**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»**

КАФЕДРА **«Программное обеспечение ЭВМ и информационные технологии»**

# ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

## Записи с вариантами. Обработка таблиц

**Студент** _____ Палладий Е.И.

**Группа** _____ ИУ7-31Б

Название предприятия: НУК ИУ МГТУ им. Н. Э. Баумана

| | |
|---|---|
| Студент | _____ Палладий Е.И. |
| Преподаватель | _____ Барышникова М.Ю. |

*2024 г.*

# СОДЕРЖАНИЕ

# 1 Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки: пузырёк и пузырёк с флагом, сортируемое поле: **столица**, Сортировка производится двумя способами: а) Используя таблицу, б) Используя массив ключей. Возможность добавления и удаления записей в ручном режиме, просмотр таблицы, просмотр таблицы в порядке расположения таблицы ключей обязательна. Осуществить поиск информации по варианту. Произвести исследование эффективности алгоритмов сортировки при использовании структур и ключей

# 2 Техническое задание

## 2.1 Исходные данные

Список стран, максимум 11000 записей. В файле каждая страна должна быть разделена переносом строки, поля разделены пробелом. Информация о стране: название, столица, материк, необходимость наличия визы, время полета до страны, минимальная стоимость отдыха, основной вид туризма:

1. Экскурсионный:
    1.1. Количество объектов
    1.2. Основной вид объектов (природа, искусство, история)
2. Пляжный:
    2.1. Основной сезон
    2.2. Температура воздуха и воды
3. Спортивный:
    3.1. Вид спорта (горные лыжи, серфинг, восхождения)

**Меню программы:**

1. Вывести информацию о программе
2. Загрузить список стран из файла
3. Сохранить список стран в файл
4. Добавить страну в конец списка
5. Удалить страну из списка по названию
6. Вывести список стран
7. Вывести список ключей
8. Вывести список стран по по списку ключей
9. Вывести список стран на выбранном материке, где можно заняться указанным видом спорта, со стоимостью отдыха меньше указанной
10. Отсортировать список ключей по столице
11. Отсортировать список фильмов по столице
12. Произвести и вывести исследование

13. Выход

### 2.2 Описание задачи, реализуемой программой

Сохранение, добавление, удаление, вывод, сортировка стран и/или ключей как в файл, так и в консоль. Проведение замерного эксперимента и исследования об эффективности использования массива структур и массива ключей

### 2.3 Способ обращения к программе

Запускается через терминал: `./app.exe`. Затем необходимо выбрать одну из предложенных опций в меню. При выборе опции №12 (Произвести и вывести исследование) для построение графиков зависимости времени выполнения сортировки от количества элементов необходимо из корневой директории прописать команду: `gnuplot research/src/*.sh`. После этого графики будут расположены в директории `research/plots`

### 2.4 Описание аварийных случаев

1. **Аварийные случаи при записи и чтении фильма**
   1.1. Ошибки при вводе любого строкового поля
      a) Пустой ввод
      b) Переполнение буфера
   1.2. Ошибки при вводе поля: Информации о необходимости наличия визы
      a) Пустой ввод
      b) Символ
      c) Число, отличное от 0 или 1
   1.3. Ошибки при вводе полей: Время полета до страны, Минимальная стоимость отдыха, Количество объектов
      a) Пустой ввод
      b) Символ
      c) Отрицательное число
   1.4. Ошибки при вводе полей: Основной вид объектов, Вид спорта:
      a) Пустой ввод

b) Символ

c) Число не из диапазона [1; 3];

1.5. Ошибки при вводе полей: Температура воздуха, Температура воды:

a) Пустой ввод

b) Символ

c) Число, выходящее за границы типа `short`

2. **Аварийный случай при добавлении страны в конец списка**

2.1. Список уже содержит максимальное количество элементов

3. **Аварийный случай при удалении страны из списка**

3.1. Страна отсутствует в списке

4. **Аварийный случай при работе с файлом**

4.1. Файл отсутствует

4.2. Нет прав доступа для работы с файлом

# 3 Описание структуры данных

Листинг 1. Структура для хранения фильма с вариативным полем

```
typedef struct
{
    char name[MAX_COUNTRY_NAME_LENGTH];
    char capital[MAX_CAPITAL_LENGTH];        // Название столицы
    char mainland[MAX_MAINLAND_LENGTH];      // Название материка
    bool visa;                               // Потребность в визу

    uint32_t flying_time;                    // Время полета в минутах
    uint32_t min_vacation_price;             // Минимальная цена отдыха

    type_of_tourism tourism;                 // Перечисление туризма
    type_t type;                             // Объекдинение структур туризма
} country_t;
```

**Объяснение полей:**

| **Поле** | **Описание** |
|---|---|
| name | Название страны |
| capital | Название столицы |
| mainland | Название материка |
| visa | потребность в визе |
| lfying_time | Время полета |
| min_vacation_price | Минимальная стоимость отдыха |
| tourism | Вид туризма |
| type | Union туризма |

Листинг 2. Структура для хранения ключей

```
    typedef struct
    {
        size_t ind;
        char capital[MAX_CAPITAL_LENGTH];
    } key_t;
```

---

**Объяснение полей:**

| Поле | Описание |
|---|---|
| ind | Индекс в исходной таблице |
| capital | Название столицы |

Листинг 3. Объединение видов туризма

```
typedef union
{
    sightseeing_t sightseeing;              // Экскурсионный
    beach_t beach;                          // Пляжный
    sport_t sport;                          // Спортивный
} type_t;
```

---

**Объяснение полей:**

| Поле | Описание |
|---|---|
| sightseeing | Структура с экскурсионным видом туризма |
| beach | Структура с пляжным видом туризма |
| sport | Структура со спортивным видом туризма |

Листинг 4. Структуры туризма

```
    // Экскурсионный вид отдыха
    typedef struct
    {
        uint32_t objects_amount;            // Количество объектов
```

```
5           type_of_objects objects_type;       // Вид объекта
6       } sightseeing_t;
7
8       // Пляжный вид отдыха
9       typedef struct
10      {
11          char season[MAX_SEASON_LENGTH];       // Сезон
12          short water_temperature;              // Темеература воды
13          short air_temperature;                // Температура воздуха
14      } beach_t;
15
16      // Спорттивный вид отдыха
17      typedef struct
18      {
19          type_of_sport sport_type;
20      } sport_t;
```

**Объяснение полей:**

| **Поле** | **Описание** |
|---|---|
| objects_amount | Количество объектов |
| objects_type | Вид объекта |
| season | Сезон |
| water_temperature | Температура воды |
| air_temperature | Температура воздуха |
| sport_type | Вид спорта |

Листинг 5. Возможные поля туризма

```
1       //Вид туризма
2       typedef enum
3       {
4           SIGHTSEEING = 1,        // Экскурсионный
5           BEACH,                  // Пляжный
6           SPORT                   // Спортивный
7       } type_of_tourism;
8
9       //Вид объектов
```

```
10      typedef enum
11      {
12          NATURE = 1,             // Природа
13          ART,                    // Искусство
14          HISTORY                 // История
15      } type_of_objects;
16
17      //Вид спорта
18      typedef enum
19      {
20          MOUNTAIN_SKIING = 1,    // Горные лыжи
21          SURFING,                // Сёрфинг
22          CLIMBING                //Скалолазание
23      } type_of_sport;
```

## 4 Описание алгоритма

1. Ввод пункта меню
2. Выбран пункт 1:
    - Информация о программе выводится на экран
3. Выбран пункт 2:
    - Вводится название файла, из которого будет происходить запись
    - Список стран из файла загружается в массив структур
4. Выбран пункт 3:
    - Вводится название файла, в который будет происходить запись
    - Список стран из массива структур загружается в файл
5. Выбран пункт 4:
    - Вводится страна из консоли
    - Страна добавляется в массив, если там есть место
6. Выбран пункт 5:
    - Вводится страна из консоли
    - Если страна есть в массиве - она удаляется
    - Длина массива уменьшается, если страна была удалена
7. Выбран пункт 6-9:
    - Список стран/ключей выводится в консоль
8. Выбран пункт 10-11:
    - Список фильмов или список ключей сортируется по столице
9. Выбран пункт 12:
    - Производится сортировка фильмов при разном количестве записей
    - Выводится статистика сортировки
    - При необходимости можно построить графики зависимости времени от количества фильмов

### 4.1 Основные функции программы

1. `int read_country(FILE *file_in, country_t *country);`
    - Читает запись страны из файла
    - **file_in**: Входной файл
    - ***country**: Указатель на страну для записи
    - Возвращает код ошибки
2. `int add_country_top(country_t countries[], const country_t country, size_t *length);`
    - Добавляет запись страны в конец массива структур
    - **countries[]**: Массив структур стран
    - **country**: Страна для добавления
    - Возвращает код ошибки
3. `void delete_in_array(country_t *countries, size_t *length, const size_t pos);`
    - Удаляет запись страны из массива структур
    - ***countries**: Массив структур стран
    - **length**: Количество элементов массива
    - **pos**: Позиция для удаления
    - Возвращает код ошибки
4. `bool find_in_array(country_t *countries, const size_t length, char *field, size_t *pos);`
    - Находит позицию страны в массиве структур по указанному полю
    - ***countries**: Массив структур стран
    - **length**: Количество элементов массива
    - ***field**: Поле для сравнения
    - **pos**: Позиция найденной страны в массиве
    - Возвращает код ошибки

5. `void bubble_sort_countries(country_t *countries, const int len)`
    - Сортирует массив структур стран
    - ***countries**: Массив структур стран
    - **len**: Количество элементов массива
6. `void bubble_sort_keys(key_t *keys, const int len)`
    - Сортирует массив ключей
    - ***countries**: Массив ключей
    - **len**: Количество элементов массива

# 5 Тестовые данные

**Позитивные тесты**

| Тест | Входные данные | Выходные данные |
|---|---|---|
| Загрузить список стран из файла | data.txt | Данные успешно загружены |
| Сохранить список стран в файл | data.txt | Данные успешно сохранены |
| Добавить страну в конец списка | name capital mainland 1 20 30 1 12 1 | Страна успешно добавлена |
| Вывести список стран | | name capital mainland + 10 20 Sightseeing: 2 - Nature |
| Вывести список ключей | | 0 0 capital |
| Вывести список стран по списку ключей | | name capital mainland + 10 20 Sightseeing: 2 - Nature |
| Вывести список стран по заданию | mainland 2 30 | name capital mainland + 10 20 Sport: 2 |
| Удалить страну из списка по названию | name | Страна успешно удалена |
| Отсортировать список ключей по столице | | Список ключей успешно отсортирован |
| Отсортировать список фильмов по столице | | Список фильмов успешно отсортирован |

| | | |
|---|---|---|
| Произвести и вывести исследование | | Исследование проведено успешно |
| Выход | | Пока! |

**Негативные тесты**

| Тест | Входные данные | Выходные данные |
|---|---|---|
| Пустой ввод строки | name capital _ | EMPTY STRING ERROR! |
| Переполнение строки | name capital qwertyuiop qwertyui | BUF OVERFLOW ERROR! |
| Символ вместо числа | name capital mainland w 20 30 1 12 1 | INPUT/OTPUT ERROR! |
| Число выходит за диапазон short | name capital mainland й 20 12345678 1 12 1 | RANGE ERROR! |
| Файл не существует (При загрузке данных) | qwerty.txt | ERR WITH FILE! |
| Переполнение массива | ... | RANGE ERROR! |
| Страна для удаления отсутствует | mainland | ERR COUNTRY NOT FOUND |
| Список стран по заданию пустой | mainland 1 200 | ERR COUNTRY NOT FOUND |

# 6 Исследование

Исследование проводилось на основе таблиц, содержащих от 1000 до 11 000 записей с шагом 1000. Количество итераций сортировок: 10. Все данные были сгенерированы случайным образом. В ходе эксперимента применялись два алгоритма сортировки: классическая сортировка пузырьком и её улучшенная версия — сортировка пузырьком с флагом. Замеры производительности выполнялись как для сортировки всей таблицы целиком, так и для сортировки ключей.

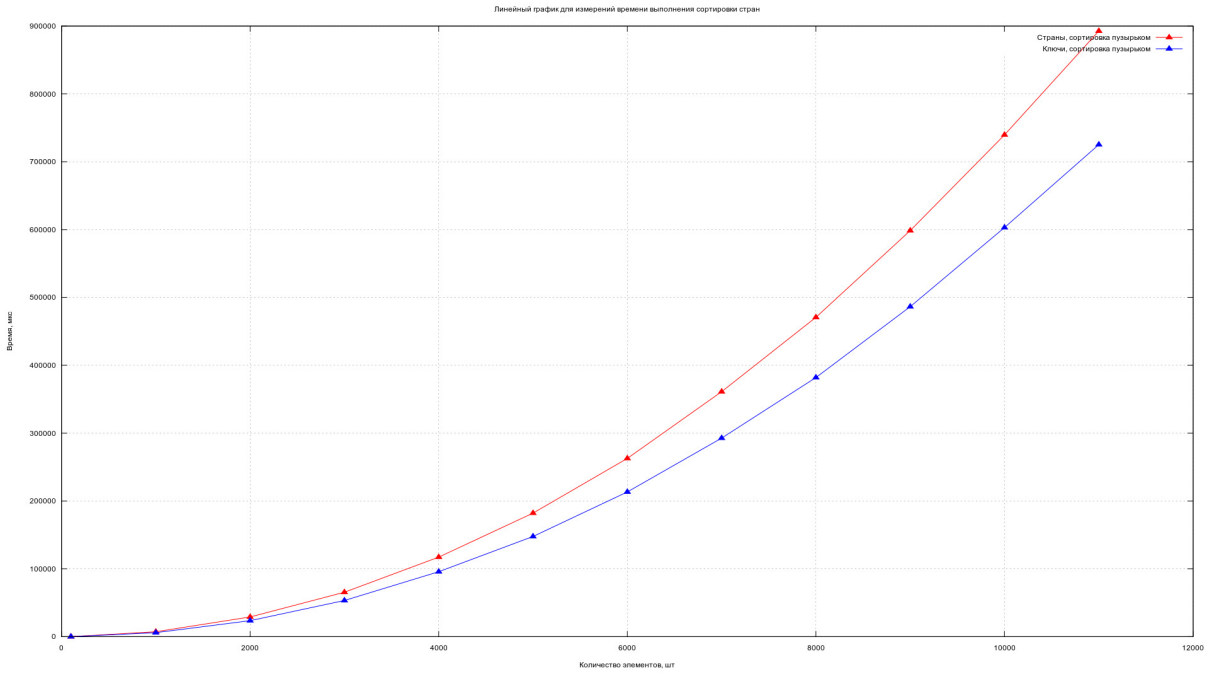


Рисунок 1. Зависимость времени между сортировкой стран и ключей при сортировке пузырьком

| Кол-во элементов | Время, мкс | Память, байт |
|---|---|---|
| 1000 | 15163 | 80 000 |
| 4000 | 117 279 | 320 000 |
| 8000 | 470 726 | 640 000 |
| 11 000 | 892 718 | 880 000 |

Таблица 4. Сортировка таблицы пузырьком

| Кол-во элементов | Время, мкс | Память, байт |
|---|---|---|
| 1000 | 12377 | 24 000 |
| 4000 | 95 769 | 96 000 |
| 8000 | 381 975 | 192 000 |
| 11 000 | 725 277 | 264 000 |

Таблица 6. Сортировка ключей пузырьком

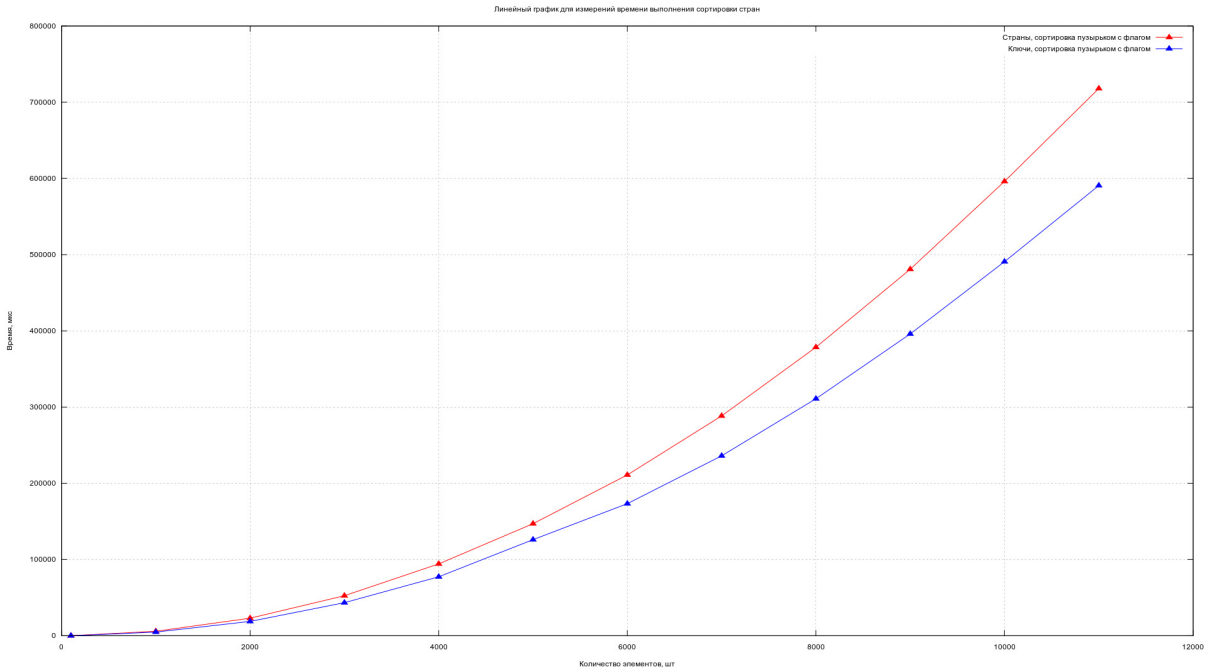


Рисунок 2. Зависимость времени между сортировкой стран и ключей при сортировке пузырьком с флагом

| Кол-во элементов | Время, мкс | Память, байт |
|---|---|---|
| 1000 | 23 028 | 80 000 |
| 4000 | 94 265 | 320 000 |
| 8000 | 378 553 | 640 000 |
| 11 000 | 717 998 | 880 000 |

Таблица 8. Сортировка таблицы пузырьком с флагом

| Кол-во элементов | Время, мкс | Память, байт |
|---|---|---|
| 1000 | 10018 | 24 000 |
| 4000 | 77 359 | 96 000 |
| 8000 | 310 965 | 192 000 |
| 11 000 | 590 672 | 264 000 |

Таблица 10. Сортировка ключей пузырьком с флагом

## Дополнительные графики

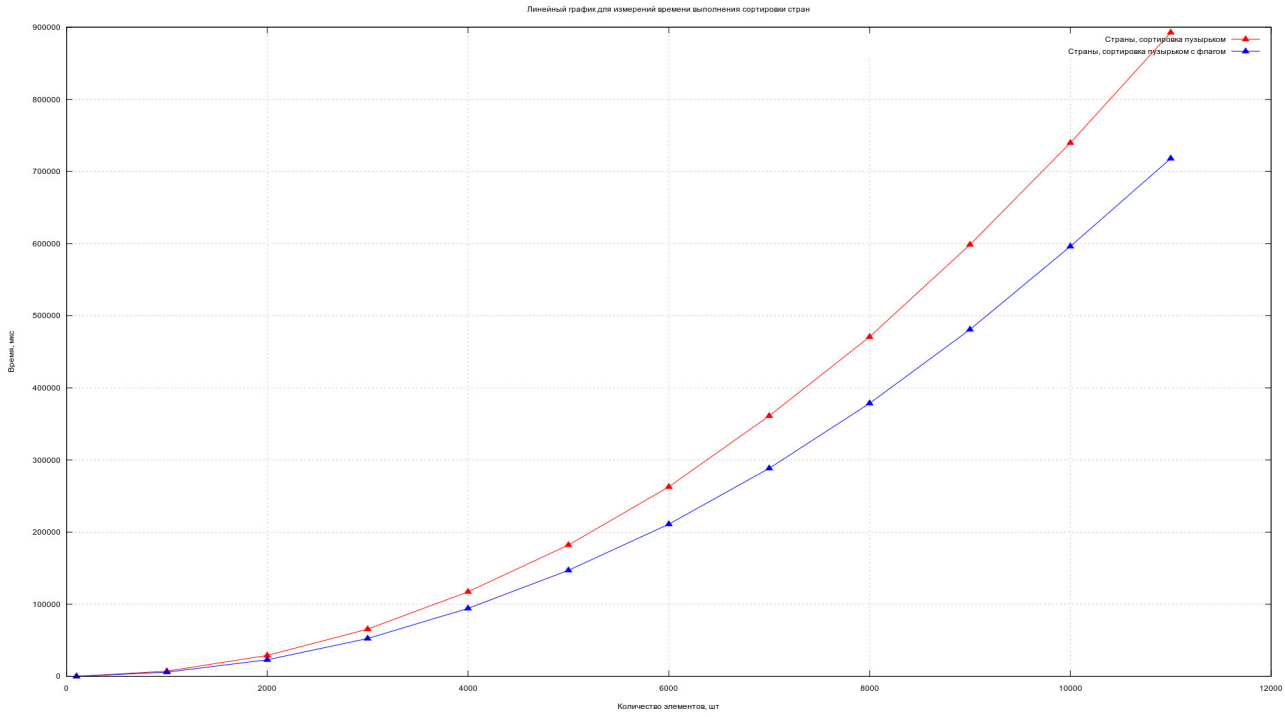


Рисунок 3. Зависимость времени между разным способом сортировки стран

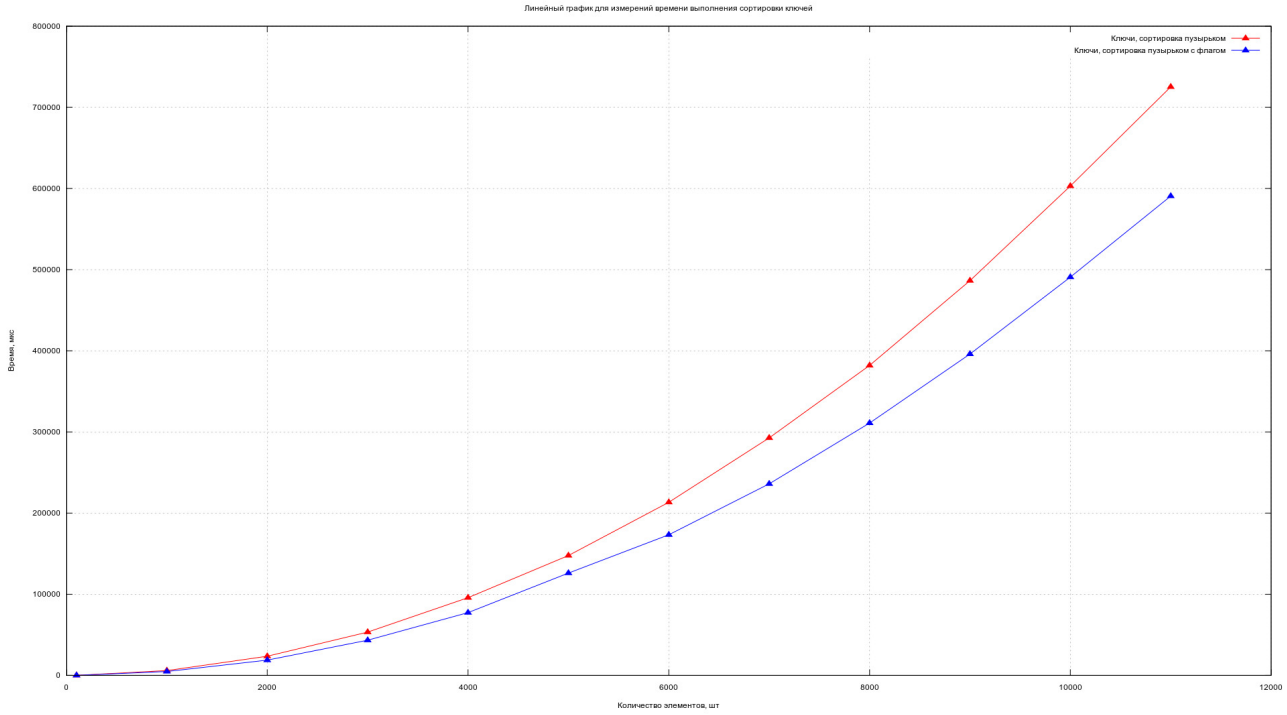


Рисунок 4. Зависимость времени между разным способом сортировки ключей

### 6.1 Выводы исследования

Как видно из полученных данных, использование ключей сокращает потребление памяти примерно на 70%, а улучшение производительности по времени при сортировке ключей составляет около 20%.

## 7 Ответы на вопросы

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

Для вариантной части записи используется объединение. Где каждое поле объединения - структура возможной записи. Объём такого объединения равен максимальному объёму одной из его структур. Такой формат обеспечивает минимизацию использования памяти.

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

Если такой ввод не вернет ошибку в программе, то дальнейший исход событий не предсказуем. Могут произойти разные ошибки, в том числе аварийное завершение программы.

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

Только программист. Он должен обеспечить правильную запись вариантной части или вывод ошибки, при неверном вводе от пользователя.

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

Таблица ключей - структура данных, содержащая ключи или идентификаторы к исходной таблице. Таблица ключей занимает меньше памяти, поэтому выгодна в при частом использовании исходной таблицы.

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

Если при работе с исходной таблицей часто происходит обмен, сравнение, вставка, удаление объектов, то использование таблицы ключей поможет сократить время работы этих подпрограмм Если происходит обращение к

большому количеству разных полей таблицы, то обрабатывать данные в таблице выгоднее

**6. Какие способы сортировки предпочтительнее для обработки таблиц и почему?**

Если данные в таблице почти отсортированы, то сортировка вставками подойдет лучше всего, так как будет осуществлено небольшое количество обменов

В общем случае, очевидно, лучше всего работают сортировки с временной сложностью $O(N \log N)$. Так как быстрее этой константы отсортировать физически невозможно. Однако есть всеми любимая **Сталинская сортировка** Которая, между прочим, работает за $O(N)$, но есть нюанс:)

Пример такой сортировки работающей за $O(N \log N)$: `Heap sort`.

# ЗАКЛЮЧЕНИЕ

Очень много информации хранятся в разных таблицах, многие из них имеют вариантные поля. В таких ситуациях программисту важно оценить, с какими операциями и данными ему придется иметь дело, чтобы подобрать лучшие по эффективности алгоритмы и методы обработки таблиц. Если необходимо сортировать таблицу, то выгодно делать это, используя ключей, такой подход даст существенное улучшение в производительности как по памяти, так и по времени