Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет
по лабораторной работе № 6

по дисциплине

«Низкоуровневое программирование»

на тему

«Строки и параметры запуска»

Выполнил:
студент гр. 143
Вербицкая И. С.

Проверил:
Антипов О.В.

Рязань 2022

Задание (вариант №5):

Задание

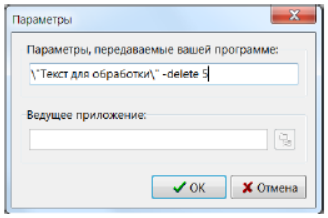
В каждом варианте задания необходимо создать программу, принимающую в качестве параметров запуска первым аргументом текст для обработки, вторым аргументом - команду обработки (одну из трёх), и после нее необходимые для работы команды дополнительные аргументы. Все команды обработки делятся на три вида: информация, создание, удаление (заданы ниже по-вариантно). Таким образом запуск программы должен иметь следующий формат:

```
Lab6.exe "Текст для обработки идёт первым аргументом." -info 5
```

Где второй аргумент задаёт команду, соответственно:
-info для команды «информация».
-create для команды «создание».
-delete для команды «удаление».
Третьим и далее аргументами идёт необходимый набор параметров для каждой из этих команд. В каждом варианте задания требуется создать минимум три функции реализующие соответствующие команды. При применении команды «информация» в консоль следует вывести искомое количество, при «создании» вывести созданный массив в консоль, а при «удалении» вывести в консоль модифицированный текст. При любом сравнении последовательностей не учитывать регистр букв. В команде «создание» использовать динамические массивы. Весь текстовый вывод в консоль должен осуществляться исключительно из основной функции **main**. Программа также должна правильно обрабатывать случай, когда аргументы запуска отсутствуют либо заданы неверно, и выводить текст ошибки, поясняющий что конкретно было сделано неправильно при задании параметров.

Важно: любой текст может состоять из *слов*, *чисел*, либо иных *последовательностей* символов. Также текст может быть разбит знаками препинания на предложения.
Слово – последовательность символов, состоящая только из букв верхнего или нижнего регистра.
Число – последовательность символов, состоящая из цифр 0…9. Числа считаются только целыми. Дробные числа следует относить к **иным последовательностям**.
Предложение в тексте может заканчиваться на символы: точка “.”, восклицательный знак “!”, вопросительный знак “?” или конец строки ‘\0’.
Следует также учитывать, что слова, числа и другие последовательности могут разделяться не только пробелами и знаками конца предложения, но и символом запятая “,”.

Примечание о параметрах запуска: если вы запускаете программу с параметрами запуска через среду Dev-C++ (выполнить -> параметры…) то текст в качестве первого аргумента следует взять в кавычки с обратным слешем:



Альтернативный вариант: для тестирования программы также можно создать в той же папке, что и сама программа, текстовый файл с расширением ***.bat** (т.е. пакетный или командный файл) и внести в него следующие содержание в любом текстовом редакторе:

```
Lab6.exe "Текст для обработки" -delete 5
pause
```

Таким образом, запуская данный пакетный файл, можно передавать приложению требуемые параметры для запуска. В пакетном файле обратный слеш перед кавычками ставить **не нужно**. Команда **pause** нужна чтобы консоль вывода не закрывалась сразу после окончания программы.

Вариант	Команды		
	Информация	**Создание**	**Удаление**
5	Функция, возвращающая максимальную длину слова, встречающуюся в тексте.	Функция, создающая массив чисел, меньше чем M.	Функция, которая удаляет каждое K-ое число из текста.

Анализ задания:

Входные данные:

- argc – количество аргументов;
- argv – массив аргументов:
  - argv[0] – путь к файлу
  - argv[1] – строка текста
  - argv[2] – требуемая команда
  - argv[3] – число-аргумент для команд создания и удаления

ОДЗ:

- argc>4 (для команд создания и удаления) / argc>3 (для команды информация);
- argv[1] – не пустая строка
- argv[2] – одна из возможных строк текста («-info», «-delete», «-create»)
- argv[3] – целое положительное число

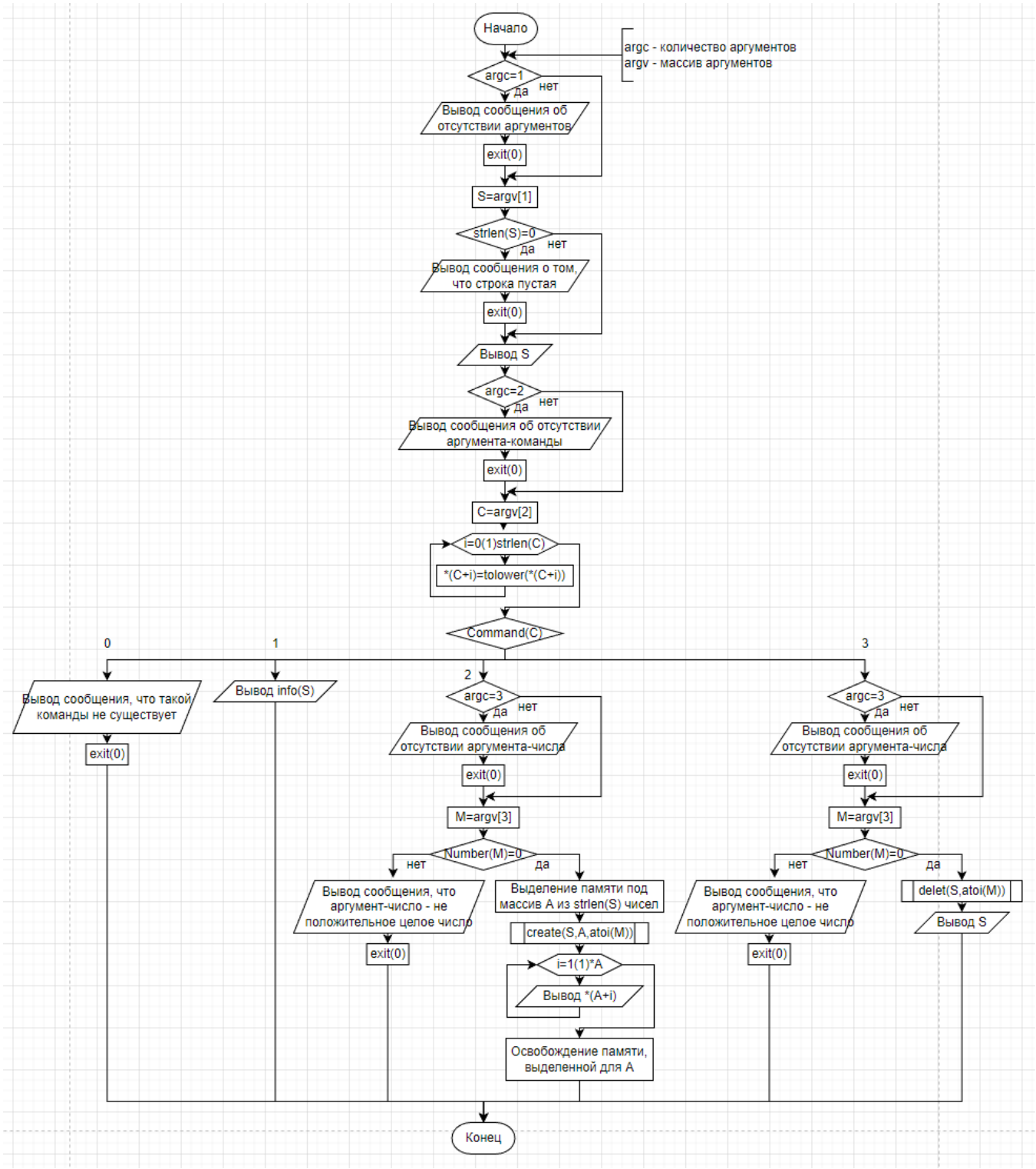
Выходные данные:

- S - строка текста до и после (с удаленным каждым K-ым словом) изменений;
- info(S) – значение функции, определяющей наибольшую длину слова в тексте;
- A – массив чисел, больших заданного числа M.

Этапы решения задачи:

1. Составить блок-схемы;
2. Составить программу;
3. Провести проверку работы программы.

Блок-схемы:


Начало
argc - количество аргументов
argv - массив аргументов
argc=1
нет
да
Вывод сообщения об отсутствии аргументов
exit(0)
S=argv[1]
strlen(S)=0
нет
да
Вывод сообщения о том, что строка пустая
exit(0)
Вывод S
argc=2
нет
да
Вывод сообщения об отсутствии аргумента-команды
exit(0)
C=argv[2]
i=0(1)strlen(C)
(C+i)=tolower((C+i))
Command(C)
0
1
2
3
Вывод сообщения, что такой команды не существует
exit(0)
Вывод info(S)
argc=3
нет
да
Вывод сообщения об отсутствии аргумента-числа
exit(0)
M=argv[3]
Number(M)=0
нет
да
Вывод сообщения, что аргумент-число - не положительное целое число
exit(0)
Выделение памяти под массив A из strlen(S) чисел
create(S,A,atoi(M))
i=1(1)*A
Вывод *(A+i)
Освобождение памяти, выделенной для A
argc=3
нет
да
Вывод сообщения об отсутствии аргумента-числа
exit(0)
M=argv[3]
Number(M)=0
нет
да
Вывод сообщения, что аргумент-число - не положительное целое число
exit(0)
delet(S,atoi(M))
Вывод S
Конец

Листинг программы:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <mem.h>
#include <ctype.h>
#include <string.h>
//функция, возвращающая ноль, если символ является маркером начала/конца слова/числа
int Marker(char S){
    int N=1;
    if ((S==' ')||(S=='.')||(S==',')||(S==';')||(S=='!')||(S=='?')||(S==':')||(S=='\"')||(S=='\0')) N=0;
    return N;}
//функция, возвращающая ноль, если последовательность является числом (целым положительным)
int Number(const char* S) {
    int i,N;
    N=0;
    for (i=0; i<strlen(S);i++)
        if (isdigit(*(S+i))==0){
            N=1;
        }
    if (N==0)
        if (atoi(S)==0) N=1;
    return N;}
//функция, возвращающая номером заданную команду
int Command(const char* S) {
    int C=0;
    if (strcmp(S,"-info")==0) C=1;
    if (strcmp(S,"-create")==0) C=2;
    if (strcmp(S,"-delete")==0) C=3;
    return C;}
/*int info(const char* S) {
    int k=0; //счетчик длины слова
    int MX=0; //максимальная длина
    int i;
    for (i=0; i<strlen(S); i++){
        if (isalpha(*(S+i))==0) { //при каждой встречей с не-буквой идет обнуление счетчика и проверка на максимум
            if (k>MX) MX=k;
            k=0;
        }
        else {  //с каждой буквой счетчик увеличивается
            k++;
            if (i==strlen(S)-1 && k>MX) MX=k;}}
    return MX;}*/
//ФУНКЦИЯ, ВОЗВРАЩАЮЩАЯ МАКСИМАЛЬНУЮ ДЛИНУ СЛОВА, ВСТРЕЧАЮЩУЮСЯ В ТЕКСТЕ
int Info(const char* S) {
    int n=0; //счетчик длины слова
    int MX=0; //максимальная длина
    int i;
    for (i=0; i<=strlen(S); i++)
    if (Marker(*(S+i))==0){ //если маркер границы слова
        if ((n!=0)&&(n>MX)) MX=n; //если до этого шло слово и его длина больше максимума
        n=0;
    }
    else {
        if (isalpha(*(S+i))!=0)
            if (n==0) {
                if (i==0) n++;
                else if (Marker(*(S+i-1))==0) n++;
                    else n=0;
            }
            else {
                n++;
            }
        else n=0; //не буква - счетчик обнуляется
    }
    return MX;
```

```
}
/*int create(const char* S, int* A, int M) {
    int i,j;
    int k=0; //количетво цифр
    int n=0; //длина цифры
    //char B[strlen(S)];
    char *B = (char*) malloc(strlen(S) * sizeof(char));
    for (i=0;i<strlen(S);i++)
        if (isdigit(*(S+i))==0) { //если не цифра
            if (n!=0) {
                for (j=0;j<n;j++)
                    *(B+j)=*(S+i-n+j);
                *(B+n)='\0';
                if (atoi(B)<M) {
                    k++;
                    *(A+k)=atoi(B);}}
            n=0;}
        else { n++;
            if (i==strlen(S)-1) {
                for (j=0;j<n;j++)
                    *(B+j)=*(S+i-n+j+1);
                *(B+n)='\0';
                if (atoi(B)<M) {
                    k++;
                    *(A+k)=atoi(B);}}}
    *A=k;
    A = (int*)realloc(A,(k+1) * sizeof(int));
    free(B);
    B=NULL;}*/
//ПОДПРОГРАММА, СОЗДАЮЩАЯ МАССИВ ЧИСЕЛ, МЕНЬШИХ ЧЕМ M
int Create(const char* S, int* A, int M) {
    int i,j;
    int k=0; //количетво цифр
    int n=0; //длина цифры
    char *B = (char*) malloc(strlen(S) * sizeof(char));// массив под текущую цифру
    if(!B) printf("Ошибка: недостаточно памяти"), exit(1);//
    for (i=0; i<=strlen(S); i++)
    if (Marker(*(S+i))==0){ //если маркер границы числа
        if (n!=0){
            for (j=0;j<n;j++)
                *(B+j)=*(S+i-n+j);
            *(B+n)='\0';
            if (atoi(B)<M){
                k++;
                *(A+k)=atoi(B);
                }
            }
        n=0;
    }
    else {
        if (isdigit(*(S+i))!=0)
            if (n==0) {
                if (i==0) n++;
                else if (Marker(*(S+i-1))==0) n++;
                    else n=0;
            }
            else n++;
        else n=0; //не цифра - счетчик обнуляется
    }
    *A=k;
    A = (int*)realloc(A,(k+1) * sizeof(int));//
    free(B); B=NULL;//
}
/*int delet(char* S, int M) {
    int i,j;
```

```
    int n=0; //счетчик длины текущего слова
    int k=0; //счетчик количества слов
    for (i=0;i<strlen(S);i++) {
        if (isalpha(*(S+i))==0) { //не буква
            if (n!=0) {
                k++;
                if (k%M==0)
                for (j=0;j<n;j++)
                    *(S+i+j-n)=' '; }
            n=0;}
        else {n++;
            if (i==strlen(S)-1) {
                k++;
                if (k%M==0)
                for (j=0;j<n;j++)
                    *(S+i+j-n+1)=' ';
    }   }   }   }*/
//ПОДПРОГРАММА УДАЛЕНИЯ КАЖДОГО К-ОГО СЛОВА В СТРОКЕ
int Delet(char* S, int M){
    int i,j;
    char* C=S; //дополнительная строчка, хранящая исходную
    int n=0; //длина текущего слова
    int k=0; //номер слова
    int Begin[strlen(S)]; //массив координат начал участков копирования в новую строку
    int End[strlen(S)]; //массив окончаний координат участков
    int w=0; //текущий элемент массивов координат
    Begin[0]=0;
    //пробегаемся по строке запоминая все вхождения
    for (i=0; i<=strlen(S); i++)
    if (Marker(*(S+i))==0){ //если маркер границы слова
        if (n!=0){ //если до этого шло слово
            k++;
            if (k%M==0){
                End[w]=i-n-1; //номер символа перед началом слова и конец участка копирования
                w++;
                Begin[w]=i; //конец слова и начало нового участка копирования
            }
        }
        n=0;
    }
    else {
        if (isalpha(*(S+i))!=0)
            if (n==0) {
                if (i==0) n++;
                else if (Marker(*(S+i-1))==0) n++;
                    else n=0;
            }
            else n++;
        else n=0;
    }
    End[w]=strlen(S); w++;
    k=0;
    for (i=0;i<w;i++)
        for (j=Begin[i];j<End[i];j++){
            *(S+k)=*(C+j);
            k++;
        }
    *(S+k)='\0';
}
//ОСНОВНАЯ ПРОГРАММА
int main(int argc, char** argv) {
    system("chcp 1251");
    if (argc==1) {
        printf("Ошибка: вы не задали входных параметров\n");
        exit(1);
```

```
    }
    char* S = argv[1];
    if (strlen(S)==0) {
        printf("Ошибка: заданная строка - пустая\n");
        exit(1);
    }
    printf("ИСХОДНЫЙ ТЕКСТ: \"%s\"\n", S);
    if (argc==2) {
        printf("Ошибка: вы не ввели команду вторым аргументом\n");
        exit(1);
    }
    char* C = argv[2];
    int i;
    for (i=0; i<strlen(C); i++)
        *(C+i)=tolower(*(C+i));

    switch (Command(C)) {
            case 1: {
                printf("КОМАНДА: Информация\n");
                printf("РЕЗУЛЬТАТ: Максимальная длина слова, встречающаяся в тексте: %d\n",Info(S));
            } break;
            case 2: {
                printf("КОМАНДА: Создание\n");
                if (argc==3) {
                    printf("Ошибка: вы не ввели число третьим аргументом\n");
                    exit(1);
                }
                char* M = argv[3];
                if (Number(M)==0) {
                    printf("РЕЗУЛЬТАТ: Массив чисел, меньших чем %d:\n",atoi(M));
                    int *A = (int*) malloc(strlen(S) * sizeof(int)); //
                    if(!A) printf("Ошибка: недостаточно памяти"), exit(1);//
                    Create(S,A,atoi(M));
                    for (i=1;i<=*A;i++)
                        printf("%*d\t",4,*(A+i));
                    free(A); A = NULL; //
                }
                else {
                    printf("Ошибка: заданный третий аргумент - не целое положительное число\n");
                    exit(1);
                }
            } break;
            case 3: {
                printf("КОМАНДА: Удаление\n");
                if (argc==3) {
                    printf("Ошибка: вы не ввели число третьим аргументом\n");
                    exit(1);
                }
                char* M = argv[3];
                if (Number(M)==0) {
                    printf("РЕЗУЛЬТАТ: Текст с каждым %d-м удалённым в нем словом:\n",atoi(M));
                    Delet(S,atoi(M));
                    printf("\"%s\"\n", S);
                }
                else {
                    printf("Ошибка: заданный третий аргумент - не целое положительное число\n");
                    exit(1);
                }
            } break;
            case 0: {
                printf("Ошибка: такой заданной  команды не существует\n");
                exit(1);
            } break;
    }
    return 0;}
```

Результаты работы программы и проверка:

Для проверки работы программа была запущена для одной и той же строки «I dont know what to print there sooooo 34 35$ 3h2 57 its_not_word 32 51 49 100» и трех различных команд (рисунок 1, 2, 3) с помощью параметров запуска. Во всех трех случаях программа справлялась с поставленной задачей, возвращая ожидаемый (в соответствии с заданной командой) результат. Программа работоспособна (рисунок 4, 5, 6).

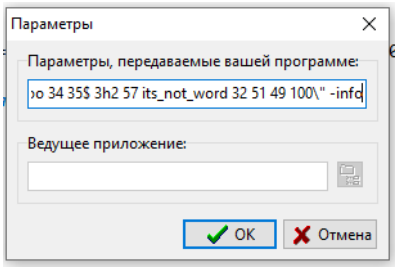


Рисунок 1

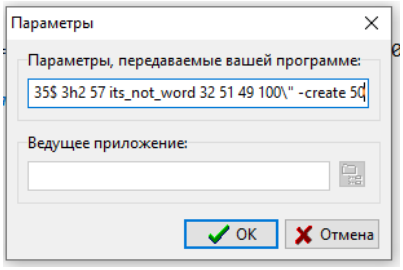


Рисунок 2

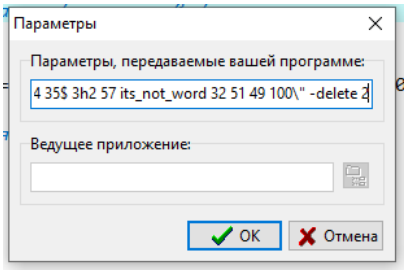


Рисунок 3

```
Текущая кодовая страница: 1251
ИСХОДНЫЙ ТЕКСТ: "I dont know what to print there sooooo 34 35$ 3h2 57 its_not_word 32 51 49 100"
КОМАНДА: Информация
РЕЗУЛЬТАТ: Максимальная длина слова, встречающаяся в тексте: 6

--------------------------------
Process exited after 0.04195 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 – результат работы команды информация

```
Текущая кодовая страница: 1251
ИСХОДНЫЙ ТЕКСТ: "I dont know what to print there sooooo 34 35$ 3h2 57 its_not_word 32 51 49 100"
КОМАНДА: Создание
РЕЗУЛЬТАТ: Массив чисел, меньших чем 50:
  34      32      49
--------------------------------
Process exited after 0.02946 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5 – результат работы команды создание

```
Текущая кодовая страница: 1251
ИСХОДНЫЙ ТЕКСТ: "I dont know what to print there sooooo 34 35$ 3h2 57 its_not_word 32 51 49 100"
КОМАНДА: Удаление
РЕЗУЛЬТАТ: Текст с каждым 2-м удалённым в нем словом:
"I know to there 34 35$ 3h2 57 its_not_word 32 51 49 100"

--------------------------------
Process exited after 0.03141 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6 – результат работы команды удаление