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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текста

Студент гр. 2384 ____ Кузьминых Е.М.

Преподаватель ____ Гаврилов А.В.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент: Кузьминых Е.М.

Группа: 2384

Тема работы: обработка текста

Исходные данные:

Вариант 15

Программе на вход подается 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.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).

Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку. Также следует предусмотреть возможность выхода из программы):

В каждом предложении заменить все слова “bomb” на “defused”.

Для каждого предложения посчитать количество слов “student” и вывести количество посчитанных слов и само предложение выделив слова “student” красным цветом.

Удалить все предложения, в которых слово “enemy” встречается больше одного раза.

Отсортировать предложения по уменьшению максимального числа в строке. Число – слово, состоящее только из цифр. Если в предложении нет числа, то для такого числа значение максимального числа равно 0.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Содержание пояснительной записки:

Разделы «Аннотация», «Содержание», «Введение», «Выполнение работы», «Примеры работы программы», «Заключение», «Список использованной литературы».

Предполагаемый объем пояснительной записки:
Не менее 24 страниц.

Дата выдачи задания: 20.10.2022

Дата сдачи реферата: 19.12.2022

Дата защиты реферата: 21.12.2022

Студент _____ Кузьминых Е.М.

Преподаватель _____ Гаврилов А.В.

АННОТАЦИЯ

В ходе выполнения курсовой работы была разработана программа на языке Си, принимающая на вход текст. Далее программа выводит подсказку пользователю о возможных изменениях текста и считывает число для выбора действия над текстом. Предложения считываются до точки, после считывания последнего предложения и нажатия enter ввод текста заканчивается. Для хранения текста и отдельных предложений использовались двумерные массивы с выделением динамической памяти. Использовались библиотеки stdlib.h, stdio.h, string.h, для работы с текстом и его выводом. Также использовалась библиотека limits.h для корректной работы функции поиска максимального числа.

СОДЕРЖАНИЕ

	Введение	6
1.	Цель и задачи	7
2.	Ход выполнения работы	8
2.1.	Функции для считывания текста	8
2.2.	Функции для обработки текста	9
2.3	Функция *main*	11
3.	Заключение	12
4.	Список использованной литературы	13
	Приложение А. Примеры работы программы	14
	Приложение Б. Инструкция по запуску программы	18

ВВЕДЕНИЕ

Цель работы – создание программы, принимающей на вход текст неизвестной длины и изменяющей его в зависимости от значения, введенного пользователем, изучение методов работы со строками и их модификаций.

Программа хранит текст с помощью двухмерных массивов, память выделяется динамически с помощью функций стандартной библиотеки stdlib.h.

1. ЦЕЛИ И ЗАДАЧИ

Цель создание программы, принимающей на вход текст неизвестной длины и изменяющей его в зависимости от значения, введенного пользователем.

Задачи:

В ходе выполнения курсовой работы были поставлены задачи:

1. Разработать способ представления и хранения текста, предложений и слов в памяти

2. Реализовать алгоритм считывания текста произвольной длины с клавиатуры и вывода его на экран

3. Освоить способ выделения из текста отдельных его частей: подстрок, являющихся словами, частями слов, предложениями.

4. Освоить способ модификации символов текста или его частей. Удаление, добавление новых символов.

5. Реализовать задания из курсовой работы на обработку текста в соответствии с условием варианта.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Функции для считывания текста:

Функция *read_sentence()* считывает посимвольно предложение введенное пользователем. В начале функции создается переменная *buffer,* отвечающая за размер буфера памяти для хранения предложения и строка sentence, под которую выделяется память. После в цикле *do while* мы посимвольно считываем текст, который вводит пользователь, записывая символ в строку *sentence*. После происходит проверка размера буфера памяти, и если он равен длине предложения, то происходит увеличение буфера на значение BUFFER_SIZE и перевыделение памяти с помощью функции *realloc()*. Цикл do while считывает предложение до того момента, пока не встретит точку или символ ‘\n’. После считывания текста на месте последнего индекса ставится символ ‘\0’ для окончания строки. В результате работы функция возвращает считанное предложение.

Функция *collect_text()* создает двухмерный массив, куда сохраняет предложения, введенные пользователем. На вход функция принимает указатель на переменную count_of_sentence, для того чтобы передать в нее количество предложений, считанных программой. В начале создается переменная *buffer,* которая хранит в себе размер буфера памяти (аналогично с функцией *read_sentence()*), сам двухмерный массив text, под который выделяется память при помощи функции *malloc()*, переменная *size,* которая будет сохранять количество предложений, которые ввел пользователь, строку sentence для хранения последнего считанного предложения. В цикле *while* мы считываем предложение, введенное пользователем с помощью вызова функции *read_sentence(),* предложение сохраняем в строку *sentence*. В случае, если пользователь ввел \n (не ввел текст, а просто нажал enter), мы выходим из цикла. После мы записываем наше предложение в двухмерный массив *text* и проверяем размер буфера, в случае нехватки памяти увеличиваем размер буфера и делаем перевыделение памяти с помощью функции *realloc()*. После выхода из цикла

функция сохраняет значение count по адресу переменной *count_of_sentence* и возвращает двухмерный массив из предложений, введенных пользователем.

2.2. Функции для обработки текста.

int is_separator(char x) – функция, возвращающая 1, если символ, переданный в функцию является разделителем (пробелом, точкой, запятой).

int len_number(int x) – функция, возвращающая «длину числа», количество символов, потребовавшихся для записи числа.

*char** null_repeat(char** text, int* count)* – функция, удаляющая предложения, равные *NULL с* помощью сдвига оставшихся предложений функцией *memmove().* Возвращает измененный текст.

*char** delete_repeats(char** text, int *count)* – функция, удаляющая одинаковые предложения, полученные при вводе от пользователя. Принимает двухмерный массив предложений и указатель на количество предложений в тексте. В начале функции создается 2 строки - *snt1* и *snt2*, в которых будут храниться сравниваемые предложения. Функция перебирает предложения и сравнивает их без учета регистра. Если предложения одинаковые, функция освобождает память предложения и приравнивает к NULL второе по порядку предложение. В конце *delete_repeats()* вызывается функция *null_repeats(),* которая предложения, равные NULL. Функция возвращает измененный текст с удаленными дубликатами предложений.

*void replace_bomb(char** text, int size)* – функция, заменяющая в тексте слово «bomb» на слово «defused». На вход принимает текст для обработки и количество предложений в тексте. С помощью функции *strstr()* находит указатель на вхождение слова «bomb» в предложении , и в случае, если это отдельное слово, а не подстрока другого слова, заменяет его на «defused» с помощью двух вызовов функций *memmove().* Затем на месте последнего индекса, увеличенного на 3 (на это количество символов увеличилась строка) в предложении ставится ‘\0’ для окончания строки и ищется повторное вхождение слова «bomb» со сдвигом

предыдущего указателя.

*void delete_enemy_sentence(char** text, int* count)* – принимает текст и количество предложений в нем. Работает схожим образом, как функция *delete_repeats()*. Создаем счетчик *count* для количества слов «enemy» в предложении. Перебираем каждое предложение в тексте, создаем его копию и разбиваем ее на слова с помощью функции *strtok()*, сравниваем каждое слово со словом «enemy» и если они совпадают, то увеличиваем счетчик на еденицу. Если для предложения счетчик больше одного, то мы освобождаем память и приравниваем предложение к *NULL*. После вызываем функцию *null_repeat()*, с помощью которой удаляем все предложения, равные NULL. Функция изменяет полученный текст.

*void count_students(char** text, int count)* – функция, которая выводит количество слов «student» для каждого предложения, а также само слово student красным цветом. В начале функции мы создаем строки *start*, *end*, в которых храним код для изменения цвета текста в консоли на красный и код для возвращения к стандартному цвету соответственно, переменную *count_stud*, в которой будет храниться количество слов «student» в предложении, и строку *stud*, хранящую в себе слово «student». После мы перебираем каждое предложение и разбиваем его на слова с помощью функции *strtok()*. Далее мы посимвольно выводим наш текст, но если встречаем отдельное слово «student», то выводим перед ним строку *start*, а после него строку *end* для смены цвета текста в консоли.

*int max_number(char *sentence)* – функция для нахождения максимального числа в предложении. В начале функции создается переменная *max*, которая хранит в себе INT_MIN - минимальное значение для типа int (для этого была использована библиотека limits.h), создается копия предложения для разбиения её на слова с помощью функции *strtok()*. Затем мы перебираем слова в предложении и с помощью функции *atoi()* сохраняем число из строки, и если оно не является подстрокой другого предложения (проверка с помощью сравнения длин числа и слова, на которые разбил предложение *strtok()*) сравниваем его с

нынешним максимальным значением и сохраняем максимальное среди двух переменных. Если в предложении нет чисел и значение max не поменялось, то оно становится равным 0.

*int comp(const void *text1, const void *text2)* – компаратор для работы функции сортировки *qsort()*. Принимает два указателя на строки, после возвращает разность максимальных чисел в строках.

*void sort_text(char** text, int count)*- функция, применяющая *qsort()* на полученный ею текст.

2.3 Функция main():

В функции *main()* создаются переменные *count_of_sentence* и *new_text*, которые будут хранить текст, введенный пользователем и количество предложений в тексте. Выводится подсказка о том, чтобы пользователь начал вводить текст. После окончания ввода программа выводит считанные предложения, а также подсказку, в которой просится ввести число от 1 до 4 для изменения полученного текста, либо ввод числа 0 для выхода из программы.

После выбора действия программа выводит текст с обработкой, зависящей от введенного пользователем числа и завершает работу. Выбор функции для использования реализован с помощью условного оператора *switch-case*.

3. ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была создана программа, принимающая на вход текст неизвестной длины и изменяющая его в зависимости от данных, введенных пользователем.

Была изучена работа с динамической памятью и указателями. Были изучены методы работ со строками, способы выделения различных подстрок из предложений и их модификаций.

4. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1.splusplus.com.

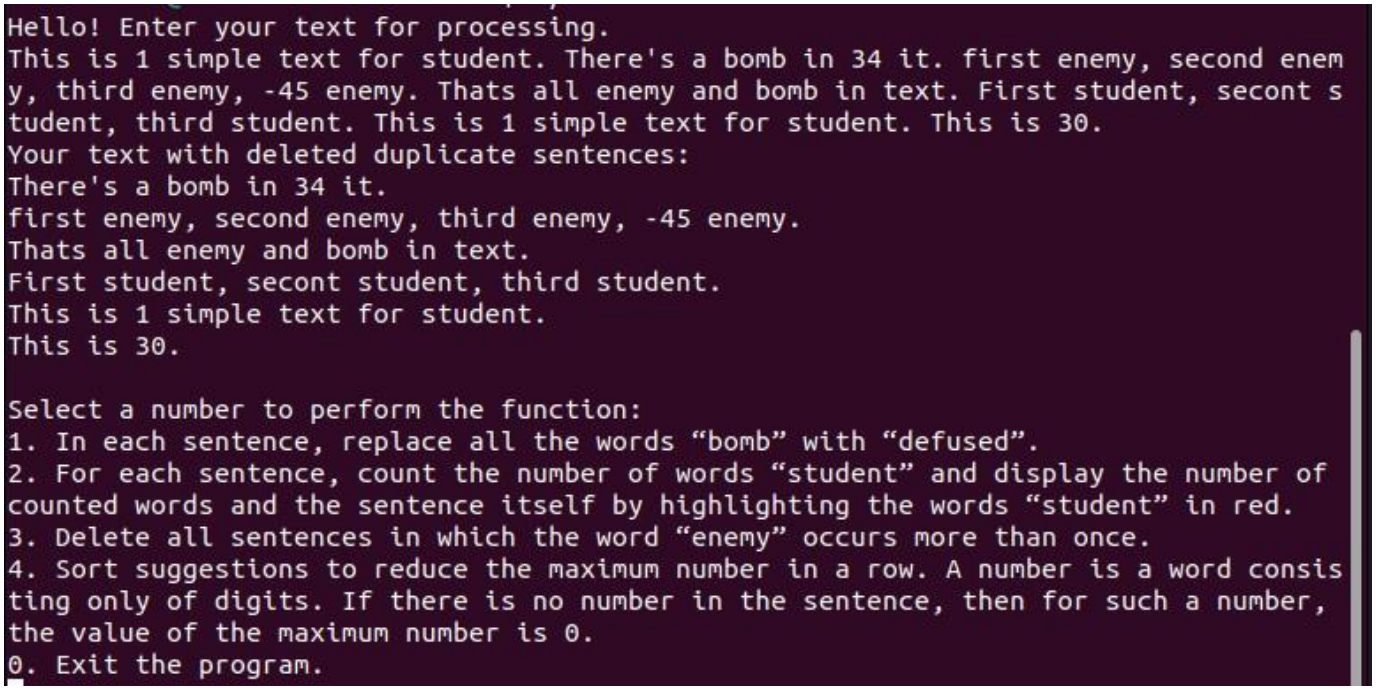
2.Язык С. Брайан Керниган и Деннис Ритчи.

3.Методические указания по выполнению курсовой и лабораторных работ по дисциплине программирование. Первый курс. Осенний семестр / сост.: Кринкин К.В., Берленко Т.А., Заславский М.М., Чайка К.В., Допира В.Е., Гаврилов А.В. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2022. 39 с.

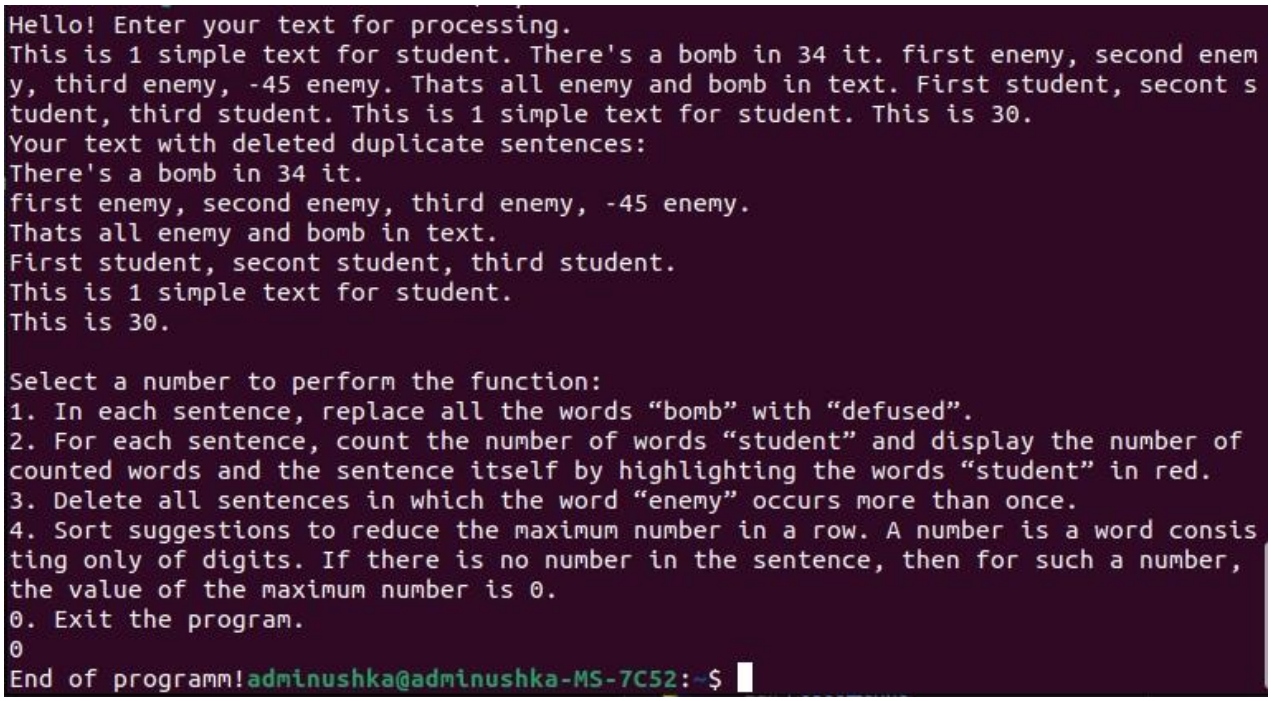
Приложение А. Примеры работы программы.

На следующих скриншотах приведены примеры работы программы. Использовались различные варианты ввода текста для демонстрации гибкости программы.

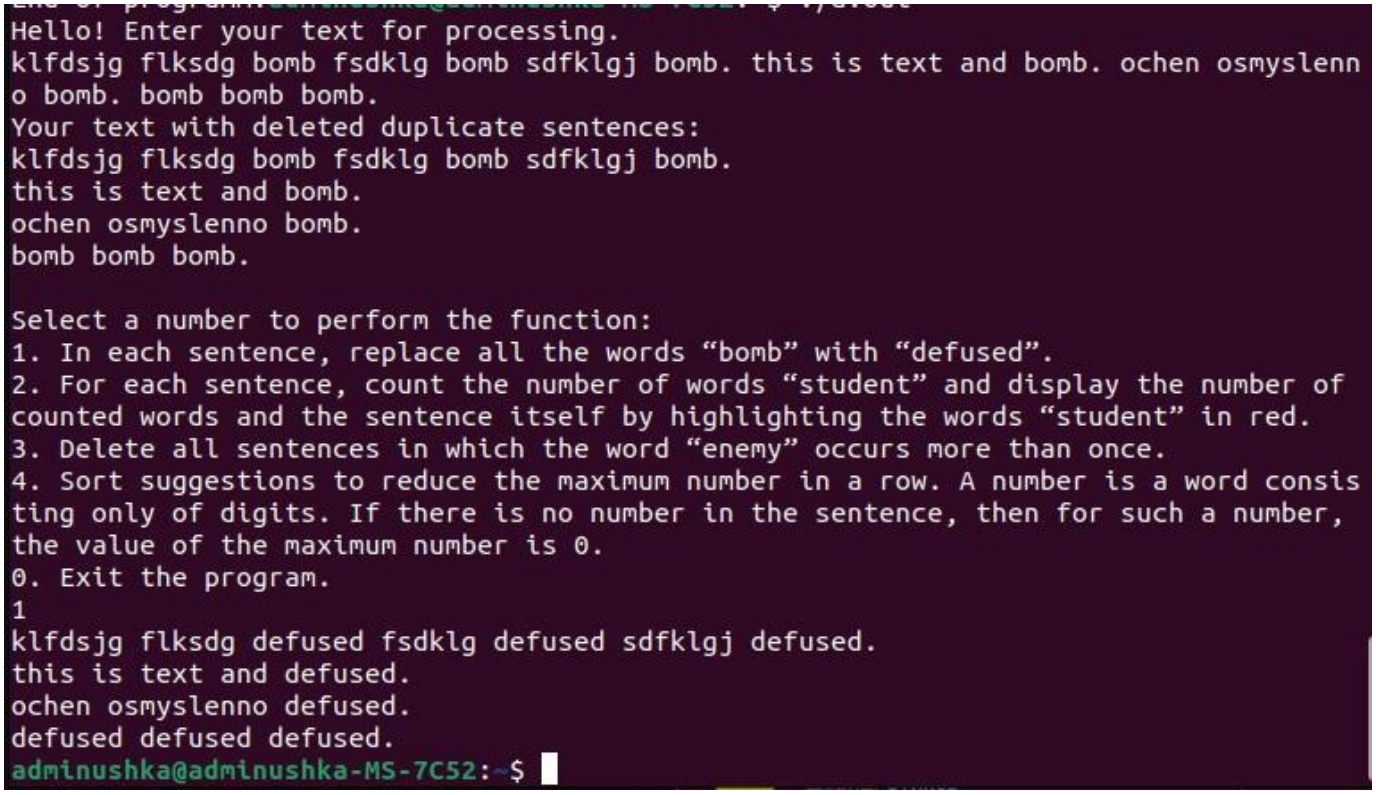
Ввод текста, удаление повторяющихся предложений:



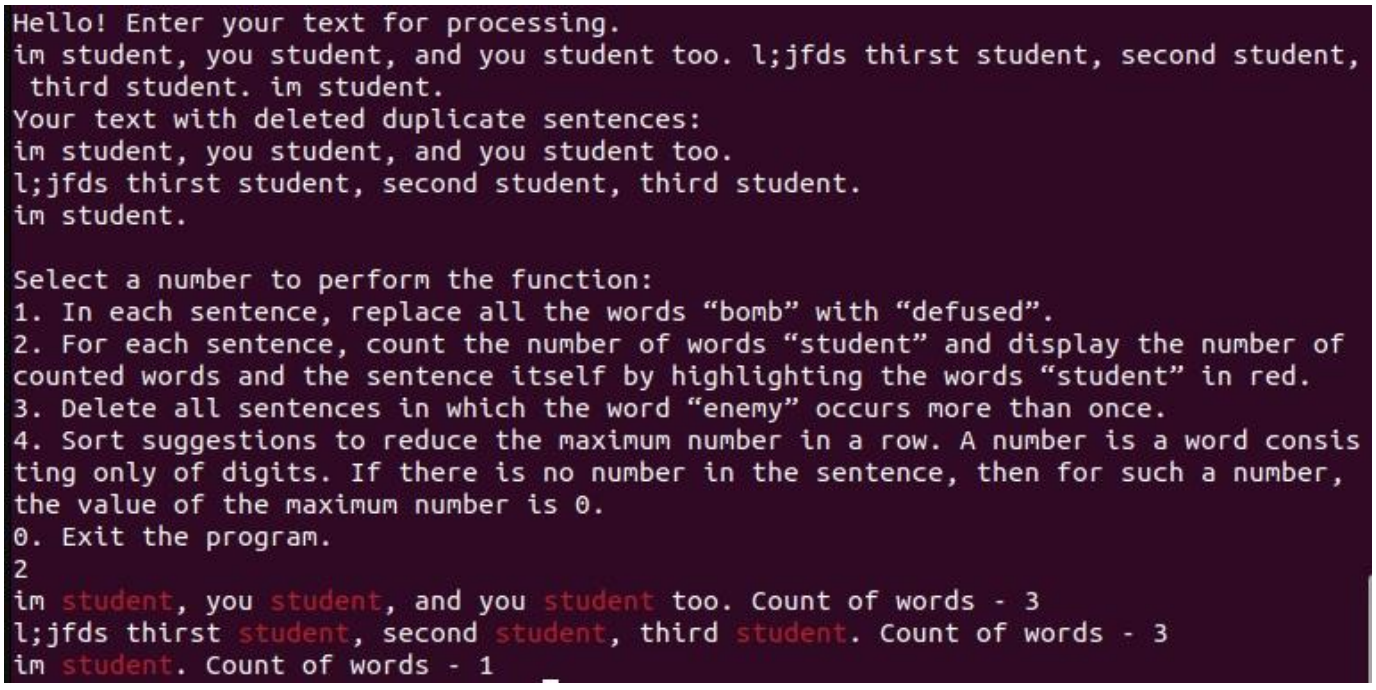
Выход из программы:



Выполнение функции №1, замена «bomb» на «defused»:



Выполнение функции №2, подсчет кол-ва слов «student» в предложении и выделение слова красным цветом:



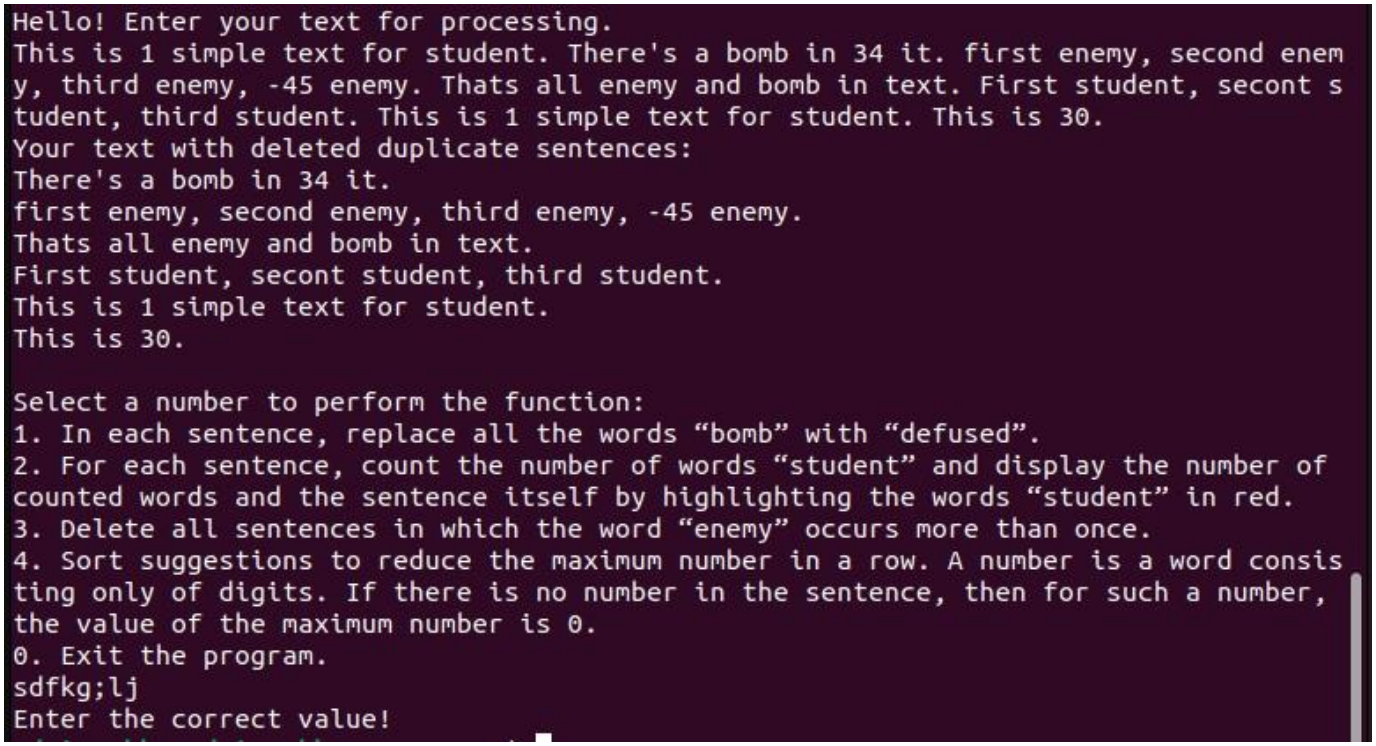
Выполнение функции №3, удаление предложений, в которых количество слов «enemy» больше 1:

```
Hello! Enter your text for processing.
this is my enemy. im your enemy, you - my enemy. enemy enemy. enemy enemy enemy. ene
my enemy enemy enemy.
Your text with deleted duplicate sentences:
this is my enemy.
im your enemy, you - my enemy.
enemy enemy.
enemy enemy enemy.
enemy enemy enemy enemy.

Select a number to perform the function:
1. In each sentence, replace all the words “bomb” with “defused”.
2. For each sentence, count the number of words “student” and display the number of
counted words and the sentence itself by highlighting the words “student” in red.
3. Delete all sentences in which the word “enemy” occurs more than once.
4. Sort suggestions to reduce the maximum number in a row. A number is a word consis
ting only of digits. If there is no number in the sentence, then for such a number,
the value of the maximum number is 0.
0. Exit the program.
3
this is my enemy.
```

Выполнение функции №4, сортировка предложений по убыванию максимального числа в предложении:

```
Hello! Enter your text for processing.
-45 abc bca 23. dfgj -1. opa pa. 56 fgdj. 123456 thirst. -56 last.
Your text with deleted duplicate sentences:
-45 abc bca 23.
dfgj -1.
opa pa.
56 fgdj.
123456 thirst.
-56 last.

Select a number to perform the function:
1. In each sentence, replace all the words “bomb” with “defused”.
2. For each sentence, count the number of words “student” and display the number of
counted words and the sentence itself by highlighting the words “student” in red.
3. Delete all sentences in which the word “enemy” occurs more than once.
4. Sort suggestions to reduce the maximum number in a row. A number is a word consis
ting only of digits. If there is no number in the sentence, then for such a number,
the value of the maximum number is 0.
0. Exit the program.
4
123456 thirst.
56 fgdj.
-45 abc bca 23.
opa pa.
dfgj -1.
-56 last.
```

Ввод некорректных данных при выборе функции для обработки текста:



ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#define BUFFER_SIZE 100

//функция считывания предложения, сделали буфер, изначально выделили
память и считываем предложение посимвольно.
char *read_sentence() {
    int buffer = BUFFER_SIZE;
    char* sentence = malloc(buffer * sizeof(char));
    int len_of_sentence = 0;
    char symbol;
    int nachalo = 1;
    do {
          symbol = getchar();
          if (nachalo == 1) {
               if (symbol == ' '||symbol=='.'||symbol==',') {
                    continue;
               } else {
                    nachalo = 0;
               }
          }
          sentence[len_of_sentence++] = symbol;
          if (len_of_sentence == buffer) {
               buffer += BUFFER_SIZE;
               sentence = realloc(sentence, buffer);
          }
    } while (symbol != '.' && symbol !='\n');
    sentence[len_of_sentence] = '\0';
    return sentence;
}

//Функция, которая собирает полученные предложения в один текст, пока
не встретит enter на вводе.
char **collect_text(int *count_of_sentence) {
    int buffer = BUFFER_SIZE;
    char **text = malloc(buffer * sizeof(char *));
    int count = 0;
    char *sentence;
    while (1) {
          sentence = read_sentence();
          if (sentence[0] == '\n') {
               break;
          }
          text[count++] = sentence;

          if (count == buffer) {
               buffer += BUFFER_SIZE;
               text = realloc(text, buffer * sizeof(char *));
          }
    }
    *count_of_sentence = count;
```

```
    return text;
}
char** null_repeat(char** text, int* count){//функция, удаляющая
предложения, если они равны null
    for (int i = 0; i < *count; i++) {
          if (text[i]==NULL){
                (*count)--;
                memmove(text+i, text+i+1, (*count-i)*sizeof (char*));
                i--;

          }

    }
    return text;
}
void delete_repeats(char **text, int *count) {//принимаем текст на
вход, если предложения равны, то делаем равным NULL и удаляем
предложения.
    char* snt1;
    char* snt2;
    for(int i = 0; i < *count; i++) {
          for(int j = i+1; j < *count; j++) {
                snt1 = text[i];
                snt2 = text[j];
                if(snt1!=NULL && snt2!=NULL) {
                      if(strcasecmp(snt1,snt2)==0) {
                            free(text[j]);
                            text[j] = NULL;
                      }
                }
          }
    }
    text = null_repeat(text,count);

}

int is_separator(char x) {//функция, которая проверяет, разделитель
ли символ
    return x == ' ' || x == '.' || x == ',';
}

void replace_bomb(char **text,int size) {
    char *bomb = "bomb";
    char *def = "defused";
    char *p;
    for (int i = 0; i < size; i++) {
          p = strstr(text[i], bomb);
          while (p) {
                if (is_separator(*(p + 4)) && (is_separator(*(p - 1))
|| p == text[i])) {
                      int len = strlen(text[i]);
                      memmove(p + strlen(def), p + strlen(bomb),
strlen(text[i])-(p-text[i])-strlen(bomb));
                      memmove(p, def, strlen(def));
                      text[i][len + 3] = '\0';
                }
                p = strstr(p+1 , bomb);
          }
```

```
    }
}

void delete_enemy_sentence(char **text, int *size) {
    char* enemy1 = "enemy";
    char* enemy2 = "enemy.";
    char* snt;
    char* istr;
    char* sep = "., ";
    int count;
    int index = 0;
    for(int i = 0; i < *size; i++) {
          count = 0;
          snt = malloc((strlen(text[i])+1)*sizeof(char));
          strcpy(snt,text[i]);
          istr = strtok(snt,sep);
          while(istr!=NULL) {
                if(strcmp(istr,enemy1)==0 || strcmp(istr,enemy2)==0) {
                      count++;
                }
                istr = strtok(NULL,sep);
          }
          if(count>1) {
                free(text[i]);
                text[i] = NULL;
          }
    }

    text = null_repeat(text, size);
}
void count_students(char **text, int count) {
    int count_stud;
    char *stud = "student";
    char* start = "\x1b[31m";
    char* end = "\x1b[0m";
    for (int i = 0; i < count; i++) {
          count_stud = 0;
          char *prev = text[i];
          char *p = strstr(text[i], stud);
          while (p != NULL) {
                if (is_separator(*(p + 7)) && (p == text[i] ||
is_separator(*(p - 1)))) {
                      count_stud++;
                      for (char *j = prev; j < p; j++) {//посимвольно
выводим текст, если встречаем отдельное слово student, то ставим
перед ним и после код для окрашивания текста в консоли.
                            printf("%c", j[0]);
                      }
                      printf("%sstudent%s",start,end);
                      prev = p + 7;
                      p = strstr(prev, stud);
                }
                else {
                      for (char *j = prev; j < p; j++) {
                            printf("%c", j[0]);
                      }
                      printf("student");
                      prev = p + 7;
```

```
                    p = strstr(prev, stud);
                }
            }
            p = text[i] + strlen(text[i]);
            for (char *j = prev; j < p; j++) {
                printf("%c", j[0]);
            }


            printf(" Count of words - %d\n", count_stud);
    }
}

int len_number(int x) {//функция для подсчета количиства символовов
для записи числа.
    int count = 0;
    if (x < 0) {
        count++;
    }
    while (x) {
        x = x / 10;
        count++;
    }
    return count;
}

int max_number(char *sentence) {//находим максимальное число в
предложении
    int max = INT_MIN;
    char *str_copy = malloc(sizeof(char) * strlen(sentence));
    strncpy(str_copy, sentence, strlen(sentence));
    char *pch = strtok(str_copy, " ,.");
    while (pch != NULL) {
        int val = atoi(pch);
        if (strlen(pch) == len_number(val)) {
            if (val > max) {
                max = val;
            }
        }
        pch = strtok(NULL, " ,.");
    }
    if (max == INT_MIN) {
        max = 0;
    }
    return max;
}

int comp(const void *text1, const void *text2) {
    return max_number(*(char **) text2) - max_number(*(char **)
text1);
}

void sort_text(char **text, int count) {//функция сортировки текста
по уменьшению максимального числа.
    qsort(text, count, sizeof(char *), comp);
}

int main() {
```

```
    int count_of_sentence;
    char **new_text;
    printf("Hello! Enter your text for processing.\n");
    new_text = collect_text(&count_of_sentence);
    delete_repeats(new_text, &count_of_sentence);
    printf("Your text with deleted duplicate sentences:\n");
    for (int i = 0; i < count_of_sentence; i++) {
         puts(new_text[i]);
    }
    printf("\nSelect a number to perform the function:\n");
    printf("1. In each sentence, replace all the words <bomb> with
“defused”.\n");
    printf("2. For each sentence, count the number of words “student”
and display the number of counted words and the sentence itself by
highlighting the words “student” in red.\n");
    printf("3. Delete all sentences in which the word “enemy” occurs
more than once.\n");
    printf("4. Sort suggestions to reduce the maximum number in a row.
A number is a word consisting only of digits. If there is no number
in the sentence, then for such a number, the value of the maximum
number is 0.\n");
    printf("0. Exit the program.\n");
    int x = getchar();
    switch (x) {
         case '0':
              printf("End of programm!");
              break;
         case '1':
              replace_bomb(new_text, count_of_sentence);
              for (int i = 0; i < count_of_sentence; i++) {
                   puts(new_text[i]);
              }
              break;
         case '2':
              count_students(new_text, count_of_sentence);
              break;
         case '3':
              delete_enemy_sentence(new_text, &count_of_sentence);
              for (int i = 0; i < count_of_sentence; i++) {
                   puts(new_text[i]);
              }
              break;
         case '4':
              sort_text(new_text, count_of_sentence);
              for (int i = 0; i < count_of_sentence; i++) {
                   puts(new_text[i]);
              }
              break;
         default:
              printf("Enter the correct value!\n");
              break;
    }
    for (int i = 0; i < count_of_sentence; i++) {
         if(new_text[i]!=NULL){
              free(new_text[i]);
         }

    }
```

```
    free(new_text);
    return 0;
}
```