

Звіт

Автор: Момот Р.Є. КІТ-119а

Дата: 25 лютого 2020

Лабораторна робота 3. ПОТОКИ

Тема: Робота з потоками: потокове введення / виведення на консоль та у файл, рядки типу *string*, *stringstream*.

Мета: отримати знання про основи роботи з поточковим введенням / виведенням на мові C++, роботу з файлами та рядками типу *string*.

1. Завдання до роботи

Загальне завдання. Поширити попередню лабораторну роботу таким чином:

- використання функцій *printf/scanf* замінити на використання *cin/cout*;
- усі конкатенації рядків замінити на використання *stringstream*;
- замінити метод виводу інформації про об'єкт на метод, що повертає рядок інформацію про об'єкт, який далі можна виводити на екран;
- замінити метод вводу інформації про об'єкт на метод, що приймає рядок з інформацією про об'єкт, обробляє його та створює об'єкт на базі цієї інформації;
- поширити клас-список, шляхом реалізації методів роботи з файлами за допомогою файлових потоків (*fstream*) (якщо використовувалися функції *fprintf/fscanf* – замінити їх на класи *ifstream/ofstream*), при цьому сигнатури методів повинні виглядати таким чином:

- читання: `void CList::readFromFile(string fileName);` де *CList* – клас-список об'єктів, при цьому слід пам'ятати, що при повторному читанні з файлу, попередні дані списку повинні бути очищені;

- запис: `void CList::writeToFile(string fileName);`

2.1 Опис класів

Базовий клас: *C_Program*

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

2.1 Опис змінних

int *TimeOfWork* – поле класу *C_Program* (час виконання програми).
int *size* – поле класу *C_Program* (розмір програми у мегабайтах).
int *AmountOfLines* – поле класу *C_Program* (кількість рядків коду).
int *index* – поле класу *C_Program* (ідентифікаційний номер).
bool *trojan* – поле класу *C_Program* (троян чи ні).
string *name* – поле класу *C_Program* (назва програми).
int *listSize* – поле класу *C_List* (розмір масиву елементів класу *C_Program*).
*C_Program** *list* – поле класу *C_Program* (масив елементів класу *C_Program*).
C_List *List* – об'єкт класу *C_List*.

`string filename` – змінна назви файлу для роботи з ним.
`C_Program newProgram` – змінна елементу програми для додавання у список.
`C_Program getProgram` – змінна елементу програми, у яку записується програма, повернена за ID.

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`).
`void Task(int)` – знаходження елементів за певним критерієм (метод класу `C_List`).
`void 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 файлу (метод класу `C_List`).
`void ReadFile(string)` – виведення даних з файлу (метод класу `C_List`).
`void SaveToFile(string)` – введення даних з файлу (метод класу `C_List`).
`~C_List()` – деструктор списку елементів (метод класу `C_List`).
`C_Program()` – конструктор без параметра (метод класу `C_Program`).
`C_Program(char*, int, int, int, int)` – конструктор класу з параметрами (метод класу `C_Program`).
`C_Program(const C_Program& other)` – конструктор копіювання (метод класу `C_Program`).
`~C_Program()` – деструктор елементу (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_DeEl(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).

3. Текст програми

test.cpp

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

void Test_GetProgramID(CList&, int&);
void Test_AddEl(CList&);
void Test_DelEl(CList&);
void Test_Task(CList&, int&);
void Test_Stringstream(CList&);
void Test_ReadFile(CList& list);

int main() {
    setlocale(LC_ALL, "Rus");
    CList list;
    list.CreateList(5);

    int values[] = { 5678, 200 };

    Test_GetProgramID(list, values[0]);
    Test_AddEl(list);
    Test_DelEl(list);
    list.PrintAll();
    Test_Task(list, values[1]);
    Test_Stringstream(list);
    list.PrintAll();
    Test_ReadFile(list);
    list.PrintAll();

    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& value)
{
    cout << "\n\nЗдесь должен быть элемент с идентификатором 5678:" << endl;
    list.GetProgramID(value);
}

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

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

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

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

void Test_Task(CList& list, int& value)
{
    cout << "\n\nЗдесь должны быть элементы размером больше 200 и не трояны:\n";
    list.Task(value);
}

void Test_Stringstream(CList& list)
{
    string nameExpected = "Скайп";
    stringstream funcResult = list.GetOneEl(1);
```

```

    string nameReal;
    funcResult >> nameReal;

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

    cout << "Если данные ниже соответствуют данным в файле, то тест пройден" << endl;
}

```

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);
    void Task(int);
    int LinesInFile(string);
    void ReadFile(string);
    void SaveToFile(string);
    stringstream GetOneEl(int) const;
    void showOneEl(stringstream&) const;

    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 << "\n    Время    Размер\tСтроки\t    Троян\tИндекс\t    Название";
    for (int i = 0; i < listSize; i++)
        PrintOneEl(i);
}
void CList::PrintOneEl(int number) const
{
    cout << endl << std::setiosflags(std::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) << std::boolalpha << list[number].getTrojan();
    cout << setw(12) << list[number].getIndex();
    cout << setw(15) << list[number].getName();
}
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;
}
void CList::Task(int minimalSize)
{
    for (int i = 0; i < listSize; i++)
        if (list[i].getSize() > minimalSize && list[i].getTrojan() == false)
            PrintOneEl(i);
}
int CList::LinesInFile(string filename)
{
    int size = 0;
    string line;

```

```

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

while (getline(fin, line)) size++;

fin.close();
return size;
}

void CList::ReadFile(string filename)
{
    std::ifstream fin(filename);
    int size2 = CList::LinesInFile(filename);

    if (!fin.is_open()) return;

    delete[] list;
    list = new C_Program[size2];
    for (int i = 0; i < size2; i++)
    {
        int TimeOfWork, size, AmountOfLines, index;
        bool trojan;
        string name;
        string trueFalse;

        fin >> name;
        fin >> index;
        fin >> size;
        fin >> TimeOfWork;
        fin >> AmountOfLines;
        fin >> trueFalse;

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

        C_Program newElement(trojan, TimeOfWork, size, AmountOfLines, index, name);
        list[i] = newElement;
    }
    setListSize(size2);
    fin.close();
    cout << endl << "Чтение из файла завершено." << endl;
}

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

    fout.setf(std::ios::left);
    fout << "\tВремя\tРазмер\t\t\tСтроки\tТроян\t\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 << std::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;
        string trueFalse;

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

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

        cout << "\n   Время   Размер\tСтроки\t   Троян\tИндекс\t   Название";
        cout << endl << std::setiosflags(std::ios::left) << setw(2) << 1 << " ";
        cout << setw(10) << TimeOfWork;
        cout << setw(12) << size;
        cout << setw(12) << AmountOfLines;
        cout << setw(12) << std::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, "Скайн");
            return Program1;
        }
        else if (valueX == 2)
        {
            C_Program Program2(true, 333, 333, 666, 5678, "Калькулятор");
            return Program2;
        }
        else if (valueX == 3)
        {

```

```

        C_Program Program3(false, 444, 444, 444, 9532, "Домино");
        return Program3;
    }
    else if (valueX == 4)
    {
        C_Program Program4(false, 555, 555, 555, 4356, "Редактор Текста");
        return Program4;
    }
    return standartProgram;
}
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;
    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;               //переменная для названия файла
    string::size_type n;
    stringstream str;
    list.CreateList(4);
    cout << endl << "Выберите команду для работы со списком: " << endl;

    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 = 3;
            else if (choise == 0) choise = 5;
            else cout << "Неверный символ." << endl;
        }
        else
        {
            cout << endl << 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 << "===== " << 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;

        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;
    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() //деструктор
{
    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>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;

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

Выберите команду для работы со списком:

```

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
=====
Ваш выбор: 1

```

	Время	Размер	Строки	Троян	Индекс	Название
1)	0	0	0	true	0	Калькулятор
2)	222	222	222	true	1234	Скайп
3)	333	333	666	true	5678	Калькулятор
4)	444	444	444	false	9532	Домино

	Время	Размер	Строки	Троян	Индекс	Название
1)	0	0	0	true	0	Калькулятор
2)	222	222	222	true	1234	Скайп
3)	333	333	666	true	5678	Калькулятор
4)	444	444	444	false	9532	Домино

```

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
=====
Ваш выбор: 2
Введите номер элемента, который надо вывести: 3

```

3)	333	333	666	true	5678	Калькулятор
-----	-----	-----	-----	------	------	-------------

	Время	Размер	Строки	Троян	Индекс	Название
1)	0	0	0	true	0	Калькулятор
2)	222	222	222	true	1234	Скайп
3)	333	333	666	true	5678	Калькулятор
4)	444	444	444	false	9532	Домино

```

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
=====
Ваш выбор: 8
Введите название файла для записи данных: data2
Запись в файл завершена.

```

	Время	Размер	Строки	Троян	Индекс	Название
1)	0	0	0	true	0	Калькулятор
2)	222	222	222	true	1234	Скайп
3)	333	333	666	true	5678	Калькулятор
4)	444	444	444	false	9532	Домино

Текстовий файл після виведення

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з потоками.

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

