

```

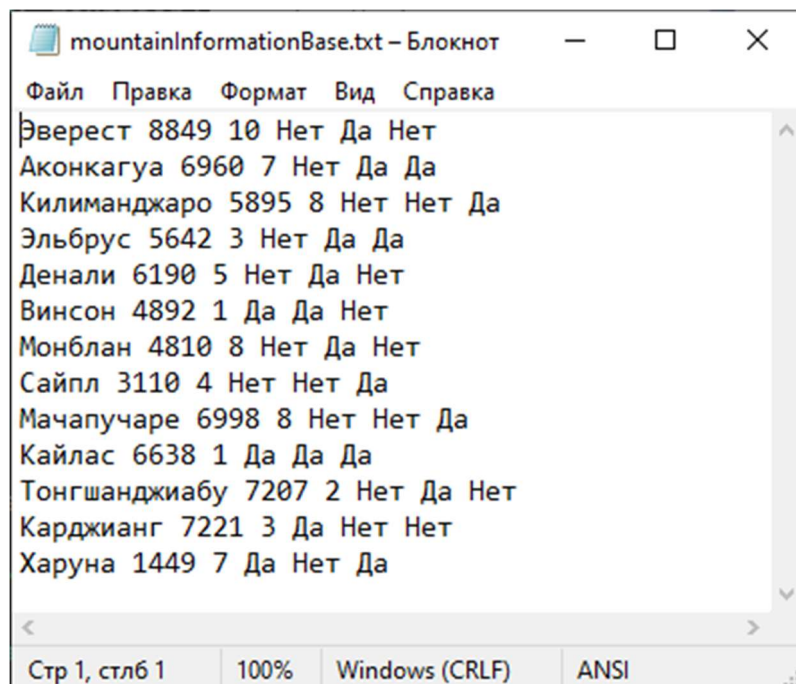
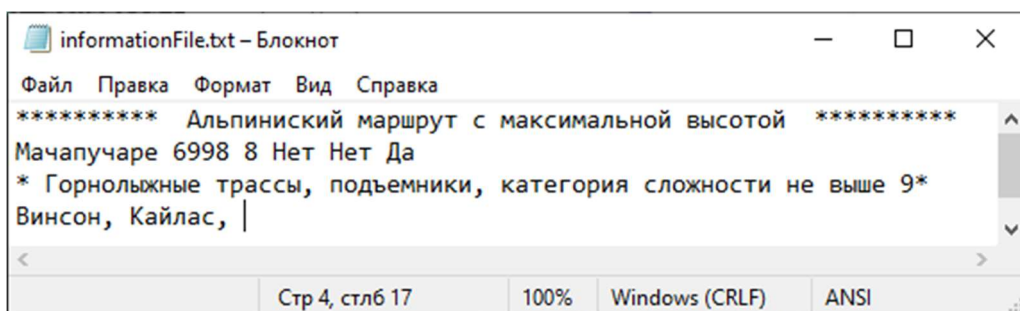
*          Высота
*          *          Сложность
*          *          *          Подъемник
*          *          *          *          Горнолыжные трассы
*          *          *          *          *          Альпинистские маршруты
*          *          *          *          *          *
Эверест      8849    10    Нет    Да    Нет
Аконкагуа    6960    7     Нет    Да    Да
Килиманджаро 5895    8     Нет    Нет   Да
Эльбрус      5642    3     Нет    Да    Да
Денали       6190    5     Нет    Да    Нет
Винсон       4892    1     Да     Да    Нет
Монблан      4810    8     Нет    Да    Нет
Сайпл        3110    4     Нет    Нет   Да
Мачапучаре   6998    8     Нет    Нет   Да
Кайлас       6638    1     Да     Да    Да
Тонгшанджиабу 7207    2     Нет    Да    Нет
Карджианг    7221    3     Да     Нет   Нет
Харуна       1449    7     Да     Нет   Да

***** Альпинистский маршрут с максимальной высотой *****
Мачапучаре   6998    8     Нет    Нет   Да

Введите максимальную категорию сложности (шкала от 0 до 10) >> 9

* Горнолыжные трассы, подъемники, категория сложности не выше 9*
>> Винсон, Кайлас,

```



```

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>

using namespace std;

struct Mountain {
    string name;           // название горы
    string height;         // высота
    string levelComplexity; // категория сложности
    string lift;           // подъемник (Да/Нет)
    string skiTrails;       // горнолыжные трассы (Да/Нет)
    string climbingRoutes;  // альпинистские маршруты (Да/Нет)
};

void ShowMountain(Mountain mountain) {
    cout << setw(15) << mountain.name;
    cout << setw(8) << mountain.height;
    cout << setw(5) << mountain.levelComplexity;
    cout << setw(5) << mountain.lift;
    cout << setw(5) << mountain.skiTrails;
    cout << setw(5) << mountain.climbingRoutes << "\n";
}

int GetNumberRows(string path) {
    int number_of_lines = 0;
    string line;
    ifstream myfile(path);
    while (std::getline(myfile, line)) {
        ++number_of_lines;
    }
    return number_of_lines;
}

ostream& operator<< (ostream& os, const Mountain& mountain) {
    os << mountain.name << " ";
    os << mountain.height << " ";
    os << mountain.levelComplexity << " ";
    os << mountain.lift << " ";
    os << mountain.skiTrails << " ";
    os << mountain.climbingRoutes << "\n";
    return os;
}

istream& operator>> (istream& is, Mountain& mountain) {
    is >> mountain.name;
    is >> mountain.height;
    is >> mountain.levelComplexity;
    is >> mountain.lift;
    is >> mountain.skiTrails;
    is >> mountain.climbingRoutes;
    return is;
}

int main()
{
    setlocale(LC_ALL, "Rus");
    string path = "mountainInformationBase.txt";

    // Открытие первого файла - общего списка всех гор
    fstream myFile;
    myFile.open(path, fstream::in | fstream::out | fstream::app);

    int N = GetNumberRows(path); // Узнаем количество записей в исходном списке

```

```

Mountain* listMountain = new Mountain[N]; // Создаем локальную копию списка в
память программы

if (!myFile.is_open()) {
    cout << "Ошибка чтения файла";
    return 0; // Завершаем работу ВСЕЙ программы если файл не открыт
}

for (int i = 0; i < N; i++) {
    myFile >> listMountain[i];
}

myFile.close();

// Печатаем шапку таблицы
cout << "\n***** Общая справка о горах
*****\n";
cout << "                Гора\n";
cout << "                *      Высота\n";
cout << "                *      *      Сложность\n";
cout << "                *      *      *      Подъемник\n";
cout << "                *      *      *      *      Горнолыжные трассы\n";
cout << "                *      *      *      *      *      Альпинистские маршруты\n";
cout << "                *      *      *      *      *      *\n";

for (int i = 0; i < N; i++) {
    ShowMountain(listMountain[i]);
}

Mountain firstSearchMountain;
int heightMountain = 0;
for (int i = 0; i < N; i++) {
    if (stoi(listMountain[i].height) > heightMountain &&
listMountain[i].climbingRoutes == "Да") {
        heightMountain = stoi(listMountain[i].height);
        firstSearchMountain = listMountain[i];
    }
}

cout << "\n***** Альпинистский маршрут с максимальной высотой
*****\n";
ShowMountain(firstSearchMountain);

fstream myFileInfo; myFileInfo.open("informationFile.txt", fstream::in |
fstream::out | fstream::app);
myFileInfo << "***** Альпинистский маршрут с максимальной высотой
*****\n";
myFileInfo << firstSearchMountain;

int myLevel;
cout << "\nВведите максимальную категорию сложности (шкала от 0 до 10) >> ";
cin >> myLevel;

if (myLevel > 10 or myLevel < 0) {
    cout << "\tОшибка ввода категории сложности!";
    return 0; // Завершаем работу ВСЕЙ программы
}

string secondSearchMountain;

for (int i = 0; i < N; i++) {
    if (stoi(listMountain[i].levelComplexity) < myLevel &&
listMountain[i].lift == "Да" && listMountain[i].skiTrails == "Да") {
        secondSearchMountain += listMountain[i].name + ", ";
    }
}

```

```
    }

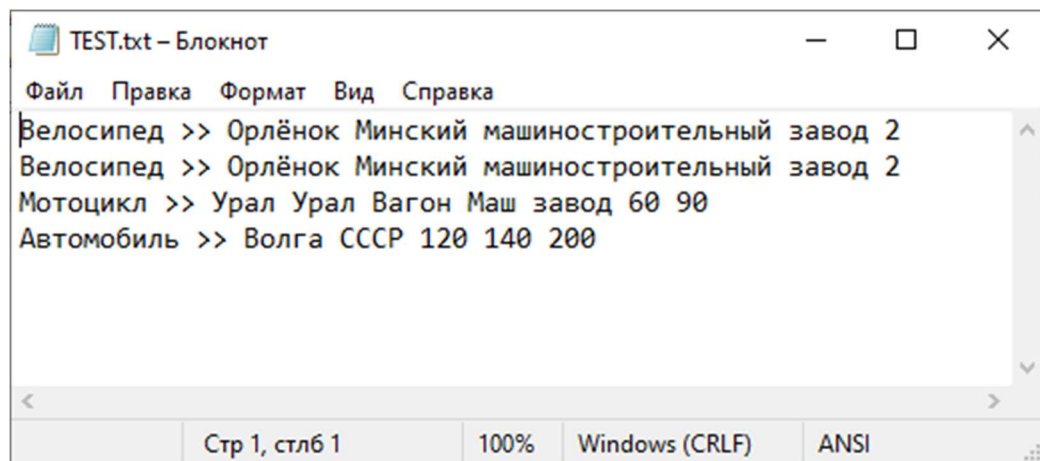
    cout << "\n* Горнолыжные трассы, подъемники, категория сложности не выше " <<
myLevel << " *\n";
    cout << "\t>> " << secondSearchMountain;

    myFileInfo << " * Горнолыжные трассы, подъемники, категория сложности не выше
" << myLevel << " *\n";
    myFileInfo << secondSearchMountain << "\n";
    myFileInfo.close();

    return 0;
}
```

Консоль отладки Microsoft Visual Studio

```
Текущая кодовая страница: 1251
Заполним данные, для продолжения нажмите Enter
Добавим велосипед, заполните поля:
Название >> Оrlёнок
Бренд >> Минский машиностроительный завод
Количество колес >> 2
Добавим мотоцикл, заполните поля:
Название >> Урал
Бренд >> Урал Вагон Маш завод
Мощность двигателя >> 60
Максимальная скорость >> 90
Добавим автомобиль, заполните поля:
Название >> Волга
Бренд >> СССР
Мощность двигателя >> 120
Максимальная скорость >> 140
Грузоподъемность >> 200
```



```
TEST.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
Велосипед >> Оrlёнок Минский машиностроительный завод 2
Велосипед >> Оrlёнок Минский машиностроительный завод 2
Мотоцикл >> Урал Урал Вагон Маш завод 60 90
Автомобиль >> Волга СССР 120 140 200
Стр 1, столб 1    100%    Windows (CRLF)    ANSI
```

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <string>

using namespace std;

string path = "TEST.txt";

class Transport abstract {
public:
    virtual void AppendTransport() = 0;
};

class Bicycle : public Transport
{
private:
    string name;
    string brand;
    int numberWheels;

    void RecordFaile() {
        fstream myFile;
        myFile.open(path, fstream::in | fstream::out | fstream::app);
        myFile << "Велосипед >> " << name << " " << brand << " " <<
numberWheels << "\n";
        myFile.close();
    }
};
```

```

    }

public:
    Bicycle(string newName = "default", string newBrand = "default", int
newNumberWheels = 0) {
        name = newName;
        brand = newBrand;
        numberWheels = newNumberWheels;

    }

    void AppendTransport() override {
        cin.ignore();
        cout << "Добавим велосипед, заполните поля:\n";
        cout << "Название >> "; getline(cin, name);
        cout << "Бренд >> "; getline(cin, brand);
        cout << "Количество колес >> "; cin >> numberWheels;
        RecordFaile();
    }
};

class Motorcycle : public Transport
{
private:
    string name;
    string brand;
    double enginePower;
    double maximumSpeed;

    void RecordFaile() {
        fstream myFile;
        myFile.open(path, fstream::in | fstream::out | fstream::app);
        myFile << "Мотоцикл >> " << name << " " << brand << " " << enginePower
<< " " << maximumSpeed << "\n";
        myFile.close();
    }

public:
    Motorcycle(string newName = "default", string newBrand = "default", int
newEnginePower = 0, int newMaximumSpeed = 0) {
        name = newName;
        brand = newBrand;
        enginePower = newEnginePower;
        maximumSpeed = newMaximumSpeed;

    }

    void AppendTransport() override {
        cin.ignore();
        cout << "Добавим мотоцикл, заполните поля:\n";
        cout << "Название >> "; getline(cin, name);
        cout << "Бренд >> "; getline(cin, brand);
        cout << "Мощность двигателя >> "; cin >> enginePower;
        cout << "Максимальная скорость >> "; cin >> maximumSpeed;
        RecordFaile();
    }
};

class Car : public Transport
{
private:
    string name;
    string brand;
    double enginePower;
    double maximumSpeed;
    double loadCapacity;

```

```

        void RecordFaile() {
            fstream myFile;
            myFile.open(path, fstream::in | fstream::out | fstream::app);
            myFile << "Автомобиль >> " << name << " " << brand << " " <<
enginePower << " " << maximumSpeed << " " << loadCapacity << "\n";
            myFile.close();
        }

    public:
        Car(string newName = "default", string newBrand = "default", int
newEnginePower = 0, int newMaximumSpeed = 0, int newLoadCapacity = 0) {
            name = newName;
            brand = newBrand;
            enginePower = newEnginePower;
            maximumSpeed = newMaximumSpeed;
            loadCapacity = newLoadCapacity;
        }

        void AppendTransport() override {
            cin.ignore();
            cout << "Добавим автомобиль, заполните поля:\n";
            cout << "Название >> "; getline(cin, name);
            cout << "Бренд >> "; getline(cin, brand);
            cout << "Мощность двигателя >> "; cin >> enginePower;
            cout << "Максимальная скорость >> "; cin >> maximumSpeed;
            cout << "Грузоподъемность >> "; cin >> loadCapacity;
            RecordFaile();
        }
};

int main()
{
    setlocale(LC_ALL, "Rus");
    system("chcp 1251");

    Transport* pointer; // указатель на базовый класс

    Bicycle _bicycle; // Экземпляр производного класса
    Motorcycle _motorcycle; // Экземпляр производного класса
    Car _car; // Экземпляр производного класса

    cout << "Заполним данные, для продолжения нажмите Enter";
    /* cin.ignore() Необходим так как после команды cin остается пробел,
    * который последняя использует как разделитель getline получает
    * на вход пробел и перестает работать */

    // Полиморфизм в действии
    pointer = &_bicycle;
    pointer->AppendTransport();

    pointer = &_motorcycle;
    pointer->AppendTransport();

    pointer = &_car;
    pointer->AppendTransport();

    return 0;
}

```