МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №12

по дисциплине Основы программирования:

«Строки и текстовые файлы»

Работу выполнил

студент гр. 4141 В.С. Сыворотнев

Санкт-Петербург

2022

***Цель лабораторной работы:*** *изучение структурной организации, способов доступа к элементам и других особенностей текстовых файлов; изучение стандартных средств языка C/C++ для работы с символами, строками и текстовыми файлами; совершенствование навыков процедурного программирования на языке C/C++ при решении задач редактирования текстовых файлов.*

***Задание на программирование:*** *используя технологию процедурного программирования разработать программу обработки текстовых файлов с числом строк не менее пяти, каждая из которых содержит не более 80 символов, в соответствии с индивидуальным заданием.*

***Вариант №31:***

Дан текст, составленный из строк. Под *словом* текста понимается последовательность букв алфавита. Между соседними словами - не менее одного пробела. Перед первым и за последним словом каждой строки произвольное число пробелов.

Найти и сохранить в каждой строке только те слова, которые встречаются в ней по одному разу. Строчные и прописные буквы считать эквивалентными.

***Описание процесса решения:***

При решении задачи, а значит в тексте программы, кроме функции *main*() необходимо реализовать следующие функции:

- функцию создания исходного файла, подлежащего обработке;

- функцию проверки существования физического файла с конкретным именем;

- функцию обработки содержимого файла в соответствии с заданием;

- функцию удаления дубликатов

- функцию сравнения слов без учета регистра

- функцию вывода содержимого файла на экран.

Имена исходного и результирующего файла задаём константными строками. При необходимости у пользователя будет возможность создать файл с любым именем.

Для конкретизации (так сказано в задании на работу) ограничиваем размер строки файла 81 символом (80 символов строки плюс нуль-символ).

Исполнение программы (функция *main*()) начинается с объявления и инициализации символьных массивов (строк), которые будут использоваться для хранения имён исходного *fname\_i* и результирующего *fname\_r* файлов.

После этого создаётся исходный файл. Для этого в функцию *main*() помещаем вызов функции *makeFile*(), параметром которой является значение указателя на начало строки с именем создаваемого исходного файла. Управление передаётся этой функции.

Исполнение функции *makeFile*() начинается с вызова функции проверки существования физического файла с указанным именем создаваемого исходного файла check*File*(). В этой функции делается попытка открытия физического файла с этим именем на чтение. Если такой файл уже существует (если он открылся на чтение), функция возвращает значение 1, в противном случае – 0.

Если физический файл с указанным именем уже существует, то пользователю об этом выдаётся сообщение и предлагается либо сохранить имя создаваемого исходного файла (т.е. уничтожить содержимое существующего файла с таким именем при создании нового файла), либо задать новое имя исходного файла. Результатом такого диалога является имя создаваемого исходного файла, которое хранится в строке *fname\_i*.

После этого в функции *makeFile* () следует оператор открытия исходного файла с именем *fname\_i* на запись. Теперь пользователь построчно вводит содержимое файла с использованием функции gets\_s(), считывает строку из стандартного потока ввода stdin.

Размер создаваемого файла при открытии не оговаривается, поэтому пользователю необходимо вывести подсказку с признаком окончания процесса создания файла. Поскольку создаётся текстовый файл, то признаком окончания его создания выбрана попытка записи в него пустой строки.

Исходный файл закрывается с использованием функции *fclose*(), и функция *makeFile*() завершает свою работу, возвращая в функцию *main*() имя созданного исходного файла.

Выводим содержимое созданного файла, чтобы впоследствии сравнить его с результатом обработки. Для этого в функцию *main*() помещаем вызов функции *outputFile*(), параметром которой является имя созданного исходного файла. Предварительно проверяем существование такого физический файл (пытаемся открыть его на чтение).

Теперь приступаем к обработке содержимого исходного файла, вызывая функцию *processFile*(), параметрами которой являются имена исходного (*fname\_i*) и результирующего (*fname\_r*) файлов.

В этой функции исходный файл открывается на чтение, а результирующий – на запись (проверка существования физического файла с указанным именем результирующего файла опущена).

Читаем первую строку исходного файла, используя функцию *fgets*(). Пока не найден конец исходного файла (*while*(!*feof*(*in*))) выполняется функция removeDuplicates(), которая принимает исходную и результирующую строку.

Очищаем текущую результирующую строку, помещая в качестве значения её первого символа нуль-символ ‘\0’. Объявляем указатель *p* на выделенное слово, затем удаляем символ переноса строки. После чего генерируем массив слов со строками, используя цикл for, выделяем память. Затем следует алгоритм поиска дубликатов. Суть алгоритма заключается поочередном переборе соседних слов и их сравнении, причем каждый элемент сравнивается только с последующими элементами. Для того, чтобы сравнить элементы без учета регистра, вызываем функцию areEqual(), которая принимает 2 строки в качестве параметров. Если элементы совпадают, то дубликат удаляем. После этого, заполняем результирующую строку словами, используя функцию *strcat*(). Убираем пробел в конце. Затем очищаем выделенную на массив память. Когда обрабатываемая строка закончится, убираем последний пробел в результирующей строке, и, если её длина не равна 0, записываем её в результирующий файл, используя функцию *fprintf*() и добавляя при этом в конец строки символ перехода на новую строку ‘\n’.

Читаем новую строку исходного файла, и, пока не найден признак конца исходного файла, повторяем обработку.

Закрываем исходный и результирующий файлы, используя функцию *fclose*(). Выполнение функции *processFile*() завершается, и в функцию *main*() возвращается имя результирующего файла.

Проверяем, создан ли результирующий файл, открывая его на чтение. Если файл создан, выводим его содержимое функцией *outputFile*().

Выполнение программы завершается. Задача решена.

Анализируем результаты теста и делаем выводы.

***Текст программы***

*/\*31.*

*Дан текст, составленный из строк. Под словом текста понимается последовательность букв алфавита.*

*Между соседними словами - не менее одного пробела. Перед первым и за последним словом каждой*

*строки произвольное число пробелов.*

*Найти и сохранить в каждой строке только те слова, которые встречаются в ней по одному разу.*

*Строчные и прописные буквы считать эквивалентными.*

*Решение с использованием функций библиотеки <stdio.h>\*/*

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include <iostream>

#include<locale.h>

#include <algorithm>

using namespace std;

#define FNAME\_I "D:\\st\_i.txt\0"    //имя исх. файла задано константной строкой

#define FNAME\_R "D:\\st\_r.txt\0"    //имя рез. файла задано константной строкой

const int RAZ = 80;            //максимальная длина строки

const char PR\_R[] = "r";        //признак открытия файла на чтение

const char PR\_W[] = "w";        //признак открытия файла на запись

void makeFile(char\*);            //функция создания файла

int checkFile(char\*, const char\*);    //проверка наличия файла

void processFile(char\*, char\*);    //обработка содержимого

void outputFile(char\*);    //вывод содержимого файла на печать

void removeDuplicates(char st\_in[RAZ], char st\_out[RAZ]);

bool areEqual(string a, string b);

int main() {

  char fname\_i[] = FNAME\_I;

  char fname\_r[] = FNAME\_R;

 setlocale(LC\_ALL,"Russian") ;

  printf(" Программа создает файл строк.**\n**");

  printf(" Затем файл построчно читается, при этом в каждой строке,**\n**");

  printf(" содержащей слова, разделенные произвольным количеством пробелов,**\n**");

  printf(" остаются только уникальные слова.**\n**");

  printf(" Создаем исходный файл:**\n**");

//создаем исходный файл

  makeFile(fname\_i);

//выводим его содержимое

  if (!checkFile(fname\_i, PR\_R)) {

    printf(" Ошибка открытия файла %s", fname\_i, " на чтение**\n**");

    printf(" Нажмите <Enter>**\n**");

    getchar();

    return 0;            //если исходный файл не создан

  }

  printf(" Содержимое исходного файла:**\n**");

  outputFile(fname\_i);

//обрабатываем файл

  processFile(fname\_i, fname\_r);

//выводим содержимое результата

  if (!checkFile(fname\_r, PR\_R)) {

    printf(" Ошибка открытия файла %s", fname\_r, " на чтение**\n**");

    printf(" Нажмите <Enter>**\n**");

    getchar();

    return 0;            //если результирующий файл не создан

  }

  printf(" Результат (в каждой строке остались только уникальные слова)**\n**");

  outputFile(fname\_r);

  printf("**\n** Для завершения нажмите <Enter>");

  getchar();

  return 0;

}

void makeFile(char\* fname\_i) {

  char st[RAZ];        //исходная строка

  FILE\* in;            //текстовый файл

  char otv;            //ответ пользователя

*/\*Открываем файл в режиме чтения (r)*

*Если файл с таким именем уже есть, то либо пользователь вводит*

*новое имя, либо новые записи будут дописаны поверх старых\*/*

  while (checkFile(fname\_i, PR\_R)) {

    printf("**\n** Файл с заданным именем уже есть! Зададите новое имя Y/N? ");

    otv = getchar();

    if (otv == 'Y' || otv == 'y') {

      printf(" Введите новое имя:**\n**");

      gets(fname\_i);

    } else {

      getchar();

      break;

    }

  }

*/\*проверяем возможность открытия файла для записи*

*поскольку полное имя файла могло быть задано с ошибкой\*/*

  if (!checkFile(fname\_i, PR\_W)) {

    printf(" Ошибка открытия файла %s", fname\_i, " на запись**\n**");

    printf(" Нажмите <Enter>**\n**");

    getchar();

    exit(0);

  }

*/\*Открываем файл в режиме записи (w) текста (по умолчанию)*

*Если файл с таким именем уже есть, то новые данные*

*будут дописаны поверх старых\*/*

  in = fopen(fname\_i, PR\_W);

  printf(" Создание файла.**\n**");

  printf(" Признак окончания ввода - ввод пустой строки**\n**");

  printf(" Введите строку и нажмите <Enter>**\n**");

  printf(" ->");

  gets(st);    */\*функция вводит строку целиком, включая*

*пробелы и символ \n\*/*

  while (strlen(st)) {

    fprintf(in, "%s**\n**", st);

    printf(" Введите строку и нажмите <Enter>**\n**");

    printf(" ->");

    gets(st);

  }

  fclose(in);        //закрываем файл

}

//проверка наличия файла

int checkFile(char\* fname, const char\* pr) {

//Открываем файл

  if (!(fopen(fname, pr)))    //если файл не существует

    return (0);

  return (1);

}

//обработка содержимого файла

void processFile(char\* fname\_i, char\* fname\_r) {

  FILE\* in, \* out;        //исходный и результирующий файлы (потоки)

  char st\_in[RAZ];        //исходная строка

  char st\_out[RAZ];        //результирующая строка

  in = fopen(fname\_i, PR\_R);        //открываем файл на чтение

  out = fopen(fname\_r, PR\_W);    //открываем файл на запись

  fgets(st\_in, RAZ, in);        //читаем строку из файла in

  while (!feof(in)) {

    removeDuplicates(st\_in, st\_out);

    st\_out[strlen(st\_out) - 1] = '**\0**';    //убираем последний пробел

    if (strlen(st\_out))

      fprintf(out, "%s**\n**", st\_out);    //записываем строку в новый файл

    fgets(st\_in, RAZ, in);        //считываем новую строку из файла

  }

  fclose(in);            //Закрываем файл in

  fclose(out);            //Закрываем файл out

}

void removeDuplicates(char st\_in[RAZ], char st\_out[RAZ]) {

  char\* p;

  char\*\* words;

  size\_t size, i, j;

  st\_out[0] = '**\0**';

  // удаляем символ переноса строки

  if (p = strrchr(st\_in, '**\n**')) {

    \*p = '**\0**';

  }

  words = NULL;

  size = 0;

  p = strtok(st\_in, " **\t**");

  words = new char\* [size + 1];

  // генерируем массив слов со строками

  for (p; p != NULL; p = strtok(NULL, " **\t**")) {

    words[size] = new char[size];

    words[size] = strdup(p);

    ++size;

  }

  // ищем и удаляем дубликаты

  for (i = 0; i < size - 1; ++i) {

    for (j = i + 1; j < size; ++j) {

      if (areEqual(words[i], words[j])) {

        \*(words[j]) = '**\0**';

      }

    }

  }

  // заполняем результирующую строку

  for (i = 0; i < size; ++i)

    if (\*(words[i])) {

      strcat(st\_out, words[i]);    //добавляем слово к результату

      strcat(st\_out, " ");    //добавляем пробел между слов

    }

  // очищаем память, задействованную на массив

  for (int i = 0; i != size; ++i)

    delete [] words[i];

  delete [] words;

}

//вывод содержимого файла на экран

void outputFile(char\* fname) {

  FILE\* in\_out;

  char st[RAZ];            //считанная строка

  in\_out = fopen(fname, PR\_R);    //открываем файл на чтение

  fgets(st, RAZ, in\_out);        //читаем строку из файла in\_out

  while (!feof(in\_out)) {

    printf("%s", st);            //выводим очередную строку на экран

    fgets(st, RAZ, in\_out);

  }

  fclose(in\_out);

}

bool areEqual(string a, string b) {

  transform(a.begin(), a.end(), a.begin(), ::tolower);

  transform(b.begin(), b.end(), b.begin(), ::tolower);

  if (a == b) {

    return true;

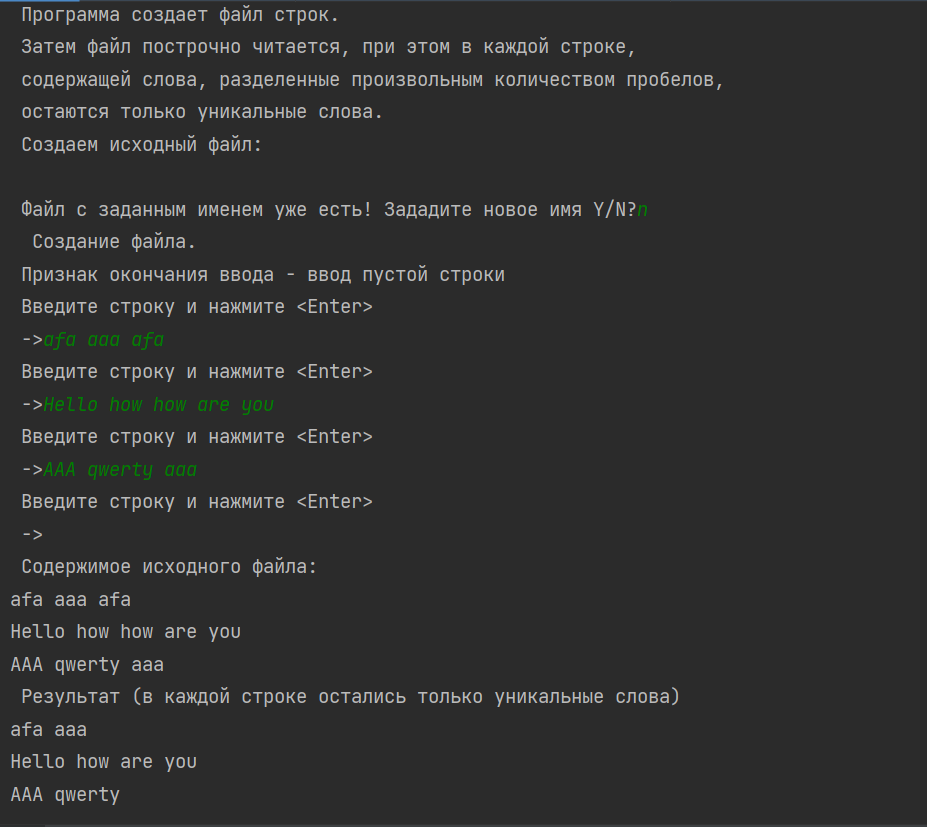
  } else {

    return false;

  }

}

***Скриншот результатов выполнения программы***



***Вывод:*** *я изучил структурную организацию, способы доступа к элементам и другие особенности текстовых файлов; изучил стандартные средства языка C/C++ для работы с символами, строками и текстовыми файлами; усовершенствовал навыки процедурного программирования на языке C/C++ при решении задач редактирования текстовых файлов.*