**Отчет**

**по лабораторной работе №2**

**Выполнил:**

студент группы ИУ7-41

Рассказов Владимир

**Проверила:**

Силантьева А.В.

**Содержание отчета**

* Как выделяется память под вариантную часть записи?
* Что будет, если в вариантную часть ввести данные, несоответствующие описанным?
* Кто должен следить за правильностью выполнения операций с вариантной частью записи?
* Что представляет собой таблица ключей, зачем она нужна?
* В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?
* Какие способы сортировки предпочтительнее для обработки таблиц и почему?

**Ответы**

* Объем памяти, необходимый для записи с вариантами складывается из длин полей фиксированной части и максимального по длине поля вариантной части.
* Произойдет ошибка. Контроль ввода неправильных данных лежит на программисте.
* Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста.
* Таблица ключей - дополнительного массив, содержащий индекс элемента в исходной таблице и выбранный ключ.
* При больших размерах таблиц поиск данных, имеющих указанный ключ, может потребовать больших затрат времени. Если же помимо поиска требуется произвести сортировку данных, то временные затраты многократно возрастут, так как потребуется осуществлять их перестановку (перемещение).
* Выбор способа сортировки зависит от размеров таблицы и ключей сортировки. Так, для больших таблиц лучше использовать сортировку по ключу. А для небольших – это будет лишней тратой памяти и практически не даст выигрыша во времени. Поэтому способ сортировки выбирает программист.

**Техническое задание**

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести сравнительный анализ реализации алгоритма поиска информации по вариантному полю при использовании 2-х языков программирования. Упорядочить таблицу по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: а) исходную таблицу; б) массив ключей. Сравнить различные алгоритмы сортировки массива при использовании таблиц записей с большим числом полей и таблиц ключей. Оценить эффективность (по времени и по используемому объему памяти) при различной реализации программы, то есть в случаях а) и б). Обосновать выбор алгоритма упорядочивания. Оценка эффективности должна быть относительной (в %).

**Техническое задание**

**Ввести список абонентов, содержащий фамилию, имя, телефон, адрес, статус (личный - дата рождения: день, месяц, год; служебный - должность, организация). Найти всех друзей, которых необходимо поздравить с днем рождения в ближайшую неделю.**

**Типы данных, используемые для хранения данных**

struct group {

QString name;

QString surname;

QString tel;

QString address;

QString status;

QVariant v;

};

struct privacy {

int dd;

int mm;

int yyyy;

};

struct work {

QString position;

QString org;

};

struct key

{

int pos;

QString surname;

};

**Сравнение реализаций поиска на двух языках**

1.Реализация на С++

void MainWindow::findBd(QVector<int> &bd)

{

bd.clear();

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

if (birthday(i))

bd.push\_back(i);

}

2.Реализация на Delphi

procedure FindSel(var Keys: TKeys; var Data: TData; var Bd: TIntAr);

var

I: Integer;

begin

SetLength(Bd, 0);

for I := 0 to Length(Keys) - 1 do

if (bithrday(i, keys, data)<8) then

begin

SetLength(Bd, Length(Bd) + 1);

Bd[Length(Bd) - 1] := i;

end;

end;

Код, написанный на С++, работает быстрее, чем код, написанный на Delphi.

Реализация на С++ более кратка и удобна.

**Сравнение скорости сортировки разными алгоритмами списка квартир и списка ключей в реализации C++**



**Основной алгоритм**

* Чтение данных из файла.
* Печать таблицы данных.
* Добавление элемента.
* Удаление элемента по индексу.
* Сортировка таблицы данных по полю «фамилия» сортировкой вставками.
* Сортировка таблицы ключей по полю «фамилия» сортировкой выбором.
* Поиск всех друзей, у которых день рождения в ближайшие 7 дней.

В результате проведения сортировок на таблицах различных размеров, можно сделать следующие выводы:

* При малом размере таблицы (менее 40-80 записей), сортировка ключей и сортировка основного массива отличаются по времени не более, чем в 3 раза.
* При большом размере списка (более 80 записей), для выигрыша в скорости лучше использовать сортировку ключей
* Список с ключами занимает на 20% больше памяти, чем таблица без ключей.
* Сравнивая сортировку вставками и сортировку выбором в любом случае по времени выигрывает сортировка выбором, которая и была выбрана в качестве рабочего алгоритма сортировки.

**Тесты к лабораторной работе**

* Пустой ввод

*Результат:* выдача сообщения

* Неверный ввод

*Результат:* Проверка ввода перехватывает его

* Чтение из файла

*Результат:* Выведенная таблица

* Чтение из файла, файл пуст

*Результат:* Сообщение о пустоте таблицы

* Добавление записи

*Результат:* Новая запись добавлена

* Удаление элемента по индексу активной строки таблицы

*Результат:* Элемент будет удалён

* Сортировка

*Результат:* Таблица данных/таблица ключей отсортирована

* Сортировка, таблица пуста

*Результат:* Вывод сообщения

* Поиск по полю

*Результат:* Вывод всех друзей, которых надо поздравить

* Поиск по полю, подходящих элементов нет

*Результат:* Вывод сообщения

* Поиск по полю, список пуст

*Результат:* Вывод сообщения