

Курс

«Объектно-ориентированное программирование на C++»

Встреча №4

*Тема: Указатель this. Побитовое копирование. Конст- руктор копирования.*

Задание.

Создайте программу, имитирующую многоквартирный дом. Необходимо иметь классы “Человек”, “Квартира”, “Дом”. Класс “Квартира” содержит динамический массив объектов класса “Человек”. Класс “Дом” содержит массив объектов класса “Квартира”.

Каждый из классов содержит переменные-члены и функ- ции-члены, которые необходимы для предметной области класса. Обращаем ваше внимание, что память под стро- ковые значения выделяется динамически. Например, для ФИО в классе “Человек”. Не забывайте обеспечить классы различными конструкторами (конструктор копирования обязателен), деструкторами. В main протестировать ра- боту полученного набора классов.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#include<direct.h>

#include<io.h>

#include<time.h>

#include<conio.h>

using namespace std;

class Human

{

private:

char\* PIB;

char\* sex;

int age;

public:

Human()

{

PIB = NULL;

sex = NULL;

}

Human(const Human& obj)

{

PIB = new char[strlen(obj.PIB) + 1];

strcpy(PIB, obj.PIB);

sex = new char[strlen(obj.sex) + 1];

strcpy(sex, obj.sex);

age = obj.age;

}

~Human()

{

if (PIB != NULL)

delete[] PIB;

if (sex != NULL)

delete[] sex;

}

char\* Get\_PIB()

{

return PIB;

}

char& Get\_PIB2()

{

return \*PIB;

}

char\* Get\_Sex()

{

return sex;

}

char& Get\_Sex2()

{

return \*sex;

}

int Get\_Age()

{

return age;

}

int& Get\_Age2()

{

return age;

}

void Get\_Age(int a)

{

age = a;

}

char\* String\_for\_anything()

{

char\* tmp = NULL;

int k = 0;

do

{

char s;

s = \_getch();

cout << s;

if (s == 13)

break;

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = s;

k++;

} while (true);

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = '\0';

cout << endl;

return tmp;

}

void Set\_PIB()

{

cout << "Введіть П.І.Б мешканця ->";

char\* tmp = String\_for\_anything();

if (PIB != NULL)

delete[] PIB;

PIB = new char[strlen(tmp) + 1];

strcpy(PIB, tmp);

delete[]tmp;

}

void Set\_PIB(char \*tmp)

{

if (PIB != NULL)

delete[] PIB;

PIB = new char[strlen(tmp) + 1];

strcpy(PIB, tmp);

}

void Set\_Sex()

{

cout << "Введіть стать мешканця ->";

char\* tmp = String\_for\_anything();

if (sex != NULL)

delete[] sex;

sex = new char[strlen(tmp) + 1];

strcpy(sex, tmp);

delete[]tmp;

}

void Set\_Sex(char\* tmp)

{

if (sex != NULL)

delete[] sex;

sex = new char[strlen(tmp) + 1];

strcpy(sex, tmp);

}

void Set\_Age()

{

cout << "Введіть вік мешканця ->";

cin >> age;

}

void Print\_PIB()

{

cout << "П.І.Б. мешканця - " << PIB << endl;

}

void Print\_Sex()

{

cout << "Стать мешканця - " << sex << endl;

}

void Print\_Age()

{

cout << "Вік мешканця - " << age << endl;

}

};

class Flat

{

private:

float s;

int number\_of\_human;

Human\* human;

public:

Flat()

{

s = 0;

number\_of\_human = 0;

human = NULL;

}

Flat(const Flat& obj)

{

s = obj.s;

number\_of\_human = obj.number\_of\_human;

human = new Human[obj.number\_of\_human];

for (int i = 0; i < obj.number\_of\_human; i++)

{

human[i].Set\_PIB(obj.human[i].Get\_PIB());

human[i].Set\_Sex(obj.human[i].Get\_Sex());

human[i].Get\_Age(obj.human[i].Get\_Age());

}

}

void Set\_members(Flat obj)

{

for (int i = 0; i < obj.number\_of\_human; i++)

{

human[i].Set\_PIB(obj.human[i].Get\_PIB());

human[i].Set\_Sex(obj.human[i].Get\_Sex());

human[i].Get\_Age(obj.human[i].Get\_Age());

}

}

~Flat()

{

if (human != NULL)

delete[] human;

}

void FillFlat(int k)

{

human = new Human[k];

}

float Get\_S()

{

return s;

}

float& Get\_S2()

{

return s;

}

int Get\_Number\_Of\_Human()

{

return number\_of\_human;

}

int& Get\_Number\_Of\_Human2()

{

return number\_of\_human;

}

Human\* Get\_Human()

{

return human;

}

Human& Get\_Human2()

{

return \*human;

}

void Set\_S()

{

cout << "Введіть площу квартири (у кв.м.)->";

cin >> s;

}

void Set\_Number\_Of\_Human()

{

cout << "Введіть кількість жителів у квартирі ->";

cin >> number\_of\_human;

human = new Human[number\_of\_human];

}

void Print\_S()

{

cout << "Площа квартири - " << s << endl;

}

void Print\_Number\_Of\_Human()

{

cout << "Жителів у квартирі - " << number\_of\_human << endl;

}

};

class Home

{

private:

char\* adres;

int number;

Flat\* flat;

public:

Home()

{

adres = NULL;

number = 0;

flat = NULL;

}

Home(const Home& obj)

{

adres = new char[strlen(obj.adres) + 1];

strcpy(adres, obj.adres);

number = obj.number;

flat = new Flat[obj.number];

for (int i = 0; i < obj.number; i++)

{

flat[i].FillFlat(obj.flat[i].Get\_Number\_Of\_Human());

}

for (int i = 0; i < number; i++)

{

flat[i].Get\_S2() = obj.flat[i].Get\_S();

flat[i].Get\_Number\_Of\_Human2() = obj.flat[i].Get\_Number\_Of\_Human();

for (int i = 0; i < number; i++)

{

flat[i].Set\_members(obj.flat[i]);

}

}

}

~Home()

{

if (adres != NULL)

delete[] adres;

if (flat != NULL)

delete[] flat;

}

char\* String\_for\_anything()

{

char\* tmp = NULL;

int k = 0;

do

{

char s;

s = \_getch();

cout << s;

if (s == 13)

break;

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = s;

k++;

} while (true);

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = '\0';

cout << endl;

return tmp;

}

void Set\_Adress()

{

cout << "Введіть адресу будинку:\n";

char\* tmp = String\_for\_anything();

if (adres != NULL)

delete[] adres;

adres = new char[strlen(tmp) + 1];

strcpy(adres, tmp);

delete[]tmp;

}

void Print\_Adres()

{

cout << "Адреса будинку: " << adres << endl;

}

void Print\_Number()

{

cout << "Квартир у будинку: " << number << endl;

}

void Set\_Flat()

{

cout << "Введіть скільки квартир у домі ->";

cin >> number;

flat = new Flat[number];

}

void Set\_Info\_By\_Home()

{

Set\_Adress();

Set\_Flat();

for (int i = 0; i < number; i++)

{

cout << i + 1 << " квартира\n";

flat[i].Set\_Number\_Of\_Human();

flat[i].Set\_S();

cout << "-------------------------------------\n";

}

for (int i = 0; i < number; i++)

{

cout << i + 1 << " квартира\n";

for (int j = 0; j < flat[i].Get\_Number\_Of\_Human(); j++)

{

cout << j + 1 << " мешканець\n";

flat[i].Get\_Human()[j].Set\_PIB();

flat[i].Get\_Human()[j].Set\_Sex();

flat[i].Get\_Human()[j].Set\_Age();

}

cout << "-------------------------------------\n";

}

}

void Print\_All()

{

Print\_Adres();

Print\_Number();

for (int i = 0; i < number; i++)

{

cout << i + 1 << " квартира\n";

flat[i].Print\_S();

flat[i].Print\_Number\_Of\_Human();

for (int j = 0; j < flat[i].Get\_Number\_Of\_Human(); j++)

{

cout << j + 1 << " мешканець\n";

flat[i].Get\_Human()[j].Print\_PIB();

flat[i].Get\_Human()[j].Print\_Sex();

flat[i].Get\_Human()[j].Print\_Age();

}

cout << "-------------------------------------\n";

}

}

void Test()

{

Set\_Info\_By\_Home();

Print\_All();

}

};

void Print\_aLL(Home a)

{

a.Print\_All();

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Home a;

a.Set\_Info\_By\_Home();

Print\_aLL(a);

return 0;

}