Министерство образования и науки РФ

Севастопольский государственный университет

Институт информационных технологий и управления в технических сетях

Лабораторная работа №3

**МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ.**

**ДРУЖЕСТВЕННЫЕ КЛАССЫ И ФУНКЦИИ.**

Выполнил:

ст.гр.ИСб-22д

Воронин И.Ю.

Проверил:

Тимофеев И.С.

Севастополь

2015

1.Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ. Освоение особенностей отладки объектно-ориентированных программ. Изучение механизма множественного наследования. Приобретение навыков работы с механизмом “дружественности”.

2.Вариант задания

Вариант 10

Базовые классы: Ткань (название, цвет, цена), Стиль (название). Класс-наследник: Костюм.

Дружественность: класс Химчистка.

3.Текст программы

Main:

#include <iostream>

#include "Chemic.h"

#include "Costum.h"

int main(){

setlocale(0, "RUS");

std::string arr[3];

double cost;

std::cout << "Ткань: ", std::cin >> arr[0];

std::cout << "Цвет: ", std::cin >> arr[1];

std::cout << "Цена: ", std::cin >> cost;

std::cout << "Стиль: ", std::cin >> arr[2];

std::cout << std::endl;

Costum c1(arr[0],arr[1],cost,arr[2]);

c1.inf();

std::cout << c1.getCost() << std::endl;

Chemic chem1(1000, 4);

chem1.setCost(2000);

chem1.clean(c1);

}

Первый родительский класс:

#include <string>

class Tkan{

protected:

std::string name;

std::string color;

double cost;

public:

Tkan();

Tkan(std::string \_name, std::string \_color, double \_cost);

std::string getName(void);

std::string getColor(void);

double getCost(void);

virtual ~Tkan();

};

#include "Tkan.h"

Tkan::Tkan(){

name = " ";

color = " ";

cost = 0;

}

Tkan::Tkan(std::string \_name, std::string \_color, double \_cost){

name = \_name;

color = \_color;

cost = \_cost;

}

std::string Tkan::getName(void) {return name;}

std::string Tkan::getColor(void) {return color;}

double Tkan::getCost(void) {return cost;}

Tkan::~Tkan(){}

Второй родительский класс:

#include <string>

class Style{

private:

std::string name;

public:

Style();

Style(std::string \_name);

std::string getName(void);

virtual ~Style();

};

#include "Style.h"

Style::Style() {name = ""; }

Style::Style(std::string \_name) {name = \_name;}

std::string Style::getName(void){return name;}

Style::~Style(){}

Дочерний класс и friend:

#include "Style.h"

#include "Tkan.h"

#include <iostream>

#include <math.h>

enum gender { male = 0, female = 1 };

class Chemic;

class Costum: public Tkan, public Style{

private:

gender pol;

public:

Costum() : Tkan("Шёлк", "Белый", 2900), Style("Готический"){

pol = male;

}

Costum(std::string tkan, std::string color, double cost, std::string style) : Tkan(tkan, color, cost), Style(style){

pol = male;

}

Costum(gender \_pol, std::string tkan, std::string color, double cost, std::string style) : Tkan(tkan, color, cost), Style(style){

pol = \_pol;

}

gender getPol(void);

friend class Chemic;

void inf();

~Costum();

};

class Chemic{

private:

double cost;

int days;

public:

Chemic(){

cost = 0;

days = 0;

}

//цена и кол-во дней в хим-ке

Chemic(double \_cost, int \_days){

cost = \_cost;

days = \_days;

}

void setCost(double \_cost){

cost = \_cost;

}

double getCost(void){

return cost;

}

void setDays(int \_days){

days = \_days;

}

int getDays(void){

return days;

}

void clean(Costum costum){

setlocale(LC\_ALL, "RUS");

//больше платишь-быстрее чистим, но чем круче материал,тем дольше

std::cout << std::endl << "Чистка " << std::endl;

std::cout<< " Количество дней: " << std::abs(((int)(costum.cost)/125 - (int)cost/800)) << std::endl;

}

~Chemic(){}

};

#include "Costum.h"

Costum::~Costum(){}

gender Costum::getPol(){

return pol;

}

void Costum::inf(){

std::cout << "Характеристика костюма: " << std::endl;

if (pol) std::cout << " Пол: женщина " << std::endl;

else std::cout << " Пол: мужчина " << std::endl;

std::cout << " Название ткани: " << Tkan::name << std::endl;

std::cout << " Цвет ткани: " << Tkan::color << std::endl;

std::cout << " Цена ткани: " << Tkan::cost << std::endl;

std::cout << " Стиль: " << Style::getName() << std::endl;

}

4.Выполнения программы

Было написано 4 класса: два родительский, дочерний и дружественный класс. В данном случае были созданы объект дочернего класса и дружественный ему.

При помощи конструктора задаются параметры, которые были унаследованы от родительских классов. После используется функция, которая выводит все заданные параметры на экран. Печать стоимости отдельной функцией. Далее создаётся объект дружественного класса, в который передаются раннее созданный объект класса, к которому он является дружественным. Далее выводятся значения, которые получаются при использовании получения доступа дружественным классом к полям исходного.

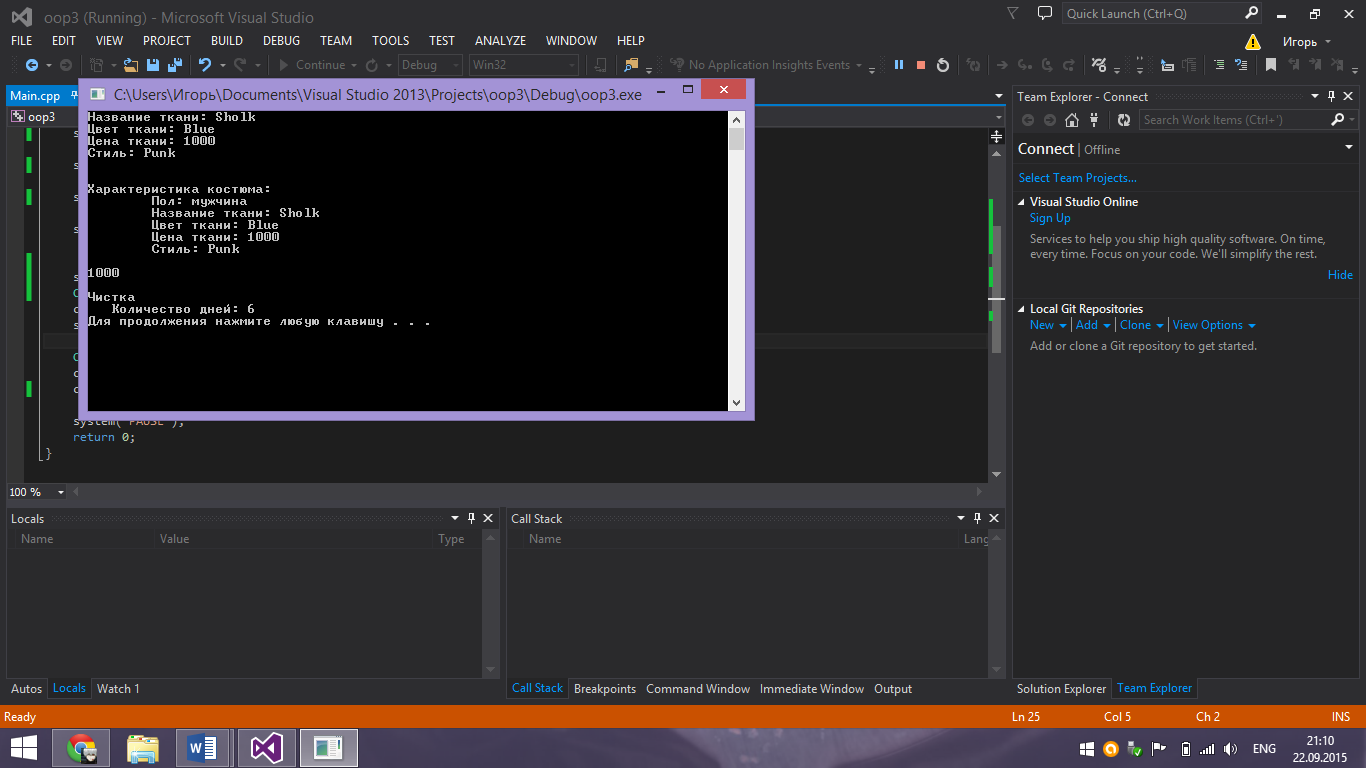


Рисунок 4.1. -Тест программы.

ВЫВОДЫ

В данной лабораторной работе были изучены понятия множественного наследования и дружественный классы и функции. Было написано 4 класса: 2 родительских, 1 дочерний и 1 дружественный. Данные методы программирования являются особенностями С++ и в последующих языках были исключены, так как данный стиль программирования применяется лишь в редких случаях, когда, к примеру, уже в готовый проект необходимо внести свои коррективы. При написании проекта с нуля необходимо минимизировать использование данных методологий.