**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по курсовой работе № 1**

**по дисциплине «Программирование»**

Тема: **Обработка текстовой информации**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0310 |  | Савенко Н.С |
| Преподаватель |  | Хахаев И.А |

Санкт-Петербург

2020**Введение.**

**Цель работы.**

Законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).

**Задание (вариант 87)**

Задан текст, содержащий произвольное количество строк, в которых отдельные слова могут разделяться одним или несколькими пробелами и знаками пунктуации (перенос слов с одной строки на другую не используется). Сформировать новый текст, который является результатом следующего преобразования исходного текста: удалить из строк слова одинаковой длины.

**Постановка задачи и описание решения**

Вводится текст. Затем в цикле производится перебор символов. После каждого пройденного символа прибавляется счетчик символов. Если встречается пробел или символ пунктуации, то нужно найти количество слов с длиной равной значению счетчика(см. описание функции stringCountByLen). Если найденных слов больше 1, то удаляются все слова такой длины из массива символов(см. описание функции clearWords) и алгоритм снова начинает перебирать символы с начала по уже обновленному тексту.

**Структура вызовов функций**

см. файл Savenko-kurs-invoke-scheme

**Описание функций.**

*int stringCountByLen(char \*string, int len);*

Находит количество слов заданной длины.

Производится перебор текста. При переборе увеличивается значение счетчика символов. При нахождении пробела или знака пунктуации сравнивается значение счетчика со значением параметра. Если они равны, то к значению найденных слов прибавляется 1. После каждого нахождении знака пунктуации счетчик символов обнуляется.

*void clearWords(char \*string, char \*result, int n)*

Очищает слова заданной длины.

Производится перебор текста. При переборе увеличивается значение счетчика символов. При нахождении пробела или знака пунктуации сравнивается значение счетчика со значением параметра. Если они равны, то запускается цикл, который стирает количество предыдущих символов равное значению счетчика символов. После каждого нахождении знака пунктуации счетчик символов обнуляется. Обновленный текст записывается по адресу входного параметра.

*void filterWords(char \*text)*

Выполняет чистку текста.

Производится перебор текста. При переборе увеличивается значение счетчика символов. При нахождении пробела или знака пунктуации идет поиск количества слов с длиной равной значению счетчика. Если их больше 1, то производится очистка слов заданной длины и перебор начинается с начала по обновленному тексту. После каждого нахождении знака пунктуации счетчик символов обнуляется.

*char \*readFromFile(FILE \*df, int \*s\_count);*

Читает текст из файла с помощью и записывает в *s\_count* количество строк в нем. Количество строк вычисляется перебором символов, если встречается символ конца строки, то счетчик строк прибавляется. Возвращает текст.

*int new\_gets(char \*s, int lim, char endSymbol)*

Читает текст из консоли, пока не упрется в ограничение, прописанное в параметре, либо не встрется символ ограничитель, указанный в параметре.

**Описание переменных**

**main()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | state | int | Вариант работы |
| 2 | str\_count | int | Количество строк |
| 3 | text | char[] | Текст для обработки |
| 4 | endSymbol | char | Символ ограничитель для ввода из консоли |
| 5 | symbolsCount | int | Ограничение по количеству символов для ввода с консоли |

**stringCountByLen(char \*string, int len)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i/p | int | Счетчики символов |
| 2 | count | int | Найденное количество |
| 3 | string | char[] | Текст для обработки |
| 4 | current | char | Текущий символ |
| 5 | len | int | Нужная длина |

**clearWords(char \*string, char \*result, int n)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i/p | int | Счетчики символов |
| 2 | count | int | Предполагаемое количество символов для удаления |
| 3 | string | char[] | Текст для обработки |
| 4 | current | char | Текущий символ |
| 5 | n | int | Нужная длина |
| 6 | result | Char[] | Результирующий текст |

**filterWords(char \*text)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i | int | Счетчик символов |
| 2 | count | int | Найденное количество символов в слове |
| 3 | text | char[] | Текст для обработки |
| 4 | current | char | Текущий символ |
| 5 | len | int | Количество слов с найденным количеством символов |
| 6 | result | Char[] | Результирующий текст |
| 7 | buffer | Char[] | Буферный текст для дальнейшего перевода в основной |

**readFromFile(FILE \*df, int \*s\_count)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i | int | Счетчик символов |
| 2 | str\_count | int | Найденное количество символов |
| 3 | text | char[] | Текст для обработки |
| 4 | current | char | Текущий символ |
| 5 | size | int | Объем текста |
| 6 | df | FILE \* | Указатель на файл для работы |
| 7 | s\_count | Int | Количество строк |

**new\_gets(char \*s, int lim, char endSymbol)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | i | int | Счетчик символов |
| 2 | lim | int | Ограничение на количество символов |
| 3 | s | char[] | Текст из консоли |
| 4 | c | char | Текущий символ |
| 5 | endSymbol | Char | Символ окончания текста |

**Схема алгоритма:**

См файл Savenko-kurs-scheme

**Контрольные примеры:**

**Пример 1:**

Входные данные:

1

60

)

y

hey

its

me

Выходные данные:

y

me

**Пример 2(выполнение зависит от содержимого input.txt)**

Входные данные:

2

Выходные данные:

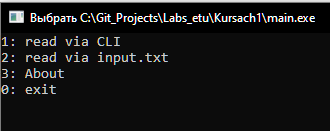
gracefull

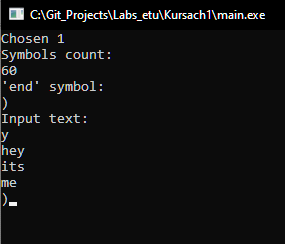
**Текст программы**

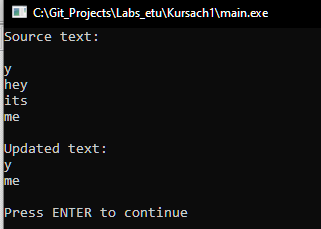
См. файл Savenko-kurs-main.c

**Примеры выполнения программы:**

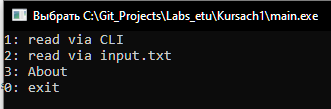
**Пример 1:**

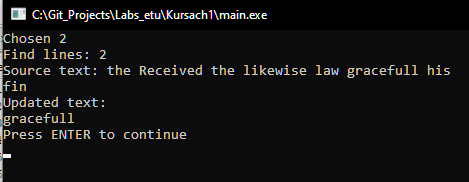
****

****

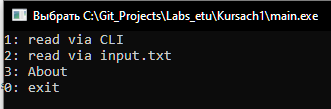
****

**Пример 2:**





Пример 3:





**Заключение**:

**Использованные заголовочные файлы:**

**stdio.h**

puts() – вывод в консоль

scanf() – получение данных из консоли

getchar() – получение символа из консоли

fopen() – открытие файла

fseek() установка положения в файле

ftell() – получение положения в файле

rewind() – установка положения в начало файла

fread() – чтение из файла

fclose() – закрытие файла

**stdlib.h**

malloc() – выделение памяти

system() - передача команды командному процессору операционной системы **string.h**

strlen() – получение длины строки

strcpy() - копирование  
**ctype.h**

isspace() – копирование строки

**Вывод:**

Научился делать законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).