

Лабораторна робота 4. РЕГУЛЯРНІ ВИРАЗИ

Тема. Регулярні вирази.

Мета: отримати знання про базові регулярні вирази та досвід роботи із застосування їх на практиці.

1. Завдання до роботи

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 28 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: `C_Program`

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

2.1 Опис змінних

`int` `timeOfWork` – поле класу `Program`(час виконання програми).
`int` `size` – поле класу `Program`(розмір програми у мегабайтах).
`int` `amountOfLines` – поле класу `Program`(кількість рядків коду).
`int` `index` – поле класу `Program`(індентифікаційний номер).
`const char*` `trojan` – поле класу `Program`(троян чи ні).
`int` `listSize` – поле класу `CList`(розмір масиву елементів класу `Program`).
`C_Program*` `list` – поле класу `C_Program`(масив елементів класу `Program`).
`C_List` `list` – об'єкт класу `CList`.
`C_Program` `newProgram`, `getProgram` – змінні для програм, необхідні для роботи програми.
`int` `choise` = 1, `value` = 0, `stop` = 1 – змінні, необхідні для роботи функції меню.
`string` `fileName` – змінна, необхідна для запису назви файлу для роботи з ним.

2.2 Опис методів

`void setListSize(int)` – запис даних у змінну розміру масиву елементів класу Program (метод класу C_List).

`int getListSize() const` – отримання даних змінної розміру масиву елементів класу Program (метод класу C_List).

`void createList(int)` – створення масиву елементів і заповнення даними (метод класу C_List).

`void printAll() const` – виведення даних елементів у консоль (метод класу C_List).

`void printOneEl(int) const` – виведення даних одного елементу у консоль (метод класу C_List).

`void addEl(C_Program&)` – додавання нового елементу в масив (метод класу C_List).

`void deleteEl(int)` – видалення елемента з масиву (метод класу C_List).

`int task(int)` – знаходження елементів за певним критерієм (метод класу C_List).

`C_Program getProgramID(int) const` – отримання даних елемента по індексу (метод класу C_List).

`C_Program programs(int)` – програми для заповнення списку (метод класу C_List).

`int linesInFile(string)` – знаходження кількості рядків у файлі (метод класу C_List).

`void readFile(string)` – читання даних з файлу (метод класу C_List).

`stringstream getOneEl(int) const` – отримання строки даних (метод класу C_List).

`void saveToFile(string)` – збереження даних у файл (метод класу C_List).

`void showOneEl(stringstream&) const` – читання даних з рядка у консоль (метод класу C_List).

`void enterNewEl()` – введення даних з клавіатури (метод класу C_List).

`void regexTask()` – виведення елементів, назва яких містить 2 слова (метод класу C_List).

`~C_List()` – деструктор списку елементів (метод класу C_List).

`C_Program()` – конструктор без параметра (метод класу C_Program)

`C_Program(bool, int, int, int, int, string)` – конструктор класу з параметрами (метод класу C_Program)

`C_Program(const C_Program& other)` – конструктор копіювання (метод класу C_Program)

`~C_Program()` – деструктор елемента (метод класу C_Program).

2.3 Опис функцій

`void Menu()` – функція меню.

`void Test_GetProgramID(C_List&, int&)` – тест функції знаходження та повернення об'єкту по індексу.

`void Test_AddEl(C_List&)` – тест функції додавання об'єкта до масиву об'єктів.

`void Test_DelEl(C_List&)` – тест функції видалення об'єкта з масиву об'єктів.

`void Test_Task(C_List&, int&)` – тест функції знаходження елементів за певними критеріями (індивідуальне завдання).

`void Test_Stringstream(C_List&)` – тест функції отримання даних зі строки.

`void Test_ReadFile(C_List&)` – тест функції читання даних з файлу.

`void Test_RegexTask(CList&)` – тест функції отримання даних програм, які містять 2 слова.

3. Текст програми

test.cpp

```
#include "Program.h"
#include "List.h"

void Test_GetProgramID(CList&);
void Test_AddEl(CList&);
void Test_DelEl(CList&);
void Test_Task(CList&);
void Test_Stringstream(CList&);
void Test_ReadFile(CList&);
void Test_RegexTask(CList&);

int main() {
    setlocale(LC_ALL, "Rus");
    CList List;
    List.createList(5);

    Test_GetProgramID(List);
    Test_AddEl(List);
    Test_DelEl(List);
    Test_Task(List);
    Test_Stringstream(List);
    Test_ReadFile(List);
    Test_RegexTask(List);

    if (_CrtDumpMemoryLeaks()) cout << "\n\nЕсть утечка памяти.\n\n";
    else cout << "\n\nУтечка памяти отсутствует\n\n.";

    return 0;
}

void Test_GetProgramID(CList& list)
{
    int expected = 5678;
    int real = list.list[2].getIndex();

    if(expected == real) cout << "Тест получения элемента по ID\t\t выполнен успешно.\n";
    else cout << "Тест получения элемента по ID\t\t не выполнен успешно.\n";
}

void Test_AddEl(CList& list)
{
    C_Program newProgram;
    int size = list.getListSize();
    list.addEl(newProgram);

    if (list.getListSize() > size) cout << "Тест добавления элемента в список\t выполнен успешно.\n";
    else cout << "Тест добавления элемента в список\t не выполнен успешно.\n";
}

void Test_DelEl(CList& list)
{
    int size = list.getListSize();
    list.deleteEl(3);

    if (size > list.getListSize()) cout << "Тест функции удаления\t\t выполнен успешно.\n\n";
    else cout << "Тест функции удаления\t\t не выполнен успешно.\n\n";
}

void Test_Task(CList& list)
{
    int expected = 2;
    int real = list.task(200);

    cout << endl;
```

```

        if(expected == real) cout << "Тест функции нахождения элементов по параметру\t\t выполнен успешно.\n";
        else cout << "Тест функции нахождения элементов по параметру\t\t не выполнен успешно.\n";
    }
void Test_Stringstream(CList& list)
{
    string nameExpected = "Skype";
    stringstream funcResult = list.getOneEl(1);
    string nameReal;
    funcResult >> nameReal;

    if (nameExpected == nameReal) cout << "Тест функции stringstream\t\t выполнен успешно." << endl;
    else cout << "Тест функции stringstream\t\t не выполнен успешно." << endl;
}
void Test_ReadFile(CList& list)
{
    string filename = "data.txt";
    list.readFile(filename);

    string expected = "Notepad ";
    string real = list.list[0].getName();

    if (expected == real) cout << "Тест функции чтения из файла\t\t выполнен успешно." << endl;
    else cout << "Тест функции чтения из файла\t\t не выполнен успешно." << endl;
}
void Test_RegexTask(CList& list)
{
    cout << "Здесь должны быть программы, содержащие в названии больше 2 слов:" << endl;
    list.regexTask();
}

```

List.h

```

#pragma once
#include "Program.h"

class CList
{ private:
    int listSize;

public:
    C_Program* list;
    void setListSize(int);
    int getListSize() const;

    void createList(int);
    void printAll() const;
    void printOneEl(int) const;
    void addEl(C_Program&);
    void deleteEl(int);
    int task(int);
    int linesInFile(string);
    void readFile(string);
    void saveToFile(string);
    stringstream getOneEl(int) const;
    void showOneEl(stringstream&) const;
    void enterNewEl();
    void regexTask();

    C_Program getProgramID(int) const;
    C_Program programs(int);
    ~CList();
};

```

List.cpp

```
#include "List.h"

void CList::createList(int value)
{
    listSize = value;
    list = new C_Program[listSize];

    for (int i = 0; i < listSize; i++)
        list[i] = programs(i);
}

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

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

void CList::printAll() const
{
    cout << endl;
    cout << "   Время\t   Размер\tСтроки\t   Троян\tИндекс\t\tНазвание";
    cout << endl;
    for (int i = 0; i < listSize; i++)
        printOneEl(i);

    cout << endl;
}

void CList::printOneEl(int number) const
{
    cout << setiosflags(ios::left) << setw(2) << number + 1 << " ";
    cout << setw(10) << list[number].getTime();
    cout << setw(12) << list[number].getSize();
    cout << setw(12) << list[number].getLines();
    cout << setw(12) << boolalpha << list[number].getTrojan();
    cout << setw(12) << list[number].getIndex();
    cout << setw(15) << list[number].getName() << endl;
}

void CList::addEl(C_Program& newProgram)
{
    C_Program* newList = new C_Program[listSize + 1];

    for (int i = 0; i < listSize; i++)
        newList[i] = list[i];
    newList[listSize++] = newProgram;
    delete[] list;

    list = new C_Program[listSize];
    for (int i = 0; i < listSize; i++)
        list[i] = newList[i];

    delete[] newList;
    cout << "Элемент добавлен." << endl;
}

void CList::deleteEl(int index)
{
    if (listSize == 0)
    {
        cout << "список программ пуст. возвращение с выбору действий." << endl;
        return;
    }
    if (index <= 0 || index > listSize)
    {
        cout << "ошибка. неверный номер элемента. возвращение." << endl;
        return;
    }
}
```

```

C_Program* newList = new C_Program[listSize - 1];

for (int i = 0; i < index - 1; i++)
    newList[i] = list[i];
for (int i = index - 1, j = index; j < listSize; i++, j++)
    newList[i] = list[j];
delete[] list;

list = new C_Program[listSize--];
for (int i = 0; i < listSize; i++)
    list[i] = newList[i];
delete[] newList;

return;
}
int CList::task(int minimalSize)
{
    int size = 0;

    for (int i = 0; i < listSize; i++)
        if (list[i].getSize() > minimalSize&& list[i].getTrojan() == false)
        {
            printOneEl(i);
            size++;
        }

    return size;
}
int CList::linesInFile(string filename)
{
    int size = 0;
    string line;
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]*)");

    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Невозможно открыть файл. Возвращение в меню." << endl;
        return 0;
    }

    while (getline(fin, line))
    {
        if (regex_match(line, regular)) size++;
        else cout << "Строка в файле не прошла проверку." << endl;
    }

    fin.close();
    return size;
}
void CList::readFile(string filename)
{
    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Неверное название файла. Повторите попытку." << endl;
        return;
    }

    string line, var;
    int size = CList::linesInFile(filename);
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]*)");
    int i = 0, a = 0, b;

    delete[] list;
    list = new C_Program[size];

```

```

while (getline(fin, line) && i < size)
{
    if (regex_match(line.c_str(), regular))
    {
        int time, size, lines, index;
        bool trojan;
        string name, name2;
        string trueFalse;
        std::istringstream temp(line);
        temp >> index;
        temp >> time;
        temp >> size;
        temp >> lines;
        temp >> trueFalse;
        temp >> name;
        temp >> name2;

        if (trueFalse == "true") trojan = true;
        else trojan = false;

        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);
    }
}

```

```

        C_Program newElement(trojan, time, size, lines, index, name);
        list[i++] = newElement;
    }
}

setListSize(size);
fin.close();
cout << endl << "Чтение из файла завершено." << endl;
}

void CList::saveToFile(string filename)
{
    std::ofstream fout(filename);

    fout.setf(ios::left);
    fout << "\tВремя\tРазмер\t\tСтроки\tТроян\t\tИндекс\tНазвание" << endl;
    for (int i = 0; i < getListSize(); i++)
    {
        fout << setw(2) << i + 1 << ") \t " << setw(9) << list[i].getTime() << setw(12);
        fout << list[i].getSize() << setw(11) << list[i].getLines() << setw(12);
        fout << boolalpha << list[i].getTrojan() << setw(11) << list[i].getIndex() <<
setw(15);
        fout << list[i].getName() << endl;
    }

    cout << "Запись в файл завершена." << endl;
    fout.close();
}

stringstream CList::getOneEl(int value) const
{
    stringstream temp;
    temp << " " << list[value].getName() << " " << list[value].getTrojan() << " " <<
list[value].getIndex() << " " << list[value].getLines() << " " << list[value].getSize() << " " <<
list[value].getTime();
    return temp;
}

void CList::showOneEl(stringstream& line) const
{
    int TimeOfWork, size, AmountOfLines, index;
    bool trojan;
    string name, name2;
    string trueFalse;

    line >> name;
    line >> name2;
    line >> trueFalse;
    line >> index;
    line >> AmountOfLines;
    line >> size;
    line >> TimeOfWork;

    if (trueFalse == "1") trojan = true;
    else trojan = false;

    if (name2 == "") name = name + " ";
    else(name = name + " " + name2);

    cout << "\n\tВремя\tРазмер\t\tСтроки\t\tТроян\t\tИндекс\t\tНазвание";
    cout << endl << setiosflags(ios::left) << setw(2) << 1 << ")";
    cout << setw(10) << TimeOfWork;
    cout << setw(12) << size;
    cout << setw(12) << AmountOfLines;
    cout << setw(12) << boolalpha << trojan;
    cout << setw(12) << index;
    cout << setw(15) << name;
    cout << endl;
}

C_Program CList::getProgramID(int id) const
{

```



```

C_Program newProgram;

for (int i = 0; i < listSize; i++)
    if (list[i].getIndex() == id)
    {
        printOneEl(i);
        newProgram = list[i];
        return newProgram;
    }
cout << "\nПрограммы с таким ID нету.\n" << endl;
return newProgram;
}
C_Program CList::programs(int valueX)
{
    C_Program standartProgram;

    if (valueX == 1)
    {
        C_Program Program1(true, 222, 222, 222, 1234, "Skype");
        return Program1;
    }
    else if (valueX == 2)
    {
        C_Program Program2(true, 333, 333, 666, 5678, "Standart Calculator");
        return Program2;
    }
    else if (valueX == 3)
    {
        C_Program Program3(false, 444, 444, 444, 9532, "Domino Super");
        return Program3;
    }
    else if (valueX == 4)
    {
        C_Program Program4(false, 555, 555, 555, 4356, "Text editor");
        return Program4;
    }
    return standartProgram;
}
void CList::enterNewEl()
{
    int time, size, lines, index;
    bool trojan;
    string name, name2, trojan2, data;
    regex regular("^[A-Z]+[\\w]* [\\w]*");

    cout << "Введите данные в линию в таком порядке:";
    cout << "Время Размер Строки Троян(true/false) Индекс Название" << endl;

    cin.ignore();
    getline(cin, data);
    std::istringstream temp(data);

    temp >> time;
    temp >> size;
    temp >> lines;
    temp >> trojan2;
    temp >> index;
    temp >> name;
    temp >> name2;

    if (name2 == "") name = name + " ";
    else(name = name + " " + name2);

    if (!regex_match(name.c_str(), regular))
    {
        cout << "Было введено неправильное имя. Формат имени:" << endl;
        cout << "Первое слово не должно начинаться с маленькой буквы." << endl;
        cout << "Не должно содержать символы." << endl;
        cout << "Завершение работы функции. Повторите попытку.";
    }
}

```

```

        return;
    }

    if (trojan2 == "true") trojan = true;
    else trojan = false;

    C_Program el(trojan, time, size, lines, index, name);
    addEl(el);
}

void CList::regexTask()
{
    regex regular("(^[A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]+)");
    int listSize = getListSize();

    for (int i = 0; i < listSize; i++)
        if (regex_match(list[i].getName(), regular))
            printOneEl(i);

    cout << endl;
}

CList::~CList()
{
    //cout << "\nВызвался деструктор" << endl;
    delete[] list;
}

```

Main.cpp

```

#include "Program.h"
#include "List.h"

void menu();

int main()
{
    setlocale(LC_ALL, "Rus");
    menu();

    if (_CrtDumpMemoryLeaks()) cout << endl << "Есть утечка памяти." << endl;
    else cout << endl << "Утечка памяти отсутствует." << endl;

    return 0;
}

void menu()
{
    CList list; //список элементов
    C_Program getProgram; //программа, которая вернётся при получении ID
    C_Program newProgram; //программа для добавления в список
    int choise = 1, value = 0, stop = 1;
    string fileName; //название файла
    string::size_type n;
    stringstream str;
    int size; //количество элементов больше определённого размера
    list.createList(4);
    cout << endl << "Выберите команду для работы со списком: ";

    while (stop != 0)
    {
        if (list.getListSize() == 0)
        {
            cout << "Список пуст. Добавить элемент(1) или закончить работу(0): " << endl;

```

```

        cin >> choise;
        cout << endl;

        if (choise == 1) choise = 11;
        else if (choise == 0) choise = 9;
        else cout << "Неверный символ." << endl;
    }
    else
    {
        cout << endl;
        cout << "1)Вывести всё на экран" << endl;
        cout << "2)Вывести 1 элемент на экран" << endl;
        cout << "3)Найти программу по индексу" << endl;
        cout << "4)Добавить элемент (в конец)" << endl;
        cout << "5)Удалить элемент" << endl;
        cout << "6)Получить список программ меньше определённого размера и не трояны"

<< endl;

        cout << "7)Получить данные из файла" << endl;
        cout << "8)Записать данные в файл" << endl;
        cout << "9)Завершение работы" << endl;
        cout << "10)Получить элемент класса из строки" << endl;
        cout << "11)Ввод данных с клавиатуры" << endl;
        cout << "12)Получить список программ, названия которых состоят из 2 слов" <<

endl;

        cout << "=====" << endl << "Ваш выбор: ";
        cin >> choise;
        cout << endl;
    }

    switch (choise)
    {
    case 1:
        list.printAll();
        break;
    case 2:
        cout << "Введите номер элемента, который надо вывести: ";
        cin >> value;
        cout << endl;

        if (value <= 0 || value > list.getListSize())
        {
            cout << "Неверный номер элемента. Повторите попытку." << endl;
            break;
        }
        list.printOneEl(value - 1);
        break;
    case 3:
        cout << "Введите id элемента, которого вы хотите получить: ";
        cin >> value;
        cout << endl;

        getProgram = list.getProgramID(value);
        break;
    case 4:
        list.addEl(newProgram);
        break;
    case 5:
        cout << "Введите номер элемента, который хотите удалить: ";
        cin >> value;
        cout << endl;

        list.deleteEl(value);
        break;
    case 6:
        cout << "Введите минимальный размер программ: ";
        cin >> value;
        cout << endl;

        size = list.task(value);
    }
}

```

```

        break;
    case 7:
        cout << "Введите название файла для чтения данных: ";
        cin >> fileName;
        cout << endl;

        n = fileName.find(".txt");
        if (n > 187) fileName += string(".txt");

        list.readFile(fileName);
        break;
    case 8:
        cout << "Введите название файла для записи данных: ";
        cin >> fileName;
        cout << endl;

        n = fileName.find(".txt");
        if (n > 187) fileName += string(".txt");

        list.saveToFile(fileName);
        break;
    case 9:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;
    case 10:
        cout << "Введите номер элемента, который вы хотите получить: ";
        cin >> value;
        cout << endl;

        str = list.getOneEl(value-1);
        list.showOneEl(str);
        break;
    case 11:
        list.enterNewEl();
        break;
    case 12:
        list.regexTask();
        break;
    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
    }
}
return;
}

```

program.cpp

```
#include "Program.h"
```

```

int C_Program::getTime() const
{
    return timeOfWork;
}
int C_Program::getSize() const
{
    return size;
}
int C_Program::getLines() const
{
    return amountOfLines;
}
int C_Program::getIndex() const
{
    return index;
}
bool C_Program::getTrojan()const

```

```

{
    return trojan;
}
string C_Program::getName()const
{
    return name;
}

void C_Program::setTime(const int valueTime)
{
    timeOfWork = valueTime;
}
void C_Program::setSize(const int valueSize)
{
    size = valueSize;
}
void C_Program::setLines(const int valueLines)
{
    amountOfLines = valueLines;
}
void C_Program::setTrojan(const bool trojanStatus)
{
    trojan = trojanStatus;
}
void C_Program::setIndex(int valueIndex)
{
    index = valueIndex;
}
void C_Program::setName(string valueName)
{
    name = valueName;
}

C_Program::C_Program(bool trojan, int time, int size, int lines, int index, string name) :
trojan(trojan), timeOfWork(time), size(size), amountOfLines(lines), index(index), name(name)
{
    //cout << "\nВызвался конструктор с параметрами";
}
C_Program::C_Program() : trojan(true), timeOfWork(0), size(0), amountOfLines(0), index(0101),
name("Basic")
{
    //cout << "\nВызвался конструктор по умолчанию.";
}
C_Program::C_Program(const C_Program& other) : trojan(other.trojan), timeOfWork(other.timeOfWork),
size(other.size), amountOfLines(other.amountOfLines), index(other.index), name(other.name)
{
    //cout << "\nВызвался конструктор копирования.";
}
C_Program::~C_Program()
{
    //cout << "\nВызвался деструктор";
}

```

Program.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <string.h>
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE__)

#include <string>
#include <iostream>
#include <iomanip>
#include <locale.h>
#include <fstream>
#include <sstream>

```

```
#include <regex>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;
using std::boolalpha;
using std::regex;
using std::ifstream;
using std::regex_match;
using std::regex_search;
using std::cmatch;
using std::setiosflags;
using std::ios;

class C_Program
{ private:
    int timeOfWork;      //average time of program execution
    int size;            //size of program
    int amountOfLines;   //number of lines in code
    int index;           //index
    bool trojan;         //trojan(yes or no)
    string name;         //name of program

public:
    int getTime() const;
    int getSize() const;
    int getLines() const;
    int getIndex()const;
    bool getTrojan()const;
    string getName() const;

    void setTime(const int);
    void setSize(const int);
    void setLines(const int);
    void setIndex(const int);
    void setTrojan(const bool);
    void setName(const string);

    C_Program();
    C_Program(bool, int, int, int, int, string);
    C_Program(const C_Program& other);
    ~C_Program();
};
```

4. Результати роботи програми

```
7)Получить данные из файла
8)Записать данные в файл
9)Завершение работы
10)Получить элемент класса из строки
11)Ввод данных с клавиатуры
12)Получить список программ, названия которых состоят из 2 слов
=====
Ваш выбор: 7

Введите название файла для чтения данных: data

Строка в файле не прошла проверку.

Чтение из файла завершено.

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Найти программу по индексу
4)Добавить элемент (в конец)
5)Удалить элемент
6)Получить список программ меньше определённого размера и не трояны
7)Получить данные из файла
8)Записать данные в файл
9)Завершение работы
10)Получить элемент класса из строки
11)Ввод данных с клавиатуры
12)Получить список программ, названия которых состоят из 2 слов
=====
Ваш выбор: 1

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Найти программу по индексу
4)Добавить элемент (в конец)
5)Удалить элемент
6)Получить список программ меньше определённого размера и не трояны
7)Получить данные из файла
8)Записать данные в файл
9)Завершение работы
10)Получить элемент класса из строки
11)Ввод данных с клавиатуры
12)Получить список программ, названия которых состоят из 2 слов
=====
Ваш выбор: 11

Введите данные в линию в таком порядке:Время Размер Строки Троян(true/false) Индекс Название
111 222 333 true 444 Hello World55
Элемент добавлен.

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Найти программу по индексу
4)Добавить элемент (в конец)
5)Удалить элемент
6)Получить список программ меньше определённого размера и не трояны
7)Получить данные из файла
8)Записать данные в файл
9)Завершение работы
10)Получить элемент класса из строки
11)Ввод данных с клавиатуры
12)Получить список программ, названия которых состоят из 2 слов
=====
Ваш выбор: 1
```

Время	Размер	Строки	Троян	Индекс	Название
1)222	333	444	false	14212	Notepad
2)666	666	666	true	666	Photo-shop; CPlus54
3)111	222	333	true	444	Hello World55

```

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Найти программу по индексу
4)Добавить элемент (в конец)
5)Удалить элемент
6)Получить список программ меньше определённого размера и не трояны
7)Получить данные из файла
8)Записать данные в файл
9)Завершение работы
10)Получить элемент класса из строки
11)Ввод данных с клавиатуры
12)Получить список программ, названия которых состоят из 2 слов
=====
Ваш выбор: 12

2 )666      666      666      true      666      Photo-shop; CPlus54
3 )111      222      333      true      444      Hello World55

```

```

1 256 7777 24 666 true cHESS super
2 14212 222 333 444 false Notepad
3 666 666 666 666 true Photo--shop;; CPlus54

```

Текстовий файл з вхідними даними

```

Тест получения элемента по ID      выполнен успешно.
Элемент добавлен.
Тест добавления элемента в список  выполнен успешно.
Тест функции удаления              выполнен успешно.

3 )444      444      444      false     9532      Domino Super
4 )555      555      555      false     4356      Text editor

Тест функции нахождения элементов по параметру выполнен успешно.
Тест функции stringstream          выполнен успешно.
Строка в файле не прошла проверку.

Чтение из файла завершено.
Тест функции чтения из файла      выполнен успешно.
Здесь должны быть программы, содержащие в названии больше 2 слов:
2 )666      666      666      true      666      Photo-shop; CPlus54

```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з регулярними виразами.

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