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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: ОБРАБОТКА СТРОК.

Студент гр. 0383 _____ Бояркин Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Бояркин Н.А.

Группа 0383

Тема работы: обработка строк.

Исходные данные:

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):

1. Преобразовать предложения так, чтобы каждое слово в нем начиналось с заглавной буквы, а остальные символы слова были строчными.
2. Удалить все предложения, в которых есть число 2018 (даже если оно внутри какого-то слова).
3. Отсортировать предложения по увеличению суммы цифр встречаемых в предложении. Если в предложении нет цифр, то сумма цифр данного предложения равняется ∞.

4. Вывести на экран все предложения, в которых встречаются все цифры хотя бы один раз.

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:

«Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 14 страниц.

Дата выдачи задания: 02.11.2020

Дата сдачи реферата: 19.12.2020

Дата защиты реферата: 21.12.2020

Студент		Боякрин Н.А.
Преподаватель		Жангиров Т.Р.

АННОТАЦИЯ

В курсовой работе была реализована обработка текста. Длина текста и каждого предложения заранее не известна. Текст сохранён в динамический массив строк и оперировался далее только с ним. Все подзадачи, включая ввода и вывода были реализованы в виде отдельных функции. В работе программы активно использовались функции стандартной библиотеки языка Си, включая функцию сортировки. Для обработки команд пользователя использовался оператор *switch*. Для выполнения команд пользователя использовались функции *delete_separating_sentence(), capital_letter(), delete2018(), digits(), qsort(), cmp()*. Пример работы программы приведён в приложении А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: создать программу, которую выполняет обработку текста в зависимости от команды пользователя. Также предусмотреть возможность выхода из программы. Использовать динамический массив строк и возможности языка Си, включая функции из стандартной библиотеки и функции сортировки, оператор switch и циклы while и for.

Программа разработана на базе операционной системе Linux Ubuntu с использованием IDLE Visual Studio Code.

1. ЦЕЛЬ И ЗАДАЧИ

1.1. Цель.

Целью данной работы является разработка программы, обрабатывающую введенный пользователем текст в зависимости от команды пользователя.

1.2. Задачи

Для достижения поставленной цели требуется решить следующие задачи:

- Считать текст пользователя.
- Написать функцию, которая выводит текст на экран.
- Реализовать функции, которые выполняют задания в зависимости от команды пользователя.
- Написать главную функцию. Вывести подсказки для пользователя.
- Реализовать очистку памяти.
- Протестировать программу.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Инициализируем наиболее важные переменные

command – команда пользователя.

count_sentence – число предложений в тексте.

capacity и capacity_two – объём текста, которые нам помогают для перевыделения памяти и их контроля.

text – динамический двумерный массив строк.

stop – вспомогательная переменная, с помощью которой пользователь может завершить программу.

2.2. Ввод данных с помощью функции *char** in(int* p_count_sentence, int capacity, int capacity_two, char** text)*

На вход функции передаются количество предложений (изначально он равен нулю), объём и сам динамический массив.

Функция динамический массив строк. С помощью цикла while мы считываем текст и разделяем его на строки. Для этого нам помогают переменные count_word и start. С помощью count_word мы записываем символ в нужный индекс и, когда символ равен ‘.’(символ точки), то мы обнуляем count_word и после точки записываем символ ‘\0’ (символ завершения строки). С помощью функции malloc мы выделяем память для следующей строки. С помощью переменной start мы понимаем, когда началось предложение и пропускаем вводимый пользователем пробел после точки. Таким образом, мы не считываем лишний символ.

Также мы проверяем не переполнился ли объём предложений с помощью переменных capacity и capacity_two. Сначала мы проверяем количество символов в предложении и если он равен capacity_two, то увеличиваем capacity_two на 100 и перевыделяем память для строки, чтобы вместилось ещё символов в строку. Потом мы проверяем хватает ли нам вместимости для предложений. Если нет, то увеличиваем память массива.

2.3. Удаляем все повторно встречающиеся предложения с помощью функции *int delete_repeating_sentence(char array, int size)***

Функция принимает на вход динамический массив строк и количество предложений в нём.

Возвращает количество предложений после удаления всех повторно встречающихся предложений.

С помощью двух циклов for мы сравниваем каждое предложение с другими предложениями и с помощью функции из стандартной библиотеки Си strcasecmp() (работает только на Linux) посимвольно сравниваем, но без учёта регистра. Если встретились повторно встречающиеся предложения, то мы их удаляем (то есть перезаписываем заново массив строк с помощью ещё одного цикла for). После этого мы перевыделяем память. (Предварительно проверив, что дополнительная переменная не вернула NULL)

2.4. Функция для преобразования предложения так, чтобы каждое слово в нем начиналось с заглавной буквы, а остальные символы слова были прописными. *void capital_letter(char array, int size)***

Функция принимает динамический массив строк и количество предложений в нём.

Функция ничего не возвращает.

С помощью цикла for и переменной i мы проходим по предложениям и после каждого прохождения мы проверяем первый символ. Если первый символ в предложении это строчная буква, то мы делаем эту букву заглавной с помощью функции toupper(). Также мы проходим по остальным символам предложения и делаем букву строчной с помощью функции tolower(), если мы встретили прописную букву в слове**.**

2.5. Функция для удаления всех предложений, в которых есть число 2018 (даже если оно внутри какого-то слова) с помощью функции *int delete(char array, int size)*.**

Функция принимает динамический массив строк и количество предложений в нём.

Функция возвращает количество предложений после удаления предложений с числом 2018.

С помощью первого цикла for мы проходим по предложениям. Если предложение меньше или равен 4 символов (включая точку), то мы его пропускаем. Если нет, то мы проходим по символам с помощью второго цикла for. Если мы нашли число 2018 (даже если оно находиться внутри какого-то слово), то мы перезаписываем динамический массив строк с помощью третьего цикла for. Переменной i присваиваем значение 0 и j присваиваем значение -1, чтобы на следующее итерации значение стало 0. Уменьшаем переменную size. Таким образом, мы проверяем все предложения заново. После этого мы перевыделяем память с помощью функции realloc и делаем проверку, что realloc не выдал нам значение NULL.

2.6. Выводим на экран все предложения, в которых встречаются все цифры хотя бы один раз с помощью функции *void digits(char array, int size)***

Функция принимает динамический массив строк и количество предложений в нём.

Функция ничего не возвращает.

С помощью цикла for проходим по предложениям в массиве и с помощью второго цикла for проходим по символам в предложении до точки. Если мы встретили в предложении одну цифру (находим её с помощью функции isdigit(), то изменяет значение переменной wasDigit и завершаем прохождение символов в этом предложении. Таким образом, мы

понимает какое предложение мы пропускаем, а какое выводим на экран. И снова изменяем значение переменной wasDigit на ноль.

2.7. Функция cmp, которая помогает функции qsort() отсортировать предложения по увеличению суммы цифр встречаемых в предложении. Если в предложении нет цифр, то сумма цифр данного предложения равняется ∞.

Функция: int cmp(const void *a, const void *b)

Функция возвращает -1, 0 или 1 в зависимости от принимаемых аргументов функции.

Изначально мы ставим переменным only_str_one и only_str_two значение равное 1 и если мы находим цифру в предложении, то меняем значение на 0. С помощью первого цикла мы суммируем все цифры в первом предложении. Аналогично делаем со вторым. Если сумма в первом предложении больше, чем во втором, то возвращается 1, если меньше, то возвращается -1. Если суммы равны, то возвращаем 0. Если в первом предложении нет цифр, то возвращаем 1, если во втором нет цифр, то возвращаем -1. Если в обоих предложениях нет цифр, то возвращаем 0.

2.8. Функция вывода *void out(char array, int size)***

Выводим все предложения на экран с помощи цикла for. В начале и в конце вывода предложений выводим символ перевода строки для более удобного чтения пользователя.

2.9. Оператор switch.

Перед каждым вводом команды пользователем, мы выводим ему подсказку. С помощью оператора switch мы выполняем ту задачу, команду которого пользователь ввёл. Чтобы выйти из программы пользователю необходимо ввести команду 0. Пользователю необходимо ввести команду 1, чтобы преобразовать предложения так, чтобы каждое

слово в нем начиналось с заглавной буквы, а остальные символы слова были строчными. Пользователю необходимо ввести команду 2, чтобы удалить все предложения, в которых есть число 2018 (даже если оно внутри какого-то слова). Пользователю необходимо ввести команду 3, чтобы отсортировать предложения по увеличению суммы цифр встречаемых в предложении. Если в предложении нет цифр, то сумма цифр данного предложения равняется ∞. Пользователю необходимо ввести команду 4, чтобы Вывести на экран все предложения, в которых встречаются все цифры хотя бы один раз. Если пользователь ввёл не команду, то мы выводим на экран “Something went wrong“

2.10. Очищаем память.

Очищаем память с помощью цикла for и функции free().

ЗАКЛЮЧЕНИЕ

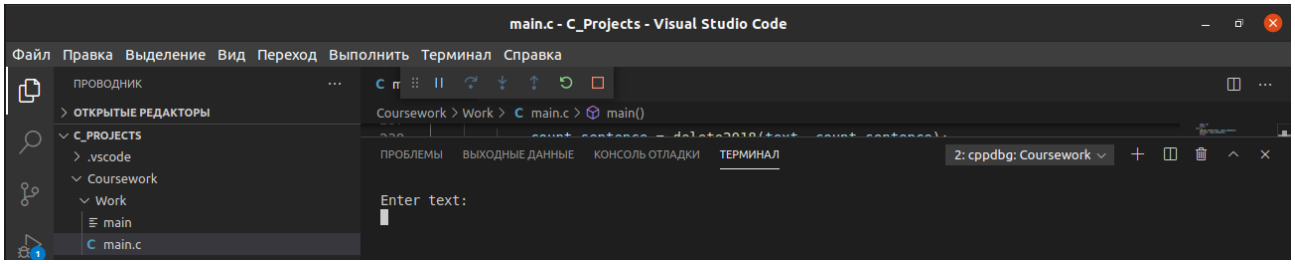
Для написания данной программы были активно применены функции из стандартной библиотеки языка Си, циклы for и while, оператор switch. Для сохранения текста был применён динамический двумерный массив строк. Создана программа, которая считывает данные пользователя и выполняет его команды. Каждая подзадача, ввод и вывод были реализованы в виде отдельных функций. Очищался массив с помощью функции *free()*. Программа выводит верный результат.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

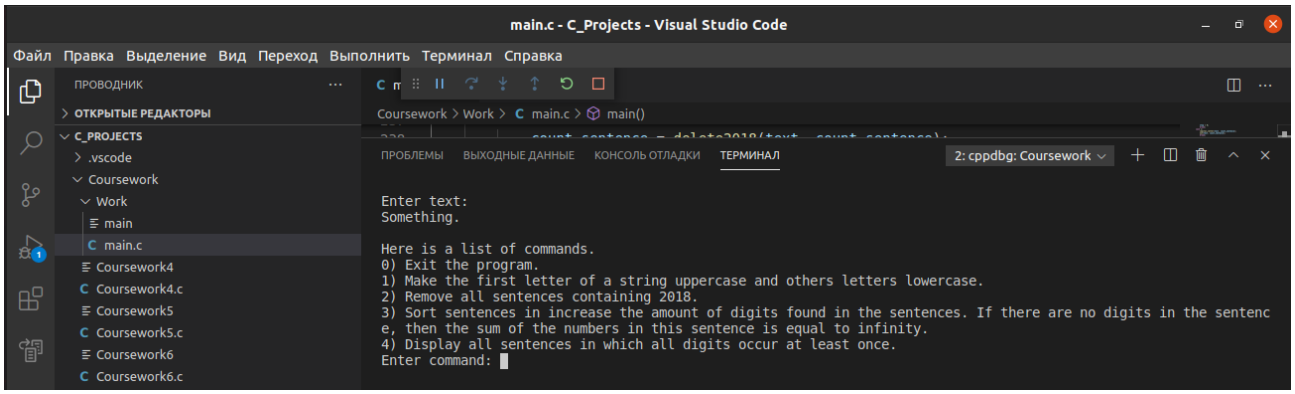
1. Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978, 288 с.
2. Сайт Cplusplus.com URL: cplusplus.com (Дата обращения 18.12.2020)
3. Сайт Программирование на C и на C++ URL: c-cpp.ru (Дата обращения 18.12.2020)

ПРИЛОЖЕНИЕ А

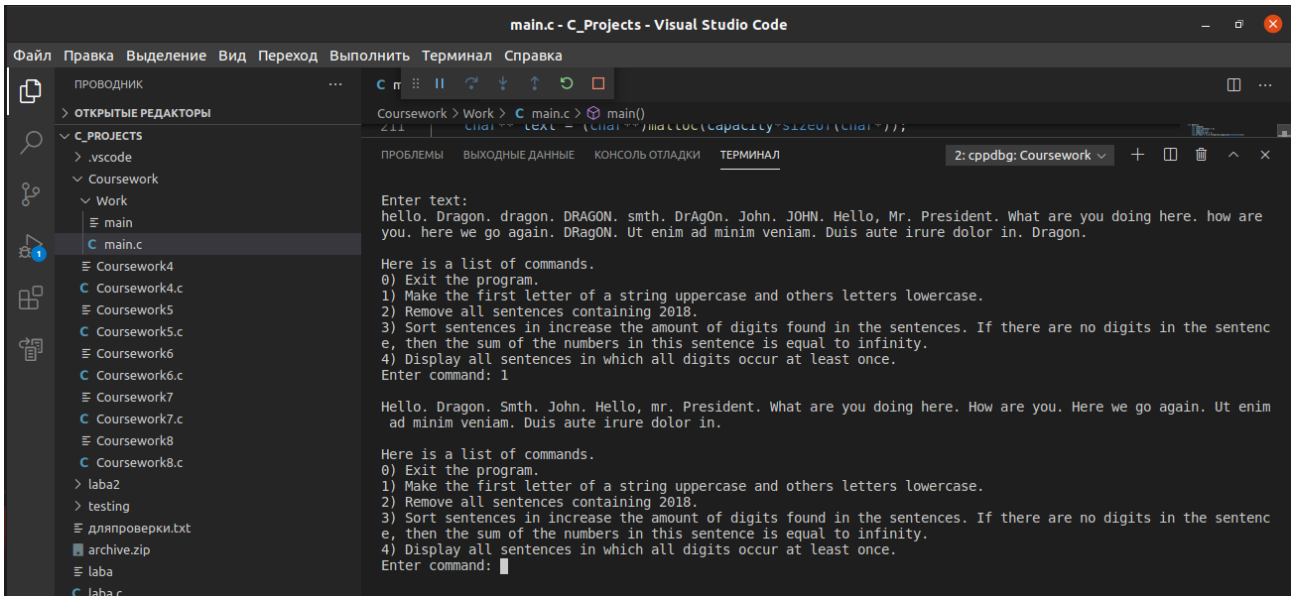
ПРИМЕР РАБОТЫ ПРОГРАММЫ



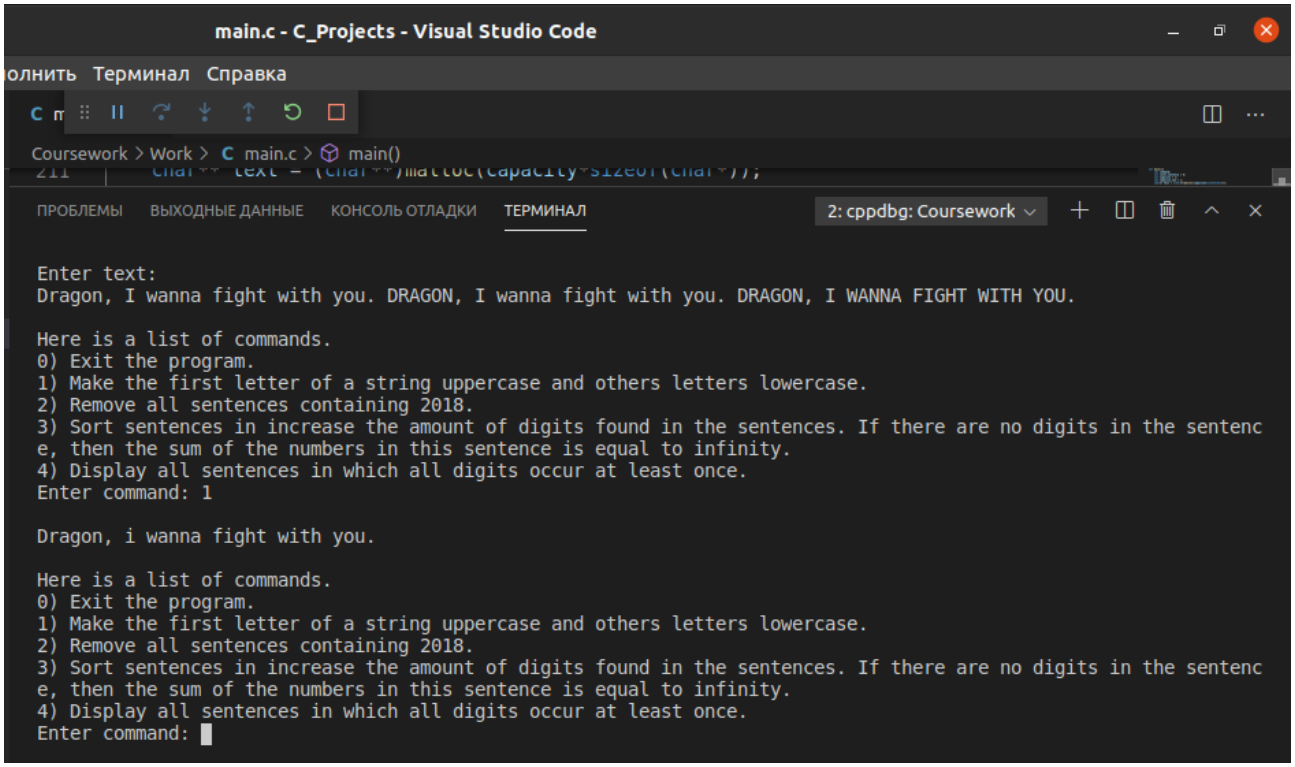
Скриншот №1. Программа выводит подсказку для пользователя сразу после запуска.



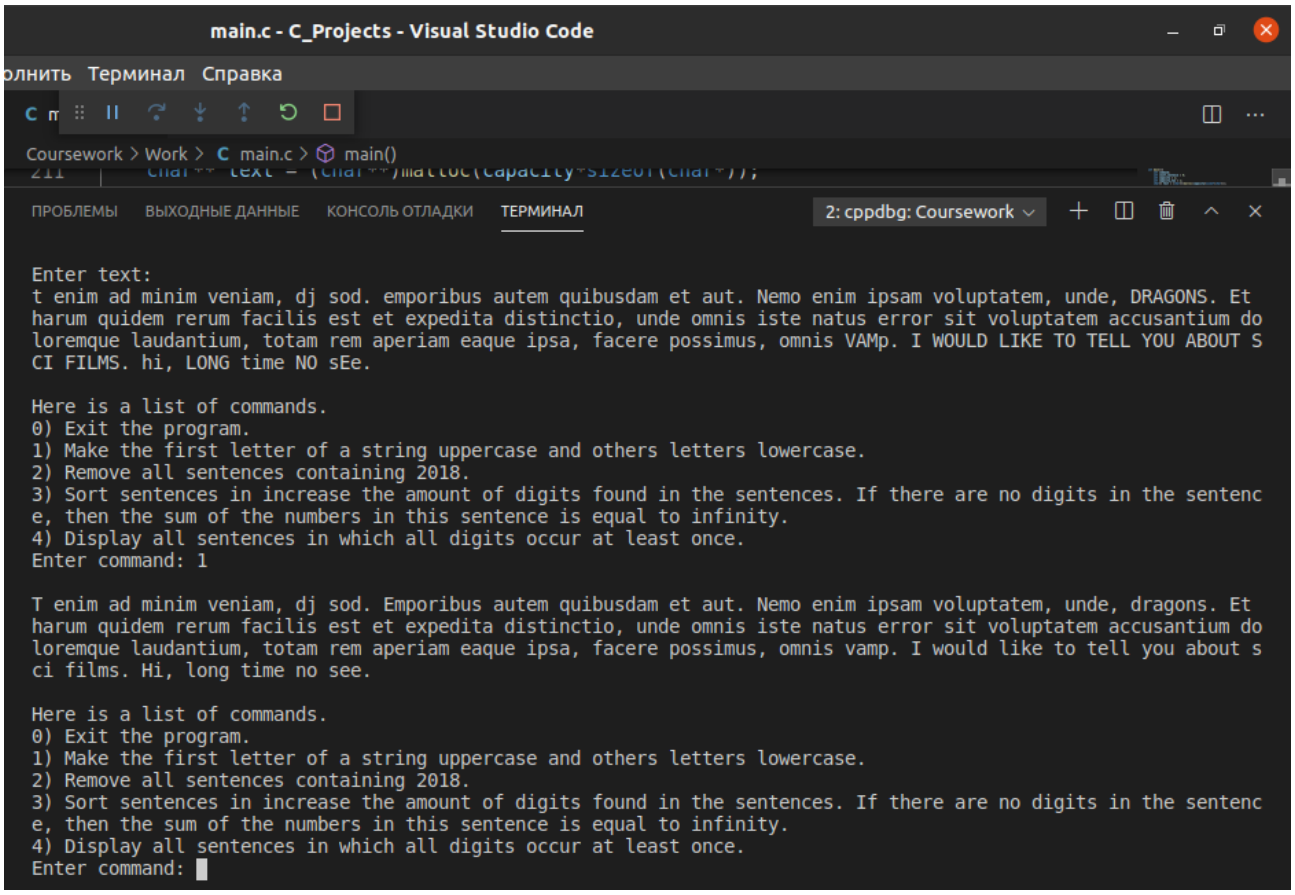
Скриншот №2. Программа ввыводит подсказку для выбора команды пользователя.



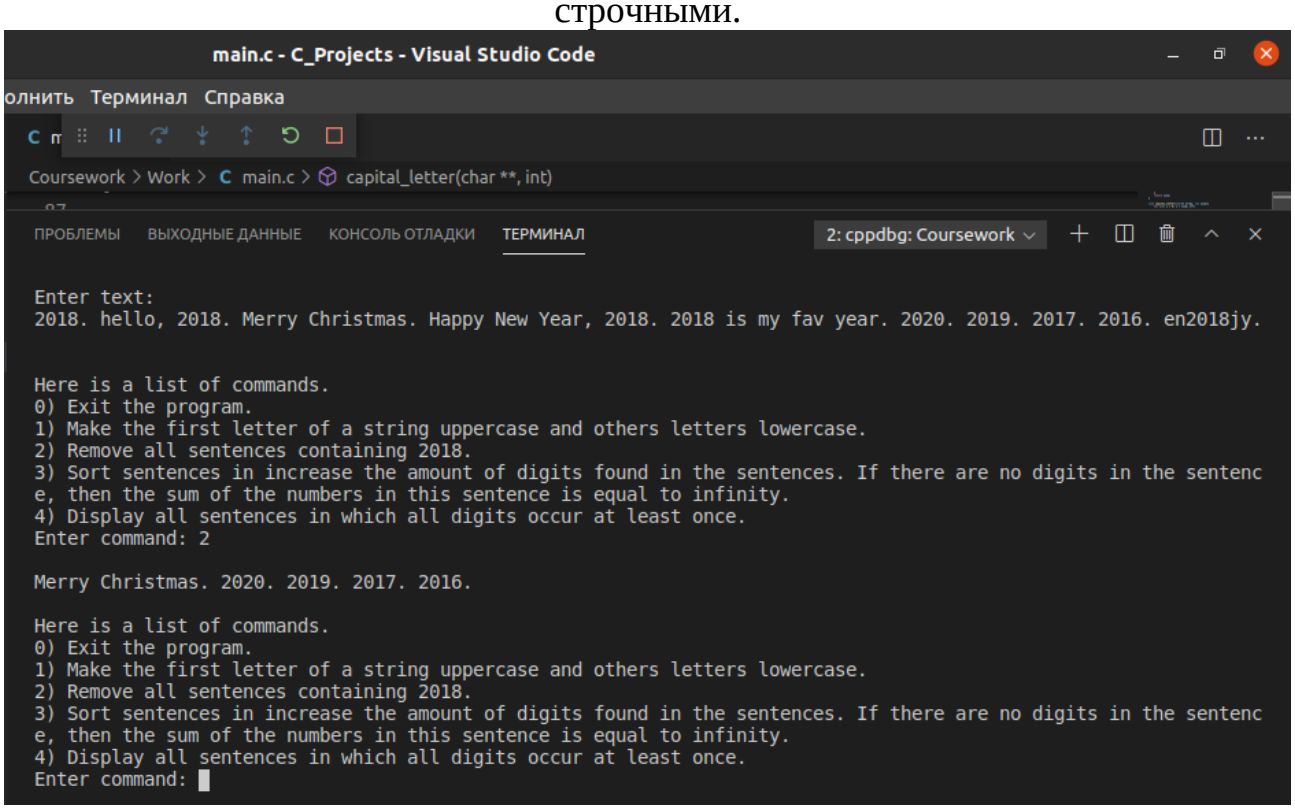
Скриншот №3. Программа удаляет все повторно всречающиеся предложение (без учёта регистра)



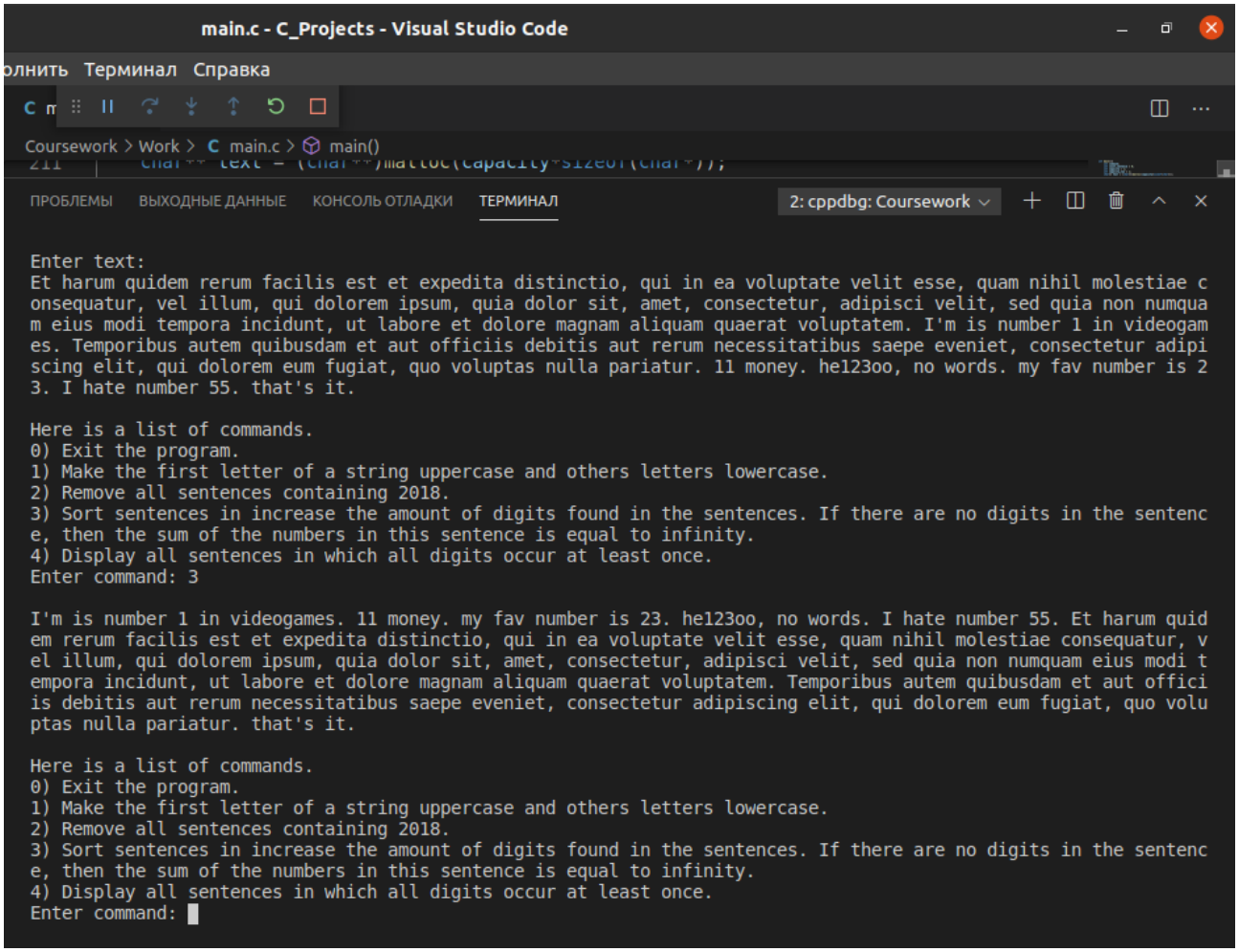
Скриншот №4. Программа удаляет все повторно всречающиеся предложение (без учёта регистра) даже предложения с пробелами или с запятыми.



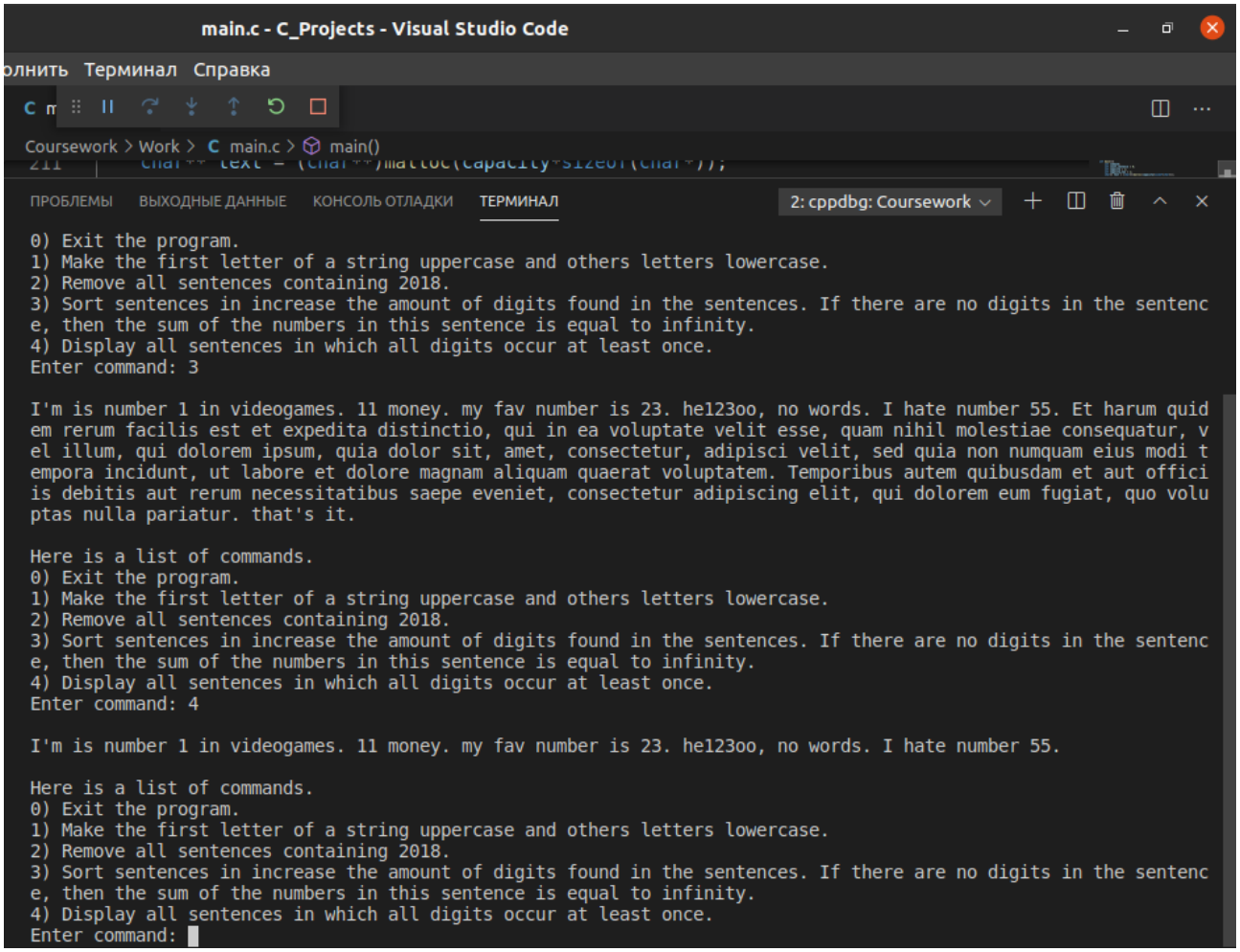
Скриншот №5. Программа преобразовывает предложения так, чтобы каждое слово в нем начиналось с заглавной буквы, а остальные символы слова были строчными.



Скриншот №6. Программа удаляет все предложения, в которых есть число 2018 (даже если оно внутри какого-то слова).



Скриншот №7. Программа сортирует предложения по увеличению суммы цифр встречаемых в предложении. Если в предложении нету цифр, то сумма цифр данного предложения равна бесконечности.



Скриншот №8 Программа выводит на экран только те предложения, в которых встречается все цифры хотя бы один раз.

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название программы: main.c

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

char** in(int* p_count_sentence, int capacity, int capacity_two, char**
text){
int count_word = 0;
char c;
int start = 1;
text[0] = (char*)malloc(capacity_two * sizeof(char));
while ((c = getchar()) != '\n'){
if (count_word == capacity_two)
{
capacity_two += 100;
char* ArrTemp = (char*)realloc(text[*p_count_sentence],
capacity_two*sizeof(char));
if (ArrTemp != NULL)
{
text[*p_count_sentence] = ArrTemp;
}
}
if (*p_count_sentence == capacity)
{
capacity += 100;
char** ArrTemp_two = (char**)realloc(text, capacity*sizeof(char*));
if (ArrTemp_two != NULL)
{
text = ArrTemp_two;
}
}
if(c == '.'){
text[*p_count_sentence][count_word++] = c;
text[*p_count_sentence][count_word++] = '\0';
*p_count_sentence += 1;
text[*p_count_sentence] = malloc(capacity_two*sizeof(char));
count_word = 0;
start = 1;
}
else if(c == ' ' && start){ ; }
else{
start = 0;
text[*p_count_sentence][count_word] = c;
```

```
count_word++;
}
}
return text;
}

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

int delete_repeating_sentence(char** array, int size){
int i, j, k;
for (i = 0; i < size - 1; i++)
{
for (j = i + 1; j < size; j++)
{
if (!(strcasecmp(array[j], array[i]))) // работает только на Линукс
{
for (k = j; k < size - 1; k++)
{
array[k] = array[k + 1];
}
i = 0;
j = 0;
size--;
}
}
}
char** ArrT = (char**)realloc(array, size*sizeof(char*));
if (ArrT != NULL)
{
array = ArrT;
}
return size;
}

void capital_letter(char** array, int size){
for (int i = 0; i < size; i++)
{
if (array[i][0] >= 'a' && array[i][0] <= 'z')
{
array[i][0] = (char)(toupper(array[i][0]));
}
for (int j = 1; array[i][j] != '.'; j++)
{ // Если сделать остальные символы строчными см в комментарии
```

```
if (array[i][j] >= 'A' && array[i][j] <= 'A')

{
array[i][j] = (char)(tolower(array[i][j]));

}
}
}
}

int delete2018(char** array, int size){
int i, j, k;
for (i = 0; i < size; i++)
{
if (strlen(array[i]) > 4){
for (j = 0; array[i][j+3] != '.'; j++)
{
if (array[i][j] == '2' && array[i][j+1] == '0' && array[i][j+2] == '1' &&
array[i][j+3] == '8')
{
for (k = i; k < size - 1; k++)
{
array[k] = array[k+1];
}
i = 0;
j = -1;
size = size - 1;
}
}
}
}
char** ArrTe = (char**)realloc(array, size*sizeof(char*));
if (ArrTe != NULL)
{
array = ArrTe;
}
return size;
}

void digits(char** array, int size){
char symbol;
char** sentence;
int was_digit = 0;
for (int i = 0; i < size; i++)
{
for (int j = 0; array[i][j] != '.'; j++){
symbol = array[i][j];
if (isdigit(symbol))
{
```

```
sentence = &array[i];
was_digit = 1;
break;
}
}
if (was_digit == 1){
printf("%s ", *sentence);
was_digit = 0;
}
}
}

int cmp(const void *a, const void *b){
char** first_sentence = (char**) a;
char** second_sentence = (char**) b;
long long int sum_first = 0;
long long int sum_second = 0;
int only_str_one = 1;
int only_str_two = 1;
for (int i = 0; (*first_sentence)[i] != '.'; i++)
{
if (isdigit((*first_sentence)[i]))
{
sum_first += (*first_sentence)[i] - '0';
only_str_one = 0;
}
}
for (int i = 0; (*second_sentence)[i] != '.'; i++)
{
if (isdigit((*second_sentence)[i]))
{
sum_second += (*second_sentence)[i] - '0';
only_str_two = 0;
}
}
if (sum_first > sum_second)
{
if (only_str_two)
{
return -1;
}
else
{
return 1;
}
}
else if (sum_first < sum_second)
{
if (only_str_one)
{
```

```
return 1;
}
else
{
return -1;
}
}
else
{
return 0;
}
}

int main(){
int command;
int count_sentence = 0;
int capacity = 1;
int capacity_two = 1;
char** text = (char**)malloc(capacity*sizeof(char*));
printf("Enter text:\n");
text = in(&count_sentence, capacity, capacity_two, text);
count_sentence = delete_repeating_sentence(text, count_sentence);
int stop = 1;
while (stop){
printf("\nHere is a list of commands.\n");
printf("0) Exit the program.\n");
printf("1) Make the first letter of a string uppercase ");
printf("and others letters lowercase.\n");
printf("2) Remove all sentences containing 2018.\n");
printf("3) Sort sentences in increase the amount of digits found in the
sentences. ");
printf("If there are no digits in the sentence,");
printf(" then the sum of the numbers in this sentence is equal to
infinity.\n");
printf("4) Display all sentences in which all digits occur at least
once.\n");
printf("Enter command: ");
scanf("%d", &command);
switch (command)
{
case 0:
stop = 0;
break;
case 1:
capital_letter(text, count_sentence);
out(text, count_sentence);
break;
case 2:
count_sentence = delete2018(text, count_sentence);
out(text, count_sentence);
```

```
break;
case 3:
qsort(text, count_sentence, sizeof(char*), cmp);
out(text, count_sentence);
break;
case 4:
printf("\n");
digits(text, count_sentence);
printf("\n");
break;
default:
printf("Something went wrong.\n");;
}
}
for (int i = 0; i < count_sentence; i++)
{
free(text[i]);
}
free(text);
return 0;
}
```