Library(const C\_Library& lib) : dynamically(lib.dynamically), name(lib.name), ID(lib.ID), year\_creating(lib.year\_creating), number\_of\_functions(lib.number\_of\_functions), function(lib.function), lang(lib.lang)/\*\*< Реалізація конструктора копіювання\*/

{

}

C\_Library::~C\_Library()/\*\*< Реалізація деструктора \*/

{

}

void C\_Expansion::setExpansion\_file(string expansion)

{

this->expansion\_file = expansion;

}

string C\_Expansion::getExpansion\_file()

{

return this->expansion\_file;

}

void C\_Expansion::setDynamically(string dynamically)

{

this->dynamically = dynamically;

}

void C\_Expansion::setName(string name)

{

this->name=name;

}

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

{

this->ID = ID;

}

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

{

this->year\_creating = year\_creating;

}

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

{

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

}

void C\_Expansion::setLanguage\_programming(string lang)

{

this->lang.setLanguage\_programming(lang);

}

void C\_Expansion::setFunction(string func)

{

this->function.setFunction(func);

}

string C\_Expansion::getDynamically()const

{

return this->dynamically;

}

string C\_Expansion::getName()const

{

return this->name;

}

int C\_Expansion::getID()const

{

return this->ID;

}

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

{

return this->year\_creating;

}

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

{

return this->number\_of\_functions;

}

string C\_Expansion::getFunction()

{

return this->function.What\_function\_is\_in\_this\_library();

}

string C\_Expansion::getLanguage\_programming()

{

return this->lang.Which\_language\_programming();

}

C\_Expansion::C\_Expansion() : expansion\_file("Default"), C\_Library()/\*\*< Реалізація конструктора по замовчуванням\*/

{}

C\_Expansion::C\_Expansion(string expansion\_file, string dynamically, string name, const int ID, const int year\_creating, const int number\_of\_functions, string func, string lang) : expansion\_file(expansion\_file), C\_Library(dynamically, name, ID, year\_creating, number\_of\_functions, func, lang)/\*\*< Реалізація конструктора з параметрами\*/

{}

C\_Expansion::C\_Expansion(const C\_Expansion& elem) : expansion\_file(elem.expansion\_file),C\_Library(elem.dynamically,elem.name,elem.ID,elem.year\_creating,elem.number\_of\_functions,elem.function,elem.lang)/\*\*< Реалізація конструктора копіювання\*/

{}

C\_Expansion::~C\_Expansion()/\*\*< Реалізація деструктора \*/

{}

C\_Expansion Capabilities::getExpansion()

{

return this->obj;

}

void Capabilities::setExpansion(C\_Expansion obj)

{

this->obj = obj;

}

string Capabilities::getExpansion\_file()

{

return obj.getExpansion\_file();

}

void Capabilities::setExpansion\_file(string expansion)

{

obj.setExpansion\_file(expansion);

}

void Capabilities::setCapabilities(string capabilities)

{

this->capabilities = capabilities;

}

string Capabilities::getCapabilities()

{

return this->capabilities;

}

void Capabilities::setDynamically(string dynamically)

{

this->dynamically = dynamically;

}

void Capabilities::setName(string name)

{

this->name = name;

}

void Capabilities::setID(const int ID)

{

this->ID = ID;

}

void Capabilities::setYear\_Creating(const int year\_creating)

{

this->year\_creating = year\_creating;

}

void Capabilities::setNumber\_of\_function(const int number\_of\_function)

{

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

}

void Capabilities::setLanguage\_programming(string lang)

{

this->lang.setLanguage\_programming(lang);

}

void Capabilities::setFunction(string func)

{

this->function.setFunction(func);

}

string Capabilities::getDynamically()const

{

return this->dynamically;

}

string Capabilities::getName()const

{

return this->name;

}

int Capabilities::getID()const

{

return this->ID;

}

int Capabilities::getYear\_Creating()const

{

return this->year\_creating;

}

int Capabilities::getNumber\_of\_function()const

{

return this->number\_of\_functions;

}

string Capabilities::getFunction()

{

return this->function.What\_function\_is\_in\_this\_library();

}

string Capabilities::getLanguage\_programming()

{

return this->lang.Which\_language\_programming();

}

Capabilities::Capabilities():capabilities("the library can do something"),C\_Library(),obj()/\*\*< Реалізація конструктора по замовчуванням\*/

{}

Capabilities::Capabilities(string capabilities, string dynamically, string name, const int ID, const int year\_creating, const int number\_of\_functions, string func, string lang,C\_Expansion obj):capabilities(capabilities),C\_Library(dynamically,name,ID,year\_creating,number\_of\_functions, func,lang),obj(obj)/\*\*< Реалізація конструктора з параметрами\*/

{}

Capabilities::Capabilities(const Capabilities& el):capabilities(el.capabilities),C\_Library(el.dynamically,el.name,el.ID,el.year\_creating,el.number\_of\_functions,el.function,el.lang),obj(el.obj)/\*\*< Реалізація конструктора копіювання\*/

{}

Capabilities::~Capabilities()/\*\*< Реалізація деструктора \*/

{}

Файл C\_List.cpp

#include "C\_List.h" /\*\*< підключення файлу C\_List.h\*/

#include"C\_Library.h" /\*\*< підключення файлу C\_Library.h\*/

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

{

this->size = size;

}

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

{

return this->size;

}

/\*\*

\* Метод Read\_file

\*

\* Короткий опис методу:

\* <ul>

\* <li> Зчитуємо інформацію з файлу, потім предаємо інформацію до методу Distribution для подальших дій</li>

\* </ul>

\* @return повертає результат відкриття файлу

\*/

int C\_List::Read\_file(string file\_name,string file\_name2)

{

ifstream file(file\_name); /\*\*< відкриття файлу\*/

if (!file) /\*\*< перевірка чи відкрився файл\*/

{

cout << "Ошибка!!! Файл не открыто." << endl;

return 1;

}

ifstream file2(file\_name2); /\*\*< відкриття файлу\*/

if (!file) /\*\*< перевірка чи відкрився файл\*/

{

cout << "Ошибка!!! Файл не открыто." << endl;

return 1;

}

string line;

string line2;

for (size\_t i = 0; i < size; i++)

{

getline(file, line); /\*\*< зчитуємо рядок з інформацією \*/

getline(file2, line2); /\*\*< зчитуємо рядок з інформацією \*/

Distribution(line,line2,i); /\*\*< виклик методу Distribution\*/

}

file.close(); /\*\*< закриття файлу\*/

return 0;

}

/\*\*

\* Метод Count\_line

\*

\* Короткий опис методу:

\* <ul>

\* <li> За допомогою функції getline перевіряємо кількість рядків</li>

\* </ul>

\* @return повертає результат відкриття файлу

\*/

int C\_List::Count\_line(string file\_name)

{

size = 0; /\*\*< обнуляємо значення розміру масиву\*/

ifstream file(file\_name); /\*\*< відкриття файлу\*/

if (!file) /\*\*< перевірка чи відкрився файл\*/

{

cout << "Ошибка!!! Файл не открыто." << endl;

return 1;

}

string line;

while (getline(file, line)) /\*\*< зчитуємо кожний рядок, доки функція не натрапить на кінець\*/

{

size++; /\*\*< кожного разу збільшуємо значення \*/

}

file.close(); /\*\*< закриття файлу\*/

return 0;

}

/\*\*

\* Метод Distribution

\*

\* Короткий опис методу:

\* <ul>

\* <li> Отриманий рядок превіряємо за допомогою регулярного виразу а потім на базі інформації з рядка створюємо обьект </li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Distribution(string line ,string line2 ,int i)

{

regex regular("[\\d]\* [\\d]\* [\\d]\* (Yes|No) [A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*[ ]?([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)? [\\w][\_]?([\\w]\*)? C[\+]{0,} [.][\\w]\*"); /\*\*< основний регулярний вираз\*/

regex replace\_reg1("([;]{2,})"); /\*\*< оголошення регулярних виразів що допоможуть уникнути повторення деяких символів\*/

regex replace\_reg2("([\_]{2,})");

regex replace\_reg3("([-]{2,})");

regex replace\_reg4("([:]{2,})");

regex replace\_reg5("([\.]{2,})");

C\_Expansion obj\_e;

int id, number\_of\_function, year\_creating;

string dynamically, name, name2, line\_res;

string name\_f, name\_l, expansion,capabilities;

auto res = regex\_match(line, regular); /\*\*< перевірка рядку основним виразом\*/

if (res)

{

if (regex\_search(line, replace\_reg1)) /\*\*< якщо виявилося повторення заміняемо символи одним таким же символом\*/

{

line = regex\_replace(line, replace\_reg1, ";");

}

if (regex\_search(line, replace\_reg2))

{

line = regex\_replace(line, replace\_reg2, "\_");

}

if (regex\_search(line, replace\_reg3))

{

line = regex\_replace(line, replace\_reg3, "-");

}

if (regex\_search(line, replace\_reg4))

{

line = regex\_replace(line, replace\_reg4, ":");

}

if (regex\_search(line, replace\_reg5))

{

line = regex\_replace(line, replace\_reg5, ".");

}

istringstream str(line); /\*\*< преретворюємо рядок у потік з інформацією\*/

str >> id >> number\_of\_function >> year\_creating >> dynamically >> name >> name2>>name\_f>>name\_l>>expansion; /\*\*< зчитуємо інформацію у відповідні змінні\*/

if (expansion == "") /\*\*< якщо ім'я данної бібліотеки складається з одного слова то остання змінна залишається пустою \*/

{

obj\_e.setName(name);

obj\_e.setDynamically(dynamically);

obj\_e.setFunction(name2);

obj\_e.setLanguage\_programming(name\_f);

obj\_e.setID(id);

obj\_e.setNumber\_of\_function(number\_of\_function);

obj\_e.setYear\_Creating(year\_creating);

obj\_e.setExpansion\_file(name\_l);

Capabilities obj\_c(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e); /\*\*< за допомогою конструктору з параметрами створюємо обьект\*/

list[i] = obj\_c; /\*\*< та заносим цей обьект в масив\*/

}

else /\*\*< якщо ім'я данної бібліотеки складається з двух слів то відповідність ззмінних та інформації зберігається \*/

{

name = name + " " + name2; /\*\*< складаємо назву бібліотеки в одну змінну\*/

obj\_e.setName(name);

obj\_e.setDynamically(dynamically);

obj\_e.setFunction(name\_f);

obj\_e.setLanguage\_programming(name\_l);

obj\_e.setID(id);

obj\_e.setNumber\_of\_function(number\_of\_function);

obj\_e.setYear\_Creating(year\_creating);

obj\_e.setExpansion\_file(expansion);

Capabilities obj\_c(line2, dynamically, name, id, year\_creating, number\_of\_function, name\_f, name\_l, obj\_e); /\*\*< за допомогою конструктору з параметрами створюємо обьект\*/

list[i] = obj\_c; /\*\*< та заносим цей обьект в масив\*/

}

}

}

/\*\*

\* Метод Create

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод створює обьекти по замовчування, але для того щоб єлементи в масиві не збігалися використаємо різну варіацію заповнення обьектів за допомогою функції rand та змінної-індекса масива,</li>

\* <li> і відповідно до цього створюються обьекти з різними данними</li>

\* </ul>

\* @return пуста функція типу void

\*/

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

{

cout << "Происходит заполнеие масива данными" << endl;

int a = 0;

int x;

for (size\_t i = 0; i < size; i++) /\*\*< за допомогою циклу пройдемо всі комірки в масиві\*/

{

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

if (i % 2 == 0) /\*\*< залежно від змінної і значення змінної х буде різним, а сама зміннна відповідає за варіацію заповнення інформацією обьекта\*/

x = rand() % 1;

else

x = 2;

if (x == 0) /\*\*< варіант 1\*/

{

string dynamically = "yes"; /\*\*< створення змінних з інфрмацією\*/

string name = "math" + i;

string func = "Default";

string lang = "default";

string expansion = ".dll";

string capabilities = "using this library you can do something";

C\_Expansion new\_el\_e(expansion, dynamically, name, 3 + i, 1999, 20, func, lang);

Capabilities new\_el\_c(capabilities, dynamically, name, 3 + i, 1999, 20, func, lang,new\_el\_e); /\*\*< створення обьекта за допомогою конструктора з параметрами \*/

list[i] = new\_el\_c; /\*\*< занесення обьекту в масив\*/

}

else if (x == 1) /\*\*< варіант 2\*/

{

string dynamically = "no"; /\*\*< /\*\*< створення змінних з інфрмацією\*/

string name = "string" + i;

string func = "Default";

string lang = "default";

string expansion = ".a";

string capabilities = "using this library you can do something";

C\_Expansion new\_el\_e(expansion, dynamically, name, 46 + i, 1998, 10, func, lang);

Capabilities new\_el\_c(capabilities, dynamically, name, 46 + i, 1998, 10, func, lang,new\_el\_e); /\*\*< створення обьекта за допомогою конструктора з параметрами \*/

list[i] = new\_el\_c; /\*\*< занесення обьекту в масив\*/

}

else if (x == 2) /\*\*< варіант 3\*/

{

C\_Expansion new\_el\_e;

int id = new\_el\_e.getID();

new\_el\_e.setID(id + 14);

Capabilities new\_el\_c; /\*\*< створюємо обьект за допомогою конструктора за замовчуванням\*/

new\_el\_c.setID(id + 14); /\*\*< змінюємо значення ідентифікатору для того щоб вони не збігалися\*/

new\_el\_c.setExpansion(new\_el\_e);

list[i] = new\_el\_c; /\*\*< занесення обьекту в масив\*/

}

}

}

/\*\*

\* Метод Add

\*

\* Короткий опис методу:

\* <ul>

\* <li> В методі створюється новий тимчасовий масив, данні з старого масиву переписуються в новий масив разом з новим елементом, далі пам'ять зі старого масиву видаляється і знову виділяється знову переписується але вже навпаки та видаляється тимчасовий</li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Add(Capabilities obj\_c, const int order)

{

size++; /\*\*< збільшуємо розмір масиву\*/

Capabilities\* new\_arr = new Capabilities[size]; /\*\*< створюємо тимчасовий масив\*/

for (size\_t i = 0, j = 0; i < size; i++) /\*\*< преписуємо данні\*/

{

if (i != order - 1)

{

new\_arr[i] = list[j]; /\*\*< пререпис данних з одного масиву в інший\*/

j++;

}

else

{

new\_arr[i] = obj\_c; /\*\*< додавання нового елементу\*/

}

}

delete[] list; /\*\*< видаляємо масив\*/

list = new Capabilities[size]; /\*\*< виділяємо пам'ять для видаленого масиву\*/

for (size\_t i = 0; i < size; i++) /\*\*< пререписуємо данні\*/

{

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

}

delete[] new\_arr; /\*\*< видаляємо тимчасивий масив\*/

}

/\*\*

\* Метод Delete

\*

\* Короткий опис методу:

\* <ul>

\* <li> В методі створюється новий тимчасовий масив, данні з старого масиву переписуються в новий маси вравуючи індек елементу що видаляєть, далі пам'ять зі старого масиву видаляється і знову виділяється знову переписується але вже навпаки та видаляється тимчасовий</li>

\* </ul>

\* @return пуста функція типу void

\*/

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

{

size--; /\*\*< зменьшення розміру масиву\*/

Capabilities\* new\_arr = new Capabilities[size]; /\*\*< створюємо тимчасовий масив\*/

cout << "Елемент который вы удалили" << endl;

stringstream str = Str\_return(list[order - 1]); /\*\*< виводимо на екран елемент що видаляється\*/

Str\_output(str, 0);

for (size\_t i = 0; i < order - 1; i++) /\*\*< преписуємо данні\*/

{

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

}

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

{

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

}

delete[] list; /\*\*< видаляємо масив\*/

list = new Capabilities[size]; /\*\*< виділяємо пам'ять для видаленого масиву\*/

for (size\_t i = 0; i < size; i++) /\*\*< пререписуємо данні\*/

{

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

}

delete[] new\_arr; /\*\*< видаляємо тимчасивий масив\*/

}

/\*\*

\* Метод Index\_return

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод знаходить елемет в якого ідентифікатор співпадає з ідентифікатором що ввів користувач</li>

\* </ul>

\* @return повертає обьект класу Capabilities

\*/

Capabilities& C\_List::Index\_return(const int index)

{

int id;

for (size\_t i = 0; i < size; i++) /\*\*< проганяємо весь масив за допомогою циклу\*/

{

id = list[i].getID(); /\*\*< дізнаємося ідентифікатор кожного елементу\*/

if (id == index)/\*\*< порівнюємо значення\*/

{

id = i; /\*\*< якщо елемент знайдено, копіюємо індекс елементу\*/

i = size;/\*\*< збільшуємо індекс для того щоб вийти з масиву\*/

}

}

return list[id];

}

/\*\*

\* Метод Str\_return

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод перетворює обьект з данними в потік інформації</li>

\* </ul>

\* @return повертає потік з данними

\*/

stringstream C\_List::Str\_return(Capabilities obj)const

{

stringstream str;

str << " " << obj.getID() << " " << obj.getNumber\_of\_function() << " " << obj.getYear\_Creating() << " " << obj.getDynamically() << " " << obj.getName() << " " << obj.getFunction() << " " << obj.getLanguage\_programming() << " " << obj.getExpansion\_file();

return str;

}

/\*\*

\* Метод Str\_output

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод виводить на екран інформацію про обьект </li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Str\_output(stringstream& str, int i)const

{

cout.setf(std::ios::right);

int number\_of\_function, year\_creating, id;

string dynamically, name1, name2, name\_f, name\_l, expansion;

str >> id; /\*\*< з потоку зчитуємо інформацію в відповідні змінні\*/

str >> number\_of\_function;

str >> year\_creating;

str >> dynamically;

str >> name1;

str >> name2;

str >> name\_f;

str >> name\_l;

str >> expansion;

if (expansion == "") /\*\*< якщо ім'я данної бібліотеки складається з одного слова то остання змінна залишається пустою \*/

cout << i + 1 << "- " << setw(10) << id << setw(13) << number\_of\_function << setw(21) << year\_creating << setw(20) << dynamically << setw(30) << name1 << setw(26) << name2 << setw(24) << name\_f << setw(20) << name\_l<<endl; /\*\*< виведення на екран\*/

else /\*\*< якщо ім'я данної бібліотеки складається з двух слів то відповідність ззмінних та інформації зберігається \*/

{

name1 = name1 + " " + name2; /\*\*< складаємо назву бібліотеки в одну змінну\*/

cout << i + 1 << "- " << setw(10) << id << setw(13) << number\_of\_function << setw(21) << year\_creating << setw(20) << dynamically << setw(30) << name1 << setw(26) << name\_f << setw(24) << name\_l << setw(20) << expansion<<endl; /\*\*< виведення на екран\*/

}

}

/\*\*

\* Метод Output

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод виводить масив на екран</li>

\* </ul>

\* @return пуста функція типу void

\*/

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

{

stringstream str;

cout << "Вивод на экран всех объектов" << endl;

cout << "№" << setw(17) << "id библиотеки" << setw(16) << "Кол-во функций" << setw(18) << "Год её создания" << setw(30) << "Динмически ли она подключена" << setw(20) << "Название библиотеки" << setw(32)<<"Одна из функций этой библиотеки"<<setw(21)<<"Язык програмирования"<< setw(29) << "Расширение файла библиоткеки" << endl;

for (size\_t i = 0; i < size; i++) /\*\*< проганяємо весь масив за допомогою циклу\*/

{

str = Str\_return(list[i]); /\*\*< викликаємо метод Str\_return\*/

Str\_output(str, i); /\*\*< викликаємо метод Str\_output\*/

}

}

/\*\*

\* Метод Sort

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод сортує масив відповідно до аргументу</li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Sort(func condition)

{

Capabilities temp; /\*\*< створення тимчасової змінної\*/

for (size\_t i = 0; i < size; i++) /\*\*< проганяємо весь масив за допомогою циклу\*/

{

for (size\_t j = 0; j < size; j++)

{

if (condition(list[i].getID(), list[j].getID())) /\*\*< перевіряємо відповідність елементів, якщо вона вірна то міняємо елементі місцями\*/

{

temp = list[i];

list[i] = list[j]; ;

list[j] = temp;

}

}

}

}

/\*\*

\* Метод Difference

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод визначає кількість кібліотек що динамічно підключаються та порівнює в скільки разів вона меньше ніж загальна кількість бібліотек в масиві</li>

\* </ul>

\* @return повертає різницю між кількістю бібліотек що динамічно підключаються та загальною іх кількість

\*/

float C\_List::Difference()

{

float count = 0;

string y;

for (size\_t i = 0; i < size; i++) /\*\*< проганяємо весь масив за допомогою циклу\*/

{

y = list[i].getDynamically();

if (y == "yes") /\*\*< перевірка чи динамічно підлючається бібліотека, якщо так збільшуємо значення змінної\*/

count++;

}

float dif = size / count; /\*\*< визначаємо частку від ділення \*/

cout << "В " << setprecision(5) << dif << " раз количество библиотек, которые динамически подключаются, меньше чем общее количество библиотек" << endl;/\*\*< повідомляємо відмінність виводячи інформацію на екран\*/

return dif;

}

/\*\*

\* Метод Info\_about\_lib

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод виводить на екран всю інформацію про бібліотеки</li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Info\_about\_lib()

{

for (size\_t i = 0; i < size; i++)

{

cout << endl << "В языке програмирования \"";

cout << list[i].getLanguage\_programming();

cout << "\" используется библиотека " << list[i].getName() << endl;

cout << "В данной библиотеке есть такая функция" << endl;

cout << "Функция - " << list[i].getFunction() << endl;

cout << "Возможности данной библиотеки" << endl << list[i].getCapabilities()<<endl;

}

}

/\*\*

\* Метод Write\_file

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод записує данні у файл</li>

\* </ul>

\* @return результат відкриття файлу

\*/

int C\_List::Write\_file(string file\_name )

{

ofstream file(file\_name); /\*\*< відкриття файлу\*/

if (!file) /\*\*< перевірка чи відкрився файл\*/

{

cout << "Ошибка!!! Файл не открыто." << endl;

return 1;

}

file << "№" << setw(17) << "id библиотеки" << setw(16) << "Кол-во функций" << setw(18) << "Год её создания" << setw(30) << "Динмически ли она подключена" << setw(20) << "Название библиотеки" << setw(32) << "Одна из функций этой библиотеки" << setw(21) << "Язык програмирования" << setw(29) << "Расширение файла библиоткеки" << endl;

for (size\_t i = 0; i < size; i++) /\*\*< записуємо інформацію у файл за допомогою циклу\*/

file << i + 1 << "- " << setw(10) << list[i].getID() << setw(13) << list[i].getNumber\_of\_function() << setw(21) << list[i].getYear\_Creating() << setw(20) << list[i].getDynamically() << setw(30) << list[i].getName() << setw(26) << list[i].getFunction() << setw(24) << list[i].getLanguage\_programming() << setw(20) << endl << list[i].getCapabilities() << endl;

file.close(); /\*\*< закриття файлу\*/

return 1;

}

/\*\*

\* Метод Check

\*

\* Короткий опис методу:

\* <ul>

\* <li> Метод виводить на екран елементи масиву в назві яких є 2 слова</li>

\* </ul>

\* @return пуста функція типу void

\*/

void C\_List::Check( )

{

regex regular("(([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)+( )+([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*))"); /\*\*< створення регулярного виразу для визначення назви з двома словами\*/

string check;

stringstream str;

int k = 0;

for (size\_t i = 0; i < size; i++) /\*\*< проганяємо весь масив за допомогою циклу\*/

{

auto check\_res = list[i].getName();

if (regex\_match(check\_res, regular)) /\*\*< преревіряємо чи містить назва 2 слова\*/

{

str = Str\_return(list[i]);

Str\_output(str, k); /\*\*< якщо відповідність знайдена виводимо на екран данний елемент\*/

k++;

}

}

}

C\_List:: ~C\_List()/\*\*< Реалізація деструктора \*/

{

delete[] list;

}

Файл Test.cpp

#include "C\_Library.h" /\*\*< підключення файлу C\_Library.h\*/

#include "C\_List.h" /\*\*< підключення файлу C\_List.h\*/

#include "Test.h" /\*\*< підключення файлу Test.h\*/

/\*\*

\* Головна функція main

\*

\* Опис дій:

\* <ul>

\* <li> Додавання російської мови в програму для діалога між програмою та користувачем</li>

\* <li> Виклик функції котрі перевіряють роботу методів</li>

\* <li> Превірка витоків пам'яті</li>

\* </ul>

\* @return повертає результат роботи програми

\*/

int main()

{

setlocale(LC\_ALL, "Rus");

bool result;

C\_List list; /\*\*< створення обьекта класу C\_List\*/

list.list = new Capabilities[3]; /\*\*< виділення пам'яті для масиву\*/

list.setSize\_array(3);

result = Test\_Add(list); /\*\*< виклик функціі-тесту Test\_Add\*/

if (result) /\*\*< перевірка тесту\*/

cout << "Test\_Add пройден успешно!" << endl;

else

cout << "Test\_Add не пройден!" << endl;

result = Test\_Delete(list); /\*\*< виклик функціі-тесту Test\_Delete\*/

if (result) /\*\*< перевірка тесту\*/

cout << "Test\_Delete пройден успешно!" << endl;

else

cout << "Test\_Delete не пройден!" << endl;

result = Test\_Index\_return(list);/\*\*< виклик функціі-тесту Test\_Index\_return\*/

if (result)/\*\*< перевірка тесту\*/

cout << "Test\_Index\_return пройден успешно!" << endl;

else

cout << "Test\_Index\_return не пройден!" << endl;

result = Test\_Difference(list);/\*\*< виклик функціі-тесту Test\_Difference\*/

if (result)/\*\*< перевірка тесту\*/

cout << "Test\_Difference пройден успешно!" << endl;

else

cout << "Test\_Difference не пройден!" << endl;

result = Test\_Sort(list);/\*\*< виклик функціі-тесту Test\_Sort\*/

if (result)/\*\*< перевірка тесту\*/

cout << "Test\_Sort пройден успешно!" << endl;

else

cout << "Test\_Sort не пройден!" << endl;

return 0;

}

Файл Test\_method.cpp

#include "C\_Library.h" /\*\*< підключення файлу C\_Library.h\*/

#include "C\_List.h" /\*\*< підключення файлу C\_List.h\*/

#include "Test.h"/\*\*< підключення файлу Test.h\*/

bool A\_more(int a, int b) { return a > b; }

bool B\_more(int a, int b) { return b > a; }

/\*\*

\* Функція Test\_Add

\*

\* Короткий опис дій:

\* <ul>

\* <li> Перевіряє метод додавання нового елементу методом перевірки розміру масиву після виконання дії </li>

\* </ul>

\* @return повертає результат виконання тесту

\*/

bool Test\_Add(C\_List& list)

{

Capabilities new\_lib; /\*\*< створення обьекту класу Capabilities\*/

list.Add(new\_lib, 4); /\*\*< виклик методу Add\*/

int size = list.getSize\_array(); /\*\*< отримаємо значення розміру масиву\*/

if (size == 4) /\*\*< перевірка чи збігається розмір з очікуваним результатом\*/

return true;

else

return false;

}

/\*\*

\* Функція Test\_Delete

\*

\* Короткий опис дій:

\* <ul>

\* <li> Перевіряє метод видалення елементу методом перевірки розміру масиву після виконання дії</li>

\* </ul>

\* @return повертає результат виконання тесту

\*/

bool Test\_Delete(C\_List& list)

{

list.Delete(1); /\*\*< виклик методу Delete\*/

int size = list.getSize\_array(); /\*\*< отримаємо значення розміру масиву\*/

if (size == 3) /\*\*< перевірка чи збігається розмір з очікуваним результатом\*/

return true;

else

return false;

}

/\*\*

\* Функція Test\_Index\_return

\*

\* Короткий опис дій:

\* <ul>

\* <li> В методі створюється обьект котрий вставляється в масив, далі визивається метод повернення по індексу що повертає нібито очікуваний обьект котрий зг0дом перевіряється</li>

\* </ul>

\* @return повертає результат виконання тесту

\*/

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

{

string dynamically = "yes"; /\*\*< створення змінних з інформацією\*/

string name = "math";

string func = "sqrt";

string lang = "C";

string expansion = ".dll";

string capabilities = "using this library you can do something";

C\_Expansion new\_el\_e(expansion, dynamically, name, 3 , 1999, 20, func, lang);

Capabilities new\_el\_c(capabilities, dynamically, name, 3 , 1999, 20, func, lang, new\_el\_e); /\*\*< створення обьекту на базі цієї інформації\*/

list.list[1] = new\_el\_c; /\*\*< занесення обьекту в масив\*/

dynamically = "", name = "", func = "", lang = "", expansion = "", capabilities = ""; /\*\*< обнуляємо змінні\*/

Capabilities return\_lib = list.Index\_return(3); /\*\*< виклик методу Index\_return\*/

dynamically = return\_lib.getDynamically(); /\*\*< зчитуємоінформацію з обьекту\*/

name = return\_lib.getName();

func = return\_lib.getFunction();

lang = return\_lib.getLanguage\_programming();

expansion = return\_lib.getExpansion\_file();

capabilities = return\_lib.getCapabilities();

int count = 0;

if (dynamically == "yes") /\*\*< перевіряємо всі поля на відповідність\*/

count++;

if (name == "math")

count++;

if (func == "sqrt")

count++;

if (lang == "C")

count++;

if (expansion == ".dll")

count++;

if (capabilities == "using this library you can do something")

count++;

if (return\_lib.getID() == 3)

count++;

if (return\_lib.getNumber\_of\_function() == 20)

count++;

if (return\_lib.getYear\_Creating() == 1999)

count++;

if (count == 9) /\*\*< перевіряємо чи дорівнює кількість вірних полів з загальною кількість \*/

return true;

else

return false;

}

/\*\*

\* Функція Test\_Difference

\*

\* Короткий опис дій:

\* <ul>

\* <li> В методі створюєть 1 обьект який не підключається динамічно,далі викликається метод Difference і превіряється результат</li>

\* </ul>

\* @return повертає результат виконання тесту

\*/

bool Test\_Difference(C\_List& list)

{

string dynamically = "no";

Capabilities new\_el\_c; /\*\*< створення обьекту класу Capabilities\*/

new\_el\_c.setDynamically(dynamically); /\*\*< замінюємо значення поля dynamically для данного обьекту (по замовчуванню завжди "yes")\*/

list.list[1] = new\_el\_c; /\*\*< вставляємо обьект в масив\*/

float diff = list.Difference(); /\*\*< виклик методу Difference\*/

if (diff == 1.5) /\*\*< перевіряємо чи збігається очікуваний результат з отриманим\*/

return true;

else

return false;

}

/\*\*

\* Функція Test\_Sort

\*

\* Короткий опис дій:

\* <ul>

\* <li> </li>

\* </ul>

\* @return повертає результат виконання тесту

\*/

bool Test\_Sort(C\_List& list)

{

int count=0;

Capabilities new\_el\_1; /\*\*< створюємо 3 обьекта класу Capabilities\*/

Capabilities new\_el\_2;

Capabilities new\_el\_3;

new\_el\_1.setID(1); /\*\*< присваюємо цим обьектам ідентифікатори \*/

new\_el\_2.setID(2);

new\_el\_3.setID(3);

list.list[0] = new\_el\_1; /\*\*< заносимо обьекти до масиву\*/

list.list[1] = new\_el\_2;

list.list[2] = new\_el\_3;

list.Sort(A\_more); /\*\*< виклик методу Sort де число А більше\*/

int id\_1 = list.list[0].getID(); /\*\*< помістимо ідентифікатори в відповідні змінні\*/

int id\_2 = list.list[1].getID();

int id\_3 = list.list[2].getID();

if (id\_1 == 3 && id\_2 == 2 && id\_3 == 1) /\*\*< перевіримо чи відбулося сортування\*/

count++;

list.Sort(B\_more); /\*\*< виклик методу Sort де число В більше\*/

id\_1 = list.list[0].getID(); /\*\*< знову помістимо ідентифікатори в відповідні змінні\*/

id\_2 = list.list[1].getID();

id\_3 = list.list[2].getID();

if (id\_1 == 1 && id\_2 == 2 && id\_3 == 3) /\*\*< перевіримо чи відбулося сортування\*/

count++;

if (count == 2)/\*\*< перевіримо чи збігається результати тестів з очікуваним результатом\*/

return true;

else

return false;

}

Файл Main.cpp

#include "C\_List.h"

#include"C\_Library.h"

typedef bool (func)(int, int);

bool A\_more(int a, int b) { return a > b; }

bool B\_more(int a, int b) { return b > a; }

int Interface(); /\*\*< Оголошення функції де проходять основні дії програми (зроблена вона для того щоб перевіряти витоки пам'яті)\*/

/\*\*

\* Головна функція main

\*

\* Опис дій:

\* <ul>

\* <li> Додавання російської мови в програму для діалога між програмою та користувачем</li>

\* <li> Виклик функції де проходять основні дії програми</li>

\* <li> Превірка витоків пам'яті</li>

\* </ul>

\* @return повертає результат роботи програми

\*/

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Rus");

int res = Interface();

if (res == 0)

return 0;

if (\_CrtDumpMemoryLeaks())

cout << endl << "Есть утечка памяти" << endl;

else

cout << endl << "Нет утечки памяти" << endl;

return 0;

}

/\*\*

\* Функція Interface

\*

\* Короткий опис дій:

\* <ul>

\* <li> Створення обьекту класу С\_List та проходження всіх його методів</li>

\* </ul>

\* @return пуста функція типу void

\*/

int Interface()

{

C\_List list\_lib; /\*\*< створення обьекта класу C\_List\*/

cout << "Cоздание массива" << endl;

int command;

cout << "Введите вариант заполнения массива 1- из файла 2-конструктором" << endl;

cin >> command; /\*\*< визначення варіації заповнення масиву інформацією\*/

string file\_read = "file\_read.txt"; /\*\*< створення змінних з назвою файлів\*/

string file\_read2 = "file\_read2.txt";

string file\_write = "file\_write.txt";

int res;

if (command == 1) /\*\*< якщо користувач обрав метод заповнення з файлу\*/

{

res = list\_lib.Count\_line(file\_read); /\*\*< визначення кількості рядків \*/

if (res == 1) /\*\*< преревірка результату роботи методу\*/

return 1;

int size = list\_lib.getSize\_array();

list\_lib.list = new Capabilities[size]; /\*\*< виділення пам'яті \*/

res = list\_lib.Read\_file(file\_read,file\_read2); /\*\*< виклик методу Read\_file\*/

if (res == 1) /\*\*< преревірка результату роботи методу\*/

return 1;

}

else /\*\*< якщо користувач обрав метод заповнення готовими данними\*/

{

int size;

cout << "Введите размер массива" << endl;

cin >> size; /\*\*< визначення розміру масиву (значення вводить користувач)\*/

list\_lib.setSize\_array(size);

list\_lib.list = new Capabilities[size]; /\*\*< виділення пам'яті \*/

list\_lib.Create(); /\*\*< виклик методу Create\*/

}

list\_lib.Output(); /\*\*< виклик методу Output\*/

int order = 0;

cout << "Введите порядковый номер который вы хотите присвоить новому элементу" << endl;

cin >> order; /\*\*< визначення індексу куди потрібно помістити новий елемент\*/

Capabilities new\_lib; /\*\*< створення нового обьекту\*/

cout << "Добавления нового элемента" << endl;

list\_lib.Add(new\_lib, order); /\*\*< виклик методу\*/

list\_lib.Output(); /\*\*< виклик методу Output\*/

cout << "Удаления элемента" << endl;

cout << "Введите номер элемента которого вы хотите удалить" << endl;

cin >> order; /\*\*< визначення індексу елемента що потрібно видалити\*/

list\_lib.Delete(order); /\*\*< виклик методу\*/

list\_lib.Output(); /\*\*< виклик методу Output\*/

cout << "Введите идентификатор елемента данные которого вы хотите получить " << endl;

cin >> order; /\*\*< визначення идентифікатора елемента що потрібно отримати\*/

Capabilities index\_lib = list\_lib.Index\_return(order); /\*\*< виклик методу Index\_return\*/

cout << "Вывод полученого элемента на екран" << endl;

stringstream str = list\_lib.Str\_return(index\_lib); /\*\*< виводимо на екран обьект що отримали\*/

list\_lib.Str\_output(str, 0);

list\_lib.Sort(A\_more); /\*\*< виклик методу Sort де число А більше\*/

list\_lib.Output(); /\*\*< виклик методу Output\*/

list\_lib.Sort(B\_more); /\*\*< виклик методу Sort де число В більше\*/

list\_lib.Output(); /\*\*< виклик методу Output\*/

cout << "Вывод на экран библиотек, в которых в названии 2 слова " << endl;

list\_lib.Check(); /\*\*< \*/

float diff = list\_lib.Difference(); /\*\*< виклик методу Difference\*/

cout << "Вывод на екран полной информации про каждую библиотеку" << endl;

list\_lib.Info\_about\_lib(); /\*\*< виклик методу Info\_about\_lib\*/

res = list\_lib.Write\_file(file\_write ); /\*\*< виклик методу Write\_file\*/

if (res == 1)

return 1;

return 0;

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

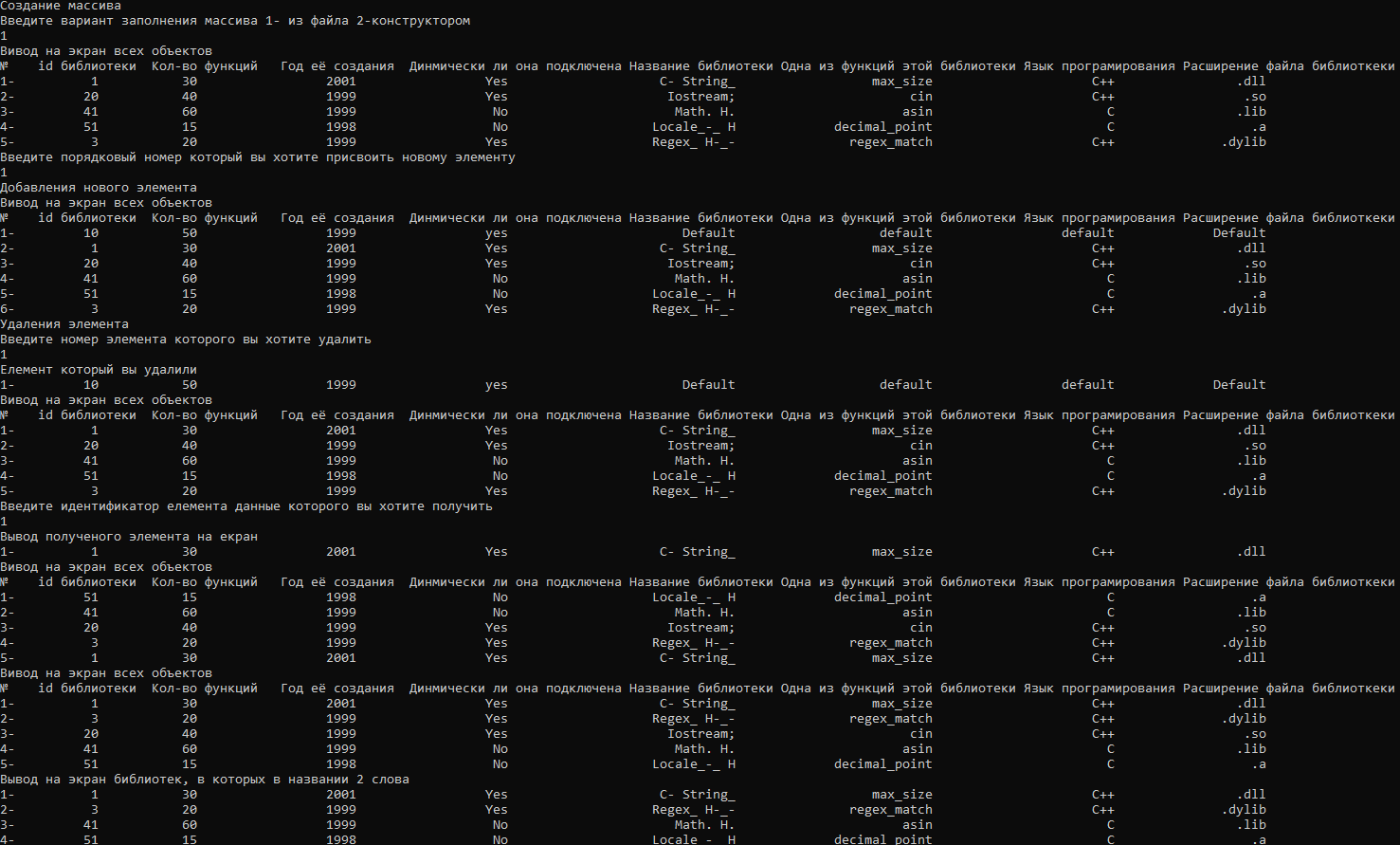


Рисунок. 1 - Результат роботи основоного завдання

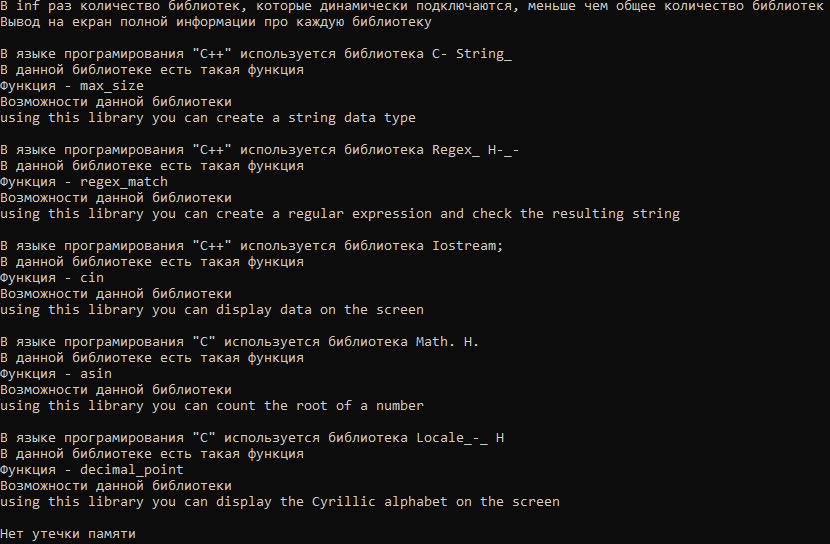


Рисунок. 2 - Результат роботи основоного завдання

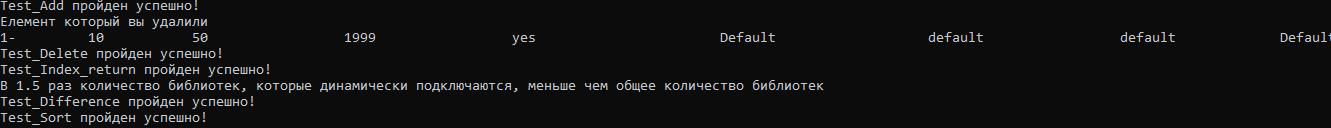


Рисунок. 3 - Результат тестування методів

ВИСНОВКИ

Отримали знання про парадигму ООП – поліморфізм; навчитися застосовувати отримані знання на практиці.