

Звіт

Автор: Богданов І.Ю. КІТ-119а Дата: 8 березня 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 потоковим введенням / виведенням на мові C++, роботу з файлами та рядками типу `string`.

1. Завдання до роботи **Індивідуальне завдання:**

Змінити попередню лабораторну роботу для роботи з потоками і строками.

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: `file`

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

2.2 Опис змінних

`int` `x_version` – поле класу `file` (64 або 32 розрядна програма.).

`int` `size` – поле класу `file` (розмір файлу у бітах).

`int` `index` – поле класу `file` (унікальний індекс).

`bool` `is_system` – поле класу `file` (чи є файл системним).

`bool` `is_hidden` – поле класу `file` (чи є файл прихованим).

`std::string` `name` – поле класу `file` (назва файлу).

`int` `next_ind` – поле класу `dir` (номер наступного файлу у директорії).

`int` `new_ind` – поле класу `dir` (індекс наступного файлу у директорії).

`file*` `files` – поле класу `dir` (масив елементів класу `file`).

`file*` `copy` – поле класу `dir` (показчик на клас `file`, використовується для правильної роботи деяких методів).

2.3 Опис методів

`int get_x() const` – отримання значення поля `x_version` змінної класу `file`(метод класу `file`).

`int get_size() const` – отримання значення поля `size` змінної класу `file`(метод класу `file`).

`int get_index() const` – отримання значення поля `index` змінної класу `file`(метод класу `file`).

`bool get_sys() const` – отримання значення поля `is_system` змінної класу `file`(метод класу `file`).

`bool get_hid() const` – отримання значення поля `is_hidden` змінної класу `file`(метод класу `file`).

`std::string get_name() const` – отримання значення поля `name` змінної класу `file`(метод класу `file`).

`void change_x(const int &x)` – зміна значення поля `x_version` змінної класу `file`(метод класу `file`).

`void change_size(const int &sz)` – зміна значення поля `size` змінної класу `file`(метод класу `file`).

`void change_index(const int &in)` – зміна значення поля `index` змінної класу `file`(метод класу `file`).

`void change_sys(const bool&)` – зміна значення поля `is_system` змінної класу `file`(метод класу `file`).

`void change_hid(const bool&)` – зміна значення поля `is_hidden` змінної класу `file`(метод класу `file`).

`void change_name(const std::string&)` – зміна значення поля `name` змінної класу `file`(метод класу `file`).

`file()` – конструктор класу `file`.

`file(const file&)` – конструктор копіювання класу `file`.

`file(const int&, const int&, const int&, const bool&, const bool&, const std::string&)` – конструктор з параметрами класу `file`.

`~file()` – деструктор класу `file`.

`void add_file(const file &f)` – додавання об'єкту класу `file` до масиву в класі `dir`(метод класу `dir`).

`void del_file(const int &index)` – видалення об'єкту класу `file` з масиву в класі `dir`(метод класу `dir`).

`void del_all()` – видалення усіх об'єктів класу `file` з масиву в класі `dir`(метод класу `dir`).

`void add_file_by_str(const std::string& str)` – додавання об'єкту класу `file` до масиву в класі `dir` за допомогою строки з інформацією про об'єкт(метод класу `dir`).

`void read_from_file(const std::string& name)` – заповнення масиву об'єктів класу `file` інформація про які буде зчитана з файлу(метод класу `dir`).

`file get_file_by_index(const int& index) const` – отримання об'єкту класу `file` з масиву в класі `dir` (метод класу `dir`).

`void get_file_to_screen(const int &index) const` – виведення об'єкту класу `file` з масиву в класі `dir` на екран(метод класу `dir`).

`void print_all() const` – виведення усіх об'єктів класу `file` з масиву в класі `dir` на екран(метод класу `dir`).

`int count_system() const` – розрахування кількості скритих і системних файлів в об'єкті класу `dir`(метод класу `dir`).

`void print_to_file(const std::string& name) const` – запис у файл інформації про об'єкти класу `file` що є в масиві(метод класу `dir`).

2.4 Опис функцій

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа 3.cpp

```
#include "menu.h"
#define _CRTDBG_MAP_ALLOC

int main() {
    menu();
    if (_CrtDumpMemoryLeaks()) {
        std::cout << "Утечка памяти обнаружена." << "\n";
    }
    else {
        std::cout << "Утечка памяти не обнаружена." << "\n";
    }
}

file.h
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>

class file {
private:
    int x_version; // x64/x32
    int size;
    int index;
    bool is_system;
    bool is_hidden;
    std::string name;
public:
    int get_x() const;
    int get_size() const;
    int get_index() const;
    bool get_sys() const;
    bool get_hid() const;
    std::string get_name() const;
    void change_x(const int&);
    void change_size(const int&);
    void change_index(const int&);
    void change_sys(const bool&);
    void change_hid(const bool&);
    void change_name(const std::string&);
    file();
    file(const file&);
```

```

        file(const int&, const int&, const int&, const bool&, const bool&, const std::string&);
        ~file();
};
file.cpp
#include "file.h"

int file::get_x() const {
    return x_version;
}
int file::get_size() const {
    return size;
}
int file::get_index() const {
    return index;
}
bool file::get_sys() const {
    return is_system;
}
bool file::get_hid() const {
    return is_hidden;
}
std::string file::get_name() const {
    return name;
}
void file::change_x(const int &x) {
    x_version = x;
}
void file::change_size(const int& sz) {
    size = sz;
}
void file::change_index(const int &in) {
    index = in;
}
void file::change_sys(const bool &sys) {
    is_system = sys;
}
void file::change_hid(const bool &hid) {
    is_hidden = hid;
}
void file::change_name(const std::string &nm) {
    name = nm;
}
file::file() {
    x_version = 32;
    size = 100;
    index = 0;
    is_system = false;
    is_hidden = false;
    name = "file";
    std::cout << "Файл создан при помощи конструктора поумолчанию." << "\n";
}
file::file(const file &f) {
    x_version = f.get_x();
    size = f.get_size();
    index = f.get_index();
    is_system = f.get_sys();
    is_hidden = f.get_hid();
    name = f.get_name();
}
file::file(const int &ver, const int &sz, const int &ind, const bool &sys, const bool &hid, const
std::string &nm) {
    x_version = ver;
    size = sz;
    index = ind;
    is_system = sys;
    is_hidden = hid;
    name = nm;
    std::cout << "Файл создан при помощи конструктора с аргументами." << "\n";
}

```

```

file::~file() {
    std::cout << "Файл уничтожен при помощи деструктора по умолчанию." << "\n";
}
dir.h
#pragma once
#include "file.h"

class dir {
private:
    file* files;
    file* copy;
    int next_ind = 0;
    int new_ind = 1;
public:
    void add_file(const file &f);
    void del_file(const int &index);
    void del_all();
    void add_file_by_str(const std::string& str);
    void read_from_file(const std::string& name);
    void get_file_to_screen(const int &index) const;
    std::string get_file_to_str(const int& index) const;
    file get_file_by_index(const int& index) const;
    void print_all() const;
    int count_system() const;
    void print_to_file(const std::string& name) const;
};
dir.cpp
#include "dir.h"

void dir::add_file(const file &f) {
    if (next_ind == 0) {
        files = new file[1];
        files[next_ind] = f;
        files[next_ind].change_index(new_ind);
        new_ind++;
        next_ind++;
    }
    else {
        copy = new file[next_ind];
        for (int i = 0; i < next_ind; i++) {
            copy[i] = files[i];
        }
        delete[] files;
        files = new file[next_ind + 1];
        for (int i = 0; i < next_ind; i++) {
            files[i] = copy[i];
        }
        delete[] copy;
        files[next_ind] = f;
        files[next_ind].change_index(new_ind);
        new_ind++;
        next_ind++;
    }
}

void dir::del_file(const int &index) {
    if (next_ind == 1) {
        delete[] files;
        next_ind--;
    }
    else {
        copy = new file[next_ind - 1];
        for (int i = 0; i < index; i++) {
            copy[i] = files[i];
        }
        for (int i = index + 1; i < next_ind; i++) {
            copy[i - 1] = files[i];
        }
        delete[] files;
        files = new file[next_ind - 1];
    }
}

```

```

        for (int i = 0; i < next_ind; i++) {
            files[i] = copy[i];
        }
        delete[] copy;
        next_ind--;
    }
}

void dir::del_all() {
    if (next_ind != 0) {
        delete[] files;
        next_ind = 0;
    }
}

void dir::read_from_file(const std::string& name) {
    del_all();
    std::ifstream f;
    char* temp;
    f.open(name);
    while (!f.eof()) {
        temp = new char[100];
        f.getline(temp, 100);
        add_file_by_str(temp);
        delete[] temp;
    }
    f.close();
}

std::string dir::get_file_to_str(const int &index) const {
    std::stringstream s;
    s << files[index].get_name() << " " << files[index].get_index() << " " <<
files[index].get_size() << " " << files[index].get_x() << " " << files[index].get_hid() << " " <<
files[index].get_sys();
    return s.str();
}

void dir::print_all() const {
    for (int i = 0; i < next_ind; i++) {
        std::cout << i + 1 << " ";
        std::string str;
        str = get_file_to_str(i);
        std::cout << str << "\n";
    }
}

int dir::count_system() const {
    int count = 0;
    for (int i = 0; i < next_ind; i++) {
        if (files[i].get_hid() && files[i].get_sys()) {
            count++;
        }
    }
    return count;
}

file dir::get_file_by_index(const int& index) const {
    for (int i = 0; i < next_ind; i++) {
        if (files[i].get_index() == index) {
            return files[i];
        }
    }
}

void dir::add_file_by_str(const std::string& str) {
    int i = str.find(" ");
    std::string name = str.substr(0, i);
    int i2 = str.find(" ", i + 1);
    std::string temp = str.substr(i + 1, i2 - i);
    std::stringstream s;
    s << temp;
    int index = 0;
    s >> index;
    int i3 = str.find(" ", i2 + 1);
    s.clear();
    temp = str.substr(i2 + 1, i3 - i2);

```

```

s << temp;
int size = 0;
s >> size;
int i4 = str.find(" ", i3 + 1);
s.clear();
temp = str.substr(i3 + 1, i4 - i3);
s << temp;
int x = 0;
s >> x;
int i5 = str.find(" ", i4 + 1);
s.clear();
temp = str.substr(i4 + 1, i5 - i4);
s << temp;
bool hid = false;
s >> hid;
int i6 = str.find(" ", i5 + 1);
s.clear();
temp = str.substr(i5 + 1, i6 - i5);
s << temp;
bool sys = false;
s >> sys;
file new_file(x, size, index, sys, hid, name);
add_file(new_file);
}
void dir::print_to_file(const std::string& name) const {
    std::ofstream f;
    f.open(name);
    std::string str;
    for (int i = 0; i < next_ind; i++) {
        str = get_file_to_str(i);
        f << str;
        if (i != next_ind - 1) {
            f << "\n";
        }
    }
    f.close();
}

```

menu.h

#pragma once

#include "dir.h"

void menu();

menu.cpp

#include "menu.h"

```

void menu() {
    setlocale(LC_ALL, "Russian");
    int n = 0, temp_i;
    dir directory;
    directory.read_from_file("data.txt");
    while (n != 4) {
        std::cout << "Выберите желаемую опцию:" << "\n";
        std::cout << "1 - добавить элемент в список." << "\n";
        std::cout << "2 - удалить элемент из списка." << "\n";
        std::cout << "3 - показать все элементы списка." << "\n";
        std::cout << "4 - завершить работу программы." << "\n";
        std::cout << "5 - посчитать количество скрытых системных файлов." << "\n";
        std::cout << "6 - прочитать данные из файла. " << "\n";
        std::cout << "7 - записать текущий список данных в файл. " << "\n";
        std::cin >> n;
        if (n == 1) {
            directory.add_file_by_str("file 0 123 64 0 0 \0");
            std::cout << "Файл добавлен." << "\n";
        }
        else if (n == 2) {
            std::cout << "Введите номер удаляемого элемента (нумерация начинается с 1): ";
            std::cin >> temp_i;
            directory.del_file(temp_i - 1);
            std::cout << "Файл удалён. " << "\n";
        }
    }
}

```

```

    }
    else if (n == 3) {
        directory.print_all();
    }
    else if (n == 5) {
        std::cout << "Количество скрытых системных файлов: " << directory.count_system() <<
"\n";
    }
    else if (n == 6) {
        directory.read_from_file("data.txt");
    }
    else if (n == 7) {
        directory.print_to_file("data.txt");
    }
}
directory.del_all();
}
test.cpp
#include "menu.h"
#define _CRTDBG_MAP_ALLOC

int main() {
    setlocale(LC_ALL, "Russian");
    file test_file;
    test_file.change_index(1);
    test_file.change_size(2);
    test_file.change_x(32);
    test_file.change_name("test");
    test_file.change_hid(true);
    test_file.change_sys(false);
    if ((test_file.get_index() == 1) && (test_file.get_size() == 2) && (test_file.get_x() == 32))
{
        std::cout << "Первый тест на работу геттеров и сеттеров базового класса пройден
успешно." << "\n";
    }
    else {
        std::cout << "Первый тест на работу геттеров и сеттеров базового класса провален." <<
"\n";
    }
    dir test_dir;
    test_dir.add_file(test_file);
    test_dir.print_all();
    std::cout << "Если перед этим сообщением на экран вывелась информация о файле методы
add_file, print_all и get_file_to_screen работают корректно." << "\n";
    test_dir.del_all();
    test_dir.print_all();
    std::cout << "Если перед этим сообщение на экран не выводились новые числа то методы del_all
и del_file работают корректно." << "\n";
    test_dir.read_from_file("data.txt");
    test_dir.print_all();
    std::cout << "Если перед этим сообщением на экран вывелась информация о файлах, то методы
чтения с файла и получение файла из строки работают правильно." << "\n";
    test_dir.add_file(test_file);
    test_dir.print_to_file("data.txt");
    test_dir.read_from_file("data.txt");
    test_dir.print_all();
    std::cout << "Если перед этим сообщение на экран вывелась информация о всех предыдущих файлах
и одном новом, то метод записи информации в файл работает правильно." << "\n";
    if (_CrtDumpMemoryLeaks()) {
        std::cout << "Утечка памяти обнаружена." << "\n";
    }
    else {
        std::cout << "Утечка памяти не обнаружена." << "\n";
    }
    int t;
    std::cin >> t;
}
data.txt
sys 2 123 32 1 1

```


qwerty 3 521 64 1 0
hello 4 289 64 0 1
water 5 10000 32 0 0

4. Результати роботи програми

Результати роботи програми:

```
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл создан при помощи конструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Выберите желаемую опцию:
1 - добавить элемент в список.
2 - удалить элемент из списка.
3 - показать все элементы списка.
4 - завершить работу программы.
5 - посчитать количество скрытых системных файлов.
6 - прочитайте данные из файла.
7 - записать текущий список данных в файл.
4
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Файл уничтожен при помощи деструктора поумолчанию.
Утечка памяти не обнаружена.
```

Результати тестів:

```

файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл создан при помощи конструктора с аргументами.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл создан при помощи конструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
файл уничтожен при помощи деструктора поумолчанию.
1 sys 7 123 32 1 1
2 qwerty 8 521 64 1 0
3 hello 9 289 64 0 1
4 water 10 10000 32 0 0
5 test 11 2 32 1 0
Если перед этим сообщение на экран вывелась информация о всех предыдущих файлах и одном новом, то метод записи информации в файл работает правильно.
/течка памяти не обнаружена.

```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з потоками, строками та файлами.

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