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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа со строками в языке СИ.

Студент гр. 1304 _____ Радионов Б. С

Преподаватель _____ Чайка К. В

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Радионов Б.С.

Группа 1304

Тема работы (проекта): работа со строками в языке СИ.

Исходные данные:

Текст, состоящий из предложений, разделенных точкой. Предложения, состоящие из слов, разделенных запятой или пробелом. Слова – набор латинских букв и цифр.

Содержание пояснительной записки:

«Содержание», «Введение», «Заключение», «Список использованных 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 _____ Радионов Б.С.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Требуется написать программу, которой подаётся текст, его необходимо сохранить в динамический массив и далее оперировать только с ним. Далее нужно выбрать, какое задание по обработке текста следует выполнить (для этого выводится подсказка на экран). После выбора задания, программа выполняет его. Так же, предусмотрен выход из программы.

СОДЕРЖАНИЕ

	Введение	5
1.	Описание программы	6
1.1.	Функция удаления дублированных строк	6
1.2.	Функция удаления подстроки “Robin”	7
1.3.	Функция сортировки массива	8
1.4.	Функция удаления всех предложений, где сумма цифр равна 13	9
1.5.	Функция вывода предложений в которых первое слово начинается с гласной или цифры	10
1.6.	Функция для поиска запятой	11
1.7.	Функция по замене “Robin” на “I am Batman”	11
1.8.	Функция для поиска символа гласной буквы	13
1.9.	Функция main	13
	Заключение	17
	Список использованных источников	18
	Приложение А. Исходный код	19
	Приложение Б. Результаты тестирования	20

ВВЕДЕНИЕ

Целью работы является работа с текстом и его обработка. Для этого используются стандартные библиотеки, указатели, динамические массивы.

Требуется реализовать следующие функции:

1. Заменить в тексте подстроки “Robin” (могут быть внутри слова) на подстроку “I am Batman”.
2. Отсортировать предложения по увеличению количества запятых в предложении.
3. Удалить все предложения в которых сумма цифр в предложении равняется 13.
4. Вывести все предложения в которых первое слово начинается с гласной буквы или цифры.

1. ОПИСАНИЕ ПРОГРАММЫ

1.1 Функция удаления дублированных строк

```
int DeleteDoubleString(char** Array,int maxCount)
{
  int duble_string = 0;
    for (int i = 0; i < maxCount; i++) {
        for (int j = 0; j < maxCount; j++) {
            if (i != j){
                if (strlen(Array[i]) == strlen(Array[j])){
                    if (strcasecmp(Array[i], Array[j]) == 0) {
                        for (int q = j; q < maxCount; q++) {
                            strcpy( Array[q] , Array[q+1]);
                        }
                        --maxCount;
                    }
                }
            }
        }
    }

    char** new_stringArray = (char**)realloc(Array, (maxCount) * sizeof(char*));

   if ( new_stringArray != NULL ) {
        Array = new_stringArray;
        return maxCount;
    } else {
        printf("Не создан массив new_stringArray \n");
        return -1 ;
    }
```

```
}
```

Создается цикл в цикле, на проверку дублей. После проверки на одну и ту же строку, сравниваем длины строк. Если длины совпали, сравниваем содержание строки безе регистра. Далее представлен цикл удаления дублированной строки с перемещением последующих строк на строку вверх. Массив пересоздается и в него копируются данные. Так же, выполняется проверка на создание нового массива.

1.2 Функция удаления подстроки Robin

```
int DeleteSubStringRobin(char** Array,int maxCount)
{
   int StartNumSubstr = 0;
   size_t sizeString;
   char *findSubtr,*OldSubString =  "Robin",*NewSubString = "I am
Batman",*Newstring = (char*)malloc(StartNumSubstr * sizeof(char));
   for (int i = 0; i <= maxCount ; i++) {

      sizeString = replaceSubstr(Array[i],Newstring, OldSubString, NewSubString);

      if ( strcasecmp(Array[i],Newstring)  != 0){
         strcpy(Array[i], Newstring);

      }
   }
}
```

Циклом перебираются все предложения главного массива сверху вниз, которые собраны в этом массиве и при помощи функции *sizeString = replaceSubstr(Array[i],Newstring, OldSubString, NewSubString);* замеянем “Robin” на “I am Batman”.

1.3 Функция сортировки массива

```
int SortArray(char** Array,int maxCount)
{
   char * Temp_String;
   for (int startIndex = 0; startIndex <= maxCount ; ++startIndex){
      int smallestIndex = startIndex;
      for (int currentIndex = startIndex + 1; currentIndex <= maxCount;
++currentIndex) {
         if (countZapToString(Array[currentIndex]) <
countZapToString(Array[smallestIndex]))
            smallestIndex = currentIndex;
      }
      Temp_String = Array[startIndex];
      Array[startIndex] = Array[smallestIndex];
      Array[smallestIndex] = Temp_String;
   }

}
```

Циклом перебираем каждый элемент массива (кроме последнего, он уже будет отсортирован к тому времени, когда мы до него доберемся). В переменной *smallestIndex* хранится индекс наименьшего значения, которое мы нашли в этой итерации. Начинаем с того, что наименьший элемент в этой итерации - это первый элемент (индекс 0), поэтому приравниваем *smallestIndex к startIndex*. Затем ищем элемент поменьше в остальной части массива. Если мы нашли элемент, который меньше нашего наименьшего элемента, то запоминаем его. *smallestIndex* становится наименьшим элементом. Меняем местами наше начальное наименьшее число с тем, которое мы обнаружили.

1.4 Функция удаления всех предложений, где сумма цифр равна 13

```
int DeleteSum13(char** Array,int maxCount)
{
   int NumberTemp = 0,Razrayd = 0;
   int sum = 0;
   for (int i = 0; i < maxCount; i++) {
      for (int j = 0; j < strlen(Array[i]); j++)
      {
         if (('1' <= Array[i][j]) && (Array[i][j]  <= '9')) {
            NumberTemp = atoi(&Array[i][j]);
            sum = sum + NumberTemp;
                     while(NumberTemp>10){
               NumberTemp = NumberTemp / 10;
               ++j;
            }

         }
      }
      if ( sum == 13){
                  for (int q = i; q < maxCount; q++)
            strcpy( Array[q] , Array[q+1]);
         --i;
         --maxCount;
      }
      sum = 0;
   }
   char** new_stringArray = (char**)realloc(Array, (maxCount) * sizeof(char*));
   if ( new_stringArray != NULL ) {
      Array = new_stringArray;
      return maxCount;
```

```
    } else {
        printf("Не создан массив new_stringArray \n");
        return -1 ;
    }
}
```

Циклом считаем сумму цифр в строке массива. Условием проверяем, что этот символ означает цифру 1 – 9. Определяем разрядности числа из строки . На сколько разрядов число больше 10 из-за особенности реализации функции atoi() Циклом двигаем указатель по строке. Если сумма равна 13, удаляем строки. Так как строки были удалены, уменьшаем итератор и уменьшаем размер массива на одну строку. Пересоздаем массив и копируем данные.

1.5 Функция вывода предложений в которых первое слово начинается с гласной или цифры

```
int PrintString(char** Array,int maxCount) //
{
    printf("\nРезультат:\n");
    for (int i = 0; i < maxCount; i++) {
        for (int j = 0; j < strlen(Array[i]); j++) {
            if (Array[i][j] != ' ') {
                if (('0' <= Array[i][j]) && (Array[i][j] <= '9') || glasnyChar(Array[i][j]))
{
                    printf("%s \n", Array[i]);
                }
                break;
            }
        }
    }
}
```

Перебираем строку. Условием и функцией glasnyChar, о которой будет написано далее, определяем, что первый символ это цифра 1 – 9 или гласная.

1.6 Функция для поиска запятой

```
int countZapToString(char *String)
{
   int count = 0;
   for (int i = 0; String[i] != '\0'; i++){
      if (String[i] == ',')
         count++;
   }
   return count;
}
```

Функция ищет символ запятая.

1.7 Функция по замене “Robin” на “ I am Batman”

```
size_t replaceSubstr(const char *srcString, char *dstString, const char
*oldSubString, const char *newSubString)
{
   size_t l_old = strlen(oldSubString), l_new = strlen(newSubString);
   size_t dst_length = 0;

   const char *src_current = srcString;
   do
   {
      const char *src_next = strstr(src_current, oldSubString);
      if (src_next == NULL)
      {
         if (dstString != NULL)
```

```
            strcpy(dstString + dst_length, src_current);
        dst_length += strlen(src_current);
        break;
    }
    size_t n = src_next - src_current;

    if (dstString != NULL)
        memcpy(dstString + dst_length, src_current, n);
    dst_length += n;
    if (dstString != NULL)
        memcpy(dstString + dst_length, newSubString, l_new);
    dst_length += l_new;
    src_current = src_next + l_old;
  } while (1);
  return dst_length;
}
```

Происходит поиск во входящей строке подстроки “Robin”. Если подстроки не найдено, то выходим из цикла. Ищется номер позиции подстроки “Robin” во входящей строке. Далее происходит копирование в исходящую строку данных из входящей строки только по позиции до слова Robin. Находится позиция, куда будет вставляться подстрока “I am Batman”. В бесконечном цикле копируется остаток входящей строки после ввода Robin для дальнейшего поиска Robin.

1.8 Функция для поиска символа гласной буквы

```
int glasnyChar(char n)
{
   int k = 0;
   char mas[12] = {'a', 'e', 'i', 'o', 'u', 'y', 'A', 'E', 'I', 'O', 'u', 'Y'};
   for (int i = 0; i < 12; i++)
   {
      if (n == mas[i])
         k++;
   }
   if (k == 1)
      return 1;
   else
      return 0;
}
```

Функция в цикле определяет символ гласной буквы. Если элемент – гласная, то k = 1 и возвращается, иначе возвращается 0.

1.9 Функция main

```
int main() {


   setlocale(LC_ALL, "Rus");


   char *iputString = (char*)malloc(MAX_SIZE_INPUT * sizeof(char*)),
subArray[MAX_SIZE_STRING], ** stringArray =
(char**)malloc(MAX_SIZE_MATRIX * sizeof(char*));


   int i=0, count=0, numberZadanie = 0,result = 0;
```

```
if ( stringArray == NULL ) {
   printf("Не создан массив stringArray \n");
   return -1;
}


printf( "Введите данные для обработки : " );


while ((subArray[i]  = getchar()) != '\n') {

   if (subArray[i]  == '.') {

      subArray[i]  = '\0';

      stringArray[count] = (char*)malloc(MAX_SIZE_MATRIX  * sizeof(char));

      strcpy(stringArray[count], subArray);

      count++;
      i = 0;
   }
   else {
      i++;

   }
}
```

```
    count =  DeleteDoubleString(stringArray,count - 1);

    if (count <0)
        printf("Массив не создан после сжатия \n");

    printf( "Выберите одно из сдедующих заданий \n" );
    printf( "1. Заменить в тексте подстроку \"Robin\" на \"I am Bathman\" \n" );
    printf( "2. Отсортировать предложения по количеству запятых \n" );
    printf( "3. Удалить все предложения в которых сумма цифр равняется 13\n"
);
    printf( "4. Вывести все предложения в которых первое слово начинается с
гласной буквы или цифры \n" );
    printf( "5. Выход . \n" );
    printf( "Ваш выбор: " );

    numberZadanie = (int)getchar();
    switch ( numberZadanie ) {
        case '1':
            printf("Выполняем задание 1\n");
            count = DeleteSubStringRobin(stringArray,count-1);
            break;
        case '2':
            printf("Выполняем задание 2\n");
            count =  SortArray(stringArray,count-1);
            break;
        case '3':
            printf("Выполняем задание 3\n");
            count =  DeleteSum13(stringArray,count-1);
            break;
        case '4':
```

```
            printf("Выполняем задание 4\n");
            count = PrintString(stringArray,count-1);
            break;
        case '5':
            printf("Выход \n");
            break;
        default:
            printf( "Неправильный ввод номера задания.\n" );
    }
    if (numberZadanie == '1' || numberZadanie == '2' || numberZadanie == '3' ){
        printf("\nРезультат:\n");
        for (int i = 0; i < count-1; i++) {
            printf("%s.\n", stringArray[i]);
        }
    }
    for (int i = 0; i < count-1; ++i) { free(stringArray[i]);  } free(stringArray);
    free(iputString);
    return 0;
}
```

Основная функция, в которой считывается текст посимвольно. Объявляем наш массив через указатель на char и выделяем память для него. Если массив не создан, на экран выводится “Не создан массив”. С помощью функции *DeleteDoubleString* удаляем дублированные строки. Пользователь должен выбрать, какое задание программе нужно выполнить. С помощью оператора *switch* выполняется вызов выбранного задания. Так же, в основной функции происходит освобождение памяти.

ЗАКЛЮЧЕНИЕ

Были изучены теоретические материалы по теме курсовой, разработан и написан программный код, проведено тестирование программы.

Исходный код в приложении А.

Результаты тестов в приложении Б.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The C Programming Language / Brian W. Kernigan, Dennis M. Ritchie Second Edition, 1988. 288c.
2. C++ Resources Network [Электронный ресурс]

URL: http://cplusplus.com/

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.c

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


#define MAX_SIZE_STRING 90
#define MAX_SIZE_MATRIX 100
#define MAX_SIZE_INPUT 1000

size_t replaceSubstr(const char *srcString, char *dstString, const char
*oldSubString, const char *newSubString);

int glasnyChar(char n);

int countZapToString(char *String);

int DeleteDoubleString(char** Array,int maxCount);

int DeleteSubStringRobin(char** Array,int maxCount);

int SortArray(char** Array,int maxCount);

int DeleteSum13(char** Array,int maxCount);

int PrintString(char** Array,int maxCount);
```

```
void delete(char *string);

int main() {


    setlocale(LC_ALL, "Rus");


    // объявляем наш массив как указатель на char и выделяем необходимую память для нее
    char *iputString = (char*)malloc(MAX_SIZE_INPUT * sizeof(char*)), subArray[MAX_SIZE_STRING], ** stringArray = (char**)malloc(MAX_SIZE_MATRIX * sizeof(char*));

    int i=0, count=0, numberZadanie = 0,result = 0;


    if ( stringArray == NULL ) {
        printf("Не создан массив stringArray \n");
        return -1;
    }


    printf( "Введите данные для обработки : " );


    while ((subArray[i]  = getchar()) != '\n') {
```

```
    if (subArray[i] == '.') { //если обнаружили точку это конец строки то записываем строку предложения в строку массива

        subArray[i] = '\0';

        stringArray[count] = (char*)malloc(MAX_SIZE_MATRIX * sizeof(char));

        strcpy(stringArray[count], subArray); //  копируем данные в массив

        count++;
        i = 0;
    }
    else {
        i++;

    }
}

//Удаляем дубликаты строк в массиве
count =  DeleteDoubleString(stringArray,count - 1);

if (count <0)
    printf("Массив не создан после сжатия \n");

// Выполнения задания

printf( "Выберите одно из сдедующих заданий \n" );
printf( "1. Заменить в тексте подстроку \"Robin\" на \"I am Bathman\" \n" );
printf( "2. Отсортировать предложения по количеству запятых \n" );
```

```
    printf( "3. Удалить все предложения в которых сумма цифр равняется 13\n"
);
    printf( "4. Вывести все предложения в которых первое слово начинается с
гласной буквы или цифры \n" );
    printf( "5. Выход . \n" );
    printf( "Ваш выбор: " );

    numberZadanie = (int)getchar();
    switch ( numberZadanie ) {
        case '1':
            printf("Выполняем задание 1\n");
            count = DeleteSubStringRobin(stringArray,count-1);
            break;
        case '2':
            printf("Выполняем задание 2\n");
            count =  SortArray(stringArray,count-1);
            break;
        case '3':
            printf("Выполняем задание 3\n");
            count =  DeleteSum13(stringArray,count-1);
            break;
        case '4':
            printf("Выполняем задание 4\n");
            count = PrintString(stringArray,count-1);
            break;
        case '5':
            printf("Выход \n");
            break;
        default:
            printf( "Неправильный ввод номера задания.\n" );
```

```
    }

// Выводим данные на экран
    if (numberZadanie == '1' || numberZadanie == '2' || numberZadanie == '3' ){
        printf("\nРезультат:\n");
        for (int i = 0; i < count-1; i++) {
            printf("%s.\n", stringArray[i]);
        }
    }

    // освобождаем память
    for (int i = 0; i < count-1; ++i) { free(stringArray[i]);  } free(stringArray);
    free(iputString);
    return 0;
}

int DeleteDoubleString(char** Array,int maxCount) //Удаление дублированных
строк
{

    int duble_string = 0;


    for (int i = 0; i < maxCount; i++) {

        for (int j = 0; j < maxCount; j++) { //перебираем элементы массива в
поисках дублей

            if (i != j){ // это не одна строка
                if (strlen(Array[i]) == strlen(Array[j])){ // сравниваем длинны строк
```

```
            // если длинны строк совпали
            if (strcasecmp(Array[i], Array[j]) == 0) { // сравниваем содержание
строки без регистра

                //Цикл удаления дублированной строки перещемением
последующих строк на строку вверх
                for (int q = j; q < maxCount; q++) {
                   strcpy( Array[q] , Array[q+1]);
                }
                --maxCount;

            }
        }
    }
}


// пересоздаем массив и копируем данные
char** new_stringArray = (char**)realloc(Array, (maxCount) * sizeof(char*));

if ( new_stringArray != NULL ) {
   Array = new_stringArray;

   return maxCount;

} else {
   printf("Не создан массив new_stringArray \n");
   return -1 ;
}
```

```
}

int DeleteSubStringRobin(char** Array,int maxCount) //Удаление подстроки Robin
{
   int StartNumSubstr = 0;
   size_t sizeString;
   char *findSubtr,*OldSubString =  "Robin",*NewSubString = "I am
Batman",*Newstring = (char*)malloc(StartNumSubstr * sizeof(char));

   for (int i = 0; i <= maxCount ; i++) {

      // заменяем Robin в строке
      sizeString = replaceSubstr(Array[i],Newstring, OldSubString, NewSubString);



      if ( strcasecmp(Array[i],Newstring)  != 0){
         strcpy(Array[i], Newstring); // заменяем в массиве данные

      }
   }
}

int SortArray(char** Array,int maxCount) //Сотрировка массива
{
   char * Temp_String;
   // Перебираем каждый элемент массива (кроме последнего, он уже будет
отсортирован к тому времени, когда мы до него доберемся)
   for (int startIndex = 0; startIndex <= maxCount ; ++startIndex) {
```

```
        // В переменной smallestIndex хранится индекс наименьшего значения, которое мы нашли в этой итерации.
        // Начинаем с того, что наименьший элемент в этой итерации - это первый элемент (индекс 0)
        int smallestIndex = startIndex;

        // Затем ищем элемент поменьше в остальной части массива
        for (int currentIndex = startIndex + 1; currentIndex <= maxCount; ++currentIndex) {
            // Если мы нашли элемент, который меньше нашего наименьшего элемента,
            if (countZapToString(Array[currentIndex]) < countZapToString(Array[smallestIndex])) // сравниваем по количеству запятых в строке
                //  запоминаем его
                smallestIndex = currentIndex;
        }

        // smallestIndex теперь наименьший элемент.
        // Меняем местами наше начальное наименьшее число с тем, которое мы обнаружили

        Temp_String = Array[startIndex];
        Array[startIndex] = Array[smallestIndex];
        Array[smallestIndex] = Temp_String;
    }

}
```

```
int DeleteSum13(char** Array,int maxCount) //Удалить все предложения где сумма цифр равна 13
{
   int NumberTemp = 0,Razrayd = 0;

   int sum = 0;
   for (int i = 0; i < maxCount; i++) {

      // считаем сумму цифр в строке массива

      for (int j = 0; j < strlen(Array[i]); j++)
      {
         if (('1' <= Array[i][j]) && (Array[i][j]  <= '9')) { // определяем что символ это цифра между 1 и 9
            NumberTemp = atoi(&Array[i][j]);
            sum = sum + NumberTemp;
            // определяем разрядности числа из строки . На сколько разрядов число больше 10 изза особенности реализации функции atoi()
            while(NumberTemp>10){
               NumberTemp = NumberTemp / 10;
               ++j; //Двигаем указатель по строке
            }

         }
      }
      if ( sum == 13){ // удаляем  в массиве строки с суммой цифр равной 13
         //Цикл удаления  строки перещемением последующих строк на строку вверх
         for (int q = i; q < maxCount; q++)
            strcpy( Array[q]  , Array[q+1]);
```

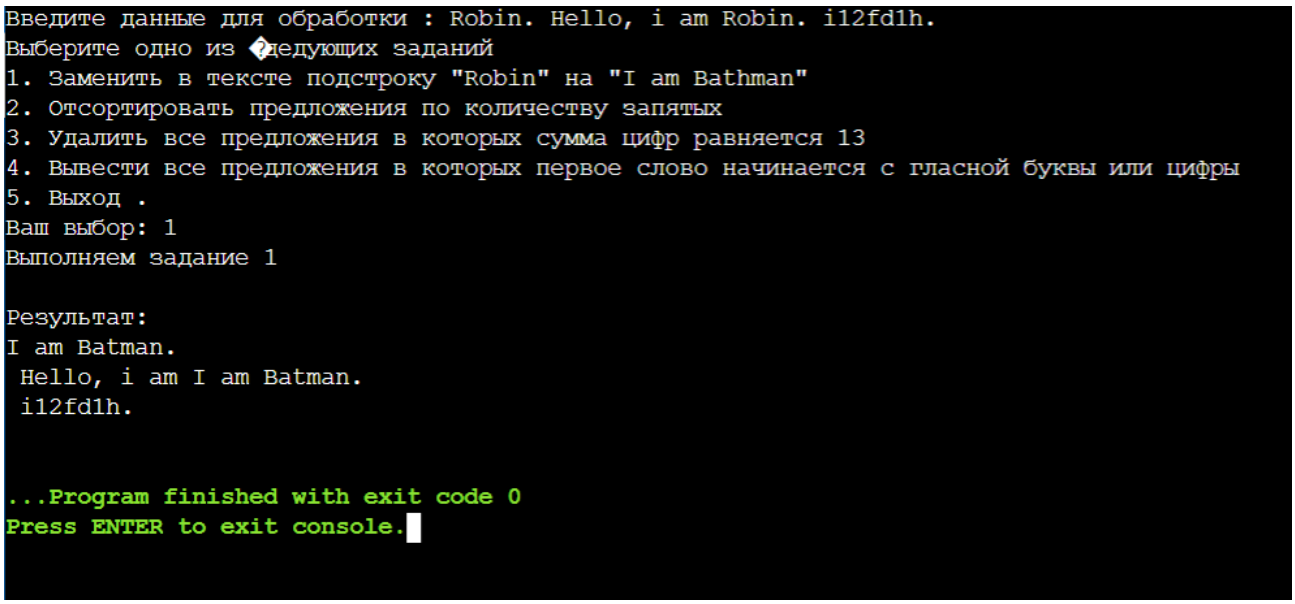
```
            --i;        // итератор уменьшаем так как строку удалили
            --maxCount; // уменьшаем размер массив
            // а на одну строку
        }
        sum = 0; // обнуляем для следующей строки
    }

    // пересоздаем массив и копируем данные
    char** new_stringArray = (char**)realloc(Array, (maxCount) * sizeof(char*));

    if ( new_stringArray != NULL ) {
        Array = new_stringArray;

        return maxCount;

    } else {
        printf("Не создан массив new_stringArray \n");
        return -1 ;
    }

}

int PrintString(char** Array,int maxCount) //Вывести все предложения где .....
{
    printf("\nРезультат:\n");
    for (int i = 0; i < maxCount; i++) {

        for (int j = 0; j < strlen(Array[i]); j++) { // перебираем строку

            if (Array[i][j] != ' ') {
```

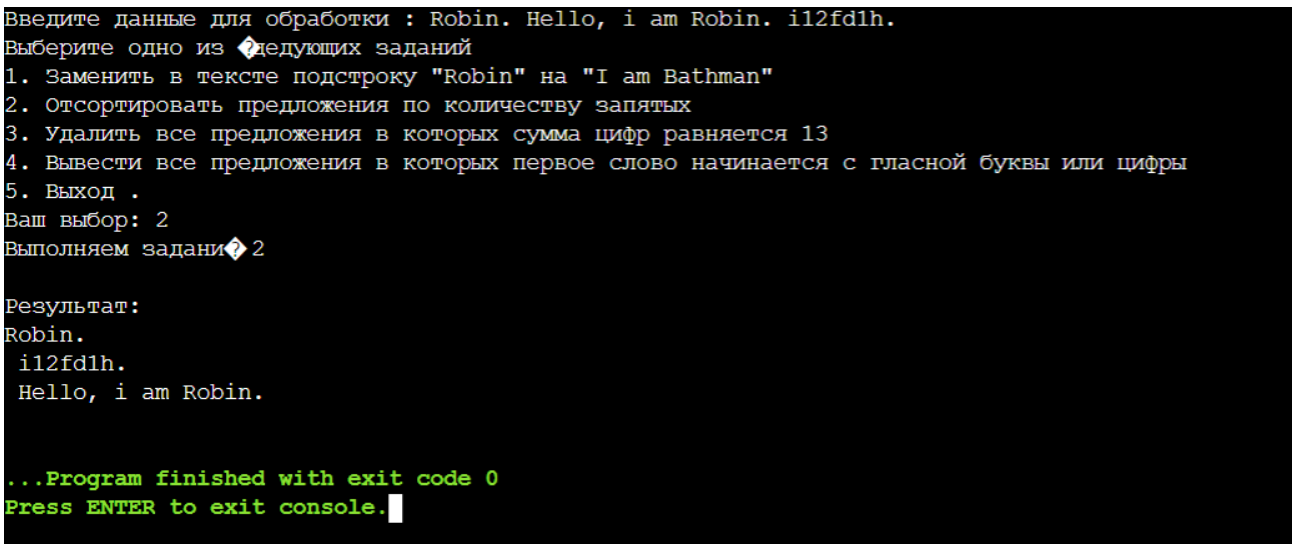
```
            if (('0' <= Array[i][j]) && (Array[i][j] <= '9') || glasnyChar(Array[i][j]))
{ // определяем что Первый символ это цифра  1 - 9 или гласная
                printf("%s \n", Array[i]);
            }
            break;
        }
    }
}

int countZapToString(char *String)
{
    int count = 0;
    for (int i = 0; String[i] != '\0'; i++){
        if (String[i] == ',') // ищем символ запятая
            count++;
    }
    return count;
}

size_t replaceSubstr(const char *srcString, char *dstString, const char
*oldSubString, const char *newSubString)
{
    size_t l_old = strlen(oldSubString), l_new = strlen(newSubString);
    size_t dst_length = 0;

    const char *src_current = srcString;
    do
    {
```

```
        const char *src_next = strstr(src_current, oldSubString);
        if (src_next == NULL)
        {
            if (dstString != NULL)
                strcpy(dstString + dst_length, src_current);

            dst_length += strlen(src_current);

            break;
        }

        size_t n = src_next - src_current;

        if (dstString != NULL)
            memcpy(dstString + dst_length, src_current, n);

        dst_length += n;

        if (dstString != NULL)
            memcpy(dstString + dst_length, newSubString, l_new);

        dst_length += l_new;

        src_current = src_next + l_old;
    } while (1);

    return dst_length;
}

int glasnyChar(char n)
```

```
{
    int k = 0;
    char mas[12] = {'a', 'e', 'i', 'o', 'u', 'y', 'A', 'E', 'I', 'O', 'u', 'Y'};
    for (int i = 0; i < 12; i++)
    {
        if (n == mas[i])
            k++;
    }
    if (k == 1)
        return 1;
    else
        return 0;
}
```

ПРИЛОЖЕНИЕ Б.
ТЕСТИРОВАНИЕ





```
Введите данные для обработки : Robin. Hello, i am Robin. i12fd1h.
Выберите одно из ?едующих заданий
1. Заменить в тексте подстроку "Robin" на "I am Bathman"
2. Отсортировать предложения по количеству запятых
3. Удалить все предложения в которых сумма цифр равняется 13
4. Вывести все предложения в которых первое слово начинается с гласной буквы или цифры
5. Выход .
Ваш выбор: 3
Выполняем задание 3

Результат:
Robin.
 Hello, i am Robin.


...Program finished with exit code 0
Press ENTER to exit console.
```

```
Введите данные для обработки : Robin. Hello, i am Robin. i12fd1h.
Выберите одно из сдедующих заданий
1. Заменить в тексте подстроку "Robin" на "I am Bathman"
2. Отсортировать предложения по количеству запятых
3. Удалить все предложения в которых сумма цифр равняется 13
4. Вывести все предложения в которых первое слово начинается с гласной буквы или цифры
5. Выход .
Ваш выбор: 4
Выполняем задание 4

Результат:
 i12fd1h


...Program finished with exit code 0
Press ENTER to exit console.
```

```
Введите данные для обработки : Robin. Hello, i am Robin. i12fd1h.
Выберите одно из ?едующих заданий
1. Заменить в тексте подстроку "Robin" на "I am Bathman"
2. Отсортировать предложения по количеству запятых
3. Удалить все предложения в которых сумма цифр равняется 13
4. Вывести все предложения в которых первое слово начинается с гласной буквы или цифры
5. Выход .
Ваш выбор: 5
Выход


...Program finished with exit code 0
Press ENTER to exit console.
```