**Отчет по лабораторной работе №2**

**Пронина Арсения, ИУ7-32Б**

**Вариант 25**

# Условия задачи

Ввести список машин, имеющихся в автомагазине, содержащий: марку автомобиля, страну-производитель, цену, цвет и состояние: новый – гарантия (в годах); нет - год выпуска, пробег, количество ремонтов, количество собственников. Вывести цены не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен.

# Техническое задание

## Входные данные

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна). Осуществить поиск информации по варианту.

## Требования к входным данным

# Структуры данных

*struct* **new\_car**

{

int guarantee;

};

*struct* **old\_car**

{

int year\_of\_release;

int probeg;

int repair\_count;

int owners\_count;

};

*union* **state**

{

new\_car new\_car\_params;

old\_car old\_car\_params;

};

*struct* **car**

{

string brand;

string manufacturer\_country;

int price;

string color;

bool is\_new;

state condition;

};

# Описание алгоритма

Вся информация о таблице записей о машинах хранится в динамическом массиве типа car\*

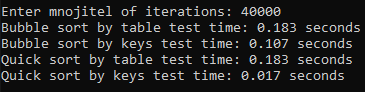
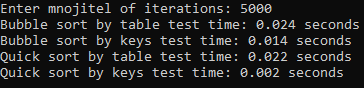
# Тесты

(проверки неккоректного ввода)(показать как работает меню)

# Тип “запись” с вариантной частью

Поскольку нет необходимости хранить оба поля объединения **state** одновременно, то можно сэкономить память на хранении этой информации используя *union*.

# Сравнительный анализ



(показать в процентах разницу в скорости и памяти (таблицей))

Чем больше таблица, тем наглядней видно, что сортировка с помощью ключей экономит значительное количество времени.

Тем не менее использование дополнительного массива влечёт за собой использование дополнительной памяти.

При всём это необходимо не забывать, что при использовании строкового типа ключа эффективность этого метода снижается из-за необходимости сравнивать поля посимвольно.

# Контрольные вопросы

**1.Как выделяется память под вариантную часть записи?**

Выделяется общий блок памяти = размеру наибольшего поля

**2.Что будет, если в вариантную часть ввести данные, несоответствующие описанным?**

Данные будут заменены, но будут нести не корректную информацию.

**3.Кто должен следить за правильностью выполнения операций с вариантной частью записи?**

Программист (При обработке записей с вариантами программисту необходимо следить за правильностью хранения и обработки данных, содержащихся в вариантной части.)

**4.Что представляет собой таблица ключей, зачем она нужна?**

Матрица индексов ключевого поля и соответствующих им значений. Нужна для экономии времени.

**5.В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?**

Таблицу ключей при большом кол-ве записей. Но следует учитывать, что если в качестве ключа используется символьное поле записи, то это влечет за собой необходимость посимвольной обработки данного поля в цикле, и, следовательно, приводит к увеличению времени выполнения любых операций. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, так же замедляет вывод этих данных.

# Вывод

(вариатные поля + таблица ключей) /\*Я приобрёл навыки работы с типом данных «запись» (структура, struct), содержащим вариантную часть (объединение, смесь, union), и с данными, хранящимися в таблицах. Произвёл сравнительный анализ реализации разных алгоритмов сортировки (bubble и qsort), при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей. А также оценил эффективность программы по используемому объему памяти при использовании объединений. \*/