Звіт

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

Лабораторна робота №1. Класи

Тема. Класи та специфікатори доступу. Інкапсуляція. Константи. **Мета:**

отримати базові знання про класи. Дослідити механізм інкапсуляції.

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

Для предметної галузі розробити два класи:

- клас, що відображає сутність «базового класу». При цьому, в даному класі повинно бути мінімум три числових поля (бажано, щоб одне з цих полів було унікальним ідентифікатором об’єкта);
- клас, що має у собі динамічний масив об’єктів базового класу та має в собі методи додавання, видалення елемента, отримання елемента по індексу (або ідентифікатору), вивід усіх елементів на екран.

Прикладна галузь: директорія ПК
Базовий клас: файл

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).

`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).

`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 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).

2.4 Опис функцій

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа 1.cpp

```
#include <iostream>
#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
```

```
#include <string>
#include <iostream>

class file {
private:
	int x_version; // x64/x32
	int size;
	int index;
public:
	int get_x() const;
	int get_size() const;
	int get_index() const;
	void change_x(const int &x);
	void change_size(const int &sz);
	void change_index(const int &in);
};
```
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;
}
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;
}
```
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();
	file get_file_by_index(const int& index) const;
	void get_file_to_screen(const int &index) const;
	void print_all() 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 && index == 0) {
                free(files);
                next_ind--;
        }
        else if (index >= 0) {
                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--;
        }
}
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::del_all() {
        free(files);
        next_ind = 0;
}
void dir::get_file_to_screen(const int &index) const {
        std::cout << files[index].get_index() << " " << files[index].get_size() << " x" <<
files[index].get_x() << "\n";
}
void dir::print_all() const {
        for (int i = 0; i < next_ind; i++) {
                std::cout << i + 1 << " ";
                get_file_to_screen(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;
    temp_file.change_size(1234);
    temp_file.change_x(32);
    directory.add_file(temp_file);
    temp_file.change_size(521);
    temp_file.change_x(64);
    directory.add_file(temp_file);
    temp_file.change_size(289);
    temp_file.change_x(64);
    directory.add_file(temp_file);
    temp_file.change_size(10000);
    temp_file.change_x(32);
    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 - получить файл по индексу." << "\n";
        std::cin >> n;
        if (n == 1) {
            temp_file.change_size(123);
            temp_file.change_x(64);
            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";
        }
        else if (n == 3) {
            directory.print_all();
        }
        else if (n == 5) {
            std::cout << "Введите индекс нового элемента. ";
            std::cin >> temp_i;
            temp_i;
            directory.get_file_by_index(temp_i);
        }
    }
    directory.del_all();
}
test.cpp
#include <iostream>
#include "menu.h"

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 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";
	int t;
	std::cin >> t;
}
```

4. Результати роботи програми

Результати роботи програми:

```
Выберите желаемую опцию:
1 - добавить элемент в список.
2 - удалить элемент из списка.
3 - показать все элементы списка.
4 - завершить работу программы.
5 - получить файл по индексу.
3
1 1 1234 x32
2 2 521 x64
3 3 289 x64
4 4 10000 x32
Выберите желаемую опцию:
1 - добавить элемент в список.
2 - удалить элемент из списка.
3 - показать все элементы списка.
4 - завершить работу программы.
5 - получить файл по индексу.
4
Утечка памяти не обнаружена.
```

Результати тестів:

```
Первый тест на работу геттеров и сеттеров базового класса пройден успешно.
1 1 2 x32
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.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з класами та їх специфікаторами доступу, інкапсуляцією, константами.

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