**«Язык программирования C++»**

**Тема: Наследование. Абстрактные классы**

**Цель:**

Закрепить у слушателей практические навыки и теоретические знания для работы классами и объектами, свойствами. Научиться создавать иерархии классов.

**Требования к отчету:** Отчет должен быть оформлен в виде электронного документа: программный код с комментариями, выводы о результатах выполняемых действий и копии экрана. Размер файла отчета до 2 МБ со скриншотами.

**Задание 1. Р**еализовать программу “**Строительство дома**”.

Реализовать:

- классы

* **House (Дом), Basement (Фундамент), Walls (Стены), Door (Дверь), Window (Окно), Roof (Крыша);**
* **Team (Бригада строителей), Worker (Строитель), TeamLeader (Бригадир),**

- Абстрактные классы

* **IWorker, IPart.**

Все части дома должны реализовать класс **IPart (Часть дома)**, для рабочих и бригадира предоставляется базовый класс **IWorker (Рабочий)**.

Бригада строителей (**Team**) строит дом (**House**). Дом состоит из фундамента (**Basement**), стен (**Wall**), окон (**Window**), дверей (**Door**), крыши (**Part**).

Согласно проекту, в доме должно быть 1 фундамент, 4 стены, 1 дверь, 4 окна и 1 крыша.

Бригада начинает работу, и строители последовательно “строят” дом, начиная с фундамента. Каждый строитель не знает заранее, на чём завершился предыдущий этап строительства, поэтому он “проверяет”, что уже построено и продолжает работу. Если в игру вступает бригадир (**TeamLeader**), он не строит, а формирует отчёт, что уже построено и какая часть работы выполнена.

Все этапы «строительства» бригадир записывать в файл (вести лог строительных работ).

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include <iostream>

#include<conio.h>

#include<time.h>

#include<fstream>

using namespace std;

class IPart

{

public:

virtual void Show()=0;

};

class Basement:public IPart

{

public:

Basement()

{

}

void Show()

{

cout << "Фундамент\n";

}

};

class Walls :public IPart

{

public:

Walls()

{

}

void Show()

{

cout << "Стіна\n";

}

};

class Door :public IPart

{

public:

Door()

{

}

void Show()

{

cout << "Двері\n";

}

};

class Window :public IPart

{

public:

Window()

{

}

void Show()

{

cout << "Вікно\n";

}

};

class Roof :public IPart

{

public:

Roof()

{

}

void Show()

{

cout << "Дах\n";

}

};

class House

{

IPart\*\* arr\_parts;

int positon;

int count\_Basement;

int count\_Walls;

int count\_Door;

int count\_Window;

int count\_Roof;

public:

int& Get\_positon()

{

return positon;

}

int Get\_count\_Basement()

{

return count\_Basement;

}

int Get\_count\_Walls()

{

return count\_Walls;

}

int Get\_count\_Door()

{

return count\_Door;

}

int Get\_count\_Window()

{

return count\_Window;

}

int Get\_count\_Roof()

{

return count\_Roof;

}

IPart\*\* Get\_arr\_parts()

{

return arr\_parts;

}

House(int count\_Basement,int count\_Walls, int count\_Door, int count\_Window, int count\_Roof)

{

/\*cout << "Конструктор House\n";\*/

positon = 0;

this->count\_Basement = count\_Basement;

this->count\_Walls = count\_Walls;

this->count\_Door = count\_Door;

this->count\_Window = count\_Window;

this->count\_Roof = count\_Roof;

arr\_parts = new IPart\*[this->count\_Basement + this->count\_Walls + this->count\_Door + this->count\_Window + this->count\_Roof];

}

~House()

{

/\*cout << "Деструктор House\n";\*/

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

{

delete arr\_parts[i];

}

delete[]arr\_parts;

}

};

class IWorker

{

public:

virtual int Get\_Type\_Of\_IWorker()=0;

};

class Worker:public IWorker

{

public:

Worker()

{

}

int Get\_Type\_Of\_IWorker()

{

return 1;

}

};

class TeamLeader :public IWorker

{

public:

TeamLeader()

{

}

int Get\_Type\_Of\_IWorker()

{

return 2;

}

};

class Team

{

House\* house;

IWorker\*\* arr\_workers;

int positon;

public:

Team(House\* house,int count\_worker, int count\_teamleader=0)

{

/\*cout << "Конструктор team\n";\*/

this->house = house;

arr\_workers = new IWorker \* [count\_worker + count\_teamleader];

positon = 0;

for (; positon < count\_worker; positon++)

{

arr\_workers[positon] = new Worker;

}

for (int i=0; i< count\_teamleader;i++, positon++)

{

arr\_workers[positon] = new TeamLeader;

}

}

~Team()

{

/\*cout << "Деструктор team\n";\*/

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

{

delete arr\_workers[i];

}

delete[] arr\_workers;

}

void Add\_Part(int type)

{

switch (type)

{

case 1:

house->Get\_arr\_parts()[house->Get\_positon()] = new Basement;

house->Get\_arr\_parts()[house->Get\_positon()]->Show();

house->Get\_positon()++;

break;

case 2:

house->Get\_arr\_parts()[house->Get\_positon()] = new Walls;

house->Get\_arr\_parts()[house->Get\_positon()]->Show();

house->Get\_positon()++;

break;

case 3:

house->Get\_arr\_parts()[house->Get\_positon()] = new Roof;

house->Get\_arr\_parts()[house->Get\_positon()]->Show();

house->Get\_positon()++;

break;

case 4:

house->Get\_arr\_parts()[house->Get\_positon()] = new Door;

house->Get\_arr\_parts()[house->Get\_positon()]->Show();

house->Get\_positon()++;

break;

case 5:

house->Get\_arr\_parts()[house->Get\_positon()] = new Window;

house->Get\_arr\_parts()[house->Get\_positon()]->Show();

house->Get\_positon()++;

break;

}

}

void Construction()

{

bool b = false;

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

{

if (arr\_workers[i]->Get\_Type\_Of\_IWorker() == 2)

b = true;

}

for (int i = 0; i < house->Get\_count\_Basement(); i++)

{

Add\_Part(1);

}

for (int i = 0; i < house->Get\_count\_Walls(); i++)

{

Add\_Part(2);

}

for (int i = 0; i < house->Get\_count\_Roof(); i++)

{

Add\_Part(3);

}

for (int i = 0; i < house->Get\_count\_Door(); i++)

{

Add\_Part(4);

}

for (int i = 0; i < house->Get\_count\_Window(); i++)

{

Add\_Part(5);

}

if (b)

{

fstream f;

f.open("e:\\file.txt", ios::out);

if (!f)

{

cout << "FILE ERROR\n";

}

else

{

int x;

x = rand() % 30+1;

f << x << ".";

x = rand() % 12 + 1;

f << x << " збудовано фундамент\n";

x = rand() % 30 + 1;

f << x << ".";

x = rand() % 12 + 1;

f << x << " збудовано стіни\n";

x = rand() % 30 + 1;

f << x << ".";

x = rand() % 12 + 1;

f << x << " збудовано дах\n";

x = rand() % 30 + 1;

f << x << ".";

x = rand() % 12 + 1;

f << x << " поставлені двері\n";

x = rand() % 30 + 1;

f << x << ".";

x = rand() % 12 + 1;

f << x << " поставлені вікна\n";

}

f.close();

}

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int window, door,walls;

int worker, teamleader;

cout << "Скільки зробити стін?\n";

cout << "->";

cin >> walls;

cout << "Скільки поставити вікон?\n";

cout << "->";

cin >> window;

cout << "Скільки поставити дверей?\n";

cout << "->";

cin >> door;

House a(1,walls,door,window,1);

cout << "Скільки додати робітників?\n";

cout << "->";

cin >> worker;

while (worker <= 0)

{

cout << "Додайте хоча б одного робітника\n";

cout << "->";

cin >> worker;

}

cout << "Додати бригадира?(1-так,2-ні)\n";

cout << "->";

cin >> teamleader;

if (teamleader == 1)

{

Team b(&a, worker, 1);

b.Construction();

}

else

{

Team c(&a, worker);

c.Construction();

}

return 0;

}