**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Программирование»**

**Тема: Использование указателей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Сергеенков М.Ю. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

**Цель работы**

Научиться использовать указатели и работать с динамической памятью, написать программу для форматирования текста.

**Ход работы**

1. В локальном репозитории создана новая ветка *sergeenkov\_lr3*

git checkout -b sergeenkov\_lr3

1. Создана папка *sergeenkov\_lr3*

mkdir sergeenkov\_lr3

1. В вышеуказанной папке создан файл *lab3.cpp*

touch lab3.cpp

1. В файл *lab3.cpp* записан алгоритм по нижеуказанному заданию. 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:

* Все предложения, которые заканчиваются на '?' должны быть удалены.
* Каждое предложение должно начинаться с новой строки.
* Табуляция в начале предложения должна быть удалена.
* 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).

1. Далее описана реализация алгоритма.

Так как для выделения памяти необходимо знать количество вводимых символов, которое заведомо не известно в поставленной задаче, было принято решение в качестве буфера обмена использовать связанный список. Каждый новый символ помещается в список с помощью функции *push()*, при этом наращивается счетчик. После этого выделяется память под весь массив, затем символы по очереди извлекаются из списка с помощью функции *pop()* и кладутся в массив.

При извлечении элемента из списка память, занимаемая им, освобождается. Далее происходит непосредственно форматирование текста по заданию с помощью функции *formatText()*, которая получает на вход количество предложений в тексте,сам текст и количество символов.

Ниже приведен код программы.

#include <stdio.h>

#include <stdlib.h>

/\*Будем использовать связный список для

\*\*хранения вводимых символов

\*/

typedef struct Node{

char value;

struct Node \*next;

} Node;

void push(Node \*\*head, char data);

char pop(Node \*\*head);

int countSentences(char \*text, int count);

void formatText(char \*text, int numberOfCharacters, int sentences);

int main(int argc, char const \*argv[])

{

char inputChar;

char character;

int numberOfCharacters;

int i;

char\* charactersArray;

int numberOfSentences;

Node \*head=NULL;

while((inputChar=getchar())!=EOF){

push(&head, inputChar);

++numberOfCharacters;

if (inputChar=='!'){

break;

}

}

charactersArray=(char\*)malloc(numberOfCharacters\*sizeof(char));

for (i=(numberOfCharacters-1); i>=0; --i){

charactersArray[i]=pop(&head);

}

numberOfSentences=countSentences(charactersArray, numberOfCharacters);

/\*Теперь мы имеем все предложения в одном массиве

\*\*Можем начать форматирование текста

\*/

formatText(charactersArray, numberOfCharacters, numberOfSentences);

return 0;

}

/\*Функция для помещения нового элемента в связный список

\*/

void push(Node \*\*head, char data) {

Node \*tmp = (Node\*)malloc(sizeof(Node));

tmp->value = data;

tmp->next = (\*head);

(\*head) = tmp;

}

/\*Функция для извлечения элемента из связного списка

\*/

char pop(Node \*\*head) {

Node\* prev = NULL;

int val;

if (head!=NULL) {

prev = (\*head);

val = prev->value;

(\*head) = (\*head)->next;

free(prev);

return val;

}

}

/\*Функция для подсчета предложений в тексте

\*/

int countSentences(char \*text, int count){

int sentences=0;

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

{

if ((text[i]==';')||(text[i]=='?')||(text[i]=='.')||(text[i]=='!')){

++sentences;

}

}

return sentences;

}

/\*Функция для форматирования текста

\*/

void formatText(char \*text, int numberOfCharacters, int sentences){

int lengthOfSentence[sentences];

int i;

int j=0;

int m=0;

int n=0;

int k=0;

int countSentencesAfter=0; //количество предложений после форматирования

int flag=0; //Флаг поднят, когда находимся в предложении

for (i = 0; i < sentences; ++i)

{

lengthOfSentence[i]=0;

}

//Выделяем память под строки

char \*\*sentence=(char\*\*)malloc(sentences\*sizeof(char\*));

//подсчитываем количество символов в каждом предложении

for (i= 0; i < numberOfCharacters; ++i)

{

if ((text[i]==';')||(text[i]=='?')||(text[i]=='.')||(text[i]=='!')){

flag=0;

++j;

}

else{

if (flag==0){

flag=1;

}

if (flag==1){

++lengthOfSentence[j];

}

}

}

//Выделяем память под каждое предложение

for (i=0; i < sentences; ++i){

sentence[i]=(char\*)malloc((lengthOfSentence[i])\*sizeof(char));

}

flag=0;

for (i = 0; i < sentences; ++i)

{

for (j = 0; j <= lengthOfSentence[i] ; ++j)

{ if (k<numberOfCharacters){

sentence[m][n]=text[k];

++n;

if ((text[k]==';')||(text[k]=='?')||(text[k]=='.')||(text[k]=='!')){

if ((text[k+1]==' ')||(text[k+1]=='\t')||(text[k+1]=='\n')){

--n;

}

++n;

sentence[m][n]='\0';

++m;

n=0;

++k;

}

++k;

}

}

}

//вывод на консоль массива строк

for (i=0; i < sentences; ++i){

if ((sentence[i][lengthOfSentence[i]-1]!='?')&&(sentence[i][lengthOfSentence[i]]!='?')&&(sentence[i][lengthOfSentence[i]-1]!='!')){

++countSentencesAfter;

for (int j = 0; j <=lengthOfSentence[i]; ++j)

{

if ((sentence[i][j]!='\t')&&(sentence[i][j]!='\n')&&(sentence[i][j]!='\0'))

putchar(sentence[i][j]);

}

putchar('\n');

}

}

printf("Количество предложений до %d и количество предложений после %d\n", sentences-1, countSentencesAfter);

}

1. Проиндексированы изменения в локальном репозитории.

git status

git add sergeenkov\_lr2

1. Сделан коммит в локальный репозиторий.

git commit -m “add menu.c”

1. Изменения отправлены в удаленный репозиторий.

git push origin sergeenkov\_lr2

1. Сделан *pull request* в ветку master.

**Вывод**

В ходе лабораторной работы получены навыки работы с памятью и указателями, реализован алгоритм для обработки текста.