**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**Курсовая РАБОТА**

**по дисциплине «Программирование»**

**Тема: Обработка строк.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2018

|  |  |  |
| --- | --- | --- |
| Студент Киреев К.А. | | |
| Группа 8381 | | |
| Тема работы: обработка строк.  Исходные данные: все подзадачи программы, ввод/вывод должны быть реализованы в виде отдельной функции. Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Также и последующая обработка данных должна производиться с использованием функций стандартных библиотек языка. Сохранение, обработка и вывод данных должны осуществляться при помощи динамической памяти. | | |
|  | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников». | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 23.11.2018 | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Киреев К.А. |
| Преподаватель |  | Жангиров Т.Р. |

**Аннотация**

В ходе выполнения курсовой работы была написана программа, которая обрабатывает введенный пользователем текст. Обработка осуществляется в соответствии с заданием. В работе реализован интерфейс для взаимодействия с пользователем. Также реализована защита от некорректного ввода. Реализован выход из программы при отсутствии текста после обработки. Для хранения текста используются динамические массивы. Сортировки и операции со строками осуществлены с использованием функций стандартной библиотеки. Каждая подзадача программы, ввод/вывод реализованы в виде отдельных функций.

**Summary**

In the course work the program that processes the text entered by the user was created. Processing is carried out in accordance with the task. The program has a simple user interface. Also the program has implemented protection against incorrect input. Implemented exit from the program with no text after processing. Text is saved in dynamic arrays. Sorts and string operations are implemented with standard library functions. Each subtask of the program, input and output are implemented as separate functions.

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Задание | 6 |
| 2. | Ход выполнения работы | 7 |
| 2.1. | Структура программы | 7 |
| 2.2. | Создание функции ввода текста | 7 |
| 2.3. | Разделение текста на массив предложений | 8 |
| 2.4. | Создание функции удаления повторяющихся предложений | 8 |
| 2.5. | Создание функции для взаимодействия с пользователем | 8 |
| 2.6. | Создание функций обработки текста | 9 |
| 2.6.1. | Создание функции для подсчета слова garbage в предложениях | 9 |
| 2.6.2. | Создание функции для замены всех цифр в предложении на строку, введенную пользователем | 10 |
| 2.6.3. | Создание функции для удаления предложений с тремя заглавными буквами подряд | 10 |
| 2.6.4. | Создание функции сортировки | 11 |
| 2.7. | Итоговый вид программы в файле main.c | 12 |
|  | Заключение | 13 |
|  | Список использованных источников | 14 |
|  | Приложение А. Исходный код программы | 15 |

**ВВЕДЕНИЕ**

**Цель работы**

Написать на языке Си программу, которая обрабатывает текст, состоящий из латинских букв, цифр, остальных символов, по указаниям пользователя.

**Основные задачи**

Применить на практике стандартные функции таких библиотек, как string.h, ctype.h, stdlib.h.

Использовать динамические массивы.

Реализовать программу, все подпрограммы которой реализованы в виде функций.

Освоить процесс тестирования программы и исправления ее поведения в исключительных случаях.

**Методы решения**

Разработка программы велась с помощью операционной системы Windows 10 и с использованием онлайн-компилятора repl.it. Отладка также производилась при помощи repl.it

**1. ЗАДАНИЕ**

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой). Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).

Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку). Также следует предусмотреть возможность выхода из программы:

1. Для каждого предложения посчитать количество слов “garbage” в нем (без учета регистра). В зависимости от количества вывести следующие строки: 0 - “Clean”, [1 5] - “Must be washed”, >5 - “It is a dump”.
2. Заменить все цифры в предложении на введенную строку.
3. Удалить все предложения, в которых есть три подряд идущие буквы в верхнем регистре.
4. Отсортировать по уменьшению количества слов начинающихся с гласной буквы.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

# 2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

**2.1. Структура программы**

Вначале прописывается первичное устройство программы. Объявляются заголовочные файлы стандартной библиотеки Си, такие как <stdio.h>, <stdlib.h>, <string.h>, <ctype.h>. Прописываются прототипы функций, используемых в программе:

1. int Text(char\*\*\* text, char\*\* str);
2. int RemoveTheSame(char\*\*\* text, int\* num);
3. int Menu(void);
4. int Garbage(char\*\*\* text, int num);
5. int Replace(char\*\*\* text, int num);
6. int Delete(char\*\*\* text, int num);
7. int Sort(char\*\*\* text, int num);
8. int cmp(const void\*a, const void \*b);

**2.2. Создание функции ввода текста**

Создаются переменные currentLetter, line, sym для корректного запоминания текста. Выделяется память с помощью функции calloc() под первоначальный текст, который вводится в виде строки и далее разделяется на массив предложений. Вначале идет посимвольное считывание текста до символа переноса строки, символы пробела и табуляции не учитываются; при нехватке памяти выделяется дополнительная память под текст при помощи функции realloc().

**2.3. Разделение текста на массив предложений**

Выделяется память под текст. Далее в цикле while() выделяется память под предложение. В следующем цикле символы из начальной строки берутся по порядку и при отсутствии соответствующих условий записываются в ячейку массива, под которую была выделена память. В конце предложения добавляется символ конца строки. Также выделяется дополнительная память при превышении заданного числа предложений в тексте и превышении заданного числа символов в предложении.

**2.4. Создание функции удаления повторяющихся предложений**

В цикле for, в котором задается текущее значение переменной j, зависящее от i, и далее в цикле while идет посимвольное сравнение всех предложений текста. Переменной flag присваивается значение 1, при котором происходит удаление предложения, если не найдется условий для его сохранения. Вначале сравниваются длины предложений, при совпадении которых flag обнуляется. Далее идет посимвольное сравнение двух предложений приведенных к нижнему регистру, и происходит обнуление flag при несовпадении символов. Затем происходит удаление предложения, посредством перемещения всех предложений после удаленного на одну ячейку влево и уменьшение значения переменной num, отвечающей за общее количество предложений, на один.

**2.5. Создание функции для взаимодействия с пользователем**

Создана функция Menu, в которой реализуется интерфейс для взаимодействия с пользователем во время использования программы. С помощью функций puts() производится вывод текста меню с выбором команд для выполнения. Например: puts("Введите 4, чтобы отсортировать предложения по уменьшению количества слов начинающихся с гласной буквы"). Далее происходит ввод выбранной команды. При вводе цифр 1,2,3,4 происходит дальнейшая обработка текста.

При вводе цифр 5 и 6 происходит вывод информации о текущем тексте. При вводе любого отличного символа происходит выход из программы. Функция возвращает значение команды, введенной пользователем.

**2.6. Создание функций обработки текста**

**2.6.1. Создание функции для подсчета слова garbage в предложениях**

Функция Garbage принимает на вход массив предложений и количество предложений. Счетчик count используется для подсчета предложений, в которых есть совпадающие со словом garbage слова. В цикле for проверяется каждое предложение. И далее в цикле while посимвольно проверяется каждое слово до символа конца строки ‘\0’. Если символ, приведенный к нижнему регистру, совпадет с первым символом искомого слова, то в цикле while пойдет проверка всех последующих символов на соответствующее совпадение с остальными символами слова garbage и увеличение счетчика count при совпадении всех символов слова garbage и последовательных символов, входящих в текущее слово. При значении count больше 5 выводится сообщение “It is a dump”. При значении равном 0 выводится сообщение “Clean”. При других значениях выводится ”Must be washed”. Например, для текста “garbage yes. GaRbaGeno. Maybe.” выведется сообщение:

garbage yes. ---> Must be washed

GaRbaGeno. ---> Must be washed

Maybe. ---> Clean

**2.6.2. Создание функции для замены всех цифр в предложении на строку, введенную пользователем**

Создана функция Replace(). Вначале в цикле for производится проверка на присутствие цифр в предложениях. Если в предложениях отсутствуют цифры, то выводится соответствующее сообщение. Иначе пользователю предлагается ввести строку, на которую надо поменять цифры. Происходит посимвольное считывание строки и сохранение ее в переменную buffer. Далее в цикле for проверяется каждое предложение и в цикле while каждое слово. Если будет найдена цифра, то сначала выделяется дополнительная память под строку. Затем происходит смещение всего предложения вправо на длину buffer. И в конце образовавшееся место посимвольно заполняется введенным словом. Например, для текста “Abcde5ff6g. Seco12n4.” и строки “INPUT” выведется сообщение:

AbcdeINPUTffINPUTg.

SecoINPUTINPUTnINPUT.

**2.6.3. Создание функции для удаления предложений с тремя заглавными буквами подряд**

Создана функция Delete. В цикле for идет проверка каждого предложения. И в последующем цикле while идет посимвольная проверка каждого слова. Идет проверка символа на заглавную букву, при успешном прохождении которой происходит увеличение переменной flag. Далее реализована проверка того, что подряд идут именно **три** заглавные буквы, при 4 и более буквам flag обнуляется. Далее при значении flag равном 3 происходит удаление предложения способом, описанным в пункте 2.4., и уменьшение счетчика предложений на один. В конце цикла происходит проверка того, что **подряд** идут именно три **заглавные** буквы. Без этой проверки функция удаляла бы предложения, в которых три заглавные буквы расположены хаотично.

При удалении всех предложений выводится сообщение “Удалены все предложения”. Иначе выводится обработанный текст. Например, для текста

“ SentenceYes. One Two Three. SEnTENce. THEEND.” выведется сообщение:

SentenceYes.

One Two Three.

THEEND.

**2.6.4. Создание функции** **сортировки**

Создана функция Sort для сортировки массива предложений по уменьшению слов, начинающихся с гласной буквы. Использована функция qsort() из библиотеки <stdlib.h> для сортировки массива. Далее осуществляется вывод измененного текста. Для функции qsort написан функция - компаратор cmp(). В функции сравниваются два предложения. В первом цикле for идет посимвольная проверка первого предложения. Если текущая буква – гласная буква и предыдущий символ – это пробел или запятая, то увеличивается счетчик слов для первого предложения. Если это первая буква предложения, то другая проверка смотрит только на то, что буква гласная. Для второго предложения выполняются аналогичные действия. Далее сравниваются счетчики для первого и второго предложений counter\_for\_s и counter\_for\_p. И возвращается значение 1, -1 или 0 при равенстве слов, начинающихся с гласной буквы.

Например, для текста “Abc DEf. OKo iju awe. opert uhg.” выведется сообщение:

OKo iju awe.

opert uhg.

Abc DEf.

**2.7. Создание главной функции main.c**

В функции main() вначале вызывается функция Text() для создания массива предложений. Она возвращает начальное количество предложений и присваивает это значение переменной sentences. Далее вызывается функция RemoveTheSame(), в которой удаляются повторяющиеся предложения. Она возвращает количество предложений после удаления и присваивает это значение переменной delete\_sentences. Затем в цикле do{}while вызывается функция Menu() для выбора текущего действия. И с помощью конструкции switch – case вызывается функция в соответствии с командой, заданной пользователем. При количестве предложений равном 0 происходит присвоение переменной flag значение единицы, при котором производится выход из цикла и окончание программы.

**ЗАКЛЮЧЕНИЕ**

В результате выполнение курсовой работы была освоена работа с динамическими массивами языка Си. Была создана программа для обработки введенного пользователем текста согласно задачам, указанным в задании. Был получен опыт в разработке программы, в которой все задачи, а также ввод и вывод выполняются в разных функциях. В обработке и сортировке использовались функции стандартных библиотек string.h, ctype.h, stdlib.h. Программа имеет интерфейс, созданный для удобства использования его пользователем. Также программа реагирует на некорректный ввод и на отсутствие В процессе выполнения был освоен процесс тестирования и исправления программы.

**список использованных источников**

1. Керниган Б. В., Ритчи Д. М. Язык программирования Си: Пер. с англ.

— 3-е изд. — СПб.: Невский Диалект, 2001. — 352 с.

# 2. Майк МакГрат Программирование на С для начинающих : Пер. с англ.- СПб.: Эксмо, 2016 – 193 c.

3. Стив Оолайн C Elements of Style М.: M&T books, 1992 456 c.

4. Banahan M., Brady D, Doran M The C Book // The C Book. URL:

<http://publications.gbdirect.co.uk/c_book/thecbook.pdf>

5. Nick Parlante Essential C // Essential C. URL: http://cslibrary.stanford.edu/101/EssentialC.pdf

**ПРИЛОЖЕНИЕ А.**

**ИСХОДЫЙ КОД ПРОГРАММЫ.**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define STRSIZE 500

#define SYMBOL 100

#define SENTENCE 50

int Text(char\*\*\* text, char\*\* str);

int RemoveTheSame(char\*\*\* text, int\* num);

int Menu(void);

int Garbage(char\*\*\* text, int num);

int Replace(char\*\*\* text, int num);

int Delete(char\*\*\* text, int num);

int Sort(char\*\*\* text, int num);

int cmp(const void\*a, const void \*b);

int main()

{

char\*\* text;

int flag = 0;

char\* str;

int sentences = Text(&text, &str);

int delete\_sentences = RemoveTheSame(&text, &sentences);

do

{

int cmdf = Menu();

printf("\n");

switch(cmdf){

case 1: delete\_sentences = Garbage(&text, delete\_sentences);

if (delete\_sentences == 0)

{

puts("<-----> Выход <----->\n");

flag = 1;

}

break;

case 2: delete\_sentences = Replace(&text, delete\_sentences);

if (delete\_sentences == 0){

puts("<-----> Выход <----->\n");

flag = 1;}

break;

case 3: delete\_sentences = Delete(&text, delete\_sentences);

if (delete\_sentences == 0){

puts("<-----> Выход <----->\n");

flag = 1;}

break;

case 4: delete\_sentences = Sort(&text, delete\_sentences);

if (delete\_sentences == 0){

puts("<-----> Выход <----->\n");

flag = 1;}

break;

case 5: for (int i = 0; i < delete\_sentences; i++)

printf("%s.\n", text[i]);

break;

case 6: printf("Текущее количество предложений равно %d.\n", delete\_sentences); break;

default: puts("<-----> Выход <----->\n"); flag = 1; break;

}

if (flag == 1){

break;}

}

while (1);

return 0;

}

int Text(char\*\*\* text, char\*\* str)

{

char currentLetter = 0;

int line = 0;

int sym = 0;

int max\_sentences = SENTENCE;

int str\_size = STRSIZE;

int max\_symbols = SYMBOL;

\*str = (char\*)calloc(str\_size, sizeof(char));

puts("Введите текст:");

for (int counter = 0; (currentLetter = getchar()) != '\n'; counter++)

{

if ((currentLetter == ' ') && (sym == 0))

{

counter--;

continue;

}

sym++;

if (currentLetter == '.')

sym = 0;

if (str\_size < counter + 1)

{

str\_size += STRSIZE;

\*str = (char\*)realloc(\*str, str\_size \* sizeof(char));

}

(\*str)[counter] = currentLetter;

}

int y = strlen(\*str);

(\*str)[++y] = '\0';

char\*\* new\_text = (char\*\*)calloc(max\_sentences, sizeof(char\*));

char c;

int i = 0;

int flag = 0;

while(1)

{

int symbol = 0;

new\_text[line] = (char\*)calloc(max\_symbols, sizeof(char));

while(1)

{

c = (\*str)[i++];

if (c == '.')

break;

if (c == '\0'){

flag = 1;

break;

}

new\_text[line][symbol] = c;

if (symbol++ == max\_symbols){

max\_symbols += SYMBOL;

new\_text[line] = (char\*)realloc(new\_text[line], max\_symbols \* sizeof(char));

}

}

if (flag == 1)

break;

new\_text[line][++symbol] = '\0';

if (line++ == max\_sentences){

max\_sentences += SENTENCE;

new\_text = (char\*\*)realloc(new\_text, max\_sentences \* sizeof(char\*));

}

}

\*text = new\_text;

return line;

}

int RemoveTheSame(char\*\*\* text, int\* num)

{

int j = 0;

int flag = 1;

for (int i = 0; i < \*num; i++)

{

j = i + 1;

while (j < \*num)

{

flag = 1;

if (strlen((\*text)[i]) != strlen((\*text)[j]))

flag = 0;

for (int k = 0; k < strlen((\*text)[i]); k++)

{

if (tolower((\*text)[i][k]) != tolower((\*text)[j][k]))

flag = 0;

}

if (flag == 1)

{

for(int new\_i = j; new\_i < \*num - 1; new\_i++)

(\*text)[new\_i] = (\*text)[new\_i + 1];

\*num -= 1;

}

else

j++;

}

}

return \*num;

}

int Menu(void)

{

int cmd;

puts("\nДействия:");

puts("Введите 1, чтобы посчитать количество слов {garbage} в каждом предложении.");

puts("Введите 2, чтобы заменить все цифры в предложении на введенную строку.");

puts("Введите 3, чтобы удалить все предложения в которых есть три подряд идущие буквы в верхнем регистре.");

puts("Введите 4, чтобы отсортировать предложения по уменьшению количества слов начинающихся с гласной буквы.");

puts("Введите 5, чтобы вывести текущий текст.");

puts("Введите 6, чтобы узнать текущее количество предложений в тексте.");

puts("Введите любой отличный символ, чтобы выйти.");

scanf("%d", &cmd);

return (cmd);

}

int Garbage(char\*\*\* text, int num)

{

int count = 0;

char garbage\_str[8] = "garbage";

for (int i = 0; i < num; i++)

{

count = 0;

int letter = 0;

int j;

while ((\*text)[i][letter] != '\0'){

if (tolower((\*text)[i][letter]) == garbage\_str[0]){

j = 0;

while (garbage\_str[j] == tolower((\*text)[i][letter + j]))

j++;

if (j == 7 || j == 8)

count++;

}

letter++;

}

printf("%s. ---> ", (\*text)[i]);

if (count > 5)

printf("It is a dump\n");

else if (count == 0)

printf("Clean\n");

else

printf("Must be washed\n");

}

return num;

}

int Replace(char\*\*\* text, int num)

{

int count = 0, symbol = 0;

int j, k;

int current = 0;

char buffer[100];

int max = SYMBOL \* 3;

for (int i = 0; i < num; i++)

{

symbol = 0;

while ((\*text)[i][symbol] != '\0')

{

if (isdigit((\*text)[i][symbol]))

count++;

symbol++;

}

}

if (count == 0)

printf("В предложениях отсутствуют цифры\n");

else

{

printf("Введите строку:\n");

char c = getchar();

c = getchar();

for (int j = 0; c != '\n'; j++){

buffer[j] = c;

c = getchar();

}

int p;

for (int i = 0; i < num; i++)

{

symbol = 0;

while ((\*text)[i][symbol] != '\0')

{

if (isdigit((\*text)[i][symbol]))

{

p = 0;

int len = strlen(buffer);

int lenstr = strlen((\*text)[i]);

(\*text)[i] = (char\*)realloc((\*text)[i], (lenstr + len) \* sizeof(char));

for (j = lenstr; j >= symbol + 1; j--) {

(\*text)[i][j + len - 2] = (\*text)[i][j - 1];

}

for (k = symbol; k < symbol + len; k++)

{

(\*text)[i][k] = buffer[p];

p++;

}

symbol += len - 1;

}

symbol++;

}

}

}

return num;

}

int Delete(char\*\*\* text, int num)

{

int flag = 0, symbol = 0;

for (int i = 0; i < num; i++)

{

symbol = 0;

flag = 0;

while ((\*text)[i][symbol] != '\0')

{

if (isupper((\*text)[i][symbol]))

flag++;

if ((isupper((\*text)[i][symbol + 1])) && (flag == 3))

continue;

if (flag == 3)

{

for (int h = i; h < num - 1; h++)

(\*text)[h] = (\*text)[h+1];

num--;

i--;

flag = 0;

break;

}

if ((isblank((\*text)[i][symbol + 1])) || (islower((\*text)[i][symbol + 1])) || (ispunct((\*text)[i][symbol + 1])) || (isdigit((\*text)[i][symbol + 1])))

flag = 0;

symbol++;

}

}

if (num == 0)

puts("Удалены все предложения");

else

for (int i = 0; i < num; i++)

printf("%s.\n", (\*text)[i]);

return num;

}

int cmp(const void\*a, const void \*b)

{

int counter\_for\_s = 0, counter\_for\_p = 0;

char\* str1 = \*((char\*\*)a), \*str2 = \*((char\*\*)b);

for (int i = 0; i < strlen(str1); i++)

{

if (i != 0)

{

if (((tolower(str1[i]) == 'a') || (tolower(str1[i]) == 'e') || (tolower(str1[i]) == 'i') || (tolower(str1[i]) == 'o') || (tolower(str1[i]) == 'u') || (tolower(str1[i]) == 'y')) && ((str1[i - 1] == ' ') || (str1[i - 1] == ',')))

counter\_for\_s++;

}

else

{

if ((((tolower(str1[i])) == 'a') || (tolower(str1[i]) == 'e') || (tolower(str1[i]) == 'i') || (tolower(str1[i]) == 'o') || (tolower(str1[i]) == 'u') || (tolower(str1[i]) == 'y')))

counter\_for\_s++;

}

}

for (int i = 0; i < strlen(str2); i++)

{

if (i != 0)

{

if (((tolower(str2[i]) == 'a') || (tolower(str2[i]) == 'e') || (tolower(str2[i]) == 'i') || (tolower(str2[i]) == 'o') || (tolower(str2[i])== 'u') || (tolower(str2[i]) == 'y')) && ((str2[i - 1] == ' ') || (str2[i - 1] == ',')))

counter\_for\_p++;

}

else

{

if (((tolower(str2[i]) == 'a') || (tolower(str2[i]) == 'e') || (tolower(str2[i]) == 'i') || (tolower(str2[i]) == 'o') || (tolower(str2[i]) == 'u') || (tolower(str2[i]) == 'y')))

counter\_for\_p++;

}

}

if (counter\_for\_s > counter\_for\_p)

return -1;

else if (counter\_for\_s < counter\_for\_p)

return 1;

else

return 0;

}

int Sort(char\*\*\* text, int num)

{

qsort(\*text, num, sizeof(char\*), cmp);

for (int i = 0; i < num; i++)

printf("%s.\n", (\*text)[i]);

return num;

}