

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Заполните поля игровой станции:
id >> 1
name >> Иванов
number >> 01
OnOff >> On
profit >> 5600
numberUsers >> 14

Выберете, куда добавить новую запись?
1 - Добавить запись в конец
2 - Добавить запись в середину
3 - Отмена
Ввод >> 1
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
1 Иванов 01 On 5600 14
2 Иванов 02 On 3200 10
3 Иванов 03 On 8250 43
4 Иванов 04 On 1200 3
5 Иванов 05 On 4012 10
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Суточная норма посетителей составляет 12 человек
Список машин, выполнивших норму:
1 Иванов 01 On 5600 14
3 Иванов 03 On 8250 43
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Суточная норма дохода составляет 5421руб.
Список машин, выполнивших норму :
1 Иванов 01 Оп 5600 14
3 Иванов 03 Оп 8250 43
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Суточная норма дохода составляет 5421руб.
Список машин, НЕ выполнивших норму :
2 Иванов 02 Оп 3200 10
4 Иванов 04 Оп 1200 3
5 Иванов 05 Оп 4012 10
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Максимальная прибыль:
3 Иванов 03 Оп 8250 43
Минимальная прибыль:
4 Иванов 04 Оп 1200 3
Поменять местами Макс и Мин записи?
1 - Да
2 - Нет
Ввод >>
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Выполнили норму прибыли 2 машин.
Сформирован список машин:
1 Иванов 01 Оп 5600 14
3 Иванов 03 Оп 8250 43
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Выберете режим сортировки:
1 - По убыванию
2 - По возрастанию
Ввод >> 1
3 Иванов 03 Оп 8250 43
1 Иванов 01 Оп 5600 14
5 Иванов 05 Оп 4012 10
2 Иванов 02 Оп 3200 10
4 Иванов 04 Оп 1200 3
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
Выберете режим сортировки:
1 - По убыванию
2 - По возрастанию
Ввод >> 2
4 Иванов 04 Оп 1200 3
2 Иванов 02 Оп 3200 10
5 Иванов 05 Оп 4012 10
1 Иванов 01 Оп 5600 14
3 Иванов 03 Оп 8250 43
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
1 test 1 1 1 1
2 test 2 2 2 2
3 test 3 3 3 3
4 4 4 4 4 4
5 test 5 5 5 5
Введите id записи которую нужно удалить >> 5
Удаление прошло успешно!
Для продолжения нажмите любую клавишу . . .
```

```
D:\Project C++\ConsoleApplication10\x64\Debug\ConsoleApplication10.exe
1 test 1 1 1 1
2 test 2 2 2 2
3 test 3 3 3 3
4 4 4 4 4 4
Для продолжения нажмите любую клавишу . . .
```

```

#include <iostream>
#include <string>
#include <Windows.h>
#include <vector>
#include <algorithm>

using namespace std;

const int dailyRateVisitors = 12; // Суточная норма по кол-ву посетителей
const int minProfit = 5421; // Минимальна прибыль

struct Model {
    string id;           // порядковый номер записи
    string name;         // имя ответственного
    string number;       // номер машины
    string OnOff;        // свободна
    string profit;       // доход с места
    string numberUsers;  // количество пользователей за сутки
};

class Controller {
private:
    vector<Model> model;
    vector<Model> propertyModel;

public:
    Controller() { }

    void ShowModel(Model _model) {
        cout << _model.id << " ";
        cout << _model.name << " ";
        cout << _model.number << " ";
        cout << _model.OnOff << " ";
        cout << _model.profit << " ";
        cout << _model.numberUsers << "\n";
    }

    void ShowController(string command) {
        vector<Model> _model;
        if (command == "model") {
            _model = model;
        } else if (command == "propertyModel") {
            _model = propertyModel;
        }

        if (_model.empty()) { cout << "Список пуст\n\n"; return; }
        for (auto i : _model) { ShowModel(i); }
        system("pause");
        system("cls");
    }

    void AddNewModel() {
        cout << "Заполните поля игровой станции:\n";
        Model newModel; newModel.id = to_string(model.size() + 1);

        cout << "id >> " << newModel.id << "\n";

        cout << "name >> "; getline(cin, newModel.name);
        cout << "number >> "; getline(cin, newModel.number);
        cout << "OnOff >> "; getline(cin, newModel.OnOff);
        cout << "profit >> "; getline(cin, newModel.profit);
        cout << "numberUsers >> "; getline(cin, newModel.numberUsers);

        cout << "\nВыберете, куда добавить новую запись?\n";
        cout << "1 - Добавить запись в конец\n";
    }
};

```

```

cout << "2 - Добавить запись в середину\n";
cout << "3 - Отмена\n";
int choice;
cout << "Ввод >> "; cin >> choice; cin.ignore(); // Чистим буфер консоли
чтобы не ломался getline()

int middle = model.size() / 2; // Выносим из switch
switch (choice) {
case 1:
    model.push_back(newModel);
    break;
case 2:
    model.insert(model.begin() + middle, newModel);
    break;
case 3:
    cout << "\nЗапись не была добавлена!\n";
    break;
default:
    cout << "\nНедопустимый ввод!\n";
    break;
}
system("pause");
system("cls");
}

void DeleteModel() {
    if (model.empty()) { cout << "Список пуст\n\n"; return; }

    for (auto i : model) { ShowModel(i); }

    cout << "Введите id записи которую нужно удалить >> ";
    int choice; cin >> choice;

    bool notFound = true;
    int n = model.size(); // Для удаления по индексу
    for (int i = 0; i < n; i++) {
        if (model[i].id == to_string(choice)) {
            auto iter = model.cbegin(); // указатель на первый элемент
            model.erase(iter + i);      // удаляем найденный элемент
            notFound = false;
        }
    }
    if (notFound) {
        cout << "Элемент с указанным интэксом не найден\n0ошибка удаления!\n";
    } else {
        cout << "Удаление прошло успешно!\n";
    }
    system("pause");
    system("cls");
}

void NormCompleted() {
    if (model.empty()) { cout << "Список пуст\n\n"; return; }

    cout << "Суточная норма посетителей составляет " << dailyRateVisitors << "
человек\nСписок машин, выполнивших норму:\n";
    bool errorNotFound = true;
    for (auto i : model) {
        if (stoi(i.numberUsers) >= dailyRateVisitors) {
            ShowModel(i);
            errorNotFound = false;
        }
    }
    if (errorNotFound) {
        cout << "Ни одна машина не выполнила суточную норму\n";
    }
}

```

```

    }

    system("pause");
    system("cls");
}

// это "функциональный объект", внесенный в основной "объект" ради удобства:
void operator()(bool even) {
    if (even) {
        cout << "Суточная норма дохода составляет " << minProfit << "руб.\n";
        bool errorNotFound = true;
        cout << "Список машин, выполнивших норму : \n";
        for (auto i : model) {
            if (stoi(i.profit) >= minProfit) {
                ShowModel(i);
                errorNotFound = false;
            }
        }
        if (errorNotFound) { cout << "Отсутсвуют машины, выполнившие норму\n"; }
    } else {
        cout << "Суточная норма дохода составляет " << minProfit << "руб.\n";
        bool errorNotFound = true;
        cout << "Список машин, НЕ выполнивших норму : \n";
        for (auto i : model) {
            if (stoi(i.profit) < minProfit) {
                ShowModel(i);
                errorNotFound = false;
            }
        }
        if (errorNotFound) { cout << "Все машины выполнили суточную норму\n"; }
    }
    system("pause");
    system("cls");
}

void ProfitMaxMin() {
    if (model.empty()) { cout << "Список пуст\n\n"; return; }

    int _maxModel = 0;
    int _minModel = 0;

    int n = model.size();
    for (int i = 0; i < n; i++) {
        if (stoi(model[i].profit) < stoi(model[_minModel].profit)) { _minModel =
i; }
        if (stoi(model[i].profit) > stoi(model[_maxModel].profit)) { _maxModel =
i; }
    }
    cout << "Максимальная прибыль:\n"; ShowModel(model[_maxModel]);
    cout << "Минимальная прибыль:\n"; ShowModel(model[_minModel]);

    cout << "\nПоменять местами Макс и Мин записи?\n";
    cout << "1 - Да\n";
    cout << "2 - Нет\n";
    int choice;
    cout << "Ввод >> "; cin >> choice; cin.ignore();

    switch (choice) {
    case 1:
        iter_swap(model.begin() + _maxModel, model.begin() + _minModel);
        cout << "\nЭлементы перезаписаны!\n";
        break;
    case 2:
        cout << "\nСохранен порядок элементов!\n";
    }
}

```

```

        break;
    default:
        cout << "\nНедопустимый ввод!\n";
        break;
    }
    system("pause");
    system("cls");
}

void NewVector() {
    if (model.empty()) { cout << "Список пуст\n\n"; return; }

    auto result{ count_if(model.begin(), model.end(), [](Model _model)
        { if (stoi(_model.profit) >= minProfit) return true; return false; }) };

    if (result < 1) { cout << "Ни одна машина не выполнила план!\n\n"; return; }

    cout << "Выполнили норму прибыли " << result << " машин.\n";
    for (auto i : model) {
        if (stoi(i.profit) >= minProfit) {
            propertyModel.push_back(i);
        }
    }
    cout << "Сформирован список машин:\n";
    ShowController("propertyModel");
}

void SortVector() {
    if (model.empty()) { cout << "Список пуст\n\n"; return; }

    cout << "\nВыберете режим сортировки:\n";
    cout << "1 - По убыванию\n";
    cout << "2 - По возрастанию\n";
    int choice;
    cout << "Ввод >> "; cin >> choice; cin.ignore(); // Чистим буфер консоли
    чтобы не ломался getline()

    // ПРЕДИКАНТЫ СПРЯТАННЫ В СТРОКЕ СОРТИРОВКИ
    switch (choice) {
        case 1:
            sort(model.begin(), model.end(), [](Model a, Model b) { return
stoi(a.profit) > stoi(b.profit); }); // Убывание
            break;
        case 2:
            sort(model.begin(), model.end(), [](Model a, Model b) { return
stoi(a.profit) < stoi(b.profit); }); // Возрастание
            break;
        default:
            cout << "\nНедопустимый ввод!\n";
            break;
    }

    for (auto i : model) { ShowModel(i); }
    system("pause");
    system("cls");
}

};

void startData(Controller& _controller) {
    _controller.AddNewModel();
    _controller.AddNewModel();
    _controller.AddNewModel();
    _controller.AddNewModel();
}

```



```

    _controller.AddNewModel();
}

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    Controller _controller;
    startData(_controller); // добавить запись в конец/середину

    _controller.ShowController("model"); // вывести на печать каждую запись
    _controller.NormCompleted(); // найти характеристики по значению меньше заданной
    константы
    _controller(true); // найти характеристики по значению больше заданной
    _controller(false); // найти характеристики по значению меньше заданной
    _controller.ProfitMaxMin(); //найти максимальное/минимальное значение
    характеристики, поменять местами

    _controller.NewVector(); // посчитать количество, используя условие и
    // переписать часть записей в другой вектор или список в соответствии с условием

    _controller.SortVector(); // отсортировать по одному типу характеристики по
    возрастанию;
    _controller.SortVector(); // отсортировать по другому типу характеристики по
    убыванию

    _controller.DeleteModel(); // удалить выбранную запись
    _controller.ShowController("model");
}

```