МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текстовых данных в си

Студент гр. 1304 _____ Сенников К.Д.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Сенников к.д.

Группа 1304

Тема работы: обработка текстовых данных в си

Исходные данные:

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.)

Содержание пояснительной записки:

Введение.

Основные теоретические положения.

Реализация программы.

Заключение.

Список используемых источников.

Предполагаемый объем пояснительной записки: Не менее 10 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 18.12.2021

Дата защиты реферата: 20.12.2021

Студент гр. 1304 _____ Сенников К.Д.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Создана программа принимающая на вход текст, сохраняющийся в динамический массив. Текст преобразовывается в соответствии с заданием. Пользователю предлагается выбрать пункт, для обработки текста, после выбора программа обрабатывает текст в соответствии с выбранным пунктом. При вводе пользователем нуля программа завершается.

СОДЕРЖАНИЕ

	Введение	5
1.	Описание кода программы	6
1.1.	Функция считывания текста	6
1.2	Функция замены всех гласных	9
1.3	Функция для нахождения необходимой подстроки	9
1.4	Функция удаления необходимых предложений	12
1.5	Функция вывода текста	14
1.6	Функция сортировки	14
1.7	Функция вывода подстрок для пункта 2	14
1.8	Функция main	15
2.	Тестирование	17
	Заключение	18
	Список использованных источников	19
	ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ	20

ВВЕДЕНИЕ

Целью данной работы является создание функций, занимающихся обработкой строк согласно заданию. Для достижения поставленной цели требуется решить следующие задачи: изучить управляющие конструкции языка Си, изучить использования указателей и массивов в языке Си, изучить особенности взаимодействия с динамической памятью, подробнее изучить функции стандартной библиотеки.

Требуется реализовать следующие функции:

1. Заменить в тексте все гласные буквы на следующую букву в алфавите.
2. Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше. Например, для предложения “To sleep or not to work” необходимо вывести “sleep”.
3. Удалить все предложения у которых длина первого слова равняется 4.
4. Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.

1. ОПИСАНИЕ КОДА ПРОГРАММЫ

1.1. Функция считывания текста

Функция readtext посимвольно считывает предложения в указатель word, затем объединяет все предложения в массиве указателей text. Ввод функция заканчивает при поступлении символа переноса строки. Перед добавлением предложения в массив функция посимвольно проверяет это предложение с уже имеющимися без учета регистра. Если подобное предложение уже имеется, функция не включает его в массив. В конце функция возвращает получившийся текст в виде массива указателей.

```
char** readtext(int *pN){
    char **text;
    char *word;
    int countword=0;
    int countsymbol=0;
    char symbol;
    int i;
    int j;
    int flag=0;
    int *size=NULL;
    text=NULL;
    printf("Введите текст\n");
    do{
        word=NULL;
        size=realloc(size,(countword+1)*sizeof(int));
        text=realloc(text,(countword+1)*sizeof(char*));
        if (size==NULL || text==NULL){
```

```
        printf("Память не выделена\n");
        return NULL;
}
countsymbol=0;
do{
        scanf("%c",&symbol);
        word=realloc(word,(countsymbol+1)*sizeof(char));
        if (word==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
        countsymbol++;
        word[countsymbol-1]=symbol;
}while (symbol!='.' && symbol!='\n');
if (symbol=='\n'){
        break;
}
word=realloc(word,(countsymbol+1)*sizeof(char));
if (word==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
word[countsymbol]='\0';
if (countword!=0){
        for (i=0;i<countword;i+=1){
                flag=1;
                if (size[i]==strlen(word)){
                        for (j=0;j<size[i];j++){
                                if ((int)text[i][j]!=(int)word[j]){
```

```
                                        if ((int)word[j]>64 && (int)word[j]<123
&& (int)text[i][j]>64 && (int)text[i][j]<123){
                                                if (abs((int)word[j]-(int)text[i][j])!
=32){
                                                    flag=0;
                                                }
                                            }
                                            else{
                                                flag=0;
                                            }
                                        }
                                    }
                                    else{
                                        flag=0;
                                    }
                                    if (flag==1){
                                        break;
                                    }
                                }

                            }
                            if (flag==0 || countword==0){
                                text[countword]=word;
                                size[countword]=strlen(word);
                                countword+=1;
                            }
                        }while (word[countsymbol-1]!='\n');
                        *pN=countword;
```

```
    return text;
}
```

1.2 Функция замены всех гласных

На вход функция получает текст и его размер. Далее функция посимвольно просматривает каждое предложение и находит гласную. При нахождении функция заменяет её на следующий символ по таблице аски. В конце функция возвращает получившийся текст.

```
char** replace(char** text, int count){
    int i;
    int j;
    char *str="AaIiOoUuEeYy";
    for (i=0;i<count;i++){
        for (j=0;j<strlen(text[i]);j++){
            if(strrchr(str,text[i][j])){
                text[i][j]=(char)((int)text[i][j]+1);
            }
        }
    }
    return text;
}
```

1.3 Функция для нахождения необходимой подстроки

Функция проверяет наличие в предложений необходимых подстрок: «To », « or not to », и сравнивает слова находящиеся между ними. Слово максимальной длины функция заносит в отдельный массив для последующего выведения всех подстрок отдельно от текста.

```
char** substring(char **text, int count,int *psub){
```

```
char **substr=NULL;
int i;
int j;
int g=0;
char *str1="To ";
char *str2=" or not to ";
for (i=0;i<count;i++){
	if (strstr(text[i],str1) && strstr(text[i],str2)){
		int v1=3;
		int v2=strstr(text[i],str2)-text[i]+11;
		int len1=strstr(text[i],str2)-text[i]-3;
		int len2=strlen(text[i])-(v2+1);
		int k=0;
		if (len1>len2){
			k=v1;
			int h=strlen(text[i])+len1;
			substr=realloc(substr,(g+1)*sizeof(char*));
			if (substr==NULL){
				printf("Память не выделена\n");
				return NULL;
			}
			for (j=0;j<len1;j++){
				substr[g]=realloc(substr[g],(k-2)*sizeof(char));
				if (substr[g]==NULL){
					printf("Память не выделена\n");
					return NULL;
				}
				substr[g][j]=text[i][k];
				k++;
```

```
        }
        substr[g]=realloc(substr[g],(k-2)*sizeof(char));
        if (substr[g]==NULL){
            printf("Память не выделена\n");
            return NULL;
        }
        substr[g][j+1]='\0';
    }
    else{
        k=v2;
        int h=strlen(text[i])+len2;
        substr=realloc(substr,(g+1)*sizeof(char*));
        if (substr==NULL){
            printf("Память не выделена\n");
            return NULL;
        }
        for (j=0;j<len2;j++){
            substr[g]=realloc(substr[g],(k-2)*sizeof(char));
            if (substr[g]==NULL){
                printf("Память не выделена\n");
                return NULL;
            }
        for (j=0;j<len2;j++){
            substr[g]=realloc(substr[g],(k-2)*sizeof(char));
            if (substr[g]==NULL){
                printf("Память не выделена\n");
                return NULL;
            }
            substr[g][j]=text[i][k];
```

```
                k++;
            }
            substr[g]=realloc(substr[g],(k-2)*sizeof(char));
            if (substr[g]==NULL){
                printf("Память не выделена\n");
                return NULL;
            }
            substr[g][j+1]='\0';
        }
        g++;
    }
}
*psub=g;
return substr;
}
```

1.4 Функция удаления необходимых предложений

Функция посимвольно просматривает первое слово каждого предложения и измеряет его длину. Если длина равна 4 функция удаляет это предложение путём смещения элементов массива указателей.

```
char** delete(char** text,int *pN){
    int i;
    int j;
    int len;
    int *index=NULL;
    int k=0;
    for (i=0;i<*pN;i++){
```

```
            len=0;
            for (j=0;j<strlen(text[i]);j++){
                if (text[i][j]!=' ' && text[i][j]!='.'){
                    len++;
                }
                else{
                    break;
                }
            }
            if (len==4){
                index=realloc(index,(k+1)*sizeof(int));
                if (index==NULL){
                    printf("Память не выделена\n");
                    return NULL;
                }
                index[k]=i-k;
                k++;
            }
        }
        for (i=0;i<k;i++){
            for (j=index[i];j<(*pN-1);j++)
                text[j]=text[j+1];
            free(text[*pN]);
            *pN-=1;
        }
        return text;
}
```

1.5 Функция вывода текста

Функция получает массив указателей и выводит его предложение за предложением.

```
void write(char **text,int N){
    for (int i=0;i<N;i++)
        printf("%s",text[i]);
    printf("\n");
}
```

1.6 Функция сортировки

Функция создана для функции qsort стандартной библиотеки. Она отвечает за сортировку по коду последнего символа предложения.

```
int sort(const void*a,const void*b){
    char *str1=*(char**)a;
    char *str2=*(char**)b;
    return (int)str1[strlen(str1)-2]-(int)str2[strlen(str2)-2];
}
```

1.7 Функция вывода подстрок для пункта 2

Функция выводит массив указателей строку за строкой.

```
void writesub(char **substr,int countsub){
    for (int i=0;i<countsub;i++)
        printf("%s\n",substr[i]);
}
```

1.8 Функция main

Функция отвечает за ввод пользователем необходимого пункта и вызов соответствующий функций. Определение соответствия между числом, введённым пользователем, и имеющихся функций происходит с помощью оператора switch.

```
int main(){
    int i;
    char **text;
    int *pN;
    int N=0;
    int fun=1;
    char **substr=NULL;
    int *psub;
    int countsub;
    psub=&countsub;
    pN=&N;
    text=readtext(pN);
    while (fun){
        printf("Введите необходимый номер подпункта. Введите 0 для выхода из программы\n");
        scanf("%d",&fun);
        switch(fun){
            case 1:
                text=replace(text,N);
                write(text,N);
                break;
            case 2:
                substr=substring(text,N,psub);
                write(text,N);
```

```
                    if (substr!=NULL){
                        writesub(substr,countsub);
                    }
                    break;
                case 3:
                    text=delete(text,pN);
                    write(text,N);
                    break;
                case 4:
                    qsort(text,N,sizeof(char*),sort);
                    write(text,N);
                    break;
                default:
                    break;
            }
        }
        for (i=0;i<N;i++)
            free(text[i]);
        free(text);
        return 0;
}
```

2. ТЕСТИРОВАНИЕ

Входные данные	Выходные данные	Комментарий
Erwe.Awe1e, ewqe.Erwe.Retqwe.erwe.To sleep or not to work.To rats or not to honest. 2 3 4 1 0	Введите текст Введите необходимый номер подпункта. Введите 0 для выхода из программы Erwe.Awe1e, ewqe.Retqwe.To sleep or not to work.To rats or not to honest. sleep honest Введите необходимый номер подпункта. Введите 0 для выхода из программы Awe1e, ewqe.Retqwe.To sleep or not to work.To rats or not to honest. Введите необходимый номер подпункта. Введите 0 для выхода из программы Awe1e, ewqe.Retqwe.To sleep or not to work.To rats or not to honest. Введите необходимый номер подпункта. Введите 0 для выхода из программы Bwf1f, fwqf.Rftqwf.Tp slffp pr npt tp wprk.Tp rbts pr npt tp hpnfst. Введите необходимый номер подпункта. Введите 0 для выхода из программы	Всё верно

ЗАКЛЮЧЕНИЕ

Были изучены возможности ввода текста. Изучена обработка множества строк, сортировка функцией qsort. Подробнее рассмотрена работа с динамической памятью, указателями, массивами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The C Programming Language / Brian W. Kernigan, Dennis M. Ritchie Second edition, 1988. 288 с.

2. The C++ Resources network [Электронный ресурс]

URL: http://cplusplus.com

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>


char** readtext(int *pN){
	char **text;
	char *word;
	int countword=0;
	int countsymbol=0;
	char symbol;
	int i;
	int j;
	int flag=0;
	int *size=NULL;
	text=NULL;
	printf("Введите текст\n");
	do{
		word=NULL;
		size=realloc(size,(countword+1)*sizeof(int));
		text=realloc(text,(countword+1)*sizeof(char*));
		if (size==NULL || text==NULL){
```

```
        printf("Память не выделена\n");
        return NULL;
}
countsymbol=0;
do{
        scanf("%c",&symbol);
        word=realloc(word,(countsymbol+1)*sizeof(char));
        if (word==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
        countsymbol++;
        word[countsymbol-1]=symbol;
}while (symbol!='.' && symbol!='\n');
if (symbol=='\n'){
        break;
}
word=realloc(word,(countsymbol+1)*sizeof(char));
if (word==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
word[countsymbol]='\0';
if (countword!=0){
        for (i=0;i<countword;i+=1){
                flag=1;
                if (size[i]==strlen(word)){
                        for (j=0;j<size[i];j++){
                                if ((int)text[i][j]!=(int)word[j]){
```

```
                                        if ((int)word[j]>64 && (int)word[j]<123
&& (int)text[i][j]>64 && (int)text[i][j]<123){
                                            if (abs((int)word[j]-(int)text[i][j])!
=32){
                                                flag=0;
                                            }
                                        }
                                        else{
                                            flag=0;
                                        }
                                    }
                                }
                            }
                            else{
                                flag=0;
                            }
                            if (flag==1){
                                break;
                            }
                        }

                    }
                    if (flag==0 || countword==0){
                        text[countword]=word;
                        size[countword]=strlen(word);
                        countword+=1;
                    }
                }while (word[countsymbol-1]!='\n');
```

```
	*pN=countword;
	return text;
}


char** replace(char** text, int count){
	int i;
	int j;
	char *str="AaIiOoUuEeYy";
	for (i=0;i<count;i++){
		for (j=0;j<strlen(text[i]);j++){
			if(strrchr(str,text[i][j])){
				text[i][j]=(char)((int)text[i][j]+1);
			}
		}
	}
	return text;
}


char** substring(char **text, int count,int *psub){
	char **substr=NULL;
	int i;
	int j;
	int g=0;
	char *str1="To ";
	char *str2=" or not to ";
	for (i=0;i<count;i++){
		if (strstr(text[i],str1) && strstr(text[i],str2)){
```

```
int v1=3;
int v2=strstr(text[i],str2)-text[i]+11;
int len1=strstr(text[i],str2)-text[i]-3;
int len2=strlen(text[i])-(v2+1);
int k=0;
if (len1>len2){
        k=v1;
        int h=strlen(text[i])+len1;
        substr=realloc(substr,(g+1)*sizeof(char*));
        if (substr==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
        for (j=0;j<len1;j++){
                substr[g]=realloc(substr[g],(k-2)*sizeof(char));
                if (substr[g]==NULL){
                        printf("Память не выделена\n");
                        return NULL;
                }
                substr[g][j]=text[i][k];
                k++;
        }
        substr[g]=realloc(substr[g],(k-2)*sizeof(char));
        if (substr[g]==NULL){
                printf("Память не выделена\n");
                return NULL;
        }
        substr[g][j+1]='\0';
}
```

```
            else{
                k=v2;
                int h=strlen(text[i])+len2;
                substr=realloc(substr,(g+1)*sizeof(char*));
                if (substr==NULL){
                    printf("Память не выделена\n");
                    return NULL;
                }
                for (j=0;j<len2;j++){
                    substr[g]=realloc(substr[g],(k-2)*sizeof(char));
                    if (substr[g]==NULL){
                        printf("Память не выделена\n");
                        return NULL;
                    }
                    substr[g][j]=text[i][k];
                    k++;
                }
                substr[g]=realloc(substr[g],(k-2)*sizeof(char));
                if (substr[g]==NULL){
                    printf("Память не выделена\n");
                    return NULL;
                }
                substr[g][j+1]='\0';
            }
            g++;
        }
}
*psub=g;
return substr;
```

```
}


char** delete(char** text,int *pN){
	int i;
	int j;
	int len;
	int *index=NULL;
	int k=0;
	for (i=0;i<*pN;i++){
		len=0;
		for (j=0;j<strlen(text[i]);j++){
			if (text[i][j]!=' ' && text[i][j]!='.'){
				len++;
			}
			else{
				break;
			}
		}
		if (len==4){
			index=realloc(index,(k+1)*sizeof(int));
			if (index==NULL){
				printf("Память не выделена\n");
				return NULL;
			}
			index[k]=i-k;			k++;
		}
	}
	for (i=0;i<k;i++){
```

```
            for (j=index[i];j<(*pN-1);j++)
                text[j]=text[j+1];
            free(text[*pN]);
            *pN-=1;
        }
        return text;
}


void write(char **text,int N){
        for (int i=0;i<N;i++)
            printf("%s",text[i]);
        printf("\n");
}


int sort(const void*a,const void*b){
        char *str1=*(char**)a;
        char *str2=*(char**)b;
        return (int)str1[strlen(str1)-2]-(int)str2[strlen(str2)-2];
}


void writesub(char **substr,int countsub){
        for (int i=0;i<countsub;i++)
            printf("%s\n",substr[i]);
}
int main(){
        int i;
```

```
char **text;
int *pN;
int N=0;
int fun=1;
char **substr=NULL;
int *psub;
int countsub;
psub=&countsub;
pN=&N;
text=readtext(pN);
while (fun){
 printf("Введите необходимый номер подпункта. Введите 0 для выхода из программы\n");
scanf("%d",&fun);
switch(fun){
case 1:
text=replace(text,N);                    write(text,N);
break;
case 2:
substr=substring(text,N,psub);
write(text,N);
if (substr!=NULL){
writesub(substr,countsub);
}
break;
case 3:
text=delete(text,pN);
write(text,N);
break;
```

```
                case 4:
                        qsort(text,N,sizeof(char*),sort);
                        write(text,N);
                        break;
                default:
                        break;
        }
    }
    for (i=0;i<N;i++)
        free(text[i]);
    free(text);
    return 0;
}
```