Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

до лабораторної роботи № 6

з дисципліни: «Програмування, частина 2 (ООП)**»**

на тему: «СПАДКУВАННЯ»

Варіант № 14

Підготував:

студент групи КІ-103

Мишак М.А.

Перевірив:

Ст. викладач

Каф. ЕОМ

Гузинець Н.В

Львів 2025

Мета: познайомитися із спадкуванням класів..

Теоретичний матеріал

Спадкування – це механізм, за допомогою якого один клас може одержувати атрибути та функціональність іншого. Спадкування дозволяє створювати ієрархію класів.

При створенні нового класу, що повністю дублює існуючий клас і дещо розширяє його новими властивостями і функціональністю програміст може недублювати існуючий клас і дописувати в нього розширену функціональність, авказати, що новий клас є спадкоємцем елементів попередньо визначеногокласу і визначити у ньому лише необхідну нову функціональність. В цьомувипадку існуючий клас, функціональність якого розширюється у новому класі,називається базовим класом (base class). Новостворений клас називаєтьсяпохідним класом (derived class), або спадкоємцем. Кожен похідний клас можебути використаним у ролі базовогокласу для майбутніх похідних класівстворюючи при цьому дерево спадкування, яке ще називають ієрархієюспадкування класів (class hierarchy). Спадкування прийнято відображати увигляді графу (дерева) уьнапрямкузверху-вниз. При цьому клас, що є у самомуверху є самим першим базовим класом і називається кореневим класом абокоренем дерева спадкування класів. Похідний клас, через проміжний, моженаслідувати характеристики базового класу. У цьому випадку говорять, щобазовий клас є непрямим базовим класом (indirect base class) для похідного.

Зокрема, корінь дерева наслідувань є непрямим базовим класом для усіх класів,які знаходяться нижче першого рівня ієрархії. Клас, який При одиночномуспадкуванні (single inheritance) клас породжується одним базовим класом. Примножинному спадкуванні (multiple inheritance) похідний клас успадковуєвластивості декількох базових класів, причому можлива ситуація коли одинбазовий клас буде успадкований кілька разів по кількох гілках. При створенніоб'єкта похідного класу в пам'яті зберігаються копії усіх класів, які становлять вітку, що породила даний клас.

Завдання:

Створити абстрактний базовий клас і похійдний від нього клас, які реалізують модель предметної області згідно варіанту. Кожен клас має мати

мінімум 3 власні елементи даних один з яких створюється динамічно, методи встановлення і читання характеристик елементів-даних класу (Set і Get), та мінімум 2 абстрактні методи обробки даних і мінімум 2 методи обробки даних
у похідному класі. Крім цього клас має містити перевантаження оператора присвоєння, конструкторів по замовчуванню і копіювання та віртуальний деструктор. Для розроблених класів реалізувати програму-драйвер, яка демонструє роботу класів.

Лістинги (тексти**) програм:**

```
//Device.h
#pragma once
#include <string>

using namespace std;

class Device {
protected:
    string type;
    string* brand;
    int power;
    string* color;

public:
    Device(string type, const string& brand, int power, const string& color);
    Device();
    Device(const Device& other);
    Device& operator=(const Device& other);

    void set_brand(const string& brand);
    string get_brand();
    void set_power(int power);
    int get_power();
    void set_color(const string& color);
    string get_color();

    virtual void process_data() = 0;
    virtual void print_info() = 0;

    virtual ~Device();
```

```
};
//Device.cpp
#include "Device.h"
#include <iostream>

using namespace std;

Device::Device(string type, const string& brand, int power, const string& color) {
    this->type = type;
    this->brand = new string(brand);
    this->power = power;
    this->color = new string(color);
}

Device::Device() {
    type = "";
    brand = new string("");
    power = 0;
    color = new string("");
}

Device::Device(const Device& other) {
    type = other.type;
    brand = new string(*other.brand);
    power = other.power;
    color = new string(*other.color);
}

Device& Device::operator=(const Device& other) {
    if (this == &other) return *this;

    delete brand;
    delete color;

    type = other.type;
    brand = new string(*other.brand);
    power = other.power;
    color = new string(*other.color);

    return *this;
}
```

```
void Device::set_brand(const string& brand) {
   *(this->brand) = brand;
}

string Device::get_brand() {
   return *brand;
}

void Device::set_power(int power) {
   this->power = power;
}

int Device::get_power() {
   return power;
}

void Device::set_color(const string& color) {
   *(this->color) = color;
}

string Device::get_color() {
   return *color;
}

Device::~Device() {
   delete brand;
   delete color;
}
//Television.h
#pragma once
#include "Device.h"

class Television : public Device {
private:
   int screen_size;
   string* resolution;

public:
   Television(string type, const string& brand, int power, const string& color,
      int screen_size, const string& resolution);
```

```
    Television(const Television& other);
    Television& operator=(const Television& other);

    void set_screen_size(int screen_size);
    int get_screen_size();

    void set_resolution(const string& resolution);
    string get_resolution();

    void process_data() override;
    void print_info() override;

    virtual ~Television();
};
//Television.cpp
#include "Television.h"
#include <iostream>

using namespace std;

Television::Television(string type, const string& brand, int power, const string& color,
    int screen_size, const string& resolution) {
    this->type = type;
    this->brand = new string(brand);
    this->power = power;
    this->color = new string(color);
    this->screen_size = screen_size;
    this->resolution = new string(resolution);
}

Television::Television(const Television& other) : Device(other) {
    screen_size = other.screen_size;
    resolution = new string(*other.resolution);
}

Television& Television::operator=(const Television& other) {
    if (this == &other) return *this;

    delete resolution;

    Device::operator=(other);
```

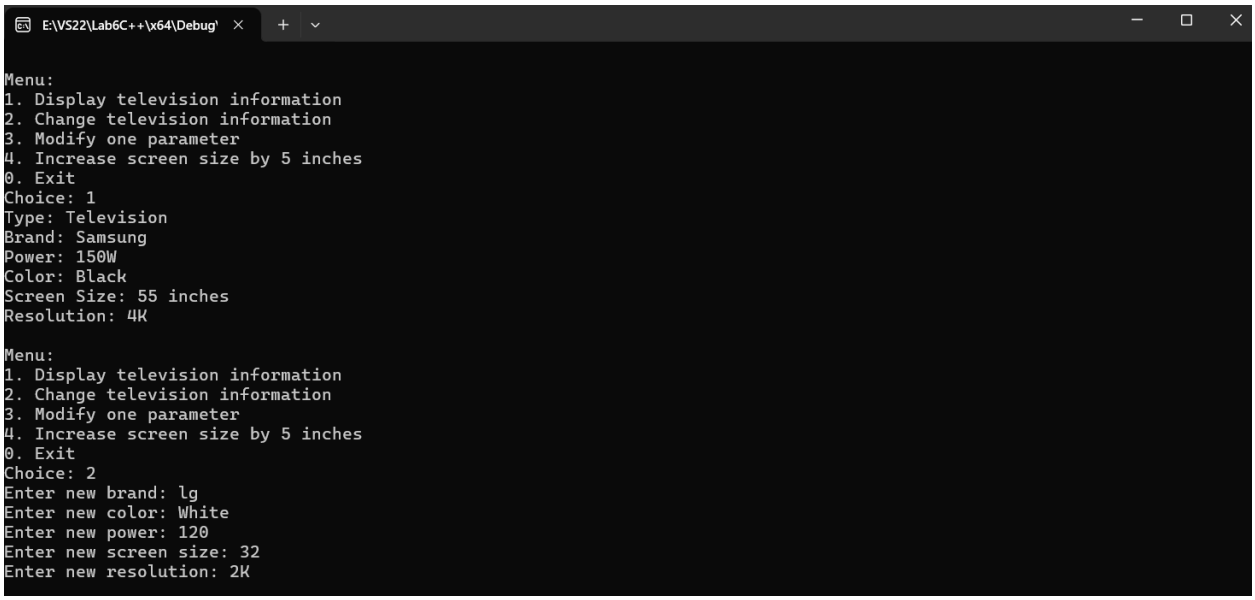
```
    screen_size = other.screen_size;
    resolution = new string(*other.resolution);

    return *this;
}

void Television::set_screen_size(int screen_size) {
    this->screen_size = screen_size;
}

int Television::get_screen_size() {
    return screen_size;
}

void Television::set_resolution(const string& resolution) {
    *(this->resolution) = resolution;
}

string Television::get_resolution() {
    return *resolution;
}

void Television::process_data() {
    this->screen_size += 5;
    cout << "Screen size increased by 5 inches" << endl;
}

void Television::print_info() {
    cout << "Type: " << type << endl
        << "Brand: " << *brand << endl
        << "Power: " << power << "W" << endl
        << "Color: " << *color << endl
        << "Screen Size: " << screen_size << " inches" << endl
        << "Resolution: " << *resolution << endl;
}

Television::~Television() {
    delete resolution;
}
//main.cpp
#include "Television.h"
```

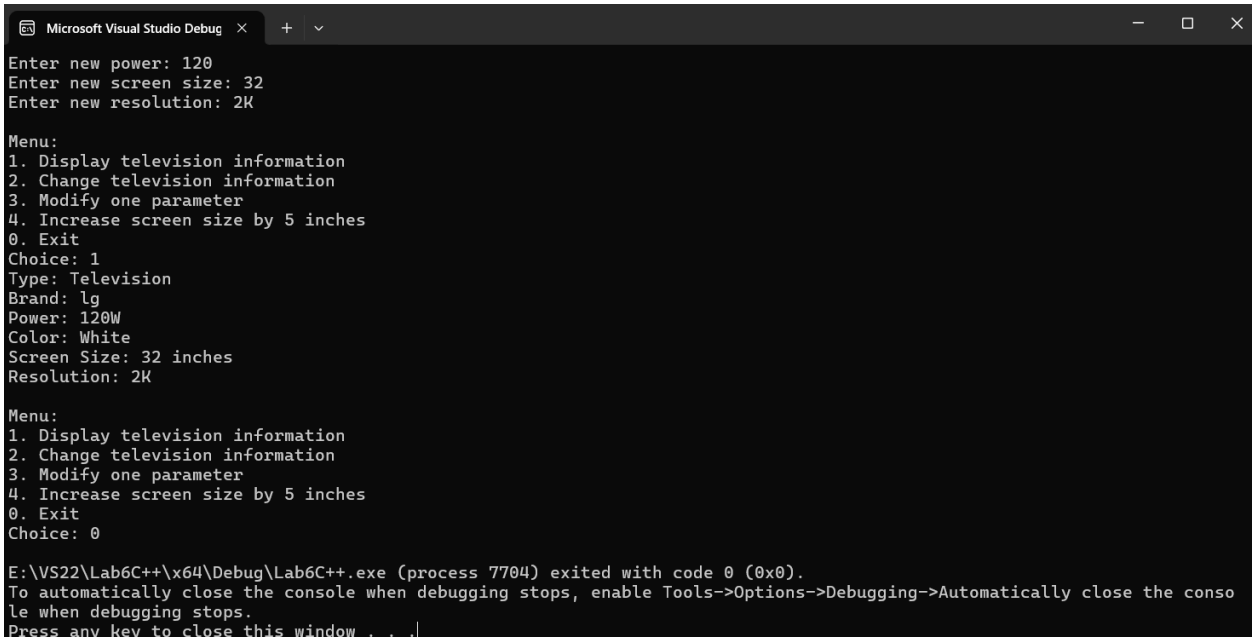
```
#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);

    Television tv("Television", "Samsung", 150, "Black", 55, "4K");
    int choice;

    while (true) {
        cout << "\nMenu:\n";
        cout << "1. Display television information\n";
        cout << "2. Change television information\n";
        cout << "3. Modify one parameter\n";
        cout << "4. Increase screen size by 5 inches\n";
        cout << "0. Exit\n";
        cout << "Choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            tv.print_info();
            break;

        case 2:
        {
            string brand, color, resolution;
            int power, screen_size;

            cout << "Enter new brand: ";
            cin.ignore();
            getline(cin, brand);

            cout << "Enter new color: ";
            getline(cin, color);

            cout << "Enter new power: ";
```

```
    cin >> power;

    cout << "Enter new screen size: ";
    cin >> screen_size;

    cin.ignore();
    cout << "Enter new resolution: ";
    getline(cin, resolution);

    tv.set_brand(brand);
    tv.set_color(color);
    tv.set_power(power);
    tv.set_screen_size(screen_size);
    tv.set_resolution(resolution);
}
break;

case 3:
{
    int param_choice;
    cout << "Which parameter do you want to change?\n";
    cout << "1. Brand\n";
    cout << "2. Power\n";
    cout << "3. Color\n";
    cout << "4. Screen Size\n";
    cout << "5. Resolution\n";
    cout << "Choice: ";
    cin >> param_choice;

    if (param_choice == 1) {
        string brand;
        cout << "Enter new brand: ";
        cin.ignore();
        getline(cin, brand);
        tv.set_brand(brand);
    }
    else if (param_choice == 2) {
        int power;
        cout << "Enter new power: ";
        cin >> power;
        tv.set_power(power);
```

```
            }
            else if (param_choice == 3) {
                string color;
                cout << "Enter new color: ";
                cin.ignore();
                getline(cin, color);
                tv.set_color(color);
            }
            else if (param_choice == 4) {
                int screen_size;
                cout << "Enter new screen size: ";
                cin >> screen_size;
                tv.set_screen_size(screen_size);
            }
            else if (param_choice == 5) {
                string resolution;
                cout << "Enter new resolution: ";
                cin.ignore();
                getline(cin, resolution);
                tv.set_resolution(resolution);
            }
            else {
                cout << "Invalid choice!\n";
            }
        }
        break;

        case 4:
            tv.process_data();
            break;

        case 0:
            return 0;

        default:
            cout << "Invalid choice!\n";
            break;
        }
    }
}
```

Результати виконання програм:





Висновок:

У ході виконання лабораторної роботи було розглянуто механізм спадкування на мові програмування С++.