

ЗВІТ

Виконав:

Заночкин Є., КІТ-119а

Дата: 24 квітня 2020 р.

Лабораторна робота №4

РЕГУЛЯРНІ ВИРАЗИ

Мета роботи. Отримати знання про базові регулярні вирази та досвід роботи із застосуванням їх на практиці.

1.Завдання до роботи

Варіант 6.

Загальне завдання: поширити попередню лабораторну роботу таким чином:

- при введенні інформації про базовий клас (немає різниці, чи з клавіатури, чи з файлу), організувати перевірку відповідності таким критеріям з використанням регулярних виразів:

- можна вводити тільки кириличні символи, латинські символи, цифри, пропуски, розділові знаки;

- не повинно бути пропусків та розділових знаків, які повторюються;

- перше слово не повинно починатися з маленького символу;

- у клас-список додати метод, що виводить на екран список усіх об'єктів, які мають одне або більше полів з щонайменше двома словами (перевірку організувати за допомогою регулярних виразів).

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: Task

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

List

2.2 Опис змінних

int mark – оцінка за роботу (Task).

int countOfDoneExercises – кількість виконаних завдань (Task).

int studentIndex – індекс студента (Task).

string name – ім'я студента (Task).

int rgz – кількість ргз (Task).

int listSize – розмір масиву елементів класу List.

Task* stud – масив елементів класу Task.

List list – об'єкт класу List.

List test – об'єкт класу List.

Task var - об'єкт класу Task.

2.3 Опис методів

void createList() – створення масиву елементів і заповнення даними (List).

void printAll() const – виведення даних елементів у консоль (List).

void printOneStudent(stringstream&) const – виведення даних одного студента у консоль (List).

void addStudent(const Task) – додавання нового елементу в масив (List).

void deleteStudent(int) – видалення елемента з масиву (List).

int getStudentID(int)const – отримання даних елемента по індексу (List).

int getStudentRGZ(int)const - отримання даних елемента по кількості ргз (List).

void ReadFile(string, int) – читання даних з файлу (List).

int FileString(string) – кількість рядків у файлі (List).

void WriteFile(string) const – запис даних у файл (List).

stringstream getObj(int) const – повернення студента у вигляді рядка (List).

Task enterNewStudent() – введення нового студента з клавіатури (List).

void regexTask() – виведення на екран об'єктів, які в полі string мають 2 слова (List).

Task() – конструктор за змовчуванням (Task).

Task(int, string, int, int, int) – конструктор з аргументами (Task).

Task(const Task& stud) – конструктор копіювання (Task).

~Task() – деструктор (Task).

~List() – деструктор (List).

2.4 Опис функцій

void Menu() – функція меню.

Task CreateStudent() – стандартний об'єкт класу Task.

Task CreateStudent2() – стандартний об'єкт класу Task.

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

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

void TestGetStudentID(List&, int) – тест функції повернення індексу студента.

3. Текст програми

Task.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#define __CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, __FILE__, __LINE__)
#define new DEBUG_NEW
#include <string>
#include <iostream>
#include <iomanip>
#include <sstream>
#include <fstream>
#include <regex>
using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;
using std::regex;
class Task
{
```

```

private:
int studentIndex;
string name;
int mark;
int countOfDoneExercises;
int rgz;
public:
int getMark() const;
void setMark(int);
int getCountOfDoneExercises() const;
void setCountOfDoneExercises(int);
int getStudentIndex() const;
void setStudentIndex(int);
int getRgz() const;
void setRgz(int);
string getName() const;
void setName(const string);
Task();
Task(int, string, int, int, int);
Task(const Task& stud);
~Task();
};
void Menu();

```

List.h

```

#pragma once
#include "Task.h"
class List
{
private:
int listSize;
public:
Task* stud;
int getListSize() const;
void setListSize(int);
void addStudent(const Task);
void deleteStudent(int);
void printAll() const;
void printOneStudent(stringstream&) const;
void createList();
int getStudentID(int)const;
int getStudentRGZ(int)const;

```

```

void ReadFile(string, int);
int FileString(string);
void WriteFile(string) const;
stringstream getObj(int i) const;
void regexTask();
Task enterNewStudent();
~List();
};
Task CreateStudent();
Task CreateStudent2();

```

Task.cpp

```

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

int Task::getMark() const { return mark; }
void Task::setMark(int mark1) { mark = mark1; }
int Task::getCountOfDoneExercises() const { return countOfDoneExercises; }
void Task::setCountOfDoneExercises(int count_of_done_exercises1)
{ countOfDoneExercises = count_of_done_exercises1; }
int Task::getStudentIndex() const { return studentIndex; }
void Task::setStudentIndex(int exercises1) { studentIndex = exercises1; }
int Task::getRgz() const { return rgz; }
void Task::setRgz(int rgz1) { rgz = rgz1; }
string Task::getName() const { return name; }
void Task::setName(string name1) { name = name1; }

Task CreateStudent()
{
    Task stud;
    stud.setMark(5);
    stud.setCountOfDoneExercises(5);
    stud.setName("Petrova Katya");
    return stud;
}

Task CreateStudent2()
{
    Task stud;
    stud.setMark(2);
    stud.setCountOfDoneExercises(2);
    stud.setName("Ivanov Petya");
    return stud;
}

```

```

Task::Task(int student_index, string name, int mark, int countOfDoneExercises, int rgz) :
studentIndex(student_index), name(name), mark(mark), countOfDoneExercises(countOfDoneExercises), rgz(rgz)
{ cout << "Constructor with parameter" << endl; }

Task::Task() : studentIndex(0), name(" "), mark(0), countOfDoneExercises(0), rgz(0)
{ cout << "Default constructor" << endl; }

Task::Task(const Task& stud) : studentIndex(stud.studentIndex), name(stud.name), mark(stud.mark),
countOfDoneExercises(stud.countOfDoneExercises), rgz(stud.rgz)
{ cout << "Copy constructor" << endl; }

Task::~~Task()
{ cout << "Destructor Task" << endl; }

```

List.cpp

```

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

int List::getListSize() const { return listSize; }

void List::setListSize(int size) { listSize = size; }

void List::addStudent(const Task task)
{
    if (task.getMark() == 0)
    {
        cout << "Empty object. Error";
        return;
    }

    setListSize(getListSize() + 1);
    Task* newstud = new Task[listSize];
    for (int i = 0; i < listSize - 1; i++)
        newstud[i] = stud[i];
    newstud[listSize - 1] = task;
    if (newstud[listSize - 1].getStudentIndex() == 0)
    {
        newstud[listSize - 1].setStudentIndex(stud[listSize - 2].getStudentIndex() + 1);
        newstud[listSize - 1].setRgz(stud[listSize - 2].getRgz() + 1);
    }
    delete[] stud;
    stud = new Task[listSize];
    for (int i = 0; listSize > i; i++)
        stud[i] = newstud[i];
    delete[] newstud;
}

void List::createList()
{
    stud = new Task[listSize];
}

```

```

for (int i = 0; listSize > i; i++)
{
    stud[i] = CreateStudent();
    stud[i].setStudentIndex(i + 1);
    stud[i].setRgz(i + 5);
}
}

void List::deleteStudent(int c)
{
    setListSize(getListSize() - 1);
    Task* newstud = new Task[listSize];
    int i = 0;
    for (; i < getListSize(); i++)
    {
        if (stud[i].getStudentIndex() == c)
            break;
        newstud[i] = stud[i];
    }
    for (; i < getListSize(); i++)
        newstud[i] = stud[i + 1];
    delete[] stud;
    stud = new Task[listSize];
    for (int i = 0; i < getListSize(); i++)
        stud[i] = newstud[i];
    delete[] newstud;
}

void List::printOneStudent(stringstream& ss) const
{
    int index;
    string name, name2;
    int mark;
    int exercises;
    int rgz;

    cout << std::left << setw(6) << "Index" << setw(15) << "Name" << setw(10) << "Mark" << setw(13) <<
    "Exercises" << setw(8) << "RGZ" << endl;

    ss >> index;
    ss >> name;
    ss >> name2;
    ss >> mark;
    ss >> exercises;
    ss >> rgz;
    if (name2 == "")

```

```

        name = name + " ";
    else (name = name + " " + name2);
    cout << std::left;
    cout << setw(6) << index;
    cout << setw(15) << name;
    cout << setw(10) << mark;
    cout << setw(13) << exercises;
    cout << setw(8) << rgz;
}

void List::printAll() const
{
    cout << std::left << setw(6) << "Index" << setw(15) << "Name" << setw(10) << "Mark" << setw(13) <<
"Exercises" << setw(8) << "RGZ" << endl;
    for (int i = 0; List::getListSize() > i; i++)
        cout << std::left << setw(6) << stud[i].getStudentIndex() << setw(15) << stud[i].getName() << setw(10) <<
stud[i].getMark() << setw(13) << stud[i].getCountOfDoneExercises() << setw(8) << stud[i].getRgz() << endl;
}

List::~~List()
{
    cout << "Destructor List" << endl;
    delete[] stud;
}

int List::getStudentID(int id) const
{
    for (int i = 0; i < listSize; i++)
        if (stud[i].getStudentIndex() == id)
            return i;
    cout << "Wrong ID" << endl;
    return -1;
}

int List::getStudentRGZ(int a) const
{
    for (int i = 0; i < listSize; i++)
        if (stud[i].getRgz() == a)
            return i;
    cout << "Wrong count of RGZ" << endl;
    return -1;
}

void List::ReadFile(string filename, int c)
{
    std::ifstream fin(filename);
    if (!fin.is_open())

```



```

return;
string line;
regex regular("(\\d)* [A-ZА-Я]+(\\wА-Яа-я,;:-)* (\\wА-Яа-я,;:-)* (\\d)* (\\d)* (\\d)*");
int i = 0, a = 0, b = 0;
delete[] stud;
stud = new Task[c];
while (getline(fin, line) && i < c)
{
if (regex_match(line.c_str(), regular))
{
int studentIndex;
string name, name2;
int mark;
int countOfDoneExercises;
int rgz;
std::istringstream fin(line);
fin >> studentIndex;
fin >> name;
fin >> name2;
fin >> mark;
fin >> countOfDoneExercises;
fin >> rgz;
if (name2 == "")
name = name + " ";
else (name = name + " " + name2);
do
{
b = 0;
a = name.find("--");
if (a != -1)
{
name.erase(a, 1);
b = 1;
}
a = name.find(" ");
if (a != -1)
{
name.erase(a, 1);
b = 1;
}
a = name.find(",");
if (a != -1)

```

```

{
name.erase(a, 1);
b = 1;
}
a = name.find("::");
if (a != -1)
{
name.erase(a, 1);
b = 1;
}
a = name.find(";;");
if (a != -1)
{
name.erase(a, 1);
b = 1;
}
a = name.find("_");
if (a != -1)
{
name.erase(a, 1);
b = 1;
}
} while (b == 1);
Task ex(studentIndex, name, mark, countOfDoneExercises, rgz);
stud[i++] = ex;
}
}
setListSize(c);
fin.close();
cout << "Data from file have written";
return;
}
int List::FileString(string filename)
{
int c = 0;
string line;
regex regular("([\\d]* [A-ZА-Я]+[\\wА-Яа-я,,:-]* [\\wА-Яа-я,,:-]* [\\d]* [\\d]* [\\d]*)");
std::ifstream fin(filename);
if (!fin.is_open())
{
cout << "Error open file";
return 0;
}

```

```

    }
    while (getline(fin, line))
    {
        if (regex_match(line, regular))
            c++;
        else cout << "String is not correct" << endl;
    }
    fin.close();
    return c;
}

void List::WriteFile(string filename) const
{
    std::ofstream fout(filename);
    if (!fout.is_open())
    {
        cout << "";
        return;
    }
    for (int i = 0; i < listSize; i++)
    {
        fout << std::left;
        fout << setw(3) << stud[i].getStudentIndex();
        fout << setw(15) << stud[i].getName();
        fout << setw(7) << stud[i].getMark();
        fout << setw(6) << stud[i].getCountOfDoneExercises();
        fout << setw(6) << stud[i].getRgz() << endl;
    }
    cout << "Write to file - correct" << endl;
    return;
}

stringstream List::getObj(int i) const
{
    stringstream ss;
    ss << stud[i].getStudentIndex();
    ss << " " << stud[i].getName();
    ss << " " << stud[i].getMark();
    ss << " " << stud[i].getCountOfDoneExercises();
    ss << " " << stud[i].getRgz();
    return ss;
}

Task List::enterNewStudent()
{

```

```

Task add, error;
int index, mark, rgz, exercises, age, rgz4Teach;
string surname, name, data;
regex regular("(\\d]* [A-Z]+[\\wA-Za-z,.;-]* [A-Z]+[\\wA-Za-z,.;-]* [\\d]* [\\d]* [\\d]*)");
cout << "Enter data in line(index, surname, name, mark, exercises, rgz)" << endl;
cin.ignore();
getline(cin, data);
std::istringstream temp(data);
temp >> index;
temp >> surname;
temp >> name;
temp >> mark;
temp >> exercises;
temp >> rgz;
if (name == "")
    surname = surname + " ";
else (surname = surname + " " + name);
if (!regex_match(data, regular))
{
    cout << "You enter wrong data";
    return error;
}
add.setStudentIndex(index);
add.setName(surname);
add.setMark(mark);
add.setCountOfDoneExercises(exercises);
add.setRgz(rgz);
return add;
}

void List::regexTask()
{
    stringstream ss;
    int index, mark, rgz, exercises;
    string name, name2;
    regex regular("(^[A-ZА-Я]+[\\wА-Яа-я,.;-]* [\\wА-Яа-я,.;-]+)");
    int listSize = getListSize();
    cout << std::left << setw(6) << "Index" << setw(15) << "Name" << setw(10) << "Mark" << setw(13) <<
    "Exercises" << setw(8) << "RGZ" << endl;
    for (int i = 0; i < listSize; i++)
        if (regex_match(stud[i].getName(), regular))
        {
            ss = getObj(i);

```

```

ss >> index >> name >> name2 >> mark >> exercises >> rgz;
if (name2 == "")
name = name + " ";
else (name = name + " " + name2);
cout << std::left;
cout << setw(6) << index;
cout << setw(15) << name;
cout << setw(10) << mark;
cout << setw(13) << exercises;
cout << setw(8) << rgz << endl;
}
}

```

Menu.cpp

```

#include "Task.h"
#include "List.h"
void Menu()
{
List list;
int c = 0, a = 0, b = 0;
int count_of_students = 2;
int menu_number = 1;
int delete_number;
string fileName;
stringstream ss;
Task var;
int file;
list.setListSize(count_of_students);
list.createList();
while (menu_number)
{
menu_number = 0;
cout << endl << "Menu:" << endl;
cout << "1.Add a new student" << endl;
cout << "2.Add a new student (enter from keyboard)" << endl;
cout << "3.Delete one student" << endl;
cout << "4.Show all student" << endl;
cout << "5.Show student via his index" << endl;
cout << "6.Individual task" << endl;
cout << "7.Read array from file" << endl;
cout << "8.Write to file" << endl;

```

```

cout << "9.Surname+Name in object" << endl;
cout << "10.End program" << endl;
cin >> menu_number;
switch (menu_number)
{
case 1:
var = CreateStudent2();
list.addStudent(var);
break;
case 2:
list.addStudent(list.enterNewStudent());
break;
case 3:
cout << "Enter a index of student who you want to delete:" << endl;
cin >> delete_number;
if (delete_number < 1)
{
cout << "Wrong student index" << endl;
break;
}
for (int i = 0; list.getListSize() > i; i++)
if (delete_number == list.stud[i].getStudentIndex())
{
list.deleteStudent(delete_number);
break;
}
break;
case 4:
list.printAll();
break;
case 5:
cout << "Enter a index of student:";
cin >> c;
b = list.getStudentID(c);
if (b == -1)
break;
ss = list.getObj(b);
list.printOneStudent(ss);
break;
case 6:
cout << "Enter a count of RGZ";
cin >> a;

```

```

b = list.getStudentRGZ(a);
if (b == -1)
break;
ss = list.getObj(b);
list.printOneStudent(ss);
break;
case 7:
cout << "Enter file name:";
cin >> fileName;
list.ReadFile(fileName, list.FileString(fileName));
break;
case 8:
cout << "Enter file name:";
cin >> fileName;
list.WriteFile(fileName);
break;
case 9:
list.regexTask();
break;
case 10:
menu_number = 0;
break;
default:
cout << "You have chosen the wrong number of the menu";
break;
}
}
return;
}

```

main.cpp

```

#include "Task.h"

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

```

Test.cpp

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

int TestAddStudent(List&, int);
int TestDeleteStudent(List&, int);
int TestGetStudentID(List&, int);
int TestReadFile(List&, int);

int main()
{
{
List test;
int count = 0;
test.setListSize(2);
test.createList();
count = TestAddStudent(test, count);
count = TestDeleteStudent(test, count);
count = TestGetStudentID(test, count);
count = TestReadFile(test, count);
if (count == 4)
cout << endl << "All tests are successful" << endl;
else
cout << endl << "Not all tests are successful" << endl;
}
if (_CrtDumpMemoryLeaks())
cout << endl << "WARNING! Memory leak" << endl;
else
cout << endl << "There is no memory leak" << endl;
return 0;
}

int TestAddStudent(List& test, int count)
{
test.addStudent(CreateStudent2());
if (test.getListSize() == 3)
{
cout << endl << "Test: Add_student - successful" << endl;
count++;
}
else
cout << endl << "Test: Add_student - unsuccessful" << endl;
return count;
}

int TestDeleteStudent(List& test, int count)
```



```

{
test.deleteStudent(test.getListSize());
if (test.getListSize() == 2)
{
cout << endl << "Test: Delete_student - successful" << endl;
count++;
}
else
cout << endl << "Test: Delete_student - unsuccessful" << endl;
return count;
}
int TestGetStudentID(List& test, int count)
{
int num = test.getStudentID(2);
if (num == 1)
{
cout << endl << "Test: GetStudentId - successful" << endl;
count++;
}
else
cout << endl << "Test: GetStudentId - unsuccessful" << endl;
return count;
}
int TestReadFile(List& test, int count)
{
string filename = "Text.txt";
int a = 3;
test.ReadFile(filename,a);
string expected = "Ivanov Vasya";
string real = test.stud[0].getName();
if (expected == real)
{
cout << endl << "Test: Readfile - successful" << endl;
count++;
}
else
cout << endl << "Test: Readfile - unsuccessful" << endl;
return count;
}

```

4. Результати роботи програми

```

Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Individual task
7.Read array from file
8.Write to file
9.Surname+Name in object
10.End program
2
Default constructor
Default constructor
Enter data in line(index, surname, name, mark, exercises, rgz)
10 zANOCHKIN Egor 5 5 10
You enter wrong dataCopy constructor
Destructor Task
Destructor Task
Empty object. ErrorDestructor Task

Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Individual task
7.Read array from file
8.Write to file
9.Surname+Name in object
10.End program
4

```

Index	Name	Mark	Exercises	RGZ
1	Petrova Katya	5	5	5
2	Petrova Katya	5	5	6

Рисунок 1 – Перевірка введення з клавіатури

```

Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Individual task
7.Read array from file
8.Write to file
9.Surname+Name in object
10.End program
7
Enter file name:Text.txt
String is not correct
Destructor Task
Destructor Task
Default constructor
Default constructor
Default constructor
Constructor with parameter
Destructor Task
Constructor with parameter
Destructor Task
Constructor with parameter
Destructor Task
Data from file have written
Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Individual task
7.Read array from file
8.Write to file
9.Surname+Name in object
10.End program
4

```

Index	Name	Mark	Exercises	RGZ
1	Ivanov Vasya	5	10	4
2	Zanochkin Egor	4	15	5
3	Plohoi Ivan	3	20	6

Рисунок 2 – Перевірка введення з файлу

1	1 Ivanov Vasya 5 10 4
2	2 Zanochkin Egor 4 15 5
3	3 Plohoi Ivan 3 20 6
4	4 zANOCHKIN Egor 5 5 12

Рисунок 3 – Файл з даними

```
Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Individual task
7.Read array from file
8.Write to file
9.Surname+Name in object
10.End program
9
Index Name          Mark Exercises RGZ
1   Ivanov Vasya    5      10      4
2   Zanochnik Egor 4      15      5
3   Plohoi Ivan     3      20      6
```

Рисунок 4 – Виведення студентів з 2 словами у полі string

5. Висновки

Під час виконання лабораторної роботи була організована перевірка на введення даних за допомогою регулярних виразів та додано метод, що виводить на екран усіх студентів, які мають у полі string 2 слова.

Програма протестована, виконується без помилок, витоків пам'яті немає.