Звіт

Автор: Богданов І.Ю. КІТ-119а Дата: 8 березня 2020

Лабораторна робота №2. Перевантаження методів.

Тема. Класи. Конструктори та деструктори. Перевантаження методів.

Мета: отримати базові знання про класи, конструктори та деструктори.

Дослідити механізм створення та видалення об’єктів.

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

Знати кількість файлів що мають атбрибут прихований і є системними.

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 программа.).

`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).

`const char* 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).

`const char* 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 char*)` – 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 char*)` – 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).

`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).

2.4 Опис функцій

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа 2.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 <cstring>
#include <iostream>

class file {
private:
	int x_version; // x64/x32
	int size;
	int index;
	bool is_system;
	bool is_hidden;
	const char* name;
public:
	int get_x() const;
	int get_size() const;
	int get_index() const;
	bool get_sys() const;
	bool get_hid() const;
	const char* 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 char*);
	file();
	file(const file&);
	file(const int&, const int&, const int&, const bool&, const bool&, const char*);
	~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;
}
const char* 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 char* 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();
	int i = 0;
	name = f.get_name();
}
file::file(const int &ver, const int &sz, const int &ind, const bool &sys, const bool &hid, const
char *nm) {
	x_version = ver;
	size = sz;
	index = ind;
	is_system = sys;
	is_hidden = hid;
	int i = 0;
	name = nm;
	std::cout << "Файл создан при помощи конструктора с аргументами." << "\n";
}
file::~file() {
	std::cout << "Файл уничтожен при помощи деструктора поумолчанию." << "\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 get_file_to_screen(const int &index) const;
	file get_file_by_index(const int& index) const;
	void print_all() const;
	int count_system() const;
};
```

dir.cpp

```
#include "dir.h"

void dir::add_file(const file &f) {
	if (next_ind == 0) {
		files = (file*)malloc(sizeof(file));
		files[next_ind] = f;
		files[next_ind].change_index(new_ind);
		new_ind++;
		next_ind++;
	}
	else {
		copy = (file*)malloc(sizeof(file) * (next_ind + 1));
		for (int i = 0; i < next_ind; i++) {
			copy[i] = files[i];
		}
		free(files);
		files = (file*)malloc(sizeof(file) * (next_ind + 1));
		for (int i = 0; i < next_ind; i++) {
			files[i] = copy[i];
		}
		free(copy);
		files[next_ind] = f;
		files[next_ind].change_index(new_ind);
		next_ind++;
		new_ind++;
	}
}
void dir::del_file(const int &index) {
	if (next_ind == 1) {
		free(files);
		next_ind--;
	}
	else {
		copy = (file*)malloc(sizeof(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];
		}
		free(files);
		files = (file*)malloc(sizeof(file) * (next_ind - 1));
		for (int i = 0; i < next_ind; i++) {
			files[i] = copy[i];
		}
		free(copy);
		next_ind--;
	}
}
void dir::del_all() {
	free(files);
	next_ind = 0;
}
void dir::get_file_to_screen(const int &index) const {
```

```
	std::cout << files[index].get_name() << " " << files[index].get_index() << " " <<
files[index].get_size() << " x" << files[index].get_x() << " " << files[index].get_hid() << " " <<
files[index].get_sys() << "\n";
}
void dir::print_all() const {
	for (int i = 0; i < next_ind; i++) {
		std::cout << i + 1 << " ";
		get_file_to_screen(i);
	}
}
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];
		}
	}
}
```

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;
	file temp_file(123, 32, 0, true, true, "sys\0");
	directory.add_file(temp_file);
	temp_file.change_size(521);
	temp_file.change_x(64);
	temp_file.change_hid(false);
	temp_file.change_sys(true);
	temp_file.change_name("qwerty\0");
	directory.add_file(temp_file);
	temp_file.change_size(289);
	temp_file.change_x(64);
	temp_file.change_hid(true);
	temp_file.change_sys(false);
	temp_file.change_name("hello\0");
	directory.add_file(temp_file);
	temp_file.change_size(10000);
	temp_file.change_x(32);
	temp_file.change_hid(false);
	temp_file.change_sys(false);
	temp_file.change_name("water\0");
	directory.add_file(temp_file);
	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::cin >> n;
		if (n == 1) {
			temp_file.change_size(123);
			temp_file.change_x(64);
```

```
                        temp_file.change_hid(false);
                        temp_file.change_sys(false);
                        temp_file.change_name("file");
                        directory.add_file(temp_file);
                        std::cout << "Файл добавлен." << "\n";
                }
                else if (n == 2) {
                        std::cout << "Введите номер удалемого элемента (нумерация начинаеться 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";
                }
        }
        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);
        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, а так же базового
конструктора базового класса пройден успешно." << "\n";
        }
        else {
                std::cout << "Первый тест на работу геттеров и сеттеров, а так же базового
конструктора 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";
        file second_test_file(1, 2, 3, true, true, "name");
        test_dir.add_file(second_test_file);
        if (test_dir.count_system() == 1) {
                std::cout << "Проверка работы метода count_system и конструктора с аргументами
пройдена." << "\n";
        }
        else {
                std::cout << "Проверка работы метода count_system и конструктора с аргументами
провалена." << "\n";
        }
        if (_CrtDumpMemoryLeaks()) {
                std::cout << "Утечка памяти обнаружена." << "\n";
        }
        else {
                std::cout << "Утечка памяти не обнаружена." << "\n";
        }
        int t;
        std::cin >> t;
}
```

4. Результати роботи програми

Результати роботи програми:

```
Файл создан при помощи конструктора с аргументами.
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.
3
1 sys 1 32 x123 1 1
2 qwerty 2 521 x64 0 1
3 hello 3 289 x64 1 0
4 water 4 10000 x32 0 0
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.
4
Файл уничтожен при помощи деструктора поумолчанию.
Утечка памяти не обнаружена.
```

Результати тестів:

```
Файл создан при помощи конструктора поумолчанию.
Первый тест на работу геттеров и сеттеров, а так же базового конструктора базового класса пройден успешно.
1 file 1 2 x32 0 0
Если перед этим сообщением на экран вывелась информация о файле методы add_file, print_all и get_file_to_screen работают
 корректно.
Если перед этим сообщение на экран не выводились новые числа то методы del_all и del_file работают корректно.
Файл создан при помощи конструктора с аргументами.
Проверка работы метода count_system и конструктора с аргументами пройдена.
Утечка памяти не обнаружена.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з класами та їх конструкторами та деструкторами.

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