Федеральное государственное автономное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

О т ч е т

по домашнему заданию № 4

Название домашнего задания: Основы объектно-ориентированного программирования

Дисциплина: Алгоритмизация и программирование.

Студент гр. ИУ6-13Б _____ **С.М Соболев**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ **О.А. Веселовская**
(Подпись, дата) (И.О. Фамилия)

Москва, 2025

Часть 1. Массив объектов.

Цель: Создать класс, описывающий крепостную башню (название, высота, проездная). Реализовать для него конструктор, вывод информации и методы. Написать тестовую программу, которая в массиве объектов найдёт самую высокую башню и посчитает все проездные.

Задание: Разработать класс для реализации указанного объекта. Построить его диаграмму и составить программный код.

Все поля классов должны быть частными (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедуры, единственной задачей которой является вывод информации об объекте на экран.

Объект – крепостная башня. Поля: название, высота, признак "проездная". Методы: конструктор, процедура вывода информации о башне на экран и функции, возвращающие значения полей по запросу.

Разработать тестирующую программу, в которой оборонные укрепления представлены массивом объектов (крепостей). Для укрепления должна определяться самая высокая башня, а также количество пропускных пунктов.

Диаграмма класса:

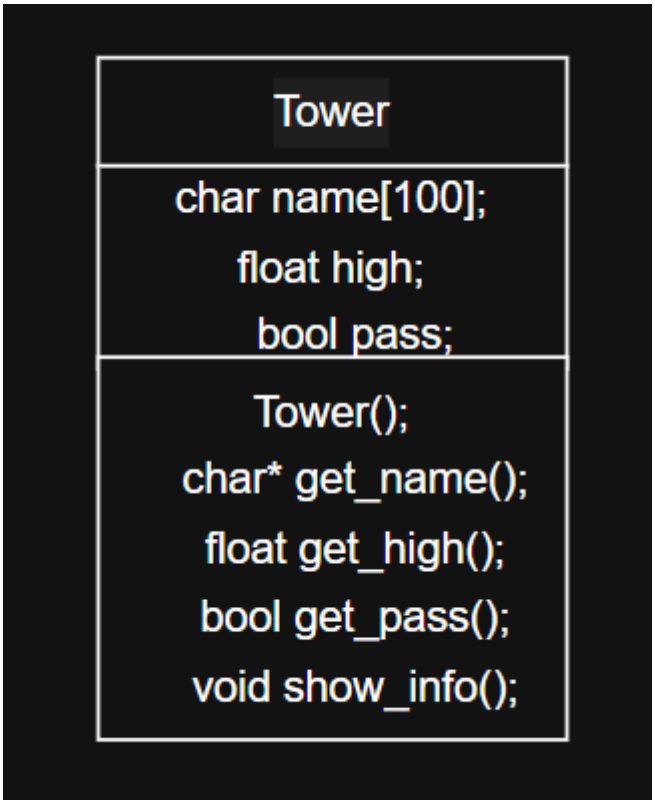


Рисунок 1 – Диаграмма класса Tower

Код файлов программы:

```
#ifndef CLASS_H
#define CLASS_H

class Tower{
protected:
    char name[100];
    float high;
    bool pass;
public:
    Tower(const char *n = "",  float h = 0, bool s = 0);
    char* get_name();
    float get_high();
    bool get_pass();
    void show_info();
};
```

Рисунок 2 – Код заголовочного файла

```cpp
#include <iostream>
#include <string.h>
#include "Class.h"
using namespace std;


Tower::Tower(const char *n ,  float h , bool s){
    strcpy(name, n);
    high = h;
    pass = s;
}
char* Tower::get_name(){
    return name;
}
float Tower::get_high(){
    return high;
}
bool Tower::get_pass(){
    return pass;
}
void Tower::show_info(){
    cout << "Название: " << get_name() << endl;
    cout << "Высота: " << high << endl;
    cout << "Есть проездная: " << (pass ? "да" : "нет") << endl;
}
```

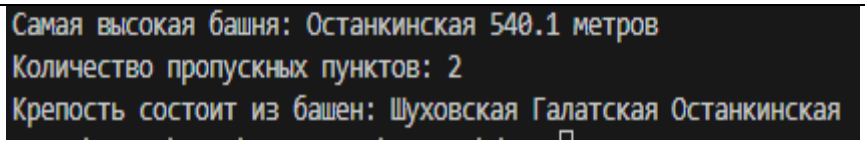
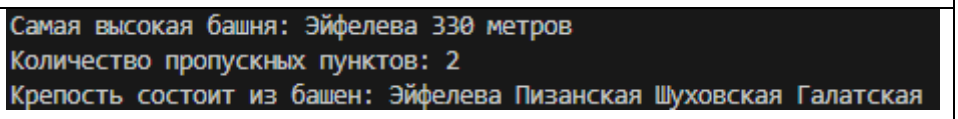
Рисунок 3 – Код файла реализации

```cpp
#include <iostream>
#include <string.h>
#include "Class.h"
#include "CLASS.cpp"

using namespace std;

int main(){
    int k, countpass = 0;
    cout << "Введите кол-во башен в крепости: ";
    cin >> k;
    char name[100];
    float high;
    bool pass;
    Tower castle[k], hmax;
    cout << "Введите название, высоту и признак проездной в башне, через пробел:" << endl;
    for (int i = 0; i < k; i++){

        cin >> name >> high >> pass;
        castle[i] = Tower(name, high, pass);
        if (high > hmax.get_high()) hmax = castle[i];
        if (pass) countpass += 1;
    }
    cout << "--------------------------" << endl;
    cout << "Самая высокая башня: " << hmax.get_name() << " " << hmax.get_high() << " метров" << endl;
    cout << "Количество пропускных пунктов: " << countpass << endl;
    cout << "Крепость состоит из башен: ";
    for (int i = 0; i < k; i++){
        cout << castle[i].get_name() << " ";
    }
    return 0;

}
```

Рисунок 4 - Код тестирующей программы

Тестирование.

Таблица 1 - Тестирование

Входные данные	Выходные данные
3 Шуховская 148.3 1 Галатская 62.5 0 Останкинская 540.1 1	
4 Эйфелева 330 1 Пизанская 55.8 0 Шуховская 148.3 1 Галатская 62.5 0	

Вывод: В ходе работы был разработан класс Tower. Все его поля приватны, методы обеспечивают доступ к данным и вывод информации. Тестовая программа корректно работает с массивом объектов, находит максимальную высоту и подсчитывает проездные башни.

Часть 2. Композиция.

Цель: На основе ранее созданного класса Tower разработать новый класс Castle, представляющий крепость как композицию башен. Построить диаграмму классов для этой композиции и реализовать программный код. В тестирующей программе продемонстрировать работу методов нового класса, инициализацию, вывод информации, подсчёт проездных башен и поиск названия самой высокой.

Задание: Используя разработанный в первой части задания класс, построить диаграмму классов композиционного объекта. Составить программный код описания класса. Разработать тестирующую программу.

Объект – крепость. Параметры: название, количество башен и сами башни. Методы объекта должны позволять: инициализировать объект (конструктор), выводить на экран информацию об объекте, определять количество проездных башен в крепости, получать название самой высокой башни.

Схема алгоритма программы:

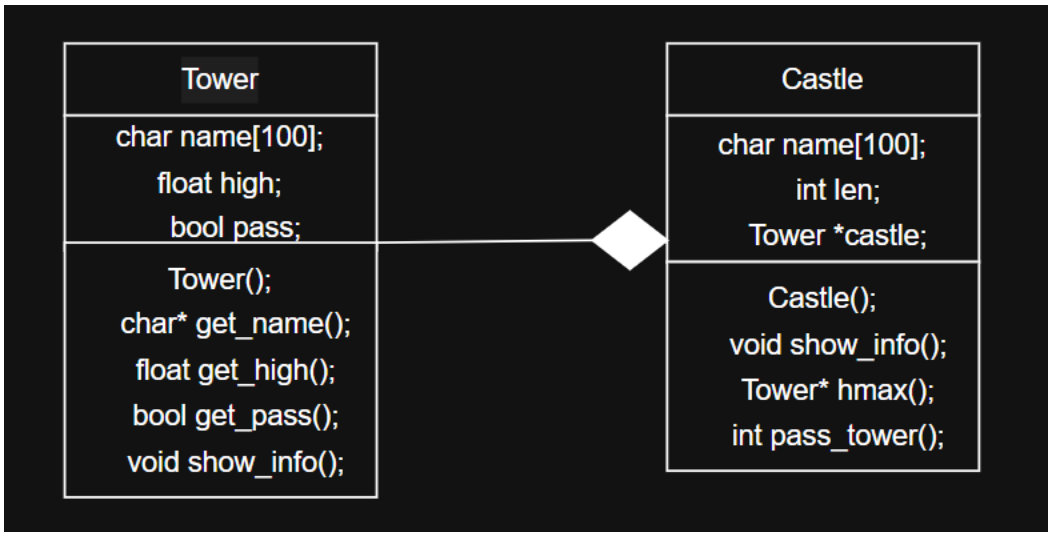


Рисунок 5 – Диаграмма композиции классов

Код файлов программы:

```
#ifndef CLASS_H
#define CLASS_H

class Tower{
protected:
    char name[100];
    float high;
    bool pass;
public:
    Tower(const char *n = "",  float h = 0, bool s = 0);
    char* get_name();
    float get_high();
    bool get_pass();
    void show_info();
};

class Castle{
protected:
    char name[100];
    int len;
    Tower *castle;
public:
    Castle();
    Castle(const char *n = "", int l = 0, Tower *c = {});
    void show_info();
    Tower* hmax();
    int pass_tower();
};


#endif
```

Рисунок 6 – Код заголовочного файла

```
#include <iostream>
#include <string.h>
#include "Class.h"
#include "CLASS.cpp"

using namespace std;

Castle input_castle(){
    int k;
    cout << "Введите кол-во башен в крепости: ";
    cin >> k;
    char namet[100], namec[100];
    cout << "Введите имя крепости: ";
    cin >> namec;
    float high;
    bool pass;
    Tower castle[k];
    cout << "Введите название, высоту и признак проездной в башне, через пробел:" << endl;
    for (int i = 0; i < k; i++){
        cin >> namet >> high >> pass;
        castle[i] = Tower(namet, high, pass);
    }
    return Castle(namec, k, castle);
}

int main(){
    Castle C = input_castle();
    C.show_info();
    cout << "Самая высокая башня: " << C.hmax()->get_name() << endl;
    cout << "Количество пропускных пунктов: " << C.pass_tower() << endl;
    return 0;
}
```

Рисунок 7 – Код тестирующей программы

```cpp
#include <iostream>
#include <string.h>
#include "Class.h"
using namespace std;


Tower::Tower(const char *n ,  float h , bool s){
    strcpy(name, n);
    high = h;
    pass = s;
}
char* Tower::get_name(){
    return name;
}
float Tower::get_high(){
    return high;
}
bool Tower::get_pass(){
    return pass;
}
void Tower::show_info(){
    cout << "Название: " << get_name() << endl;
    cout << "Высота: " << high << endl;
    cout << "Есть проездная: " << (pass ? "да" : "нет") << endl;
}


Castle::Castle() {
    name[0] = '\0';
    len = 0;
    castle= nullptr;
}
Castle::Castle(const char *n, int l, Tower *c){
    strcpy(name, n);
    len = l;
    for (int i = 0; i < len; i++){
        castle[i] = c[i];
    }
}
void Castle::show_info(){
    cout << "Название: " << name << endl;
    cout << "Кол-во башен: " << len << endl;
    cout << "Крепость состоит из башен: ";
    for (int i = 0; i < len; i++){
        cout << (castle+i)->get_name() << " ";
    }
    cout << endl;
}
Tower* Castle::hmax(){
    Tower* max = castle;
    for (int i = 0; i < len; i++){
        if ((castle+i)->get_high() > max->get_high()) max = (i+castle);
    }
    return max;
}
int Castle::pass_tower(){
    int c = 0;
    for (int i = 0; i < len; ++i){
        if (castle[i].get_pass()) c++;
    }
    return c;
}
```

Рисунок 8 – Код файла реализации

Тестирование.

Таблица 2 – Тестирование программы

Входные данные	**Выходные данные**
5 Castle100 Останкинская 540.1 1 Пизанская 55.8 0 Шуховская 148.3 1 Эйфелева 330 1 Галатская 62.5 0	Название: Castle100 Кол-во башен: 5 Крепость состоит из башен: Останкинская Пизанская Шуховская Эйфелева Галатская Самая высокая башня: Останкинская Количество пропускных пунктов: 3

Вывод: Разработан новый класс Castle, использующий в своём составе объекты класса Tower, что подтверждает отношения композиции на построенной диаграмме. Для класса реализованы все требуемые методы: конструктор, вывод полной информации о крепости, расчёт количества проездных башен и поиск самой высокой. Тестирующая программа наглядно показала работоспособность всех функций нового объекта. Работа закрепила понимание принципов композиции и повторного использования существующих классов при проектировании более сложных структур.