**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

**Тема: Использование указателей**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Чайка К.В. |

Санкт-Петербург

2021

## Цель работы.

## Освоение работы с указателями и динамической памятью.

## Задание.

## Напишите программу, которая форматирует некоторый текст и выводит результат на консоль.

## На вход программе подается текст, который заканчивается предложением "Dragon flew away!".

## Предложение (кроме последнего) может заканчиваться на:

## . (точка)

## ; (точка с запятой)

## ? (вопросительный знак)

## Программа должна изменить и вывести текст следующим образом:

## Каждое предложение должно начинаться с новой строки.

## Табуляция в начале предложения должна быть удалена.

## Все предложения, в которых есть цифра 7 (в любом месте, в том числе внутри слова), должны быть удалены.

## Текст должен заканчиваться фразой "Количество предложений до n и количество предложений после m", где n - количество предложений в изначальном тексте (без учета терминального предложения "Dragon flew away!") и m - количество предложений в отформатированном тексте (без учета предложения про количество из данного пункта).

## \* Порядок предложений не должен меняться

## \* Статически выделять память под текст нельзя

## \* Пробел между предложениями является разделителем, а не частью какого-то предложения.

## Выполнение работы.

В начале программы объявлены стандартные библиотеки языка С, необходимые для работы кода: *#include <stdio.h>, #include <stdlib.h> и #include <string.h>.*

Далее объявлены две функции #*define*, позволяющие определить макроопределения: благодаря первой везде в коде, где написано слово *size,* в результате работы препроцессора это значение будет заменено на *5*, во второй *end\_sent* будет заменяться на *“Dragon flew away!”*.

В главной функции *int main()* объявлен двумерный массив *text*, в котором хранится результат работы функции *readText()* – введённый с клавиатуры текст. В целочисленной переменой *count\_before* хранится результат работы функции *countSent()*, на вход которой поступает исходный текст, а результатом работы является количество предложений этого же текста. Двумерный массив *text* подвергается изменением и принимает новое значение – результат работы функции *RightText()*, на вход которой поступает первоначальный текст и количество предложений данного текста, результатом работы функции является измененный текст с учетом требований задач лабораторной работы. В целочисленной переменной *count\_after* хранится результат работы функции *countSent()*, на вход поступает уже измененный текст, результат работы – количество предложений в нем. С помощью цикла *for* на экран выводится отредактированный текст с применением функции *printf()*, где каждое предложение выводится с новой строки. Отдельно происходит вывод предложения, говорящего об окончании текста. С помощью функции *printf()* выводятся переменные *count\_before* и *count\_after* – количество предложений до и количество предложений после соответственно. В конце с помощью функции *freeText()*, принимающей на вход *text* и *count\_before+1*, происходит освобождение раннее выделенной динамической памяти.

**Функции:**

Функция *char\*\* ReadText()*. В функции объявлены: целочисленная переменная *sent\_cnt = 0*, отвечающая за количество предложений в вводимом тексте, двумерный массив *char \*\*txt*, в котором будет хранится текст, и для него выделяется динамическая память *malloc(size \* sizeof(char\*))*, символьная переменная *symb*. С помощью цикла *do {} while*, пока проверка с помощью функции *strcmp(txt[sent\_cnt-1], end\_sent)* не покажет, что последнее предложение идентично предложению *Dragon flew away!* в цикле происходит считывание текста. Если выделенной раньше для двумерного массива памяти не хватает, происходит увеличение *txt = realloc(txt, (sent\_cnt + 1) \* sizeof(char\*))*. Для предложения *txt[sent\_cnt]* динамически выделяем память *malloc(size \* sizeof(char))*, объявляется целочисленная переменная *symb\_cnt = 0*, отвечающая за количество символов в предложении. С помощью функции *scanf (" ")*, считывается пробел в начале предложения, если он есть – это позволяет удалить табуляцию для корректного вывода конечного результата текста. В этом же цикле(внешнем) начинает работать другой цикл(внутренний) *do {} while*, до тех пор пока введённый символ не будет одним из *((symb != ';') && (symb != '.') && (symb != '!') && (symb != '?'))*. С помощью функции *scanf()* считывается символ, для предложения происходит увеличение памяти в случае, если выделенной раннее будет не достаточно *txt[sent\_cnt] = realloc(txt[sent\_cnt], (symb\_cnt + 1) \* sizeof(char))*. Значению двумерного массива присваивается символ *txt[sent\_cnt][symb\_cnt] = symb*, переменная отвечающая за количество слов в предложении увеличивается *symb\_cnt += 1*. После завершения работы внутреннего цикла, во внешнем происходит изменение памяти под конкретный объем равный количеству символов + 1(учитывается обязательный символ строк) *txt[sent\_cnt] = realloc(txt[sent\_cnt], (symb\_cnt + 1) \* sizeof(char))*, последнему значению присваивается этот символ *txt[sent\_cnt][symb\_cnt] = '\0'*. Переменная, отвечающая за количество предложений в тексте, увеличивается *sent\_cnt += 1*. По окончании работы внешнего цикла функция возвращается двумерный массив *return txt*, в котором хранится введённый с клавиатуры текст.

Функция *int countSent(char \*\*txt)*. Функция принимает на вход двумерный массив – введённый текст и считает количество предложений в нем. Объявляется целочисленная переменная *cnt\_bef = 0*. С помощью цикла *while()* до тех пор пока предложение с индексом количества предложений не идентично с *Dragon flew away!*(это проверяет функция *strcmp()*)переменная *cnt\_bef* увеличивается. По завершению работы цикла возвращается значение равное количеству предложений в исходном тексте *return cnt\_bef.*

Функция *char \*\*rightText(char \*\*txt, int sent\_bef)*, принимает на вход количество предложений в исходном тексте и сам текст, который в ходе работы функции будет отредактирован учитывая задачи лабораторной работы. Объявляется целочисленная переменная *sent\_aft = 0*, которая используется в цикле *while()*, который работает до тех пора пока количество этих предложений не равно конечному *sent\_aft != sent\_bef*. В цикле объявляется целочисленная переменная *seven = 0*, которая отвечает за нахождение *7* в предложениях текста. С помощью цикла *for (int i = 0; i < strlen(txt[sent\_aft]); i++)* осуществляется проход по всем предложениям и символам каждого предложения. Проверяется условие на наличие символов 7 в тексте *if (txt[sent\_aft][i] == '7')*. Если оно выполняется, то переменная *seven += 1* увеличивается. При выходе из цикла *for* если значение переменной *if (seven > 0)*, то память, выделенная для этого предложения, очищается *free (txt[sent\_aft])* и происходит с помощью цикла for сдвиг предложений *for (int i = sent\_aft ; i < sent\_bef - 1; i++)* *txt[i] = txt[i+1]*. Иначе происходит переход к следующему предложению *sent\_aft += 1*. По завершению работы цикла программа возвращает исправленный в соответствии с заданием текст *return txt.*

Функция *void freeText(char\*\* txt, int cnt\_bef).* Принимает на вход текст и количество предложений в нем. С помощью цикла *for(int i = 0; i < cnt\_bef; i++)* происходит освобождение динамически выделенной памяти на каждое предложение *free(txt[i])*. В конце очищается память выделенная на весь текст динамически *free(txt)*.

Разработанный программный код см. в приложении А.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | Mama live in Mos7cow. I love pizza. We eat burger. Dragon flew away! | I love pizza.  We eat burger.  Dragon flew away!  Количество предложений до 3 и количество предложений после 2 | Корректный результат работы программы. |
|  | 7. Kupi i pomogi. Dragon flew away! | Kupi i pomogi.  Dragon flew away!  Количество предложений до 2 и количество предложений после 1 | Корректный результат работы программы. |
|  | Feels like me. Koko chanel. Mother. Dragon flew away! | Feels like me.  Koko chanel.  Mother.  Dragon flew away!  Количество предложений до 3 и количество предложений после 3 | Корректный результат работы программы. |

## Выводы.

## Произошло ознакомление с понятием «указатель» и их дальнейшее использование в языке С. Изучены способы работы с динамической памятью в языке C. Была написана программа, считывающая и записывающая текст в динамический массив строк и обрабатывающая его в соответствии с требованиями задач лабораторной работы.

# Приложение А Исходный код программы

Название файла: Chernyakova\_Valeria\_lb3.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define size 5

#define end\_sent "Dragon flew away!"

char\*\* readText(){

int sent\_cnt = 0;

char \*\*txt = malloc(size \* sizeof(char\*));

char symb;

do{

txt = realloc(txt, (sent\_cnt + 1) \* sizeof(char\*));

txt[sent\_cnt] = malloc(size \* sizeof(char));

int symb\_cnt = 0;

scanf (" ");

do{

scanf("%c", &symb);

txt[sent\_cnt] = realloc(txt[sent\_cnt], (symb\_cnt + 1) \* sizeof(char));

txt[sent\_cnt][symb\_cnt] = symb;

symb\_cnt += 1;

}while((symb != ';') && (symb != '.') && (symb != '!') && (symb != '?'));

txt[sent\_cnt] = realloc(txt[sent\_cnt], (symb\_cnt + 1) \* sizeof(char));

txt[sent\_cnt][symb\_cnt] = '\0';

sent\_cnt += 1;

}while(strcmp(txt[sent\_cnt-1], end\_sent) != 0);

return txt;

}

int countSent(char \*\*txt){

int cnt\_bef = 0;

while (strcmp(txt[cnt\_bef], end\_sent) != 0){

cnt\_bef+=1;

}

return cnt\_bef;

}

char \*\*rightText(char \*\*txt, int sent\_bef){

int sent\_aft = 0;

while ( sent\_aft != sent\_bef){

int seven = 0;

for (int i = 0; i < strlen(txt[sent\_aft]); i++ ){

if (txt[sent\_aft][i] == '7')

seven += 1;

}

if (seven > 0){

free (txt[sent\_aft]);

for (int i = sent\_aft ; i < sent\_bef - 1; i++)

txt[i] = txt[i+1];

}

else

sent\_aft += 1;

}

return txt;

}

void freeText(char\*\* txt, int cnt\_bef){

for(int i = 0; i < cnt\_bef; i++)

free(txt[i]);

free(txt);

}

int main(){

char \*\*text = readText();

int count\_before = countSent(text);

text = rightText(text, count\_before+1);

int count\_after = countSent(text);

for(int i = 0; i < count\_after; i++)

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

printf ("%s\n", end\_sent);

printf("Количество предложений до %d и количество предложений после %d\n", count\_before, count\_after);

freeText(text, count\_after+1);

return 0;

}