МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

«Наследование классов, базовый класс, производный класс»

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4831 |  |  |  | К.А. Корнющенков |
| к |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

Задание:

Изучить механизм создания нового класса на основе уже существующего, варианты доступа к элементам базового класса из производного.

Создать класс Жилище(название, общая площадь, базовая стоимость проживания человека в сутки, количество жильцов). В классе должен быть метод вывода характеристик жилья на экран. Создать на основе класса Жилище производные классы: Дом (из какого материала, частный или аренда, стоимость аренды), Коттедж(площадь земельного участка, количество этажей), Хостел(на какое количество жителей). В каждом производном классе предусмотреть метод расчета стоимости проживания в сутки.

Листинг программы:

Main.cpp

#include <iostream>

#include <string>

#include "Home.hpp"

#include "House.hpp"

#include "Cottage.hpp"

#include "Hostel.hpp"

using namespace std;

//MARK: main

int main() {

Home home("дом",76,2600,4);

home.print();

House house("дерево", "аренда", 4500);

house.print();

Cottage cottage(32.2, 2);

cottage.print();

Hostel hostel(13);

hostel.print();

return 0;

}

Cottage.cpp

#include "Cottage.hpp"

#include <iostream>

using namespace std;

Cottage::Cottage(){

this->area = 0;

this->countFloor = 0;

}

Cottage::Cottage(float area,int countFloor){

this->area = area;

this->countFloor = countFloor;

}

int Cottage::finalCost(){

return peopleCount\*750;

}

void Cottage::print(){

cout << "Коттедж" << endl;

cout << "площадь земельного участка " << area << endl;

cout << "количество этажей " << countFloor << endl;

cout << endl;

}

Cottage.hpp

#ifndef Cottage\_hpp

#define Cottage\_hpp

#include <stdio.h>

#include "Home.cpp"

//MARK:Cottage

class Cottage: public Home{

private:

int countFloor;

public:

Cottage(float area,int countFloor);

Cottage();

int finalCost();

void print() override;

};

#endif /\* Cottage\_hpp \*/

Home.cpp

#include "Home.hpp"

#include <iostream>

using namespace std;

Home::Home(){

this->name = "";

this->area = 0;

this->cost = 0;

this->peopleCount = 0;

}

Home::Home(string name,float area,int cost,int peopleCount){

this->name = name;

this->area = area;

this->cost = cost;

this->peopleCount = peopleCount;

}

void Home::print(){

cout << "Жилище" << endl;

cout << "название " << name << endl;

cout << "общая площадь " << area << endl;

cout << "базовая стоимость проживания человека в сутки " << cost << endl;

cout << "количество жильцов " << peopleCount << endl;

cout << endl;

}

Home.hpp

#ifndef Home\_hpp

#define Home\_hpp

#include <stdio.h>

#include <string>

using namespace std;

class Home{

protected:

string name;

float area;

int cost;

int peopleCount;

public:

Home(string name,float area,int cost,int peopleCount);

Home();

virtual void print();

};

#endif /\* Home\_hpp \*/

Hostel.cpp

#include "Hostel.hpp"

#include <iostream>

using namespace std;

Hostel::Hostel(int peopleCount){

this->peopleCount = peopleCount;

}

Hostel::Hostel(){

this->peopleCount = 0;

}

int Hostel::finalCost(){

return peopleCount\*250;

}

void Hostel::print(){

cout << "Хостел" << endl;

cout << "количество жителей " << peopleCount << endl;

cout << endl;

}

Hostel.hpp

#ifndef Hostel\_hpp

#define Hostel\_hpp

#include <stdio.h>

#include "Home.hpp"

//MARK:Hostel

class Hostel: public Home{

private:

public:

Hostel(int peopleCount);

Hostel();

int finalCost();

void print() override;

};

#endif /\* Hostel\_hpp \*/

Hostel.cpp

#include "House.hpp"

#include <iostream>

using namespace std;

House::House(string material,string privateOrRent,int cost){

this->cost = cost;

this->material = material;

this->privateOrRent = material;

}

House::House(){

this->cost = 0;

this->material = "";

this->privateOrRent = "";

}

void House::print(){

cout << "Дом" << endl;

cout << "цена " << cost << endl;

cout << "материал " << material << endl;

cout << "частный или аренда - " << privateOrRent << endl;

cout << endl;

}

int House::finalCost(){

return 3500;

}

House\_hpp

#ifndef House\_hpp

#define House\_hpp

#include <stdio.h>

#include <string>

#include "Home.cpp"

using namespace std;

class House: public Home{

private:

string material;

string privateOrRent;

public:

void print() override;

House(string material,string privateOrRent,int cost);

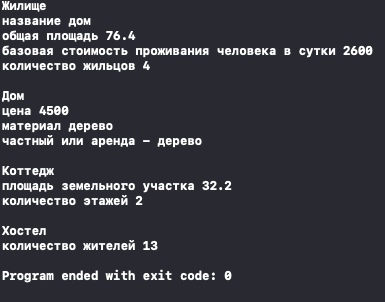
House();

int finalCost();

};

#endif /\* House\_hpp \*/

Скриншоты с результатами:



Вывод:

В ходе выполнения лабораторной работы научились изучили механизм создания нового класса на основе уже существующего и варианты доступа к элементам базового класса из производного.