**ИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Языки программирования

|  |
| --- |
| Контейнеры и алгоритмы STL |

Руководитель А.В. Проскурин

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219040 К.В. Трифонов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Получение практических навыков разработки и отладки программ по обработке строк c использованием стандартной библиотеки шаблонов (STL).

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.

2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как номер\_в\_списке\_группы - количество\_заданий).

3. Разработать классы согласно варианту задания.

4. Написать и отладить программу на подготовленных наборах тестовых данных.

5. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:

• титульный лист;

• цель лабораторной работы;

• постановку задачи;

• схему наследования классов (UML диаграмма классов);

• текст программы с комментариями;

• демонстрацию работы программы (Снимки экрана при выполнении действий программы с описанием).

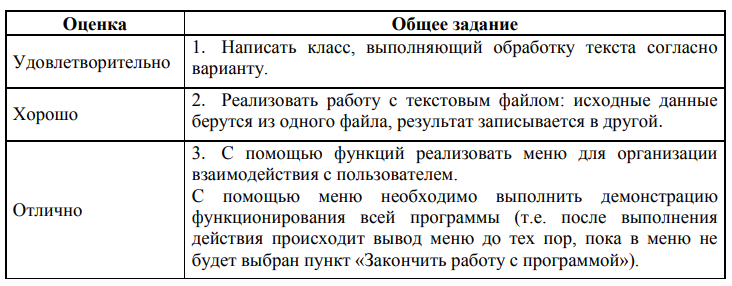
• краткие ответы на контрольные вопросы;

• выводы по лабораторной работе.

6. Защитить лабораторную работу перед преподавателем.

# постановка задачи

Необходимо реализовать класс (а также при необходимости шаблонные функции), который будет выполнять обработку текстового файла согласно варианту задания. При реализации необходимо использовать контейнеры и алгоритмы стандартной библиотеки шаблонов (STL) подходящие для решения задачи (например, контейнеры: vector, map, set, list, и алгоритмы: find, sort, и т.п.), а также методы класса string. В зависимости от оценки, на которую вы претендуете, необходимо выполнить следующие задания (Для каждой следующей оценки нужно выполнить ВСЕ предыдущие задания, если обратное не указано явно):

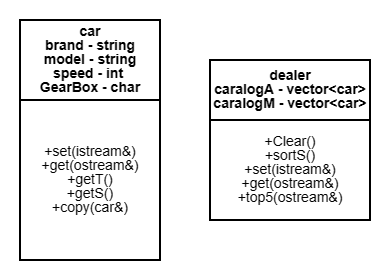


**Вариант №21.**

**Формулировка задания**: на основе каталога автосалона требуется определить пять самых быстрых автомобилей с автоматической коробкой передач и пять с механической. Вывод данных на экран осуществлять с группировкой по коробке передач и отображать в алфавитном порядке по формату: Марка автомобиля Модель (Скорость)

**Формат входных данных**: Марка автомобиля, Модель, Скорость автомобиля и тип коробки передач.

# Схема классов



# ХОД РАБОТЫ

**Класс автомобиля car:**

*class* car{

*private:*

        string brand; *// марка автомобиля*

        string model; *// модель автомобиля*

*int* speed; *// скорость автомобиля*

*char* GearBox; *// 1 буква коробки передач автомобиля*

*public:*

*// установить значения из входного потока*

*void* set(istream*&* *in*);

*// получить значения в выходной поток*

*void* get(ostream*&* *out*) *const*;

*// получить тип коробки передач*

*char* getT() *const* { return GearBox; }

*// получить скорость*

*int* getS() *const* { return speed; }

*// скопировать значения из другого объекта car*

*void* copy(*const* car*&* *B*);

*// перегрузка оператора присваивания*

        car*&* operator= (*const* car*&* *B*){ copy(*B*); return \**this*; }

*// перегрузка оператора меньше*

*bool* operator<(*const* car*&* *B*) *const*;

*// перегрузка оператора вывода в поток*

*friend* ostream*&* operator<<(ostream*&*, *const* car*&* *B*);

*// перегрузка оператора ввода из потока*

*friend* istream*&* operator>>(istream*&*, car*&*);

};

Описание методов:

Метод ввода из потока:

*void* car::set(istream*&* *in*){

        in>>brand;

        in>>model;

        in>>speed;

        in>>GearBox;

}

Метод вывода в поток:

*void* car::get(ostream*&* *out*)*const*{

    out<<brand<<" ";

    out<<model<<" ";

    out<<speed<<" ";

    out<<GearBox<<" ";

}

Перегрузка операторов:

ostream*&* operator<<(ostream*&* *out*, *const* car*&* *B*){

    B.get(out);

    return out;

}

istream*&* operator>>(istream*&* *in*, car*&* *B*){

    B.set(in);

    return in;

}

Метод копирования

*void* car::copy(*const* car*&* *B*){

    brand = B.brand;

    model = B.model;

    speed = B.speed;

    GearBox = B.GearBox;

}

Перегрузка оператора меньше для работы функции sort

*bool* car::operator<(*const* car*&* *B*)*const*{

    if (GearBox < B.GearBox)

        return true;

    else if (GearBox == B.GearBox){

        if (brand<B.brand){

            return true;

        }

        else if (brand == B.brand){

            if (model < B.model)

                return true;

            else if (model == B.model){

                if (speed < B.speed)

                    return true;

                else

                    return false;

            }

            else

                return false;

        }

        else

            return false;

    }

    else

        return false;

}

**Класс автодилер dealer:**

*class* dealer{

*private:*

        vector<car> catalogA; *// каталог автомобилей с автоматом*

        vector<car> catalogM; *// каталог автомобилей с механикой*

*public:*

*// очистить каталоги*

*void* Clear();

*// отсортировать каталог по скорости*

*void* sortS();

*// установить значения из входного потока*

        istream*&* set(istream*&* *in*);

*// получить значения в выходной поток*

        ostream*&* get(ostream*&* *out*) *const*;

*// вывести топ 5 автомобилей*

        ostream*&* top5(ostream*&* *out*);

};

Описание методов:

Ввод из потока: проверяется какого типа temp и записывается в соответствующий каталог с помощью push\_back()

istream*&* dealer::set(istream*&* *in*){

    Clear();

    if (&*in* == &cin){

*int* size;

        cout<<"Введите размер каталога:\n";

        cin >> size;

        cout<<"Каталог из "<<size<<" авто - марка, модель,  максимальную скорость и коробку передач:\n";

        car temp;

        for (*int* i = 0; i < size; i++){

*in*>>temp;

            if (temp.getT() == 'A'){

                catalogA.push\_back(temp);

            }

            else{

                catalogM.push\_back(temp);

            }

        }

    }

    else{

        car temp;

        while(*in*>>temp){

            if (temp.getT() == 'A'){

                catalogA.push\_back(temp);

            }

            else{

                catalogM.push\_back(temp);

            }

        }

    }

    sort(catalogA.begin(),catalogA.end());

    sort(catalogM.begin(),catalogM.end());

    return *in*;

}

Вывод в поток:

ostream*&* dealer::get(ostream*&* *out*)*const*{

    if (&*out* == &cout){

        cout<<"Каталог:\n";

        cout<<"Автомобили с автоматической коробкой передач:\n";

    }

    for (*int* i = 0; i < catalogA.size(); i++){

*out*<<catalogA[i]<<"\n";

    }

    if (&*out* == &cout)

        cout<<"Автомобили с механической коробкой передач:\n";

    for (*int* i = 0; i < catalogM.size(); i++){

*out*<<catalogM[i]<<"\n";

    }

    return *out*;

}

Сортировка каталогов по полю скорости с помощью метода getS():

*void* dealer::sortS(){

    for (*int* i = 0; i < catalogA.size(); i++)

    {

*bool* flag = true;

        for (*int* j = 0; j < catalogA.size() - 1; j++)

        {

            if (catalogA[j].getS()<catalogA[j+1].getS())

            {

                flag = false;

                swap(catalogA[j],catalogA[j+1]);

            }

        }

        if (flag)

        {

            break;

        }

    }

    for (*int* i = 0; i < catalogM.size(); i++)

    {

*bool* flag = true;

        for (*int* j = 0; j < catalogM.size() - 1; j++)

        {

            if (catalogM[j].getS()<catalogM[j+1].getS())

            {

                flag = false;

                swap(catalogM[j],catalogM[j+1]);

            }

        }

        if (flag)

        {

            break;

        }

    }

}

Вывод в поток 5 самых быстрых автомобилей с автоматической и механической коробкой передач:

ostream*&* dealer::top5(ostream*&* *out*){

    if (&*out* == &cout){

        cout<<"5 самых быстрых авто на автоматической коробке передач:\n";

    }

    sortS();

    car temp;

*int* c = 0;

    for (*int* i = 0; i < catalogA.size() && c<5; i++){

        cout<<catalogA[i]<<"\n";

        c++;

    }

    if (&*out* == &cout){

        cout<<"5 самых быстрых авто на механической коробке передач:\n";

    }

    c = 0;

    for (*int* i = 0; i < catalogM.size() && c<5; i++){

        cout<<catalogM[i]<<"\n";

        c++;

    }

    return *out*;

}

Очистка векторов для перезаписи:

*void* dealer::Clear(){

    catalogA.clear();

    catalogM.clear();

}

# Тестирование

Для тестирования программы было создано меню:

     while (f){

        cout<<"Главное меню\n"

        <<"1) Прочитать каталог из файла input.txt\n"

        <<"2) Прочитать каталог из консоли\n"

        <<"3) Вывести каталог в файл output.txt\n"

        <<"4) Вывести каталог в консоль\n"

        <<"5) Вывести 5 самых быстрых автомобилей каждой категории\n"

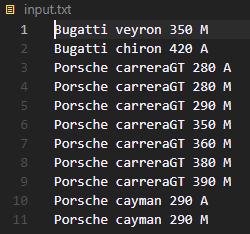
        <<"0) <<< Выход\n";

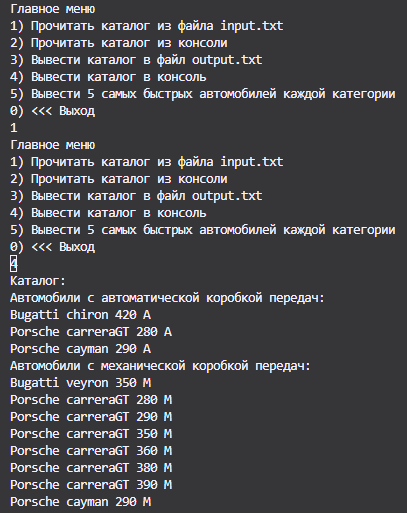
        cin>>sw;

        switch(sw){

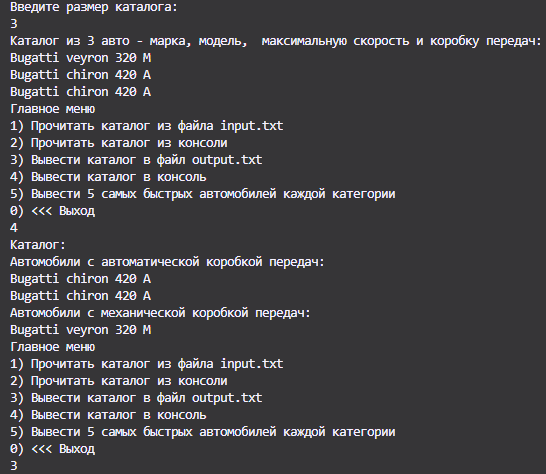
Пример чтения с последующим выводом на экран в алфавитном порядке файла input.txt:

Содержимое файла input.txt:

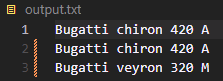




Пример чтения каталога из консоли с последующим выводом в файл output.txt:

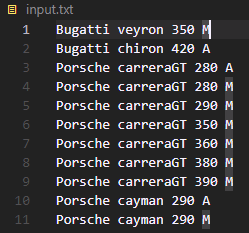


Содержимое файла output.txt после работы программы:

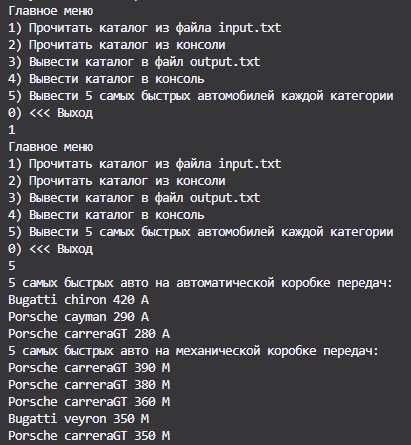


Чтение из файла input.txt и вывод 5 самых быстрых автомобилей каждой категории (АКПП и МКПП):

Содержимое файла input.txt:



Результат работы програмы:



Из 8 авто на МКПП было выведено только 5 самых быстрых, 3 авто на АКПП были выведены без исключений.

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. **Охарактеризуйте класс vector, как происходит обращение к элементам?**

Элементы вектора можно обращать по индексу, используя оператор "[]", или с помощью метода "at()".

1. **Как происходит добавление/удаление элементов в классе vector?**

Элементы можно добавлять в конец вектора с помощью метода "push\_back()", удалять с конца с помощью "pop\_back()". Вставка и удаление элементов в произвольной позиции осуществляется с помощью методов "insert()" и "erase()".

1. **При помощи какого метода можно указать количество элементов в классе vector?**

Чтобы указать количество элементов в векторе, используется метод "resize()".

1. **При помощи какого метода можно изменить количество элементов в классе vector, как при этом будут происходить изменения?**

Количество элементов в векторе можно изменить с помощью методов "resize()" и "reserve()". При добавлении элементов в вектор его размер автоматически увеличивается.

1. **Охарактеризуйте класс array, как осуществляется его передача в функцию.**

Класс array передается в функцию по значению или по ссылке, аналогично другим объектам.

1. **Поясните, что из себя представляют контейнеры set/multiset, приведети плюсы и минусы использования set и multiset.**

Контейнеры set и multiset представляют собой отсортированные уникальные значения. Основное различие между ними - multiset позволяет хранить дублирующиеся элементы, в то время как set - нет.

1. **Охарактеризуйте контейнер list. Приведите методы для добавления/удаления и обращения элементам.**

Контейнер list представляет собой двусвязный список. Элементы добавляются в начало или конец списка с помощью методов "push\_front()" и "push\_back()", удаляются с помощью "pop\_front()" и "pop\_back()".

1. **Что такое итератор, для чего он нужен. Приведите пример (с применением синтаксиса) использования итераторов.**

Итератор - это объект, позволяющий перебирать элементы контейнера. Например:

for (*int* j = 0; j < catalogA.size() - 1; j++){

            if (catalogA[j].getS()<catalogA[j+1].getS())

            {

                flag = false;

                swap(catalogA[j],catalogA[j+1]);

            }

        }

1. **Охарактеризуйте контейнер map. Как происходит работа с контейнером map?**

Контейнер map представляет собой ассоциативный массив, состоящий из пар ключ-значение. Элементы добавляются с помощью метода "insert()" и доступны по ключу.

1. **Как проверить существование ключа в контейнере map?**

Чтобы проверить существование ключа в контейнере map, можно воспользоваться методом "count()", который возвращает 1, если ключ присутствует, и 0 в противном случае.

1. **Поясните принцип и работу циклов foreach, какая форма записи у данных циклов?**

Цикл foreach (или цикл диапазона) позволяет перебирать элементы коллекции. Например:

for (*auto*& elem : collection) {

*// действия с элементом*

}

1. **Охарактеризуйте методы substr и replace класса string. Поясните принцип объединения строк.**

Метод substr() позволяет извлекать подстроку из строки, а метод replace() заменяет часть строки другой строкой. Объединение строк осуществляется с помощью оператора "+" или метода append().

1. **При помощи каких методов происходит добавление и вставка фрагмента в классе string. Перечислите методы подсчета количества символов строке string.**

Для добавления фрагмента в строку используются оператор "+=" и метод append(), вставка фрагмента осуществляется методом insert(). Методы подсчета количества символов включают size(), length() и метод find() для поиска подстроки.

1. **Приведите синтаксис поиска первого вхождение другой строки (подстроки) в заданную строку.**

Для поиска первого вхождения подстроки в строку используется метод find():

string str = "Hello, World!";

*size\_t* found = str.find("World");

if (found != string::npos) {

*// подстрока найдена*

}

# ВЫВОД

При выполнении лабораторной работы были успешно применены концепции работы с векторами, включая добавление элементов в вектор с помощью метода push\_back(), определение размера вектора при помощи метода size(), а также доступ к элементам вектора по индексу, также проведено ознакомление с контейнерами map. Использование векторов позволяет эффективно управлять строковыми данными, обеспечивая динамическое изменение размера контейнера в зависимости от потребностей программы.