Міністерство освіти і науки України

НТУУ „КПІ”

ТЕФ

**Кафедра АПЕПС**

***Звіт***

***До лабораторної роботи №4,5***

***З дисципліни***

***"Об’єктно-орієнтоване програмування"***

Виконав: **Артамонов О.Ю.**

Студент групи: **ТВ-61**

Перевірив викладач: **Карпенко С.Г.**

**Київ – 2017**

Код програми:

// 4ооп.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <cstring>

#pragma warning(disable : 4996)

class Date {

private:

int year, week;

public:

Date() : year(2017), week(40) {

printf("\nDate default constructor\n");

}

Date(int year1, int week1) : year(year1), week(week1) {

printf("\nDate init constructor\n");

}

Date(const Date &d) {

year = d.year;

week = d.week;

}

virtual void Show() {

printf("\nShow Date: ");

printf("\nyear - %d, week - %d\n", year, week);

}

virtual ~Date() {

printf("\nDate destructor\n");

}

};

class Flower {

private:

char \*name, \*color;

public:

Flower() {

name = new char[strlen("Flower") + 1];

strcpy(name, "Flower");

color = new char[strlen("Color") + 1];

strcpy(color, "Color");

printf("\nFlower default constructor\n");

}

Flower(char \*name1, char \*color1) {

name = new char[strlen(name1) + 1];

strcpy(name, name1);

color = new char[strlen(color1) + 1];

strcpy(color, color1);

printf("\nFlower init constructor\n");

}

Flower(const Flower &f) {

name = new char[strlen(f.name) + 1];

strcpy(name, f.name);

color = new char[strlen(f.color) + 1];

strcpy(color, f.color);

}

virtual void Show() {

printf("\nShow Flower\n");

printf("\nName of flower: \"%s\"\nColor of flower: %s\n", name, color);

}

~Flower() {

if (name) {

delete[] name;

}

if (color) {

delete[] color;

}

printf("\nFlower destructor\n");

}

};

class Flower\_Product : virtual public Flower {

private:

Flower f;

char \*TypeOfSale;

public:

Flower\_Product() : f() {

printf("\nFlower\_Product default constructor\n");

}

Flower\_Product (char \*name1, char \*color1, char \*TypeOfSale1) : f(name1, color1) {

printf("\nFlower\_Product init constructor\n");

TypeOfSale = new char[strlen(TypeOfSale1) + 1];

strcpy(TypeOfSale, TypeOfSale1);

}

Flower\_Product(const Flower\_Product &fp) : f(fp) {

TypeOfSale = new char[strlen(fp.TypeOfSale) + 1];

strcpy(TypeOfSale, fp.TypeOfSale);

}

void Show() {

printf("\n\t\tShow F\_P\n");

f.Show();

printf("\nType of sale: %s\n", TypeOfSale);

}

~Flower\_Product() {

if (TypeOfSale) {

delete[] TypeOfSale;

}

printf("\nFlower\_Product destructor\n");

}

};

class Consignment : public Flower\_Product {

private:

Flower\_Product fp;

int count;

public:

Consignment() : fp(), count(10) {

printf("\nConsignment default constructor\n");

}

Consignment(char \*name1, char \*color1, char \*TypeOfSale1, int count1) : fp(name1, color1, TypeOfSale1), count(count1) {

printf("\nConsignement init constructor\n");

}

Consignment(const Consignment &c) : fp(c) {

count = c.count;

}

void Show() {

printf("\n\t\tShow Consignmet\n");

fp.Show();

printf("\nCount of this type of flowers = %d\n", count);

}

~Consignment() {

printf("\nConsignement destructor\n");

}

};

class Delivery : virtual public Date, public Consignment {

private:

Date d;

Consignment c;

public:

Delivery() : d(), c() {

printf("\nDelivery default constructor\n");

}

Delivery(int year1, int week1, char \*name1, char \*color1, char \*TypeOfSale1, int count1) : d(year1, week1), c(name1, color1, TypeOfSale1, count1) {

printf("\nDelivery init constructor\n");

}

Delivery(const Delivery &d) : d(d), c(d) {}

void Show() {

printf("\n\t\tShow Delivery\n");

d.Show();

c.Show();

}

~Delivery() {

printf("\nDelivery destructor\n");

}

};

class Sale : virtual public Date, public Consignment {

private:

Date d;

Consignment c;

public:

Sale() : d(), c() {

printf("\nSale default constructor\n");

}

Sale(int year1, int week1, char \*name1, char \*color1, char \*TypeOfSale1, int count1) : d(year1, week1), c(name1, color1, TypeOfSale1, count1) {

printf("\nSale init constructor\n");

}

Sale(const Sale &s) : d(s), c(s) {}

void Show() {

printf("\n\t\tShow Sale\n");

d.Show();

c.Show();

}

~Sale() {

printf("\nSale destructor\n");

}

};

class Daily\_Balance : public Delivery, public Sale {

private:

Delivery d;

Sale s;

public:

Daily\_Balance() : d(), s() {

printf("\nDaily\_Balance default constructor\n");

}

Daily\_Balance(int year1, int week1, char \*name1, char \*color1, char \*TypeOfSale1, int count1) : d(year1, week1, name1, color1, TypeOfSale1, count1), s(year1, week1, name1, color1, TypeOfSale1, count1) {

printf("\nDaily\_Balance init constructor\n");

}

Daily\_Balance(const Daily\_Balance &db) : d(db), s(db) {}

void Show() {

printf("\n\t\tShow D\_B\n");

d.Show();

s.Show();

}

~Daily\_Balance() {

printf("\nDaily\_Balance destructor\n");

}

};

int main()

{

Date \*d;

int c = 1;

while (c == 1) {

Daily\_Balance \*db = new Daily\_Balance(2001, 21, "Aloe", "Green", "Vase", 37);

delete db;

printf("\n-------------------------Date-------------------------\n");

d = new Date(2007, 47);

d->Show();

printf("\n------------------------------------------------------\n\n");

delete d;

printf("\n-----------------------Delivery-----------------------\n");

d = new Delivery(2005, 17, "Aster", "Blue", "By the piece", 17);

d->Show();

printf("\n------------------------------------------------------\n\n");

delete d;

printf("\n-------------------------Sale-------------------------\n");

d = new Sale(2007, 39, "Camellia", "Pink", "Bouquet", 73);

d->Show();

printf("\n------------------------------------------------------\n");

delete d;

printf("\nIf you want to continue - press 1, if not - press 0.\nEnter your choice: ");

while (scanf\_s("%d", &c) != 1 || c < 0 || c > 1) {

printf("Invalid value, press 1 or 0: ");

while (getchar() != '\n');

}

}

return 0;

}

Контрольні запитання:

1. Якщо шаблон функції має кілька параметрів-типів, то кожному з них повинно передувати ключове слово class або typename.
2. class Імя\_Похідного\_Класу : специфікатор доступу Ім’я\_Базового\_Класу {};
3. Механізм віртуальних функцій реалізує поліморфізм часу виконання: який віртуальний метод викликається буде відомо тільки під час виконання програми.
4. Різниці між class та typename немає, але для простоти читання коду для класів(структур) використовується class, а для функцій typename