

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Отчет по выполнению практического задания №10

**Тема:** Применение алгоритмов поиска по ключу

**Дисциплина: «**Структуры и алгоритмы обработки данных**»**

Выполнил студент Антонов А.Д.

Группа ИКБО-01-20

**Москва 2021**

**Содержание**

1. [Задание 1 3](#_bookmark0)
2. [Отчет по заданию 2 10](#_bookmark1)
3. [Отчет по заданию 3 13](#_bookmark2)
4. [Анализ эффективности рассмотренных алгоритмов поиска в файле 17](#_bookmark3)

[ВЫВОДЫ 18](#_bookmark4)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 18](#_bookmark5)

## Задание 1

* 1. **Постановка задачи**

Создать текстовый файл из записей. Заполнить файл данными, используя для поля ключа генератор случайных чисел.

* 1. **Описание подхода к решению Структура записи файла**

Согласно варианту №2 индивидуального задания запись файла представляет собой: счет в банке: номер счета – 7 разрядное число, ФИО, адрес. Сортировка производится по номеру счета.

Размер записи в байтах

Каждая запись представлена строковой переменной и её размер зависит от длины строки, поэтому определить точный размер записи не представляется возможным. Однако можно посчитать средний размер записи, зная количество строки вес одного файла с записями. Так, для 200 записей вес файла составляет около 5400 байт (рис. 1), соответственно вес одной записи около 27 байт.

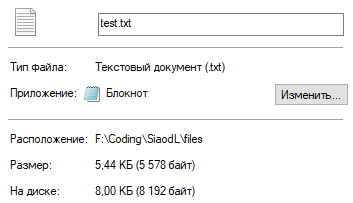
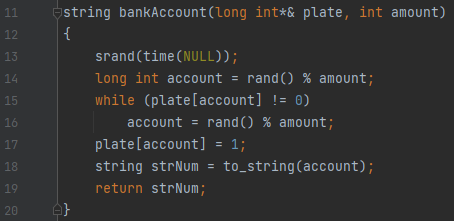
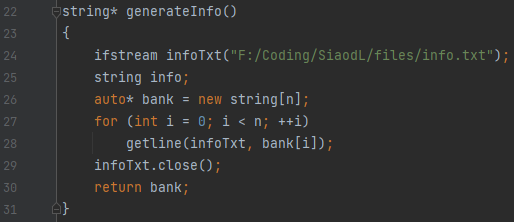


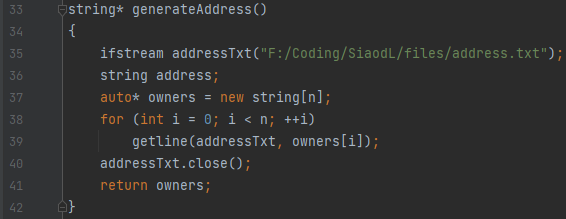
Рис. 1. Свойства файла, содержащего 200 записей

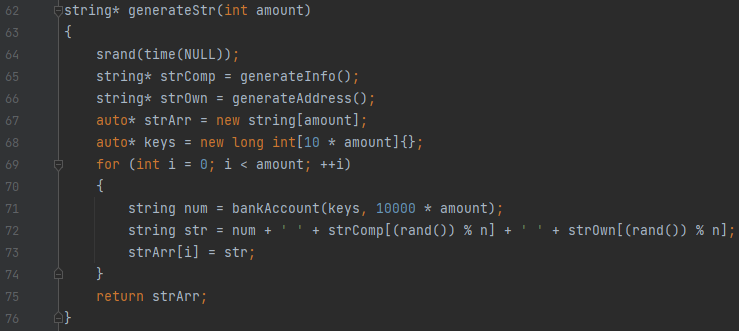
Алгоритмы, реализованные в форме функций

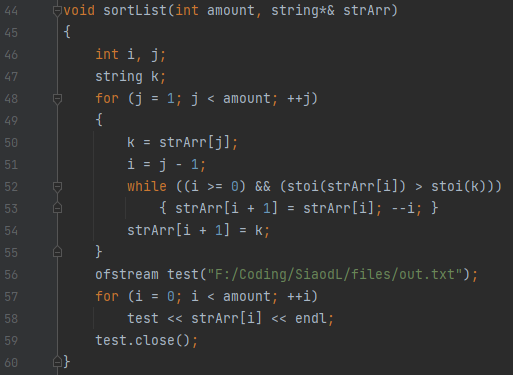
Для генерации текстового файла были созданы следующие функции:

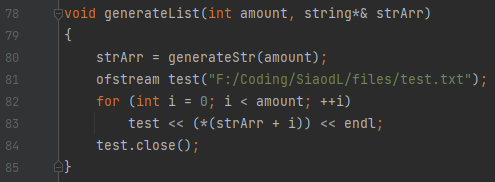
****

****

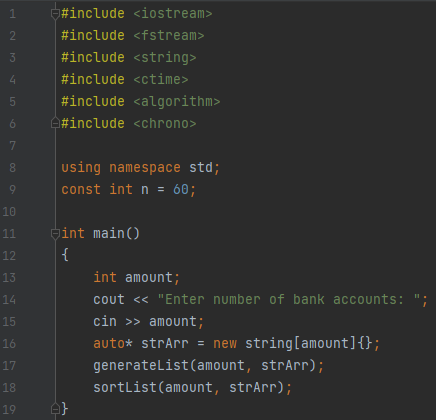
****

****

****

****

* 1. **Код программы**



* 1. **Тестирование программы**

Таблица 2. Тестирование генератора записей

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат (вес файла) |
| 1 | 10 | ~260 байт |  |
| 2 | 100 | ~2600 байт |  |
| 3 | 1000 | ~26000 байт |  |

Как видно из таблицы 2 – все функции работают и генерируют файл необходимого размера.

## Отчет по заданию 2

* 1. **Постановка задачи**

Разработать программу поиска записи по ключу в текстовом файле с применением алгоритма линейного поиска.

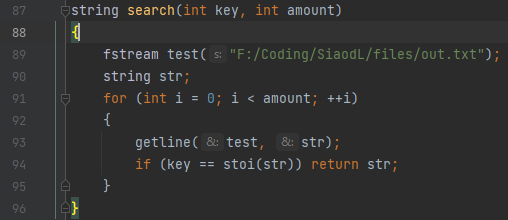
* 1. **Алгоритм**

Таблица 3. Алгоритм линейного поиска на псевдокоде

|  |
| --- |
| *key* – ключ записи, *amount –* количество записей, *str –* конкретная строка (запись) из файла |
| search(key, amount)  {  fstream test ("F:/Coding/SiaodL/files/test.txt");  string str;  for i←0 to amount **do**  getline(test, str);  if (key == stoi(str)) then return str;  **od**  } |

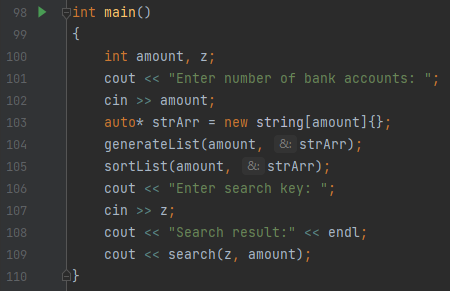
* 1. **Код функции поиска**

Функция линейного поиска открывает файл, в который были записаны данные и после этого производит последовательное сравнение ключа каждой записи с ключом, переданным в функцию.



* 1. **Код программы линейного поиска записи по ключу**

Генерируется файл с записями и после этого вводится ключ, по которому нужно найти запись. Ключ и количество записей передаются в функцию поиска. Найденная запись печатается в консоль.



* 1. **Результат тестирования программы**

Таблица 4. Тестирование функции линейного поиска

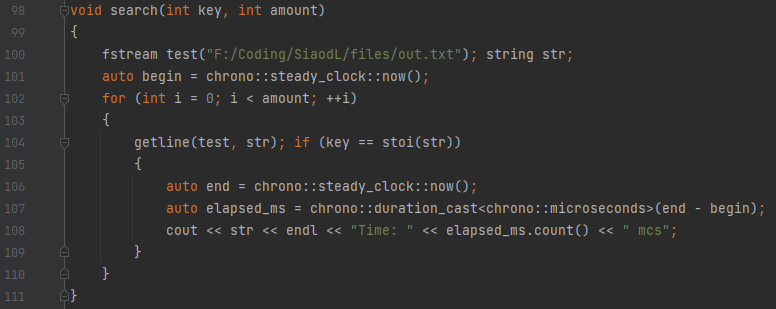
|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат (вес файла) |
| 1 | Кол-во: 10 Ключ: 1234567 | 1234567  Pupkin Vasily Ivanovich  Pushkina, 6 |  |
| 2 | Кол-во: 100  Ключ: 5553535 | 5553535  John Smith Jr.  Park Avenue, 2A |  |
| 3 | Кол-во: 1000  Ключ: 2234456 | 6969696  Buyanov Nikita Geneburn  Tarkov, Labs |  |

Как видно из таблицы 4 – функция линейного поиска работает верно и находит запись согласно введенному ключу.

* 1. **Замеры времени поиска записи по заданному ключу**

Чтобы оценить скорость работы функции линейного поиска произведем тестирование функции с замером времени. Тестирование будет производится на файлах из 100 и 1000 записей. В функцию будут передаваться последовательно ключи 3-х записей – записи, расположенной в начале, в середине и в конце файла.

Код программы:



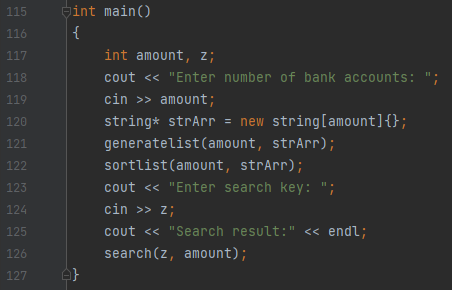


Таблица 5. Замеры времени поиска записи по ключу

|  |  |  |
| --- | --- | --- |
| Расположение записи | 100 записей | 1000 записей |
| Начало | 123 мкс | 1562 мкс |
| Середина | 742 мкс | 6132 мкс |
| Конец | 1659 мкс | 11585 мкс |

Как видно из таблицы 5 время поиска соответствует линейной сложности алгоритма: при увеличении значения в два раза время поиска по файлу увеличивается аналогично в два раза.

## Отчет по заданию 3

* 1. **Постановка задачи**

Разработать функцию бинарного поиска записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.

* 1. **Алгоритм**

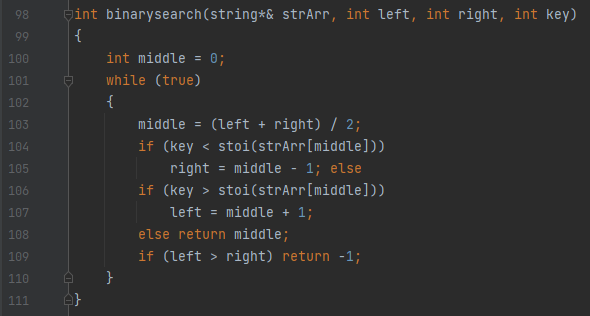
Алгоритм бинарного поиска записи с ключом в файле на псевдокоде представлен в таблице 6.

Таблица 6. Алгоритм бинарного поиска на псевдокоде

|  |
| --- |
| *key* – ключ записи, *left –* индекс левой границы, *strArr –* массив строк, *right –* индекс правой границы |
| binarysearch(strArr, left, right, key)  {  middle←0; while (true) **do**  middle←((left+right)/2);  if (key < stoi(strArr[middle])) **then** right←middle-1;  else if (key > std::stoi(strArr[middle])) **then** left←middle+1; else return middle;  if (left> right) **then** return -1;  **od**  } |

* 1. **Код функции бинарного поиска**

В функцию бинарного поиска передается таблица записей (массив со строками), индексы левой и правой границы, а также ключ, по которому ищется запись.



* 1. **Результат тестирования программы для 100 записей**

Для того чтобы убедиться, что функция бинарного поиска работает верно произведем тестирование функции (табл. 7).

Таблица 7. Тестирование функции бинарного поиска

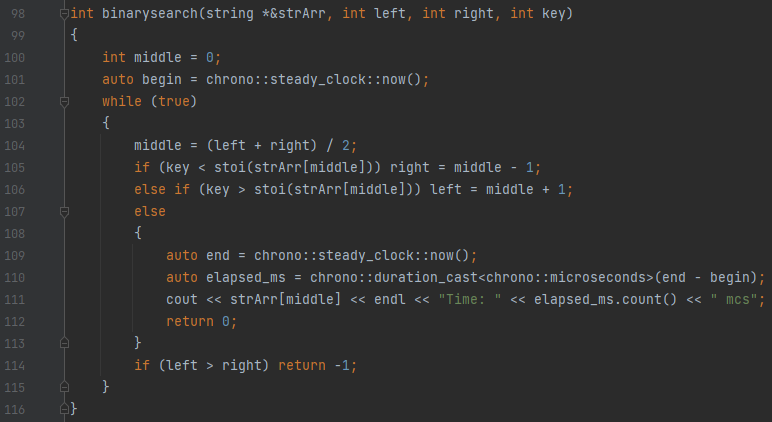
|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат (вес файла) |
| 1 | Кол-во: 10 Ключ: 1234567 | 1234567  Pupkin Vasily Ivanovich  Pushkina, 6 |  |
| 2 | Кол-во: 100 Ключ: 5553535 | 5553535  John Smith Jr.  Park Avenue, 2A |  |
| 3 | Кол-во: 1000  Ключ: 6969696 | 6969696  Buyanov Nikita Geneburn  Tarkov, Labs |  |

Как видно из таблицы 7 – функция бинарного поиска работает верно и находит запись согласно введенному ключу.

* 1. **Таблица с замерами времени бинарного поиска записи**

Чтобы оценить скорость работы функции бинарного поиска произведем тестирование функции с замером времени. Тестирование будет производится на файлах из 100 и 1000 записей. В функцию будут передаваться последовательно ключи 3-х записей – записи, расположенной в начале, в середине и в конце файла.

Код программы:



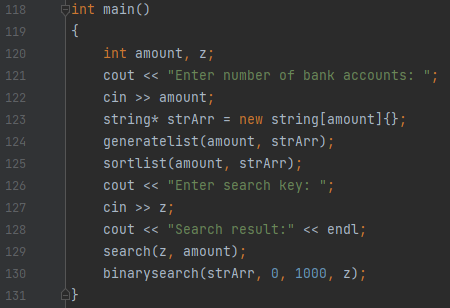
****

Таблица 8. Замеры времени бинарного поиска записи по ключу

|  |  |  |
| --- | --- | --- |
| Расположение записи | Время поиска в файле из 100 записей | Время поиска в файле из 1000 записей |
| Начало | 8 мкс | 7 мкс |
| Середина | 5 мкс | 8 мкс |
| Конец | 11 мкс | 14 мкс |

Как видно из таблицы 8, время бинарного поиска не зависит от количества записей и их расположения линейно.

## Анализ эффективности рассмотренных алгоритмов поиска в файле

Произведем анализ эффективности рассмотренных алгоритмов поиска в файле. Как видно из результатов замера времени для каждого алгоритма (табл. 5, табл. 8) – алгоритм бинарного поиска с использованием дополнительной памяти гораздо быстрее, чем алгоритм линейного поиска. Алгоритм бинарного поиска меньше зависит от длины массива и во многих частных случаях показывает себя быстрее (если элемент расположен в самой середине списка).

Таблица 9. Сравнение результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во записей | Алгоритм поиска | Расположение | | |
| Начало | Середина | Конец |
| 100 | Линейный | 123 мкс | 742 мкс | 1659 мкс |
| Бинарный | 8 мкс | 5 мкс | 11 мкс |
| 1000 | Линейный | 1562 мкс | 6132 мкс | 11585 мкс |
| Бинарный | 7 мкс | 8 мкс | 14 мкс |

# ВЫВОДЫ

В ходе данной практической работы были получены знания и практические навыки по разработке и реализации алгоритмов поиска в таблице. В первом задании был разработан генератор таблицы данных в соответствии с вариантом индивидуального задания, описаны прототипы используемых функций и приведены результаты тестирования алгоритма. Во втором задании был разработан алгоритм линейного поиска, приведены результаты тестирования алгоритма и произведен замер времени для разных случаев. В третьем задании был разработан алгоритм бинарного поиска, приведены результаты тестирования алгоритма и произведен замер времени для разных случаев. На основе замеров времени были проанализированы результаты эффективности изученных алгоритмов (время их выполнения) и сделан вывод, что алгоритм бинарного поиска является наиболее эффективным, из всех представленных в данной работе алгоритмов.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Теоретический материал по структурам и алгоритмам обработки данных
2. Мейерс, С. Эффективный и современный C++[Текст] / Скотт Мейерс. – O’REILLY, 2016. – 306 с.