****

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 №5

**Тема: «РАБОТА С ДАННЫМИ ИЗ ФАЙЛА»**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Кузнецов Л. А.

группа ИКБО-20-23

**Москва 2024**

**СОДЕРЖАНИЕ**

[Задание 1 части 5.1…………....…………………………………………………...2](#Задние1части51)

[Задание 1.а…………………………………………………………………………2](#Задание1а)

[Задание 1.б…………………………………………………………………………3](#Задание1б)

[Задание 1.в…………………………………………………………………………4](#Задание1в)

[Задание 2…………………………………………………………………………...4](#ЗАДАНИЕ2)

[Задание 2.а…………………………………………………………………………6](#Задание2а)

[Задание 2.б…………………………………………………………………………7](#Задание2б)

[Задание 2.в…………………………………………………………………………9](#Задание2в)

[Задание 3………………………………………………………………………….11](#Задание3)

[Задание 3.а………………………………………………………………………..11](#Задание3а)

[Задание 3.б………………………………………………………………………..11](#Задание3б)

[Задание 1 части 5.2……………………………………………………………….13](#Задание1части52)

[Задание 1.1………………………………………………………………………..13](#Задание11)

[Задание 1.2………………………………………………………………………..13](#Задание12)

[Задание 1.3………………………………………………………………………..15](#Задание13)

[ВЫВОДЫ………………………………………………………………………...16](#ВЫВОД)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ…………………………..16](#СПИСОКИНФОРМАЦИОННЫХИСТОЧНИКОВ)

**Задание 1 части 5.1**

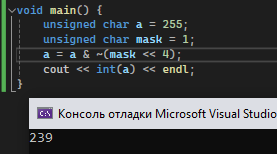
**Задание 1.а.**

**Условие**: Реализуйте вышеприведённый пример (рис. 1), проверьте правильность результата в том числе и на других значениях х.



**Рисунок 1 – Вышеприведённый пример**

Как и требовалось, реализуем пример из рисунка 1 с выводом на экран получаемого значения.



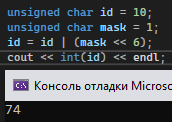
**Рисунок 2 – Реализация кода из рисунка 1**

Ответ соответствует ожиданиям, что означает – код был написан корректно.

**Задание 1.б.**

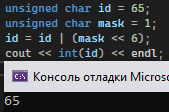
**Условие:** Реализуйте по аналогии с предыдущим примером установку 7-го бита числа в единицу.

Проделав небольшие изменения в коде, выполним требуемое. Результаты различных тестов приведены на рисунках 2-4.

****

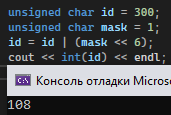
**Рисунок 3 – Установка 7-го бита единицей id=10**

В следующем примере на рис.4 id примет 64, чтобы увидеть, как число, уже имеющее 7-ой бит равный единице, не изменится.

****

**Рисунок 4 – Установка 7-го бита единицей id=65**

А в примере на рис.5 id =300, чтобы показать как будет изменяться число при превышении своего численного лимита.

****

**Рисунок 5 – Установка 7-го бита единицей id=300**

**Задание 1.в.**

**Условие:** Реализуйте код листинга 1, объясните выводимый программой результат.

Рассмотрим код на рис.6 и напишем подобный данному, изобразив его на рис.7 с выводимыми результатами.

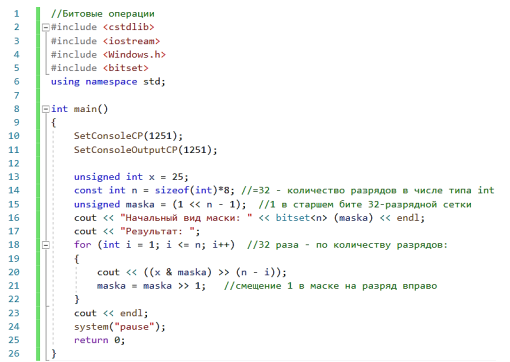
Код работает следующим образом: инициализируется число и вместе с ним маска, которая будет по нему проходить.

Маска приводится в старший разряд 32-ух битов для последующей проверки.

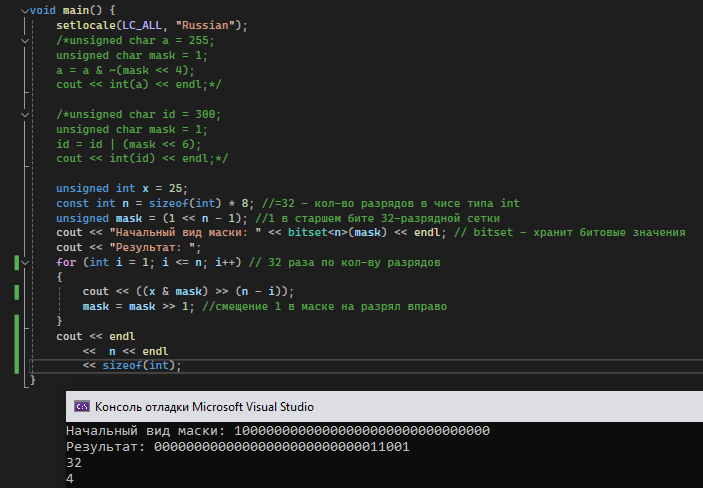
Во время проверки мы проходимся по всем разрядам проверяемого числа, начиная со старшего, постепенно продвигаясь к самому младшему.

i =1, потому что n = 32, а кол-во проходов цикла должно быть равно 32 (32 – 1 + 1 = 32), что мы и получаем при данных условиях.

При этом маска уменьшает свой старший разряд, чтобы осуществить проверку всех разрядов рассматриваемого числа. И по окончанию проверки мы выведем на экран рассматриваемое число в двоичном виде.

****

**Рисунок 6 – Листинг из задания 1.в.**

****

**Рисунок 7 – Реализованный из рис.6 код с полученными результатами**

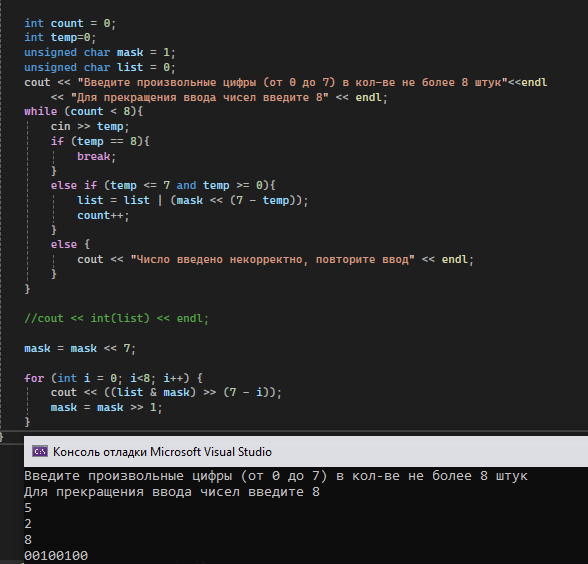
**ЗАДАНИЕ 2**

**Задание 2.а.**

**Условие:** Реализуйте вышеописанный пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа unsigned char. Проверьте работу программы. Если количество чисел в исходной последовательности больше 8 и/или значения превосходят 7, можно подобрать тип беззнакового числа для битового массива с подходящим размером разрядной сетки – до 64 в типе unsigned long long (см. табл. 1).

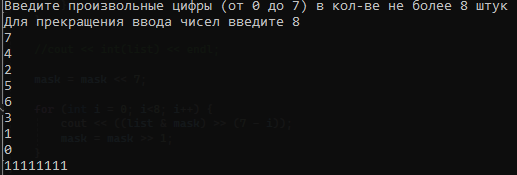
**Математическая модель**: создаём наш список в виде переменной типа unsigned char и маску, после чего вводим значения, проходящие через проверку, и полученный результат выводим на экран.

На рис. 7 предоставлен как сам код, так и его результат при вводе только 2 чисел.

****

**Рисунок 8 – Код для задания 2.а**

А на рис. 9 показан результат работы программы при вводе всех возможных чисел.

****

**Рисунок 9 – Результат работы программы при вводе всех возможных чисел**

**Задание 2.б**

**Условие:** Адаптируйте вышеприведённый пример для набора из 64-х чисел (со значениями от 0 до 63) с битовым массивом в виде числа типа unsigned long long.

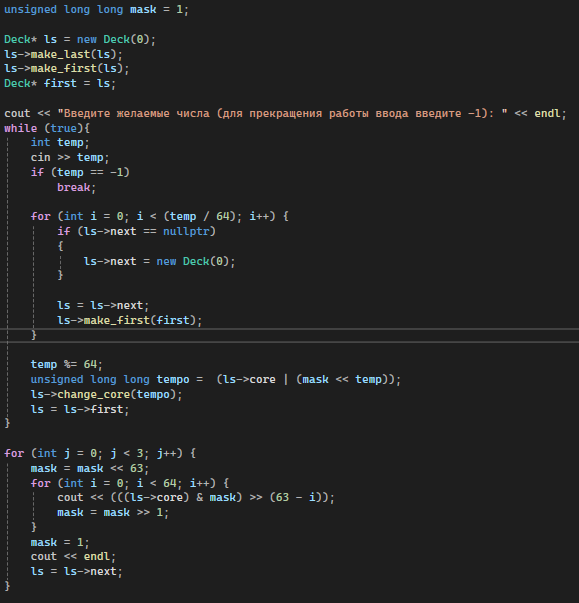
Для решения задачи и последующего усовершенствования кода пришлось зайти совершенно с другой стороны: надо создать линейный взаимосвязный список при помощи структур.

В самом начале решения создадим маску типа unsigned long long и создадим всё необходимое для корректной работы программы, после чего приступим к введению различных чисел. Если полученные числа будут превосходить разрядную сетку, то тогда будут создаваться новые элементы структур с пустым ядром (числом) до тех пор, пока разрядная сетка не расширится на достаточный уровень. И в конце расширения сетки введённое число в неё добавится и, что самое главное, сохранится в ней.

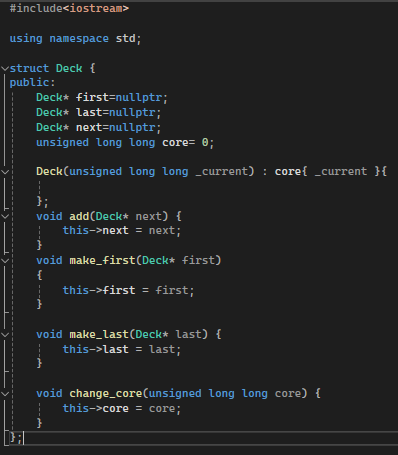
Вышеописанный код представлен на рис. 10-11.

Также можно заметить, что на рис. 10 присутствуют лишние строчки кода в конце – это импровизированный вывод до 192 не включительно с условием того, что их до этого ввёл пользователь. И результаты работы данной программы можно наблюдать на рис. 12.

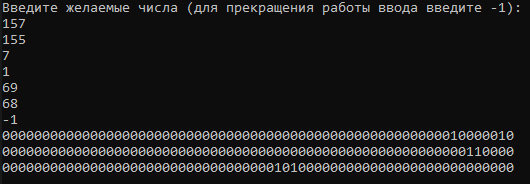
Все недостатки предыдущего метода было по большей степени устранены и программа приняла более усовершенствованный вид.



**Рисунок 10 – 1-ая часть кода программы для задания 2.б.**

****

**Рисунок 11 - 2-ая часть кода программы для задания 2.б.**

****

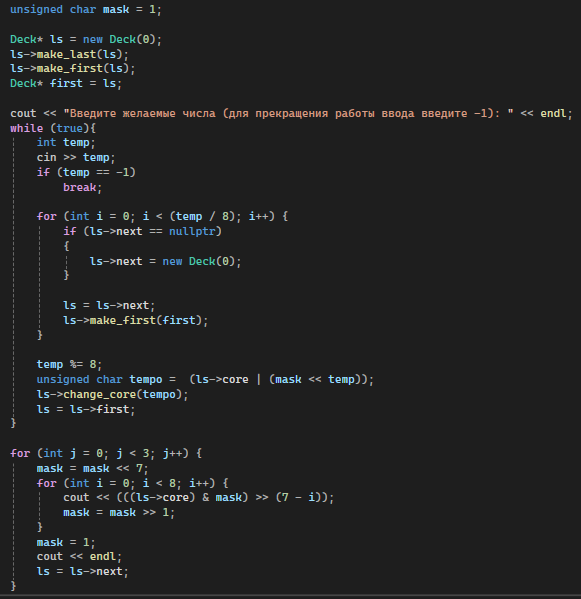
**Рисунок 12 – Результат работы программы задания 2.б.**

**Задание 2.в.**

**Условие:** Исправьте программу задания 2.б, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа unsigned long long, а линейный массив чисел типа unsigned char.

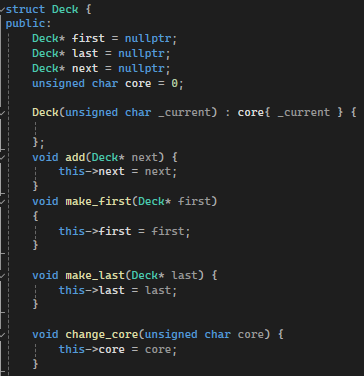
Суть программы сильно не поменялась, единственно что изменилось, так это переменные непосредственно зависящие от типа данных, с которыми необходимо работать, в остальном же – код остался без изменений.

Код программы представлен на рис. 13-14, а результат – на рис. 15.

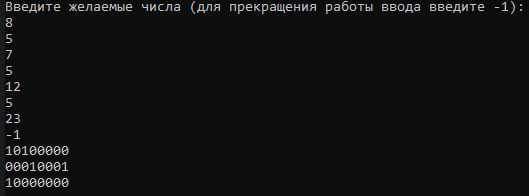


**Рисунок 13 – 1-ая часть кода программы для задания 2.в.**

Можно заметить, что на рис. 15 число 5 повторяется, однако это совершенно никак не меняет конечного результата – число 5 введено, а значит, оно будет в нашем списке. Аналогично ко всем прочим числам.



**Рисунок 14 – 2-ая часть кода программы для задания 2.в.**



**Рисунок 15 – Результат работы программы задания 2.в.**

**Задание 3**

**Задание 3.а.**

**Условие**: Реализуйте задачу сортировки числового файла с заданными условиями. Добавьте в код возможность определения времени работы программы.

**Задание 3.б.**

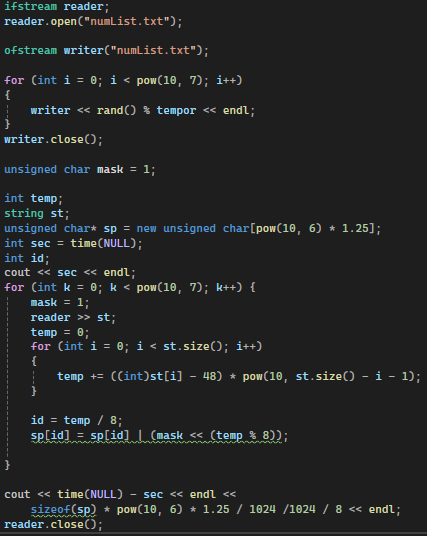
**Условие**: Определите программно объём оперативной памяти, занимаемый битовым массивом.

Метод с конструктами является не самым эффективным, когда речь заходит о большом кол-ве данных, поэтому в данной программе будет использоваться обычный линейный список типа unsigned char.

Для начала мы создадим файл numList.txt, куда запишем значения в диапазоне от 0 включительно до 10^7 не включительно. Записав значения в файл, можно начать их считывание и последующую запись в наш список.

В программе также реализованы выводы времени сортировки файла и веса массива.

Код программы представлен на рис. 17.

****

**Рисунок 16 – Код работы программы для 3.а. и 3.б.**

Также на рис. 17 отображён результат работы программы с затраченным на сортировку временем в секундах и веса массива.



**Рисунок 17 – Результат работы кода программы заданий 3.а. и 3.б.**

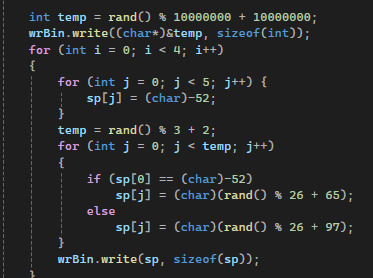
**Задание 1 части 5.2**

**Условие:** Разработать программу поиска записей с заданным ключом в двоичном файле с применением различных алгоритмов.

**Задание 1.1**

**Условие:** Создать двоичный файл из записей (структура записи определена вариантом). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны.

Как и требуется в задании, создадим бинарный файл (рис.18). Можно было бы сначала создать текстовый файл, однако, разобравшись с бинарными файлами, я решил, что будет лучше создавать конечный вид файла без каких-либо промежуточных состояний.

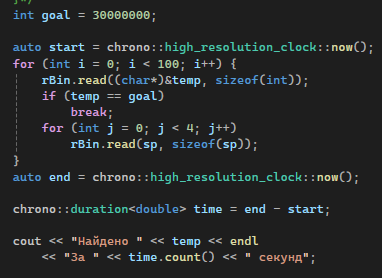
****

**Рисунок 18 – Создание бинарного файла**

**Задание 1.2**

**Условие:** Разработать программу поиска записи по ключу в бинарном файле с применением алгоритма линейного поиска. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей. Составить таблицу с указанием результатов замера времени.

Разработать программу линейного поиска было не сложно – всё-таки это самый примитивный способ поиска значений (рис. 19).

****

**Рисунок 19 – Программа линейного поиска**

Теперь же начнём экспериментировать со значениями от 100 до 10000 записей и отобразим соответствующие результаты на рис. 20-22.



**Рисунок 20 – Результат работы с 100 значений**

****

**Рисунок 21 – Результат работы с 1000 значений**

****

**Рисунок 22 – Результат работы с 10000 значений**

Из полученных результатов в таблице 1 можно заметить, что время увеличивается линейно в зависимости от кол-ва переменных.

Таблица 1 – Результаты замера времени при работе с разл-ым кол-вом пер-ых

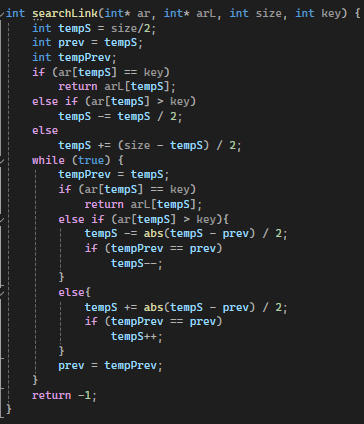
|  |  |
| --- | --- |
| Кол-во переменных | Время, с |
| 100 | 7\* |
| 1000 | 0.001 |
| 10000 | 0.006 |

**Задание 1.3**

**Условие:** Для оптимизации поиска в файле создать в оперативной памяти структур данных – таблицу, содержащую ключ и ссылку (смещение) на запись в файле. Разработать функцию, которая принимает на вход ключ и ищет в таблице элемент, содержащий ключ поиска, а возвращает ссылку на запись в файле. Алгоритм поиска определен в варианте. Разработать функцию, которая принимает ссылку на запись в файле, считывает ее, применяя механизм прямого доступа к записям файла. Возвращает прочитанную запись как результат. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей. Составить таблицу с указанием результатов замера времени.

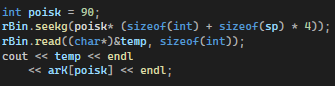
В задании требуется сделать таблицу, однако же данное понятие можно упростить до одномерного массива, где индексы элементов – это ссылки на место в файле, а сами элементы – это ключи. В реализации данного способы решения задачи можно ограничиться всего лишь одним новым полем, а именно массивом целочисленных значений **ar**.

В следующем же задании требуется отсортированный массив, поэтому придётся добавить ещё один массив **arLink**, который будет хранить ссылки на соответствующие элементы из **ar**. Реализация требуемой функции представлена на рис. 23.

****

**Рисунок 23 – Реализация функции бинарного поиска ссылки по ключу**

Теперь же разработаем функцию поиска ключа по ссылке, делается это нетрудно, а вид полученной функции отображён на рис.24.

****

**Рисунок 24 – Функция поиска ключа по ссылке**

Далее мы определим время, затрачиваемое на работу с 100, 1000 и 10000 переменными в крайних случаях в таблице 2.

Таблица 2 – Время работы с разл-ым кол-вом пер-ых.

|  |  |  |
| --- | --- | --- |
| Кол-во переменных | Поиск ссылки по ключу, с | Поиск ключа по ссылке, с |
| 100 | 1\* | 8.6 \* |
| 1000 | 1.5\* | 1,1 \* |
| 10000 | 3\* | 1.8 \* |

Как можно заметить из таблицы, оба способа сортировки получились достаточно эффективными.

**ВЫВОД**

В 5.1 знакомились с битовой сортировкой значений, а также инструментами осуществляющими данную сортировку. Программно определили время, затрачиваемое на сортировку 10^7 чисел и изучили эффективный метод сортировки.

В 5.2 ознакомились с бинарными файлами, изучили работу с ними: запись, хранение и изменение в них данных. Кроме того опробовали различные методы нахождения разных переменных отличающимися друг от друга способами.

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.

2. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 01.09.2021).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 01.09.2021).