Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«ЗАПИСИ С ВАРИАНТАМИ. ОБРАБОТКА МАТРИЦ»

по курсу:

«ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Вариант: 7

Студент:

Авдейкина Валерия Павловна, группа ИУ7-33Б _____
(подпись, дата)

Руководители:

Преподаватель ИУ7 _____

Силантьева Александра Васильевна (подпись, дата)
Преподаватель ИУ7

Барышникова Марина Юрьевна _____
(подпись, дата)

Проверяющий:

_____
(подпись, дата)

Оценка: _____

2022 г.

Оглавление

Описание условий задачи....3
Техническое задание....4
1 Входные данные....4
2 Выходные данные....5
3 Задача, реализуемая программой....7
4 Способ обращения к программе....7
5 Возможные аварийные ситуации и ошибки со стороны пользователя....7
Описание внутренних структур данных....9
Алгоритм....11
Тестирование....13
Контрольные вопросы....15
1 Каков возможный диапазон чисел, представляемых в ПК?....15
2 Какова возможная точность представления чисел?....15
3 Какие стандартные операции возможны над числами?....15
4 Какой тип данных может выбрать программист, если обрабатываемые числа превышают возможный диапазон представления чисел в ПК?....15
5 Как можно осуществить операции над числами, выходящими за рамки машинного представления?....16
Выводы....17

Описание условий задачи

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.

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 %).

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 (без учета смены года).

Техническое задание

1 Входные данные

- Входные данные при вводе номера выбранной опции меню:
  - Целые беззнаковые числа [0; 13]
- Входные данные для различных опций меню:

Номер опции	Название опции	Входные данные
0	Выход	-
1	Заполнить таблицу абонентов из файла	Имя файла, количество записей
2	Вывести таблицу с данными об абонентах	-
3	Вывести таблицу с ключами (фамилиями абонентов)	-
4	Добавить абонента в конец таблицы	Поля: «Имя», «Фамилия», «Телефон», «Адрес», «Код Статуса», «Статус-Личный» / «Статус-Официальный»
5	Удалить абонента, фамилия которого совпадает с введенной	Поля: «Фамилия»
6	Отсортировать и вывести таблицу абонентов	-
7	Отсортировать и вывести таблицу ключей (фамилий)	-
8	Отсортировать таблицу абонентов с помощью таблицы ключей	-
9	Вывести результаты сортировки пузырьком и быстрым алгоритмом (qsort)	Имя файла, количество записей
10	Вывести список абонентов, чей Д.Р. -- в ближайшую неделю от введенной даты	Поля: «День», «Месяц»
11	Очистить таблицы	-
12	Вывести меню	-
13	Вывести правила ввода	-

Таблица 1: Входные данные опций меню

- Имя файла: символы, не более 255
- Количество записей: целое беззнаковое число, [40; 3000]
- Поля:
  - «Имя» — латинские символы / цифры, не более 21
  - «Фамилия» — латинские символы / цифры, не более 21
  - «Телефон» — беззнаковое целое число, [1,400,000,00,00; 1,999,999,99,99]
  - «Адрес» — латинские символы / цифры, не более 36
  - «КодСтатуса» — целое беззнаковое: 0 — «Статус-Официальный», 1 — «Статус-Личный»
    - «Статус-Официальный»:
      - «Компания» — латинские символы / цифры, не более 31
      - «Должность» — латинские символы / цифры, не более 21
    - «Статус-Личный»:
      - «День» — целое беззнаковое число, [1; 31]
      - «Месяц» — целое беззнаковое число, [1; 12]
      - «Год» — целое беззнаковое число, [1800; 2500]

2 Выходные данные

- Выходные данные при вводе номера выбранной опции меню:
  - Выходные данные команды в случае корректного ввода
- Выходные данные для различных опций меню:

Номер опции	Название опции	Выходные данные
0	Выход	Сообщение об успехе
1	Заполнить таблицу абонентов из файла	Сообщение об успехе / сообщение об ошибке
2	Вывести таблицу с данными об абонентах	Таблица* / сообщение об ошибке
3	Вывести таблицу с ключами (фамилиями абонентов)	Таблица** / сообщение об ошибке
4	Добавить абонента в конец таблицы	Сообщение об успехе / сообщение об ошибке
5	Удалить абонента, фамилия которого совпадает с введенной	Сообщение об успехе с количеством удаленных записей
6	Отсортировать и вывести таблицу абонентов	Таблица* (отсортированная в лексикографическом порядке) / сообшение об ошибке
7	Отсортировать и вывести таблицу ключей (фамилий)	Таблица** (отсортированная в лексикографическом порядке) / сообщение об ошибке
8	Отсортировать таблицу абонентов с помощью таблицы ключей	Таблица* (отсортированная в лексикографическом порядке) / сообшение об ошибке
9	Вывести результаты сортировки пузырьком и быстрым алгоритмом (qsort)	Информация об эффективности / сообщение об ошибке
10	Вывести список абонентов, чей Д.Р. -- в ближайшую неделю от введенной даты	Таблица* с искомыми записями (без учета смены года) / сообщение об ошибке
11	Очистить таблицы	Сообщение об успехе / сообщение об ошибке
12	Вывести меню	Список опций
13	Вывести правила ввода	Список правил

Таблица 2: Выходные данные опций меню

- Таблица*:
  - Индекс, Имя, Фамилия, Телефон, Адрес, Тип статуса, Статус
- Таблица** (ключей) :
  - Индекс, Индекс в исходной таблице, Ключ (фамилия)

3 Задача, реализуемая программой

Программа реализует обработку массива записей с вариантами: сортировку по строковому полю («Фамилия») в лексикографическом порядке, поиск по полю структурного типа («Статус-Личный»), вывод в виде отформатированной таблицы, расчет и анализ времени сортировки массива различными методами (быстрым и медленным — пузырьком).

4 Способ обращения к программе

Программа вызывается в командной строке без каких-либо аргументов. Ввод данных производится с клавиатуры после соответствующего приглашения к вводу («Команда...» / «Введите..»).

5 Возможные аварийные ситуации и ошибки со стороны пользователя

1. Некорректный ввод имени файла:
    1.1. Пустой ввод
    1.2. Недопустимые символы
    1.3. Превышение допустимого количества символов *(поведение программы не определено)*
    1.4. Имя несуществующего файла или директории
2. Некорректные данные в обрабатываемом файле
3. Некорректный ввод поля «Имя» / «Фамилия» / «Адрес» / «Статус-Личный» / «Статус-Официальный»:
    3.1. Пустой ввод
    3.2. Недопустимые символы (*поведение программы не определено*)
    3.3. Превышение допустимого количества символов *(поведение программы не определено)*
4. Некорректный ввод поля «Телефон»:
    4.1. Пустой ввод
    4.2. Недопустимые символы
    4.3. Превышение допустимого количества символов *(поведение программы не определено)*
5. Некорректный ввод номера опции:

5.1. Пустой ввод

5.2. Недопустимые символы

5.3. Недопустимый номер

6. Некорректный ввод количества записей:

6.1. Недопустимые символы *(поведение программы не определено)*

6.2. Недопустимое число

7. Некорректный ввод поля «Статус-Личный»:

7.1. Одновременный ввод данных вместе с некорректными *(поведение программы не определено)*

7.2. Ввод несуществующей даты *(поведение программы не определено)*

Описание внутренних структур данных

В ходе работы были составлены две структуры данных:

1. `struct abonent_info` (хранение информации об абоненте):
   1.1. `surname` («Фамилия») — массив из `LEN_SURNAME` элементов типа `char`
   1.2. `name` («Имя») — массив из `LEN_NAME` элементов типа `char`
   1.3. `phone` («Телефон») — число типа `uint64_t`
   1.4. `address` («Адрес») — массив из `LEN_ADDRESS` элементов типа `char`
   1.5. `status_code` («КодСтатуса») — число типа `int`
   1.6. `union status` (хранение информации о статусе)
      1.6.1. `struct official_status` («Статус-Официальный»)
         1.6.1.1. `year` — число типа `uint16_t`
         1.6.1.2. `month` — число типа `uint8_t`
         1.6.1.3. `day` — число типа `uint8_t`
      1.6.2. `struct personal_status` («Статус-Личный»)
         1.6.2.1. `office` («Должность») — массив из `LEN_OFFICE` элементов типа `char`
         1.6.2.2. `company` («Статус-Компания») — массив из `LEN_COMPANY` элементов типа `char`

```c
#define LEN_SURNAME 20 + 2
#define LEN_NAME 20 + 2
#define LEN_ADDRESS 35 + 2
#define LEN_OFFICE 30 + 2
#define LEN_COMPANY 30 + 2

/* Личный статус */
struct personal_status
{
   uint16_t year;
   uint8_t day;
   uint8_t month;
};

/* Служебный статус */
struct official_status
{
   char office[LEN_OFFICE];
   char company[LEN_COMPANY];
};

/* Статус абонента */
typedef union status
{
   struct official_status official;
   struct personal_status personal;
} status_t;

/* Информация об абоненте */
typedef struct abonent_info
{
   char surname[LEN_SURNAME]; // фамилия
   char name[LEN_NAME]; // имя
   uint64_t phone; // номер телефона
   char address[LEN_ADDRESS]; // адрес
   int status_code; // код статуса
   status_t status; // статус
} abonent_t;
```

Листинг 1: Описание структуры abonent_info

2. `struct key_info` (хранение информации о ключе абонента)

   2.1. `table_ind` — число типа `int`

   2.2. `key` — массив из `LEN_SURNAME` элементов типа `char`

```c
typedef struct key_info
{
   int table_ind;
   char key[LEN_SURNAME];
} key_row_t;
```

Листинг 2: Описание структуры key_info

Итоговый размер структур:

1. `struct abonent_info`: 168 байт
2. `struct key_info`: 28 байт

Алгоритм

1. Сравнение различных методов сортировок

В ходе работы будут сравниваться два метода сортировки (обычный «пузырек» и qsort из stdlib.h) с использованием двух способов сортировки: сортировки исходной таблицы с абонентами и сортировки таблицы ключей. Сортировка будет производиться по полю «Фамилия».

Рассмотрим сортировку таблиц на примере простой сортировки (пузырьком).

- Сортировка исходной таблицы будет осуществляться с помощью сравнения требуемых полей и перестановки структур целиком в случае, если поле второй структуры «меньше», чем поле первой
- Таблица ключей (состоящая из структур, первое поле которых — исходный индекс в таблице с абонентами, второе — ключ-фамилия абонента, хранящегося под соответствующим индексом) будет сортироваться посредством сравнения ключей между собой и перестановки индексов исходной таблицы

2. Поиск ближайшего дня рождения:

В ходе работы будет использоваться поиск обычным перебором с анализом разницы между датой рождения очередного абонента и исходной датой.

В реализации решения задачи были использованы следующие основные функции:

Название модуля	Заголовок основной функции
commands.c	`void command_file_fill_tables(void);`
	`void command_add_abon_by_user(void);`
	`void command_show_abon_table(void);`
	`void command_show_key_table(void);`
	`void command_del_abon_by_value(void);`
	`void command_sort_key_table(void);`
	`void command_sort_abon_table(void);`
	`void command_keysort_abon_table(void);`
	`void command_show_diff_sorts(void);`
	`void command_show_curr_week(void);`
	`void command_show_menu(void);`
	`void command_clear_tables(void);`
	`void command_show_rules(void);`
table.c	`void add_to_table(abonent_t *table, const int size, abonent_t abonent, const int ind);`
	`void add_to_surname_key_table(key_row_t* key_table, const int ind, key_t *new_key, const int table_ind);`
	`void reset_key_table(abonent_t *table, key_row_t *key_table, const int size);`
	`void remove_from_table(abonent_t *table, int *const size, const int ind);`
	`int fget_table(FILE *const f, abonent_t *table, key_row_t *key_table, const int size);`
	`void fput_table(FILE *const f, abonent_t *table, const int size, const int code);`
	`void fput_table_by_key_table(FILE *const f, abonent_t *table, key_row_t *key_table, const int size, const int code);`
	`void fput_key_table(FILE *const f, key_row_t *key_table, const int size, const int code);`
	`void mysort_key_table(key_row_t *key_table, const int size);`
	`void mysort_abon_table(abonent_t *table, const int size);`
	`int is_inaweek_date(struct personal_status original, struct personal_status data);`

Таблица 3. Модули и основные функции, используемые в алгоритме

Функция	Описание
`void command_file_fill_tables(void)`	Заполнение матрицы с абонентами и матрицы с ключами
`void command_show_abon_table(void)`	Вывод таблицы с абонентами на экран
`void command_show_key_table(void)`	Вывод таблицы ключей на экран
`void command_add_abon_by_user(void)`	Добавление информации о новом абоненте в таблицы с клавиатуры
`void command_del_abon_by_value(void)`	Удаление абонента с фамилией, совпадающей с введенной с клавиатуры
`void command_sort_key_table(void)`	Сортировка таблицы ключей методом пузырька
`void command_sort_abon_table(void)`	Сортировка таблицы абонентов методом пузырька
`void command_keysort_abon_table(void)`	Сортировка таблицы ключей методом пузырька и вывод с ее использованием исходной таблицы на экран
`void command_show_diff_sorts(void)`	Сбор и аналих данных о времени сортировки таблиц данных с введенным количеством записей из указанного файла методами пузырька и qsort (stdlib.h)
`void command_show_curr_week(void)`	Вывод абонентов, день рождения которых — в ближайшую неделю от введенной даты (без учета смены года)
`void command_show_menu(void)`	Вывод меню на экран
`void command_clear_tables(void)`	Очищение таблиц
`void command_show_rules(void)`	Вывод правил ввода на экран
`void add_to_table(abonent_t *table, const int size, abonent_t abonent, const int ind)`	Добавить абонента abonent на место под индексом ind в таблицу абонентов table размера size
`void`	Добавить ключ new_key с

`add_to_surname_key_table(key_row_t* key_table, const int ind, key_t *new_key, const int table_ind)`	исходным индексом table_ind на место под индексом ind в таблицу ключей key_table
`void reset_key_table(abonent_t *table, key_row_t *key_table, const int size)`	Обновить таблицу ключей key_table в соответствии с таблицей абонентов table размера size
`void remove_from_table(abonent_t *table, int *const size, const int ind)`	Удалить абонента с позиции ind в таблице table размера size
`int fget_table(FILE *const f, abonent_t *table, key_row_t *key_table, const int size)`	Заполнить таблицы table и key_table размера size записями из файла f Код возврата: ERR_DATA / EXIT_SUCCESS
`void fput_table(FILE *const f, abonent_t *table, const int size, const int code)`	Вывести таблицу абонентов table размера size в файл f в соответствии с кодом вывода (VERBOSE / QUIET)
`void fput_table_by_key_table(FILE *const f, abonent_t *table, key_row_t *key_table, const int size, const int code)`	Вывести таблицу абонентов table размера size с помощью таблицы ключей key_table в файл f в соответствии с кодом вывода (VERBOSE / QUIET)
`void fput_key_table(FILE *const f, key_row_t *key_table, const int size, const int code)`	Вывести таблицу ключей key_table размера size в файл f в соответствии с кодом вывода (VERBOSE / QUIET)
`void mysort_key_table(key_row_t *key_table, const int size)`	Отсортировать таблицу ключей key_table размера size методом пузырька
`void mysort_abon_table(abonent_t *table, const int size)`	Отсортировать таблицу абонентов table размера size методом пузырька
`int is_inaweek_date(struct personal_status original, struct personal_status data)`	Определить, опережает ли дата data пользовательскую дату original на 7 дней без учета лет Возвращает: Признак (1 / 0)

Таблица 4. Описание основных функций

Тестирование

- Позитивные тесты:

Перед каждым тестом таблица заполняется из файла in.txt на 40 записей.

№	Описание	Входные данные	Сообщение / состояние
КОМАНДА 4			
1	Добавление нового в конец с личным статусом	Test1 Test1 14000000000 TestAddress 1 10 06 2003	Новый абонент был успешно добавлен в конец таблицы / записан под 40 индексом
2	Добавление нового в конец с официальным статусом	Test2 Test2 14999999999 TestAddress 0 TestCompanny TestJob	Новый абонент был успешно добавлен в конец таблицы / записан под 40 индексомё
КОМАНДА 5			
3	Нет абонентов для удаления	Test	Записи с фамилией Test были успешно удалены (0) / кол-во записей: 40
4	Один абонент для удаления	Yarwood	Записи с фамилией Yarwood были успешно удалены (1) / кол-во записей: 39
5	Несколько абонентов для удаления	Ross	Записи с фамилией Ross были успешно удалены (2) / кол-во записей: 38
КОМАНДА 6			
6	Сортировка в лексикографическом порядке по фамилиям	-	Таблица*, отсортированная / Исходная таблица отсортирована
КОМАНДА 7			
7	Сортировка ключей в	-	Таблица**,

	лексикографическом порядке по фамилиям		отсортированная / Исходная таблица неизменна, ключей — отсортирована
КОМАНДА 8			
8	Сортировка в лексикографическом порядке по фамилиям	-	Таблица*, отсортированная / Исходная таблица нетронута, ключей — отсортирована
КОМАНДА 10			
9	Несколько дат подходят	10 07 2001	Записи с индексами 7, 20
10	Ни одна дата не подходит	31 12 2001	Пустая таблица

- Негативные тесты:

№	Описание	Входные данные	Сообщение
1	Некорректный номер команды: 1) пустой ввод 2) буквы 3) выходящий за границы	1) - 2) abc 3) 64	1) Ошибка: Пустой ввод 2) Ошибка: Невверный формат данных 3) Ошибка: Невверный формат данных
КОМАНДА 1			
2	Некорректное имя файла: 1) пустой ввод 2) файла не существует 3) данные в файле некорректны	1) - 2) test 3) a.txt	1) Ошибка: Пустой ввод 2) Ошибка: Невозможно открыть файл 3) Ошибка: Невверный формат данных (после ввода кол-ва записей)
3	Некорректное количество записей: 1) пустой ввод 2) выходящее за нижнюю границу 3) выходящее за верхнюю границу	1) - 2) 1 3) 7000	1) Ошибка: Пустой ввод 2) Ошибка: Невверный формат данных 3) Ошибка: Невверный формат данных

КОМАНДА 2			
4	Вывод при пустой таблице	-	Ошибка: Таблица пуста
КОМАНДА 3			
5	Вывод при пустой таблице	-	Ошибка: Таблица пуста
КОМАНДА 4			
6	Некорректный ввод: 1) пустой ввод, фамилия 2) пустой ввод, имя 3) номер выходит за нижнюю границу 4) номер выходит за верхнюю границу 5) пустой ввод, адрес 6) код статуса: не 0 или 1 7) статус-личный: числа выходящие за границы 8) статус-официальный 1) пустой ввод, компания 2) пустой ввод, должность 9) переполнение матрицы	1) - 2) test, - 3) test, test, 1 4) test, test, 2000000000 5) test, test, 14000000000, - 6) 1) test, test, 14000000000, abc, a 2) test, test, 14000000000, abc, 3 7) соответствующий ввод 8) 1) 2) соответствующий ввод 9) ввод	1) Ошибка: Пустой ввод 2) Ошибка: Пустой ввод 3) Ошибка: Невверный формат данных 4) Ошибка: Невверный формат данных 5) Ошибка: Пустой ввод 6) Ошибка: Невверный формат данных 7) Ошибка: Невверный формат данных 8) … 1) Ошибка: Пустой ввод 2) Ошибка: Пустой ввод 9) Ошибка: Достигнуто макс. кол-во записей
КОМАНДА 5			
7	Фамилия: пустой ввод	-	Ошибка: Пустой ввод
КОМАНДА 9			
8	Некорректное имя файла: 4) пустой ввод 5) файла не существует 6) данные в файле некорректны	4) - 5) test 6) a.txt	1) Ошибка: Пустой ввод 2) Ошибка: Невозможно открыть файл 3) Ошибка: Невверный формат данных (после ввода кол-ва записей)

9	Некорректное количество записей: 4) пустой ввод 5) выходящее за нижнюю границу 6) выходящее за верхнюю границу	4) - 5) 1 6) 7000	1) Ошибка: Пустой ввод 2) Ошибка: Невверный формат данных 3) Ошибка: Невверный формат данных
КОМАНДА 10			
10	Матрица пуста	-	Ошибка: Таблица пуста
11	Числа выходят за границу	Соответствующий ввод	Ошибка: Неверный формат данных

Оценка эффективности

Эффективность оценивалась в двух плоскостях: сравнение быстрого и медленного методов сортировки, сортировка исходной таблицы и таблицы ключей. Результаты тестов приведены ниже (количество итераций для каждого варианта: 200; в выигрыше 100% -- сортировка mysort-исходная/ключи или qsort-исходная). mysort — метод пузырька (простой).

Метод	Кол-во записей	Размер таблицы, Б	Размер таблицы ключей, Б (+ % от исходной)	Среднее время сортировки исходной, мкс	Среднее время сортировки ключей (+ время доступа), мкс
mysort	40	6720	1120 (16,7%)	51	18 (0) Выигрыш (mysort исх.): **77,778%**
	200	33600	5600 (16,7%)	2788	908 (1) Выигрыш (mysort исх.): **67,432%**
	500	84000	14000 (16,7%)	17350	5807 (2) Выигрыш (mysort исх.): **66,530%**
	1000	168000	28000 (16,7%)	50449	24036 (3) Выигрыш (mysort исх.): **52,356%**
qsort	40	6720	1120 (16,7%)	3 Выигрыш (mysort): **94,118%**	4 (0) Выигрыш (qsort исх.): **-33,333%** Выигрыш (mysort): **64,706%**
	200	33600	5600 (16,7%)	49 Выигрыш (mysort): **98,242%**	57 (0) Выигрыш (qsort исх.): **-16,327%** Выигрыш (mysort): **93,722**
	500	84000	14000 (16,7%)	180 Выигрыш (mysort): **98,963%**	179 Выигрыш (qsort исх.): **0,556%** Выигрыш (mysort): **96,918%**

	1000	168000	28000 (16,7%)	408 Выигрыш (mysort): **99,191%**	413 (3) Выигрыш (qsort исх.): **-1,225%** Выигрыш (mysort): **98,282%**

Исходя из результатов можно сделать следующие выводы:

- Несмотря на дополнительные затраты памяти при использовании таблицы ключей (16,7% от исходной), выполнение сортировки по строковому полю с ее помощью более эффективно как в случае с быстрым алгоритмом, так и в случае с медленным
- В силу того, что qsort из stdlib.h использует досттаочно быстрый алгоритм, а измерения выполняются в микросекундах, а также в силу особенностей состояния процессора возможно появление достаточно больших погрешностей, которые искажают измерения и эффективность сортировки таблицы ключей по сравнению с сортировкой исходной таблицы выходит отрицательной
- Быстрый алгоритм сортировки выигрывает по времени в сравнении с медленным во всех рассмотренных случаях

Контрольные вопросы

1 Как выделяется память под вариантную часть записи?

Размер памяти равен памяти, занимаемой максимальным по длине полем. Эта память является общей для всех полей вариантной части записи.

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

Будет сообщение о том, что данные невозможно прочитать.

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

Программист.

4 Что представляет собой таблица ключей, зачем она нужна?

При больших размерах таблиц поиск данных и их сортировка может потребовать больших затрат времени. В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

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

Если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует. Однако, для размещения таблицы ключей требуется дополнительная память. Кроме того, 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.

6 Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Сортировка эффективнее выполняется с помощью таблицы ключей. Любая сортировка подразумевает замену, перестановку строк. Это затратно по памяти. Таблицы ключей позволяет свести эти операции к заменам и перестановка ключей.

Выводы

В ходе работы было выяснено, что хранение длинных чисел в виде определенных структур данных и использование длинной арифметики позволяют работать с длинными числами быстро и эффективно, однако алгоритмы вычислений становятся сложнее.