Автор: Наймитенко С. КІТ-119а

Дата: 20 травня 2020

# **Лабораторна робота №14**

**СОРТУВАННЯ**

Мета роботи: Алгоритми зміни послідовності. Сортування. Функтори.

1 ЗАВДАННЯ ДО РОБОТИ

Поширити попередню лабораторну роботу, додаючи такі можливості діалогового меню:

- об’єднання двох STL-контейнерів типу vector;

- сортувати заданий контейнер з використанням функтора.

Додаткове завдання на оцінку «відмінно»

Додати можливість об’єднання двох STL-контейнерів типу map. При цьому, якщо в обох контейнерах існують однакові ключі, то значення повинні конкатенуватися, наприклад, якщо є дві мапи для країн:

- Мапа1:

- Україна : Харків, Київ;

- Росія: Москва, Бєлгород;

- Білорусь: Мінськ, Бобруйськ.

- Мапа2:

- Польща: Варшава;

- Росія: Санкт-Петербург;

- Україна: Харків, Запоріжжя;

то об’єднана мапа повинна містити таке:

- Україна: Харків, Київ, Запоріжжя;

- Росія: Москва, Бєлгород, Санкт-Петербург;

- Білорусь: Мінськ, Бобруйськ;

- Польща: Варшава.

**Опис змінних**

vector<Capabilities> conteiner – контейнер STL vector;

list<Capabilities> conteiner – контейнер STL list;

set<Capabilities> conteiner – контейнер STL set;

map<Capabilities> conteiner – контейнер STL map;

Capabilities res\_id – об’єкт шо повернувся;

Опис функцій

T-один із типів контейнерів

T<Capabilities> Read\_file(string file\_name, string file\_name2, int size);

T <Capabilities> Distribution(string line, string line2, int i, T <Capabilities>);

T <Capabilities> Add(T <Capabilities>);

T <Capabilities> Delete(T <Capabilities>);

Capabilities Index\_return(const int index, T <Capabilities>);

void Output(vector<Capabilities>);

float Difference(T <Capabilities>);

void Info\_about\_lib(T <Capabilities>);

int Write\_file(T <Capabilities>, string file\_name);

void Check(T <Capabilities>);

**Текст програми**

Файл C\_Language.h

#pragma once

#include "C\_Library.h"

using std::string;

using std::istream;

using std::ostream;

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()const;

friend istream& operator>> (istream& input, C\_Language& obj);

friend ostream& operator<< (ostream& output, const C\_Language& obj);

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

/\*\*

\* @file C\_Library.h

\* Оголошення класу C\_Library класу що зовбражуе відношення композиція - C\_Function та класів спадкоемців C\_Expansion та Capabilities

\* @author Naimytenko Serhiy

\* @version 1.0

\* @date 2020.04.05

\*/

#pragma once

#define CRTDBG\_MAP\_ALLOC

#include <crtdbg.h>

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

#include <iostream>

#include <clocale>

#include <string>

#include <ctime>

#include <iomanip>

#include <fstream>

#include <sstream>

#include <regex>

#include <Windows.h>

#include <stddef.h>

#include <cstddef>

#include "C\_Language.h"

#include <vector>

#include <list>

#include <set>

#include <map>

using std::vector;

using std::list;

using std::set;

using std::map;

using std::pair;

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;

using std::ostream;

using std::istream;

using std::sort;

typedef bool (func)(int, int);

bool A\_more(int a, int b);

bool B\_more(int a, int b);

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()const;

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()const = 0;

virtual string getLanguage\_programming()const = 0;

virtual string getInfo() const = 0;

virtual void input(istream&) = 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\*/

virtual stringstream getSsObj() const = 0;

friend ostream& operator<< (ostream&, const C\_Library&);

friend ofstream& operator<< (ofstream&, const C\_Library&);

virtual bool operator==(const int) const;

C\_Library& operator= (C\_Library&);

friend istream& operator>> (istream&, C\_Library&);

};

class C\_Expansion final : 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()const;

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()const override;

string getLanguage\_programming()const 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\*/

string getInfo() const override final;

stringstream getSsObj() const override final;

C\_Expansion& operator= (C\_Expansion& temp);

void input(istream& input) override final;

bool operator==(const int) const override final;

};

class Capabilities final : 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()const;

void setExpansion\_file(string);

void setCapabilities(string);

string getCapabilities()const;

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()const override;

string getLanguage\_programming()const override;

string getInfo() const override final;

stringstream getSsObj() const override final;

Capabilities& operator= (Capabilities& temp);

void input(istream& input) override final;

bool operator==(const int) const override final;

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

bool operator<(Capabilities const& rhs) const

{

// Use whatever makes sense for your application.

return (ID < rhs.ID);

}

operator Capabilities()const

{

Capabilities obj;

return obj;

}

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 Conteiner.h

#pragma once

#include"C\_Library.h"

int List();

int Vector();

int Set();

int Map();

int Count\_line(string file\_name);

stringstream Str\_return(Capabilities obj);

void Str\_output(stringstream& str, int i);

Файл Test.h

#pragma once

#include"C\_Library.h"

bool Testvector();

bool Testlist();

bool Testset();

bool Testmap();

Файл C\_Language.cpp

/\*\*

\* @file C\_Language.cpp

\* Реалізація класу C\_Language

\* @author Naimytenko Serhiy

\* @version 1.0

\* @date 2020.04.05

\*/

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

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

{

language = name;

}

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

{

return language;

}

istream& operator>> (istream& input, C\_Language& obj)

{

input >> obj.language;

return input;

}

ostream& operator<< (ostream& output, const C\_Language& obj)

{

output << obj.language;

return output;

}

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)

{

function = name;

}

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

{

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()const

{

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

}

string C\_Library::getLanguage\_programming()const

{

return lang.Which\_language\_programming();

}

ostream& operator<< (ostream& output, const C\_Library& obj)

{

output << obj.getInfo();

return output;

}

ofstream& operator<< (ofstream& output, const C\_Library& obj)

{

output << obj.getInfo();

return output;

}

istream& operator>> (istream& input, C\_Library& obj)

{

obj.input(input);

return input;

}

bool C\_Library::operator==(const int ID) const

{

return this->ID == ID;

}

C\_Library& C\_Library::operator= (C\_Library& temp)

{

if (this == &temp)

return \*this;

this->ID = temp.ID;

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

this->year\_creating = temp.year\_creating;

this->dynamically = temp.dynamically;

this->function = temp.function;

this->lang = temp.lang;

this->name = temp.name;

return \*this;

}

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()const

{

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()const

{

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

}

string C\_Expansion::getLanguage\_programming()const

{

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

}

stringstream C\_Expansion::getSsObj() const

{

stringstream temp;

temp << ID << " " << year\_creating << " " << number\_of\_functions << " " << dynamically << " " << name << " " << getFunction() << " " << lang << " " << expansion\_file;

return temp;

}

string C\_Expansion::getInfo() const

{

stringstream temp;

temp << setw(13) << ID << setw(10) << number\_of\_functions << setw(10) << year\_creating << setw(13) << dynamically << setw(9) << name << setw(10) << getFunction() << setw(15) << lang << expansion\_file;

return temp.str();

}

C\_Expansion& C\_Expansion::operator= (C\_Expansion& temp)

{

if (this == &temp)

return \*this;

this->C\_Library::operator=(temp);

this->expansion\_file = temp.expansion\_file;

return \*this;

}

bool C\_Expansion::operator==(const int ID) const

{

return this->ID == ID;

}

void C\_Expansion::input(istream& input)

{

string function;

input >> ID >> number\_of\_functions >> year\_creating >> dynamically >> name >> function >> lang >> expansion\_file;

C\_Function func(function);

}

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()const

{

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()const

{

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()const

{

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

}

string Capabilities::getLanguage\_programming()const

{

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

}

stringstream Capabilities::getSsObj()const

{

stringstream temp;

temp << ID << " " << year\_creating << " " << number\_of\_functions << " " << dynamically << " " << name << " " << getFunction() << " " << lang << " " << capabilities;

return temp;

}

string Capabilities::getInfo() const

{

stringstream temp;

temp.setf(std::ios::left);

temp << setw(13) << ID << setw(10) << number\_of\_functions << setw(10) << year\_creating << setw(13) << dynamically << setw(9) << name << setw(10) << getFunction() << setw(15) << lang << capabilities;

return temp.str();

}

Capabilities& Capabilities::operator= (Capabilities& temp)

{

if (this == &temp)

return \*this;

this->C\_Library::operator=(temp);

this->obj = temp.obj;

this->capabilities = temp.capabilities;

return \*this;

}

bool Capabilities::operator==(const int ID) const

{

return this->ID == ID;

}

void Capabilities::input(istream& input)

{

string function;

input >> ID >> number\_of\_functions >> year\_creating >> dynamically >> name >> function >> lang >> capabilities;

C\_Function func(function);

}

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 List.cpp

#include "Conteiner.h"

list<Capabilities> Read\_file(string file\_name, string file\_name2, int size);

list<Capabilities> Distribution(string line, string line2, int i, list<Capabilities>);

list<Capabilities> Add(list<Capabilities>);

list<Capabilities> Delete(list<Capabilities>);

Capabilities Index\_return(const int index, list<Capabilities>);

void Output(list<Capabilities>);

float Difference(list<Capabilities>);

void Info\_about\_lib(list<Capabilities>);

int Write\_file(list<Capabilities>, string file\_name);

void Check(list<Capabilities>);

void Criterion(list<Capabilities>);

void Search(list<Capabilities>);

int List()

{

list<Capabilities> conteiner;

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

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

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

int size = Count\_line(file\_read);

if (size == 0)

return 404;

conteiner = Read\_file(file\_read, file\_read2, size);

if (conteiner.size() == 0)

return 404;

Output(conteiner);

conteiner = Add(conteiner);

Output(conteiner);

conteiner = Delete(conteiner);

Output(conteiner);

float diff = Difference(conteiner);

int id;

cout << "Введите айди элемента которого хотите получить" << endl;

cin >> id;

Capabilities res\_id = Index\_return(id, conteiner);

Str\_return(res\_id);

cout << "Вывод объектов в названии которых 2 слова" << endl;

Check(conteiner);

Info\_about\_lib(conteiner);

Criterion(conteiner);

Search(conteiner);

int res = Write\_file(conteiner, file\_write);

if (res == 1)

return 404;

conteiner.clear();

return 0;

}

list<Capabilities> Read\_file(string file\_name, string file\_name2, int size)

{

list<Capabilities> cont;

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

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

{

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

return cont;

}

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

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

{

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

return cont;

}

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 \*/

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

}

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

return cont;

}

list<Capabilities> Distribution(string line, string line2, int i, list<Capabilities> cont)

{

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);

cont.push\_back(Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

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);

cont.push\_back(Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

}

return cont;

}

list<Capabilities> Add(list<Capabilities> cont)

{

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

auto it = cont.begin();

advance(it, index\_el - 1);

Capabilities add\_el;

cont.insert(it, add\_el);

return cont;

}

list<Capabilities> Delete(list<Capabilities> cont)

{

auto it = cont.begin();

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

advance(it, index\_el - 1);

cont.erase(it);

return cont;

}

Capabilities Index\_return(const int index, list<Capabilities> cont)

{

int id;

Capabilities value;

for (auto var : cont)

{

id = var.getID(); /\*\*< дізнаємося ідентифікатор кожного елементу\*/

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

{

value = var;

}

}

return value;

}

void Output(list<Capabilities> cont)

{

stringstream str;

int i = 0;

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\_each(cont.begin(), cont.end(), Str\_return);

}

float Difference(list<Capabilities> cont)

{

float count = 0;

string y;

for (auto var : cont)

{

y = var.getDynamically();

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

count++;

}

float dif = cont.size() / count; /\*\*< визначаємо частку від ділення \*/

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

return dif;

}

void Info\_about\_lib(list<Capabilities> cont)

{

for (auto var : cont)

{

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

cout << var.getLanguage\_programming();

cout << "\" используется библиотека " << var.getName() << endl;

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

cout << "Функция - " << var.getFunction() << endl;

cout << "Возможности данной библиотеки" << endl << var.getCapabilities() << endl;

}

}

int Write\_file(list<Capabilities> cont, string file\_name)

{

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

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

{

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

return 1;

}

int i = 0;

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 (auto var : cont)

{

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

i++;

}

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

return 0;

}

void Check(list<Capabilities> cont)

{

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 (auto var : cont)

{

auto check\_res = var.getName();

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

{

Str\_return(var);

}

}

}

void Criterion(list<Capabilities> cont)

{

int command, count;

bool flag = true;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << "404 - завершение поиска количества по критериям" << endl;

cin >> command;

while (flag)

{

count = 0;

switch (command)

{

case 1:

{

regex regular("(([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)+( )+([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*))");

for (auto var : cont)

{

auto check\_res = var.getName();

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

{

count++;

}

}

cout << "Количество бииблиотек, в названии которых 2 слова - " << count << endl;

break;

}

case 2:

{

int year;

cout << "Введите год создания" << endl;

cin >> year;

for (auto var : cont)

{

if (var.getYear\_Creating() == year) /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотеки" << year << " года создания - " << count << endl;

break;

}

case 3:

{

cout << "1 - язык С, 2 - С++" << endl;

cin >> command;

if (command == 1)

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С - " << count << endl;

}

else

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C++") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С++ - " << count << endl;

}

break;

}

case 4:

{

for (auto var : cont)

{

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

count++;

}

cout << "Количество динамически подключаемых библиотек - " << count << endl;

break;

}

case 404:

{

flag = false;

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

break;

}

}

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

}

}

void Search(list<Capabilities> cont)

{

int command;

bool flag = true;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl << "3 - Расширение файла библиотеки" << endl;

cin >> command;

while (flag)

{

switch (command)

{

case 1:

{

string name;

cout << "Введите имя искомого объекта" << endl;

cin >> name;

for (auto var : cont)

{

if (var.getName() == name)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 2:

{

int id;

cout << "Введите ID искомого объекта" << endl;

cin >> id;

for (auto var : cont)

{

if (var.getID() == id)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 3:

{

string exp;

cout << "Введите расширение искомого объекта" << endl;

cin >> exp;

for (auto var : cont)

{

if (var.getExpansion\_file() == exp)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl;

cin >> command;

break;

}

}

}

}

Файл Main.cpp

#include "Conteiner.h"

bool A\_more(int a, int b)

{

return a > b;

}

bool B\_more(int a, int b)

{

return b > a;

}

int Interface();

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Rus");

int res = Interface();

if (res == 1)

return 1;

return 0;

}

int Interface()

{

int command;

cout << "Введите команду котора соответствует контейнеру" << endl << "1 - vector " << endl << "2 - list " << endl << "3 - set " << endl << "4 - map " << endl << "0 - завершение роботы програмы " << endl;

cin >> command;

while (command != 0)

{

switch (command)

{

case 1:

command = Vector();

if (\_CrtDumpMemoryLeaks())

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

else

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

break;

case 2:

command = List();

if (\_CrtDumpMemoryLeaks())

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

else

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

break;

case 3:

command = Set();

if (\_CrtDumpMemoryLeaks())

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

else

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

break;

case 4:

command = Map();

if (\_CrtDumpMemoryLeaks())

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

else

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

break;

default:

cout << "Вы ошиблись в выборе, попробуйте еще раз" << endl;

break;

}

if (command == 404)

return 1;

cout << "Введите команду котора соответствует контейнеру" << "1 - vector " << endl << "2 - list " << endl << "3 - set " << endl << "4 - map " << endl << "0 - завершение роботы програмы " << endl;

cin >> command;

}

return 0;

}

void Str\_return(Capabilities obj)

{

stringstream str;

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

Str\_output(str);

}

int Count\_line(string file\_name )

{

int 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 0;

}

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 size;

}

void Str\_output(stringstream& str)

{

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 << "- " << 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 << "- " << 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\*/

}

}

Файл Map.cpp

#include "Conteiner.h"

map<int, Capabilities> Read\_file(string file\_name, string file\_name2, int size);

map<int, Capabilities> Distribution(string line, string line2, int i, map<int, Capabilities>);

map<int, Capabilities> Add(map<int, Capabilities>);

map<int, Capabilities> Delete(map<int, Capabilities>);

Capabilities Index\_return(const int index, map<int, Capabilities>);

void Output(map<int, Capabilities>);

float Difference(map<int, Capabilities>);

void Info\_about\_lib(map<int, Capabilities>);

int Write\_file(map<int, Capabilities>, string file\_name);

void Check(map<int, Capabilities>);

void Criterion(map<int, Capabilities>);

void Search(map<int, Capabilities>);

int Map()

{

map<int, Capabilities> conteiner;

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

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

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

int size = Count\_line(file\_read);

conteiner = Read\_file(file\_read, file\_read2, size);

if (conteiner.size() == 0)

return 404;

Output(conteiner);

conteiner = Add(conteiner);

Output(conteiner);

conteiner = Delete(conteiner);

Output(conteiner);

float diff = Difference(conteiner);

int id;

cout << "Введите айди элемента которого хотите получить" << endl;

cin >> id;

Capabilities res\_id = Index\_return(id, conteiner);

Str\_return(res\_id);

Check(conteiner);

cout << "Вывод объектов в названии которых 2 слова" << endl;

Info\_about\_lib(conteiner);

Criterion(conteiner);

Search(conteiner);

int res = Write\_file(conteiner, file\_write);

if (res == 1)

return 404;

conteiner.clear();

return 0;

}

map<int, Capabilities> Read\_file(string file\_name, string file\_name2, int size)

{

map<int, Capabilities> cont;

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

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

{

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

return cont;

}

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

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

{

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

return cont;

}

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 \*/

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

}

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

return cont;

}

map<int, Capabilities> Distribution(string line, string line2, int i, map<int, Capabilities> cont)

{

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);

cont.emplace(i, Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

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);

cont.emplace(i, Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

}

return cont;

}

map<int, Capabilities> Add(map<int, Capabilities> cont)

{

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

Capabilities add\_el;

cont.emplace(index\_el, add\_el);

return cont;

}

map<int, Capabilities> Delete(map<int, Capabilities> cont)

{

auto it = cont.begin();

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

advance(it, index\_el - 1);

cont.erase(it);

return cont;

}

Capabilities Index\_return(const int index, map<int, Capabilities> cont)

{

int id;

Capabilities value;

for (auto var : cont)

{

id = var.second.getID(); /\*\*< дізнаємося ідентифікатор кожного елементу\*/

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

{

value = var.second;

}

}

return value;

}

void Output(map<int, Capabilities> cont)

{

stringstream str;

int i = 0;

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\_each(cont.begin(), cont.end(), [](pair<int, Capabilities> obj)

{

stringstream str;

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

Str\_output(str);

});

}

float Difference(map<int, Capabilities> cont)

{

float count = 0;

string y;

for (auto var : cont)

{

y = var.second.getDynamically();

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

count++;

}

float dif = cont.size() / count; /\*\*< визначаємо частку від ділення \*/

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

return dif;

}

void Info\_about\_lib(map<int, Capabilities> cont)

{

for (auto var : cont)

{

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

cout << var.second.getLanguage\_programming();

cout << "\" используется библиотека " << var.second.getName() << endl;

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

cout << "Функция - " << var.second.getFunction() << endl;

cout << "Возможности данной библиотеки" << endl << var.second.getCapabilities() << endl;

}

}

int Write\_file(map<int, Capabilities> cont, string file\_name)

{

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

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

{

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

return 1;

}

int i = 0;

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 (auto var : cont)

{

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

i++;

}

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

return 0;

}

void Check(map<int, Capabilities> cont)

{

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 (auto var : cont)

{

auto check\_res = var.second.getName();

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

{

Str\_return(var.second);

k++;

}

}

}

void Criterion(map<int, Capabilities> cont)

{

int command, count;

bool flag = true;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << "404 - завершение поиска количества по критериям" << endl;

cin >> command;

while (flag)

{

count = 0;

switch (command)

{

case 1:

{

regex regular("(([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)+( )+([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*))");

for (auto var : cont)

{

auto check\_res = var.second.getName();

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

{

count++;

}

}

cout << "Количество бииблиотек, в названии которых 2 слова - " << count << endl;

break;

}

case 2:

{

int year;

cout << "Введите год создания" << endl;

cin >> year;

for (auto var : cont)

{

if (var.second.getYear\_Creating() == year) /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотеки" << year << " года создания - " << count << endl;

break;

}

case 3:

{

cout << "1 - язык С, 2 - С++" << endl;

cin >> command;

if (command == 1)

{

for (auto var : cont)

{

if (var.second.getLanguage\_programming() == "C") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С - " << count << endl;

}

else

{

for (auto var : cont)

{

if (var.second.getLanguage\_programming() == "C++") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С++ - " << count << endl;

}

break;

}

case 4:

{

for (auto var : cont)

{

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

count++;

}

cout << "Количество динамически подключаемых библиотек - " << count << endl;

break;

}

case 404:

{

flag = false;

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

break;

}

}

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

}

}

void Search(map<int, Capabilities> cont)

{

int command;

bool flag = true;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl << "3 - Расширение файла библиотеки" << endl;

cin >> command;

while (flag)

{

switch (command)

{

case 1:

{

string name;

cout << "Введите имя искомого объекта" << endl;

cin >> name;

for (auto var : cont)

{

if (var.second.getName() == name)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var.second);

flag = false;

break;

}

}

break;

}

case 2:

{

int id;

cout << "Введите ID искомого объекта" << endl;

cin >> id;

for (auto var : cont)

{

if (var.second.getID() == id)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var.second);

flag = false;

break;

}

}

break;

}

case 3:

{

string exp;

cout << "Введите расширение искомого объекта" << endl;

cin >> exp;

for (auto var : cont)

{

if (var.second.getExpansion\_file() == exp)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var.second);

flag = false;

break;

}

}

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl;

cin >> command;

break;

}

}

}

}

Файл Set.cpp

#include "Conteiner.h"

set<Capabilities> Read\_file(string file\_name, string file\_name2, int size);

set<Capabilities> Distribution(string line, string line2, int i, set<Capabilities>);

set<Capabilities> Add(set<Capabilities>);

set<Capabilities> Delete(set<Capabilities>);

Capabilities Index\_return(const int index, set<Capabilities>);

void Output(set<Capabilities>);

float Difference(set<Capabilities>);

void Info\_about\_lib(set<Capabilities>);

int Write\_file(set<Capabilities>, string file\_name);

void Check(set<Capabilities>);

void Criterion(set<Capabilities>);

void Search(set<Capabilities>);

int Set()

{

set<Capabilities> conteiner;

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

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

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

int size = Count\_line(file\_read);

conteiner = Read\_file(file\_read, file\_read2, size);

if (conteiner.size() == 0)

return 404;

Output(conteiner);

conteiner = Add(conteiner);

Output(conteiner);

conteiner = Delete(conteiner);

Output(conteiner);

float diff = Difference(conteiner);

int id;

cout << "Введите айди элемента которого хотите получить" << endl;

cin >> id;

Capabilities res\_id = Index\_return(id, conteiner);

Str\_return(res\_id);

cout << "Вывод объектов в названии которых 2 слова" << endl;

Check(conteiner);

Info\_about\_lib(conteiner);

Criterion(conteiner);

Search(conteiner);

int res = Write\_file(conteiner, file\_write);

if (res == 1)

return 404;

conteiner.clear();

return 0;

}

set<Capabilities> Read\_file(string file\_name, string file\_name2, int size)

{

set<Capabilities> cont;

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

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

{

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

return cont;

}

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

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

{

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

return cont;

}

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 \*/

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

}

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

return cont;

}

set<Capabilities> Distribution(string line, string line2, int i, set<Capabilities> cont)

{

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;

auto it = cont.begin();

advance(it, i);

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);

cont.insert(it, Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

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);

cont.insert(it, Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

}

return cont;

}

set<Capabilities> Add(set<Capabilities> cont)

{

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

Capabilities add\_el;

auto it = cont.begin();

advance(it, index\_el);

cont.insert(it, add\_el);

return cont;

}

set<Capabilities> Delete(set<Capabilities> cont)

{

auto it = cont.begin();

int index\_el;

cout << "Введите порядковый индекс елемента которого хотите удалить" << endl;

cin >> index\_el;

advance(it, index\_el - 1);

cont.erase(it);

return cont;

}

Capabilities Index\_return(const int index, set<Capabilities> cont)

{

int id;

Capabilities value;

for (auto var : cont)

{

id = var.getID(); /\*\*< дізнаємося ідентифікатор кожного елементу\*/

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

{

value = var;

}

}

return value;

}

void Output(set<Capabilities> cont)

{

stringstream str;

int i = 0;

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\_each(cont.begin(), cont.end(), Str\_return);

}

float Difference(set<Capabilities> cont)

{

float count = 0;

string y;

for (auto var : cont)

{

y = var.getDynamically();

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

count++;

}

float dif = cont.size() / count; /\*\*< визначаємо частку від ділення \*/

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

return dif;

}

void Info\_about\_lib(set<Capabilities> cont)

{

for (auto var : cont)

{

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

cout << var.getLanguage\_programming();

cout << "\" используется библиотека " << var.getName() << endl;

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

cout << "Функция - " << var.getFunction() << endl;

cout << "Возможности данной библиотеки" << endl << var.getCapabilities() << endl;

}

}

int Write\_file(set<Capabilities> cont, string file\_name)

{

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

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

{

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

return 1;

}

int i = 0;

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 (auto var : cont)

{

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

i++;

}

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

return 0;

}

void Check(set<Capabilities> cont)

{

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

stringstream str;

int k = 0;

for (auto var : cont)

{

auto check\_res = var.getName();

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

{

Str\_return(var);

}

}

}

void Criterion(set<Capabilities> cont)

{

int command, count;

bool flag = true;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << "404 - завершение поиска количества по критериям" << endl;

cin >> command;

while (flag)

{

count = 0;

switch (command)

{

case 1:

{

regex regular("(([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)+( )+([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*))");

for (auto var : cont)

{

auto check\_res = var.getName();

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

{

count++;

}

}

cout << "Количество бииблиотек, в названии которых 2 слова - " << count << endl;

break;

}

case 2:

{

int year;

cout << "Введите год создания" << endl;

cin >> year;

for (auto var : cont)

{

if (var.getYear\_Creating() == year) /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотеки" << year << " года создания - " << count << endl;

break;

}

case 3:

{

cout << "1 - язык С, 2 - С++" << endl;

cin >> command;

if (command == 1)

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С - " << count << endl;

}

else

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C++") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С++ - " << count << endl;

}

break;

}

case 4:

{

for (auto var : cont)

{

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

count++;

}

cout << "Количество динамически подключаемых библиотек - " << count << endl;

break;

}

case 404:

{

flag = false;

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

break;

}

}

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

}

}

void Search(set<Capabilities> cont)

{

int command;

bool flag = true;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl << "3 - Расширение файла библиотеки" << endl;

cin >> command;

while (flag)

{

switch (command)

{

case 1:

{

string name;

cout << "Введите имя искомого объекта" << endl;

cin >> name;

for (auto var : cont)

{

if (var.getName() == name)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 2:

{

int id;

cout << "Введите ID искомого объекта" << endl;

cin >> id;

for (auto var : cont)

{

if (var.getID() == id)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 3:

{

string exp;

cout << "Введите расширение искомого объекта" << endl;

cin >> exp;

for (auto var : cont)

{

if (var.getExpansion\_file() == exp)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl;

cin >> command;

break;

}

}

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl;

cin >> command;

}

}

Файл Test.cpp

#include "Test.h"

int main()

{

setlocale(LC\_ALL, "Rus");

bool result;

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

if (result) /\*\*< перевірка тесту\*/

cout << "Testvector пройден успешно!" << endl;

else

cout << "Testvector не пройден!" << endl;

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

if (result) /\*\*< перевірка тесту\*/

cout << "Testlist пройден успешно!" << endl;

else

cout << "Testlist не пройден!" << endl;

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

if (result)/\*\*< перевірка тесту\*/

cout << "Testset пройден успешно!" << endl;

else

cout << "Testset не пройден!" << endl;

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

if (result)/\*\*< перевірка тесту\*/

cout << "Testmap пройден успешно!" << endl;

else

cout << "Testmap не пройден!" << endl;

return 0;

}

Файл Test\_mothody.cpp

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

#include"Conteiner.h"

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

bool Testvector()

{

vector <int> vector = { 1, 2, 5, 3};

int vectorSize = vector.size();

int count=0;

vector.push\_back(1233);

if (vectorSize < vector.size())

count++;

auto vectorItForDelete = vector.begin();

vector.erase(vectorItForDelete);

if (vectorSize == vector.size())

count++;

if (vector[1] == 5)

count++;

if (count == 3)

return true;

else

return false;

}

bool Testlist()

{

list <int> List = { 122, 13, -1215, 9345, 45 };

int listSize = List.size();

int count = 0;

List.push\_back(234);

if (listSize < List.size())

count++;

std::list<int>::iterator listItForDelete = List.begin();

List.erase(listItForDelete);

if (listSize == List.size())

count++;

std::list<int>::iterator listItForGetObj = List.begin();

if (List.begin() == listItForGetObj)

count++;

if (count == 3)

return true;

else

return false;

}

bool Testset()

{

set <int> Set = { 122, 13, -1215, 9345, 45 };

int setSize = Set.size();

int count = 0;

Set.insert(234);

if (setSize < Set.size())

count++;

std::set<int>::iterator setItForDelete = Set.begin();

Set.erase(setItForDelete);

if (setSize == Set.size())

count++;

std::set<int>::iterator setItForGetObj = Set.begin();

if (Set.find(13) == setItForGetObj)

count++;

if (count == 3)

return true;

else

return false;

}

bool Testmap()

{

map <int, int> Map;

Map.insert(std::pair<int, int>(1, 21));

Map.insert(std::pair<int, int>(21, 2));

Map.insert(std::pair<int, int>(-12, 221));

Map.insert(std::pair<int, int>(78, 98));

int mapSize = Map.size();

int count = 0;

Map.insert(std::pair<int, int>(12, 34));

if (mapSize < Map.size())

count++;

std::map<int, int>::iterator mapItForDelete = Map.begin();

Map.erase(mapItForDelete);

if (mapSize == Map.size())

count++;

std::map<int, int>::iterator mapItForGetObj = Map.begin();

if (Map.find(1) == mapItForGetObj)

count++;

if (count == 3)

return true;

else

return false;

}

Файл Vector.cpp

#include "Conteiner.h"

vector<Capabilities> Read\_file(string file\_name, string file\_name2, int size);

vector<Capabilities> Distribution(string line, string line2, int i, vector<Capabilities>);

vector<Capabilities> Add(vector<Capabilities>);

vector<Capabilities> Delete(vector<Capabilities>);

Capabilities Index\_return(const int index, vector<Capabilities>);

void Output(vector<Capabilities>);

float Difference(vector<Capabilities>);

void Info\_about\_lib(vector<Capabilities>);

int Write\_file(vector<Capabilities>, string file\_name);

void Check(vector<Capabilities>);

void Criterion(vector<Capabilities>);

void Search(vector<Capabilities>);

int Vector()

{

vector<Capabilities> conteiner;

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

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

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

int size = Count\_line(file\_read);

conteiner = Read\_file(file\_read, file\_read2, size);

if (conteiner.size() == 0)

return 404;

Output(conteiner);

conteiner = Add(conteiner);

Output(conteiner);

conteiner = Delete(conteiner);

Output(conteiner);

float diff = Difference(conteiner);

int id;

cout << "Введите айди элемента которого хотите получить" << endl;

cin >> id;

Capabilities res\_id = Index\_return(id, conteiner);

Str\_return(res\_id);

cout << "Вывод объектов в названии которых 2 слова" << endl;

Check(conteiner);

Info\_about\_lib(conteiner);

Criterion(conteiner);

Search(conteiner);

int res = Write\_file(conteiner, file\_write);

if (res == 1)

return 404;

conteiner.clear();

return 0;

}

vector<Capabilities> Read\_file(string file\_name, string file\_name2, int size)

{

vector<Capabilities> cont;

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

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

{

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

return cont;

}

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

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

{

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

return cont;

}

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 \*/

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

}

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

return cont;

}

vector<Capabilities> Distribution(string line, string line2, int i, vector<Capabilities> cont)

{

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);

cont.emplace\_back(Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

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);

cont.emplace\_back(Capabilities(line2, dynamically, name, id, year\_creating, number\_of\_function, name2, name\_f, obj\_e));

}

}

return cont;

}

vector<Capabilities> Add(vector<Capabilities> cont)

{

cout << "Добавление елемента" << endl;

Capabilities add\_el;

cont.push\_back(add\_el);

return cont;

}

vector<Capabilities> Delete(vector<Capabilities> cont)

{

auto it = cont.begin();

cout << "Удаление єлемента" << endl;

cont.pop\_back();

return cont;

}

Capabilities Index\_return(const int index, vector<Capabilities> cont)

{

int id;

Capabilities value;

for (auto var : cont)

{

id = var.getID(); /\*\*< дізнаємося ідентифікатор кожного елементу\*/

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

{

value = var;

}

}

return value;

}

void Output(vector<Capabilities> cont)

{

stringstream str;

int i = 0;

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\_each(cont.begin(), cont.end(), Str\_return);

}

float Difference(vector<Capabilities> cont)

{

float count = 0;

string y;

for (auto var : cont)

{

y = var.getDynamically();

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

count++;

}

float dif = cont.size() / count; /\*\*< визначаємо частку від ділення \*/

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

return dif;

}

void Info\_about\_lib(vector<Capabilities> cont)

{

for (auto var : cont)

{

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

cout << var.getLanguage\_programming();

cout << "\" используется библиотека " << var.getName() << endl;

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

cout << "Функция - " << var.getFunction() << endl;

cout << "Возможности данной библиотеки" << endl << var.getCapabilities() << endl;

}

}

int Write\_file(vector<Capabilities> cont, string file\_name)

{

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

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

{

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

return 1;

}

int i = 0;

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 (auto var : cont)

{

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

i++;

}

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

return 0;

}

void Check(vector<Capabilities> cont)

{

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 (auto var : cont)

{

auto check\_res = var.getName();

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

{

Str\_return(var);

}

}

}

void Criterion(vector<Capabilities> cont)

{

int command, count;

bool flag = true;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" <<endl<<"404 - завершение поиска количества по критериям" << endl;

cin >> command;

while (flag)

{

count = 0;

switch (command)

{

case 1:

{

regex regular("(([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*)+( )+([A-ZА-Я]+[\\wА-Яа-я\-|\_|:|;|\.]\*))");

for (auto var : cont)

{

auto check\_res = var.getName();

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

{

count++;

}

}

cout << "Количество бииблиотек, в названии которых 2 слова - " << count << endl;

break;

}

case 2:

{

int year;

cout << "Введите год создания" << endl;

cin >> year;

for (auto var : cont)

{

if (var.getYear\_Creating() == year) /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотеки" << year << " года создания - " << count << endl;

break;

}

case 3:

{

cout << "1 - язык С, 2 - С++" << endl;

cin >> command;

if (command == 1)

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С - " << count << endl;

}

else

{

for (auto var : cont)

{

if (var.getLanguage\_programming() == "C++") /\*\*< преревіряємо чи містить назва 2 слова\*/

{

count++;

}

}

cout << "Количество библиотек, основаных на языке програмирования С++ - " << count << endl;

}

break;

}

case 4:

{

for (auto var : cont)

{

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

count++;

}

cout << "Количество динамически подключаемых библиотек - " << count << endl;

break;

}

case 404:

{

flag = false;

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек"<< endl << "404 - завершение поиска количества по критериям" << endl;;

cin >> command;

break;

}

}

if (flag )

{

cout << "Введите номер критерия по которому хотите узнать количество элементов" << endl << "1 - Имя библиотеки состоит из 2х слов" << endl << "2 - Год создания" << endl << "3 - Язык програмирования" << endl << "4 - Количество динамически подключаемых библиотек" << endl << "404 - завершение поиска количества по критериям" << endl;

cin >> command;

}

}

}

void Search(vector<Capabilities> cont)

{

int command;

bool flag = true;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl << "3 - Расширение файла библиотеки" << endl;

cin >> command;

while (flag)

{

switch (command)

{

case 1:

{

string name;

cout << "Введите имя искомого объекта" << endl;

cin >> name;

for (auto var : cont)

{

if (var.getName() == name)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 2:

{

int id;

cout << "Введите ID искомого объекта" << endl;

cin >> id;

for (auto var : cont)

{

if (var.getID() == id)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

case 3:

{

string exp;

cout << "Введите расширение искомого объекта" << endl;

cin >> exp;

for (auto var : cont)

{

if (var.getExpansion\_file() == exp)

{

cout << "Вивод объекта которого вы искали" << endl;

Str\_return(var);

flag = false;

break;

}

}

break;

}

default:

{

cout << "Вы такого критерия не существует" << endl;

cout << "Введите номер критерия поиск по которому хотите весли" << endl << "1 - Имя библиотеки" << endl << "2 - ID библиотеки" << endl;

cin >> command;

break;

}

}

}

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

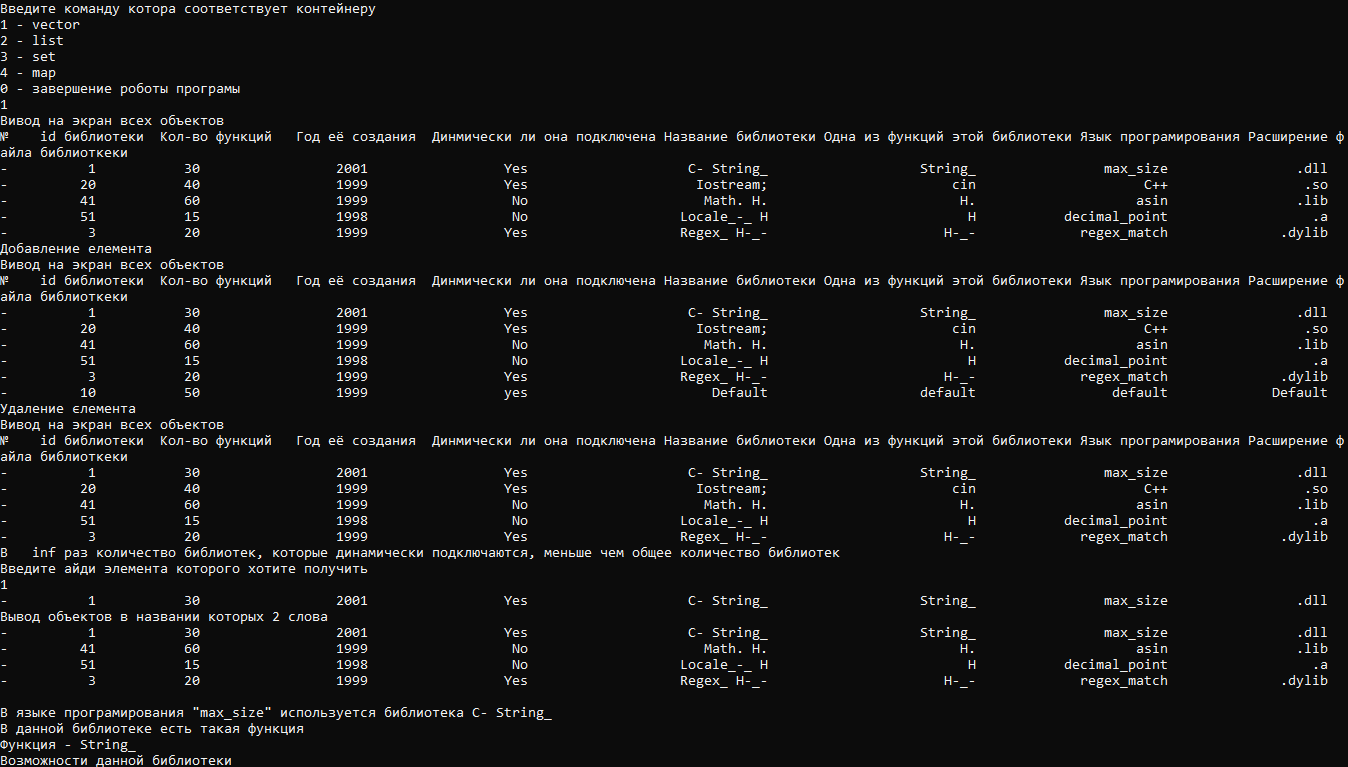
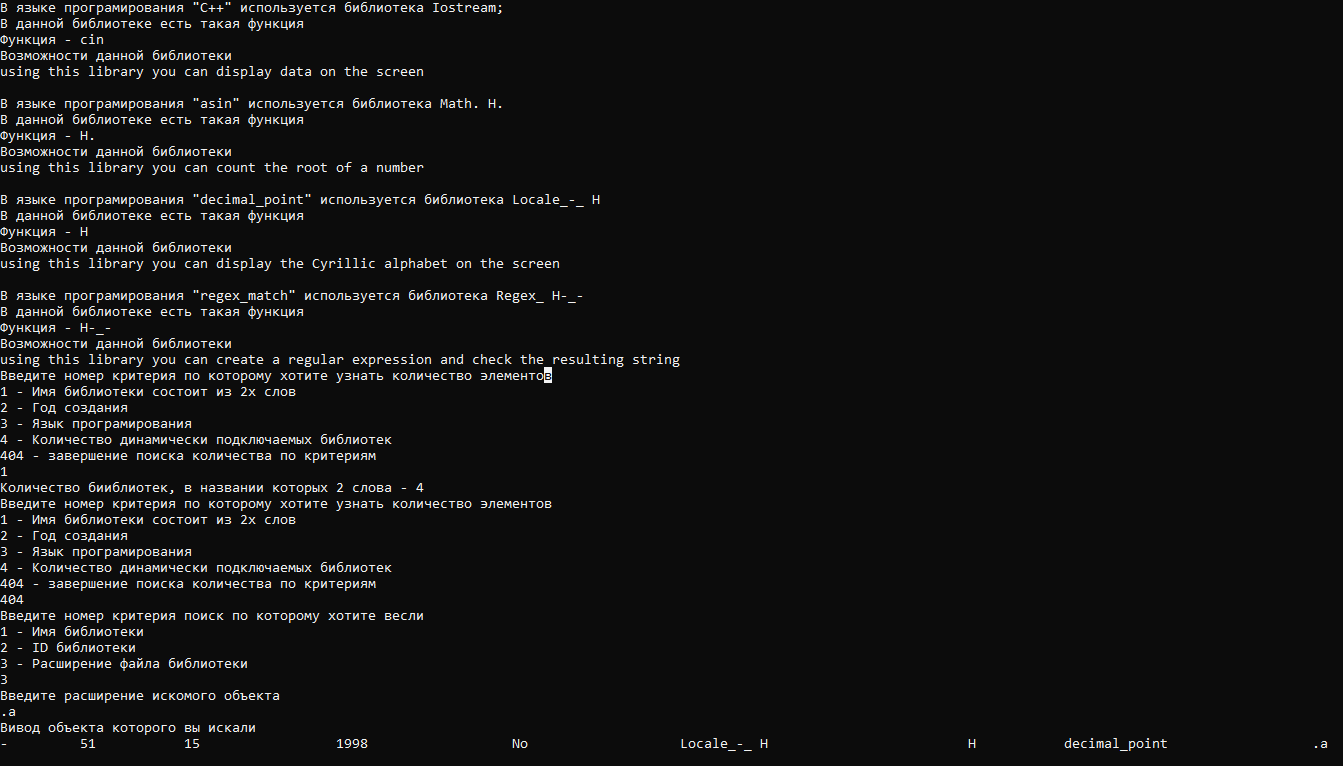


Рис. 1 - Результат роботи основоного завдання



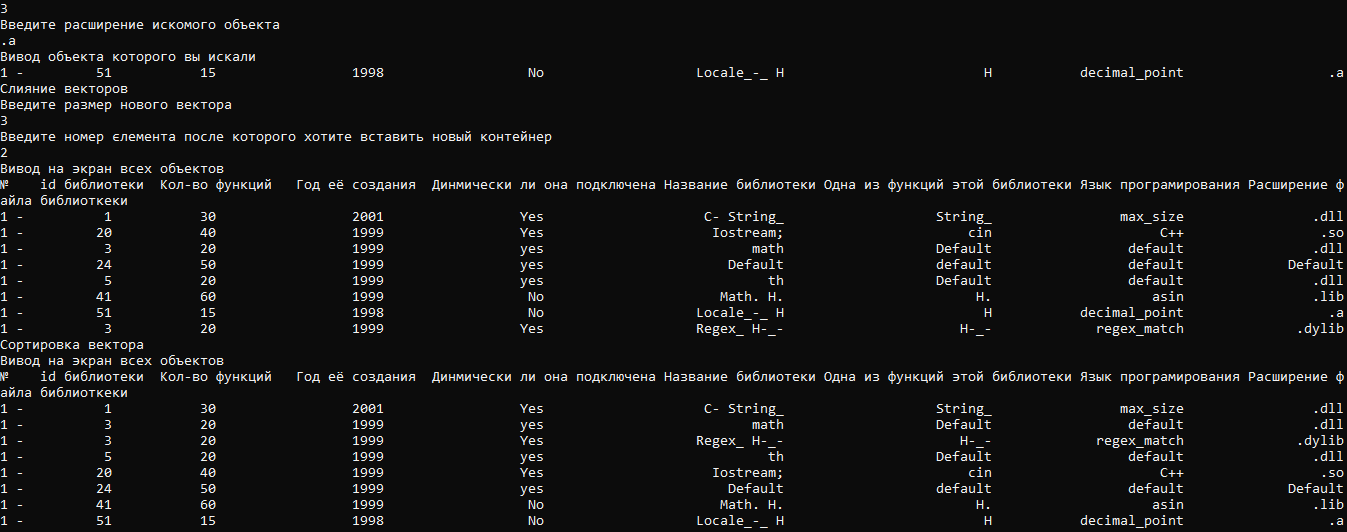


Рис. 2 - Результат роботи основоного завдання

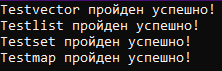


Рис. 3 - Результати тестування

ВИСНОВКИ

В результаті виконання лабораторної роботи №14 було виконано загальне завдання, а саме – об’єднання 2х контейнерів, модернізувати функцію сортування, та виконане додаткове завдання – особливе об’єднання контейнеру типу мар.