 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №2**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Робота з об’єктами, використання покажчика this»

Варіант №15

**Виконав:**

студент гр. БС-71

Орлівський С. П.

**Перевірив:**

асистент каф. БМК

Рисін С.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2019

**🞏 Практичне заняття без зауважень**

**🞏 Практичне заняття має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 потужність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 відсутні списки ініціалізації в конструкторах**

**🞏 константні методи**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Завдання:**

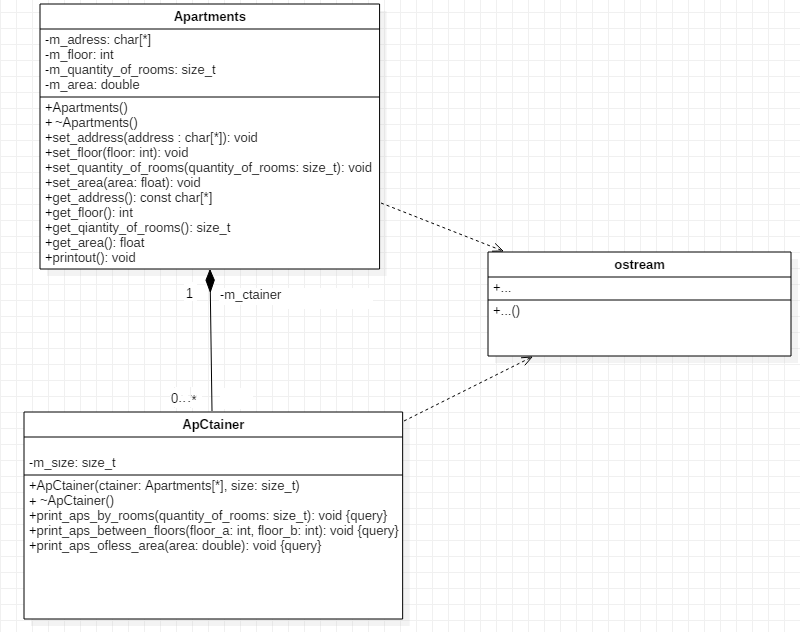
1. Вивчити теоретичні основи роботи з об’єктами та використання покажчика this.
2. Скористатися розробленим в попередній лабораторій роботі класом для розширення його функціональності (нові методи класу мають працювати з покажчиком this).
3. Визначення класів та їх реалізації мають бути розташовані у файлах \*.h та \*.cpp відповідно.
4. Відповідно до свого варіанту написати програму, яка містить клас для роботи із запитами користувача (методи класу дозволяють відповідно до вибору користувача вивести на екран задані в завданні списки) та інкапсулює в собі масив об’єктів розробленого в попередній лабораторій роботі класу:

***Створити масив об’єктів та вивести на екран:***

* ***список квартир, що мають задане число кімнат;***
* ***список квартир, що розташовані на поверсі, який знаходиться в певному проміжку;***
* ***список квартир, які мають площу, що перевершує задану.***

1. Скласти і захистити звіт.

**UML діаграма:**

****

**Код програми:**

**Stdafx.h:**

#pragma once

#include "targetver.h"

#include "conio.h"

#include <string>

#include <iostream>

#include <stdio.h>

#include <tchar.h>

#pragma once

#include <iostream>

#include <cstring>

class **Apartments**

{

private:

char \*m\_address;

int m\_floor;

size\_t m\_quantity\_of\_rooms;

double m\_area;

public:

**Apartments**(const char \*address, const int floor, const size\_t quantity\_of\_rooms,

const double area);

**Apartments**();

void **set\_address**(const char \*address);

void **set\_floor**(const int floor);

void **set\_quantity\_of\_rooms**(const size\_t quantity\_of\_rooms);

void **set\_area**(const double area);

const char \* **get\_address**() const;

int **get\_floor**() const;

double **get\_area**() const;

size\_t **get\_quantity\_of\_rooms**() const;

void **printout**() const;

~**Apartments**();

};

**apartments.cpp:**

#include "apartments.h"

Apartments::**Apartments**(const char \*address, const int floor, const size\_t quantity\_of\_rooms,

const double area):m\_address(new char [strlen(address)+1]),

m\_floor(floor), m\_quantity\_of\_rooms(quantity\_of\_rooms), m\_area(area)

{

strcpy(m\_address, address);

}

Apartments::**Apartments**() :m\_address(nullptr), m\_floor(0), m\_quantity\_of\_rooms(0),

m\_area(0)

{

}

Apartments::~**Apartments**()

{

delete[] m\_address;

}

void Apartments::**set\_address**(const char \*address)

{

delete[] m\_address;

m\_address = new char[strlen(address) + 1];

strcpy(m\_address, address);

}

void Apartments::**set\_floor**(const int floor)

{

m\_floor = floor;

}

void Apartments::**set\_quantity\_of\_rooms**(const size\_t quantity\_of\_rooms)

{

m\_quantity\_of\_rooms = quantity\_of\_rooms;

}

void Apartments::**set\_area**(const double area)

{

m\_area = area;

}

const char \*Apartments::**get\_address**() const

{

return (!m\_address) ? "" : m\_address;

}

int Apartments::**get\_floor**() const

{

return m\_floor;

}

double Apartments::**get\_area**() const

{

return m\_area;

}

size\_t Apartments::**get\_quantity\_of\_rooms**() const

{

return m\_quantity\_of\_rooms;

}

void Apartments::**printout**() const

{

std::cout << "Address: " << this->get\_address() << std::endl

<< "Floor: " << this->get\_floor() << std::endl

<< "Quantity of rooms: " << this->get\_quantity\_of\_rooms() << std::endl

<< "Area: " << this->get\_area() << std::endl;

}

**appctainer.h:**

#pragma once

#include "apartments.h"

struct **AppStruct**

{

char \*address;

int floor;

size\_t quantity\_of\_rooms;

double area;

};

class **AppCtainer**

{

private:

Apartments \*m\_ctainer;

size\_t m\_size;

public:

**AppCtainer**(AppStruct \*ctainer, size\_t size);

void **print\_apps\_by\_rooms**(size\_t quantity\_of\_rooms) const;

void **print\_apps\_between\_floors**(int floor\_a, int floor\_b) const;

void **print\_apps\_ofless\_area**(double area) const;

~**AppCtainer**();

};

**appctainer.cpp:**

#include "appctainer.h"

AppCtainer::**AppCtainer**(AppStruct \*ctainer, size\_t size)

{

m\_ctainer = new Apartments[size];

m\_size = size;

for (size\_t i = 0; i < m\_size; ++i)

{

m\_ctainer[i].set\_address(ctainer[i].address);

m\_ctainer[i].set\_area(ctainer[i].area);

m\_ctainer[i].set\_floor(ctainer[i].floor);

m\_ctainer[i].set\_quantity\_of\_rooms(ctainer[i].quantity\_of\_rooms);

}

}

void AppCtainer::**print\_apps\_by\_rooms**(size\_t quantity\_of\_rooms) const

{

for (size\_t i = 0; i < m\_size; ++i)

{

if (m\_ctainer[i].get\_quantity\_of\_rooms() == quantity\_of\_rooms)

{

std::cout << "------------------------------------\n";

m\_ctainer[i].printout();

std::cout << "------------------------------------\n";

}

}

}

void AppCtainer::**print\_apps\_between\_floors**(int floor\_a, int floor\_b) const

{

if (floor\_a > floor\_b)

{

floor\_a = floor\_b + floor\_a;

floor\_b = floor\_a - floor\_b;

floor\_a -= floor\_b;

}

for (size\_t i = 0; i < m\_size; ++i)

{

if (floor\_a <= m\_ctainer[i].get\_floor() && floor\_b >= m\_ctainer[i].get\_floor())

{

std::cout << "------------------------------------\n";

m\_ctainer[i].printout();

std::cout << "------------------------------------\n";

}

}

}

void AppCtainer::**print\_apps\_ofless\_area**(double area) const

{

for (size\_t i = 0; i < m\_size; ++i)

{

if (m\_ctainer[i].get\_area() < area)

{

std::cout << "------------------------------------\n";

m\_ctainer[i].printout();

std::cout << "------------------------------------\n";

}

}

}

AppCtainer::~**AppCtainer**()

{

delete[] m\_ctainer;

}

**main.cpp:**

#include "apartments.h"

#include "appctainer.h"

int **main**()

{

AppStruct a[] = {{(char \*)"address", 3, 3, 120}, {(char \*)"address2", 1, 1, 100}, {(char \*)"address3", 3, 3, 333}, {(char \*)"address4", 4, 4, 444}};

AppCtainer ctainer(a, sizeof(a)/sizeof(a[0]));

std::cout << "Apps between 4 and 9 floors: \n";

ctainer.print\_apps\_between\_floors(4, 9);

std::cout << "\n\nApps with 3 rooms: \n";

ctainer.print\_apps\_by\_rooms(3);

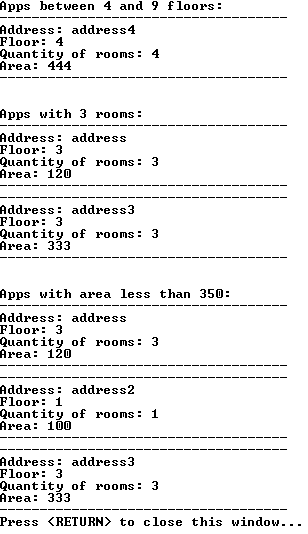
std::cout << "\n\nApps with area less than 350: \n";

ctainer.print\_apps\_ofless\_area(350);

return 0;

}

**Результат роботи програми:**

****

**Контрольні запитання**

* 1. ***Що таке покажчик this?***

Покажчик this використовується у методах класу. Покажчик this – покажчик на об’єкт, що викликав данний метод.

* 1. ***Навести приклад в якому не можна організувати код без використання покажчика this***.

Class Class::method(type par, …)

{

/…/

return \*this;  
}

* 1. ***Два способи ініціалізації об’єктів у програмі.***

За допомогою конструкторів за замовчуванням (якщо він має усі параметри за замовчуванням) або конструктор з параметрами. Або сеттерами після виклику конструктору за замовчуванням

* 1. ***Як ініціалізувати масив об’єктів?***

1) Як статичний масив (використовується конструктор за замовчуванням):

Сlass array[10];

2) Як проініціалізований статичний масив (використовується: конструктор з параметрами):

Сlass array[3] = {{-1.4}, {8.},{-7.}}; або

Сlass array[3] = {Class(-1.4), Class (8.), Class (-7.)};

3) Як динамічний масив (використовується конструктор за замовчуванням):

Сlass \*array= new Сlass [length];

4) Як динаміний масив (використовується конструктор за замовчуванням)

Tovar\* array[len];

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

array[i]=new Сlass;

5) Як динаміний масив покажчиків на динамічну область пам’яті (використовується конструктор за замовчуванням)

Сlass \*\* array=new Сlass \*[length];

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

array[i]=new Сlass;

6) Як статичний масив покажчиків на динамічну область пам’яті (використовується конструктор за замовчуванням)

Сlass \*array[3];

for(int i=0; i<3; i++)

array[i]=new Сlass;

5. ***Чому не можна ініціалізуати поле, що є масивом під час визначення класу?***

Тому що масив, як параметр представлений покажчиком на 0-овий елемен.

Можна, якщо поле статичне і відповідно не прив’язане до жодного об’экту – спільне для всіх.

1. ***Перелічити випадки, коли один клас використовує інший?***

* наслідування (екземпляр підтипу є також екземпляром надтипу)
* залежність (зміна класу-постачальника може вплинути на клас-приймач, та не навпаки)
* асоціація (показує відношення між об’єктами класу)

- бінарна: одному екземпляру одного класу відповідає лише один екземпляр іншого класу («громадянин» - «паспорт»)

- н-арна: : одному екземпляру одного класу може відповідати один і більше екземплярів іншого класу («громадянин» - «документ»)

* агрегація (один клас є контейнером для об’єктів іншого класа).

1. ***Які є види відношень між класами?***
   * + 1. Залежність
       2. Асоціація
       3. Агрегація
       4. Композиція
       5. Наслідування
       6. Узагальнення
       7. Реалізація
2. ***Які види відношень між класами визначені в мові моделювання UML?***
   * + 1. 1. Залежність
       2. 2. Асоціація(Агрегація, Композиція)
       3. 3. Узагальнення
       4. 4. Реалізація
3. ***Який тип відношення встановлено між класами в даній лабораторній работі?***

Як би я не намагався зрозуміти – якось не дуже виходить.

1. ***Що означає оператор (::)?***

Область видимості. Тобто область, де функція/змінна існують (визначені).