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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа со строками в языке СИ

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Павлов Д.Р.

Группа 1304

Тема работы : Работа со строками в языке СИ

Исходные данные:

Строка состоящая из латинский букв или цифр. Ввод строки заканчивается символом переноса строки .

Содержание пояснительной записки:

Введение.

Основные теоретические положения.

Описание кода программы.

Заключение.

Список используемых источников.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 14.12.2021

Дата защиты реферата: 16.12.2021

Студент _____ Павлов Д.Р.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Курсовая работа представляет из себя программу, предназначенную для работы со строками. Форматирование строк происходит с помощью шести функций. Пользователю дается выбор из трех функций, каждая из которых отвечает за разное форматирование строк. Код программы написан на языке программирования Си, запуск программы выполняется на операционных системах семейства Linux. При написании программы использовались управляющие конструкции и стандартные библиотеки языка программирования Си. Для ввода строки использовалась отдельная функции. Для проверки правильной работы программы проводилось тестирование. Исходный код, скриншоты, показывающие корректную работу программы, и результаты тестирования представлены в приложениях.

СОДЕРЖАНИЕ

	Введение	4
1.	Основные теоретические положения	7
1.1.	Основные управляющие конструкции языка Си.	7
1.2.	Функции стандартных библиотек языка Си.	8
2.	**Описание кода программы**	10
2.1.	Функция проверки строки(химической формулы) на корректность	10
2.2.	Функция считывания строки	12
2.3.	Функция возврата сокращенной формулы	14
2.4.	Функция раскрытия сокращенной химической формулы	17
2.5.	Функция сравнения сокращенной и раскрытой формулы	22
2.6.	Функция main	23
	Заключение	27
	Список использованных источников	28
	Приложение А. Название приложения	29
	Приложение Б. Результаты тестирования программы	43

ВВЕДЕНИЕ

Цель работы — написание программы для считывания и редактирования строки. Для достижения поставленной цели необходимо:

1) Изучить теоретический материал
2) Разработать код
3) Написать код
4) Протестировать программу

Задание:

Запись химической реакции всегда содержит описания нескольких веществ. В свою очередь, описание одного химического вещества - строка, в которой входящие в него атомы химических элементов перечисляются в определенном порядке. При этом последовательности из двух и более одинаковых атомов, идущих подряд, группируются: записывается сокращенное название химического элемента и количество одинаковых элементов подряд. Например, вместо HH пишут H2. Обозначения химических элементов состоят из одной или двух английских букв, из которых первая - прописная, а вторая - строчная.

Требуется самостоятельно (без использования стандартных и сторонних библиотек) реализовать следующие функции:

1. Функция принимающая строку представляющую химическую формулу и возвращающая ее сокращенную формулу.
2. Функция принимающая строку представляющая сокращенную формулу и возвращающая ее полное представление.
3. Функция принимающая строку представляющую полную химическую формулу и строку представляющую сокращенную химическую формулу. Функция должна сообщить, являются ли данные формулы идентичными.

   Функции должны поддерживать защиту от выхода за границу буфера и иметь механизм уведомления о некорректной ситуации.

Для демонстрации работы функций, требуется написать диалоговую программу, которая должна предлагать пользователю выбрать тестируемую

функцию, а после - запрашивать у пользователя строку (строки), которую надо передать в качестве аргумента (аргументов). Одним из вариантов выбора следует предусмотреть завершение программы.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

1.1. Основные управляющие конструкции языка Си.

Инструкция if-else используется для принятия решения. Формально ее синтаксисом является:

if (выражение) инструкция1

Else

инструкция2

причем **else**-часть может и отсутствовать. Сначала вычисляется выражение, и, если оно истинно (т. е. отлично от нуля), выполняется инструкция1. Если выражение ложно (т. е. его значение равно нулю) и существует else-часть, то выполняется инструкция2.

Оператор множественного выбора **switch** (<выражение>)

{ **case** <*константное выражение 1*>: <*операторы 1*>

...

case <*константное выражение N*>: <*операторы N*>

[**default:** <*операторы*>]
}

Выполняет поочередное сравнение выражения со списком константных выражений. При совпадении, выполнение программы начинается с соответствующего оператора. В случае, если совпадений не было, выполняется необязательная ветка **default.** Важно помнить, что операторы после первого совпадения будут выполняться далее один за другим. Чтобы этого избежать, следует использовать оператор **break.**

Цикл с предусловием
while (<*выражение*>) <*оператор*>
На каждой итерации цикла происходит вычисление выражения и если оно истинно, то выполняется тело цикла.

Цикл со счетчиком
for ([<*начальное выражение*>]; [<*условное выражение*>]; [<*выражение приращения*>])

<*оператор*>

Условием продолжения цикла, как и в цикле с предусловием, является некоторое выражение, однако в цикле со счетчиком есть еще 2 блока — начальное выражение, выполняемое один раз перед первым началом цикла и выражение приращения, выполняемое после каждой итерации цикла. Любая из трех частей оператора for может быть опущена.

Оператор break — досрочно прерывает выполнение цикла.
Оператор continue — досрочный переход к следующей итерации цикла.

1.2. Функции стандартных библиотек языка Си.

stdio.h:

• getchar() - считывает и возвращает символ из данного потока и изменяет указатель позиции файла.

- printf() - используются для вывода в стандартный поток вывода.
- sprintf() - используются для вывода в массив типа char
- puts() – выводит символьную строчку в stdout.
- scanf() – используется для вывода из стандратного потока ввода.

Stdlib.h:

Malloc() - принимает один аргумент: размер блока памяти в байтах. Возвращает указатель на выделенный блок памяти.

Realloc() - выполняет перераспределение блоков памяти.

Free() - принимает один аргумент: указатель на блок памяти. Очищает данный блок памяти.

Atoi() - переводит строку в число

String.h:

Strcat() - склеивает две строки

Strlen() - возвращает длину строки

Strcmp() - проверяет являются ли две строки идентичными

Ctype.h:

Isupper() - проверяет состоит ли строка из заглавных букв

Islower() - проверяет состоит ли строка из маленьких букв

Isdigit() - проверяет состоит ли строка из цифр

2. ОПИСАНИЕ КОДА ПРОГРАММЫ

2.1. Функция проверки строки(химической формулы) на корректность

```
int is_string_correct(char* substance, int option)
{
   if (option == 1) {
      int i = 0;
      int flag = 1;
      while (substance[i] != '\0') {
         if (!((isupper(substance[i])) && !(isupper(substance[i + 1])) ||
              (!(isupper(substance[i])) && (isupper(substance[i - 1])) && (i != 0)) ||
              (isupper(substance[i]) && isupper(substance[i+1])))) {
            flag = 0;
            break;
         }
         i++;
      }

      if (flag == 1) {
         return 1;
      } else {
         return 0;
      }


   }else if (option == 2){         // 2nd option
      int i = 0;
      int flag = 1;
      while (substance[i] != '\0') {
         if (!(((isupper(substance[i])) && !(isupper(substance[i + 1])) ||
              (!(isupper(substance[i])) && (isupper(substance[i - 1])) && (i != 0)) ||
```

```
                    (isupper(substance[i]) && isupper(substance[i+1]))) ||
(isdigit(substance[i]) )) ) {
                        flag = 0;
                        break;
                    }
                    i++;
                }
                if (flag == 1) {
                    return 1;
                } else {
                    return 0;
                }



            }

        }
```

Данная функция проверяет на корректность нашу строку, которая из себя представляет химическую формулу. Другими словами, если строка не представлена в виде традиционной хим. Формулы, она возвращает 0. К примеру: строка, состоящая из хотя бы одно случая рядом стоящих маленьких букв, будет возвращать 0. Данная функция имеет два аргумента — само вещество (строка) и опция, представленная в виде целого числа. Опция зависит от выбранной нами функцией, поскольку в случае, если мы выбираем функция сокращения (первая задача)(option = 1) — наша исходная строка не должна состоять из цифр и не иметь случаев, когда две маленькие буквы стоят рядом. Для второй задачи(option = 2) наша исходная строка должна состоять из химических элементов, либо же из цифр, для нее все так же будет действовать правило, что две маленькие буквы не стоят рядом.

Работа данной функции заключается в следующем — сначала создаем условия, при опции == 1 выполняется один алгоритм, а для 2 — другой.

1) мы создаем две переменный типа int — i и flag(=1). Переменная flag будет отвечать за вхождение некорректной ситуации в нашу исходную строку. Далее мы задаем цикл, который будет выполнятся пока наш символ в строке не станет равным символу конца строки, и уже в этом цикле мы задаем условие со всеми случаями некорректности. Если в строке встречается некорректная ситуация — flag приравнивается нулю. Далее после цикла идет условие — если flag по прежнему равен единице — мы возвращаем 1, в противном случае — возвращается 0.

2) для второй опции все аналогично первой. Единственное — добавляется условие в некорректности строки, которое разрешает использование в строке чисел(если только число не стоит на первой позиции).

2.2. Функция считывания строки

```
char* read_substance()
{
   int size = 30;
   char *sub = malloc(sizeof(char)*size);
   char* temp;
   int i = 0;
   char symbol;
   scanf(" ");

   do{
      symbol = getchar();
      if (symbol == '\n'){
```

```
            //sub[i] = '\0';
            break;

        }

        sub[i] = symbol;
        i++;

        if (i == size){
            size += 5;
            temp = realloc(sub, sizeof(char)*size);
            if (temp != NULL){
                sub = temp;
            }else{
                free(sub);
                return NULL;
            }


        }

    }while (1);
    sub[i] = '\0';

    return  sub;
}
```

Данная функция будет отвечать за считывание строки. Она посимвольно считывает символы и заносит их в динамический массив *sub. Память в *sub выделяется блоками по 30 байт, в целях экономии времени.

2.3.Функция возврата сокращенной формулы

```
char* first_task_cut(char* sub)
{
   if (is_string_correct(sub, 1) == 1) {
      char **new_sub = NULL;
      int i = 0;
      int j = 0;
      while (1) {
         new_sub = realloc(new_sub, sizeof(char *) * (i + 1));
         new_sub[i] = NULL;
         new_sub[i] = realloc(new_sub[i], sizeof(char));
         new_sub[i][0] = sub[j];
         j++;
         if (sub[j] == '\0') {
            i++;
            new_sub = realloc(new_sub, sizeof(char*) * (i + 1));
            new_sub[i] = malloc(sizeof(char)*3);
            sprintf(new_sub[i], "%s", "!!!");
            break;
         } else if ( !( isupper(sub[j]) ) ) {
            new_sub[i] = realloc(new_sub[i], sizeof(char) * 3);
            new_sub[i][1] = sub[j];
            new_sub[i][2] = '\0';
            j++;
         } else {
            new_sub[i] = realloc(new_sub[i], sizeof(char) * 2);
            new_sub[i][1] = '\0';
         }
         i++;
```

```
}


//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


char *result = malloc(sizeof(char) * 300);
char *tmp = malloc(sizeof(char));
int n = 0;
while (strcmp(new_sub[n], "!!!") != 0) {

    int counter = 1;
    int m = n + 1;
    while (strcmp(new_sub[n], new_sub[m]) == 0) {
        counter++;
        m++;
    }

    if (counter != 1) {
        sprintf(tmp, "%s%d", new_sub[n], counter);
        strcat(result, tmp);
        n = m;
        continue;
    } else {
        sprintf(tmp, "%s", new_sub[n]);
        strcat(result, tmp);
    }
```

```
            n++;
        }


        return result;
    }else{
        return NULL;
    }
}
```

Данная функция будет отвечать за решение первой задачи — сокращение химической формулы. Сначала мы проверяем нашу строку на корректность при помощи нашей написанной ранее функции (2.1.), вторым аргументом которого является число 1. Если оно не верное — мы просто возвращаем NULL, в противном случае мы поступаем следующим образом:
Сначала мы поэлементно разбиваем нашу строку и заносим результат в двумерный массив. Для этого мы инициализируем переменную в который будем записывать наши элементы далее так же инициализирвуем две переменные типа int i и j — i будет отвечать за порядок в двумерном массиве, а j за индекс элемента в нашей исходной строке. Далее мы создаем бесконечный цикл, внутри мы выделяем память для нашего двумерного массива и далее приравниваем new_sub[i][0] значение sub[j], после прибавляем 1 к j. Далее мы проверяем является ли sub[j] символом конца строки, если это так, то мы увеличиваем i на 1, и еще добавляем память для двумерного массива. Далее мы записываем в i массив строку «!!!», она нам понадобится для дальнейших действий. Далее идет другое условие, в случае если прошлое неверно — если j-ы символ начальной строки является маленькой буквой. В таком случае мы выделяем память для i-го массив и записываем в индекс 1 j-ый элемент начальной строки. После этого добавляем в конец символ конца строки. Если оба условия неверный — мы так же выделяем доп память для iго массива и

записываем в конец символ конца строки. После всех условий увеличиваем i на 1.

Далее идет часть функции, которая отвечает за подсчет количества повторений в нашем двумерном массиве. Для этого я инициализирую два массива типа char(result, tmp) и выделяю для них память. После этого инициализирую переменную типа int (n), равную нулю(n будет значить индекс элемента, который будет проверяться на повторения). Далее мы создаем цикл, который будет выполнятся пока н-ый элемент не будет равен «!!!». В цикле инициализируем две переменных типа int — counter(равную 1), счетчик повторений; и m (равную на 1 больше чем n), индекс элемента который позже будет сравниваться с n-ым элементом. Далее мы заводим уже другой цикл, который будет сравнивать элементы с индексом n и m и если вдруг окажутся неравными — заканчиваем цикл. Внутри этого цикла просто увлеичиваем counter и m на 1. После цикла мы проверяем чему равно значение counter: если оно вдруг окажется не равному одному, т.е прошлый цикл прошелся хотя бы 1 раз, то тогда мы заносим в наш массив tmp элемент с числом повторений, после к нашему результату так же добавляем уже tmp, после — приравниваем значение m к n. Делаем continue дабы избежать дальнейшего увеличения на 1. Если данное условие не выполнилось — делаем то же самое, только без continue и «склеиваем» без перменной counter. В конце нашего всего цикла «подсчета повторений» мы прибавлем к n 1. После цикла и в конце функции мы возвращаем массив result.

2.4.Функция раскрытия сокращенной химической формулы

```
char* second_task_uncover(char* sub)
{
   if (is_string_correct(sub, 2) == 1) {
      int size_of_result = 200;
      char* result = malloc(sizeof(char)*size_of_result);
```

```
char* tmp = malloc(sizeof(char)*50);
int i = 0;
int j;
while (i < strlen(sub) ) {


   if (isupper(sub[i]) && islower(sub[i+1])){
      int flag_one = 1;
      j = i+2;
      char* number_str = malloc(sizeof(char)*4);

      if (isdigit(sub[j])){
         strcat(number_str, &sub[j]);
         flag_one = 0;
      }

      if (flag_one == 0) {
         int number_int = atoi(number_str);
         for (int n = 0; n < number_int; n++) {//||||||||
            if (strlen(result) >= size_of_result){
               size_of_result += 200;
               result = realloc(result, sizeof(char)*size_of_result);
            }
            sprintf(tmp, "%c%c", sub[i], sub[i+1]);
            strcat(result, tmp);
         }
      }else{//||||||||||||||||||||||||||||||||||||||||||
         if (strlen(result) >= size_of_result){
            size_of_result += 200;
            result = realloc(result, sizeof(char)*size_of_result);
```

```
            }
            sprintf(tmp, "%c%c", sub[i], sub[i+1]);
            strcat(result, tmp);

          }//|||||||||||||||||||||||||||||||||||||||||||||||||||||



// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        }else if (isupper(sub[i]) && !(islower(sub[i+1]))){
          int flag_two = 1;
          j = i+1;
          char* number_str = malloc(sizeof(char)*4);

          if (isdigit(sub[j])){
            strcat(number_str, &sub[j]);
            flag_two = 0;
          }

          if (flag_two == 0){
            int number_int = atoi(number_str);
            for (int n = 0; n < number_int; n++){//******

              if (strlen(result) >= size_of_result){
                size_of_result += 200;
                result = realloc(result, sizeof(char)*size_of_result);
              }

              sprintf(tmp, "%c", sub[i]);
              strcat(result, tmp);
            }//*********************************************
```

```
        }else if (!(isdigit(sub[i]))){//^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*
}

            if (strlen(result) >= size_of_result){
                size_of_result += 200;
                result = realloc(result, sizeof(char)*size_of_result);
            }

            sprintf(tmp, "%c", sub[i]);
            strcat(result, tmp);
          }

        }//^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*

        if (strlen(result) >= size_of_result){
            size_of_result += 200;
            result = realloc(result, sizeof(char)*size_of_result);
        }
        i++;


    }

    //* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *



    return result;
  }else{
    return NULL;
  }
}
```

Данная функция будет отвечать за второе задание — преобразование сокращенной формулы в полную. Сначала мы как и с прошлой функцией будем проверять начальную формулу на корректность при помощи функции проверки (2.1.) только в качестве второго аргумента(option) будет число два, если она неверна — возвращаем NULL. Если формула все же корректно написана делаем следующее:

сначала инициализируем все нужные переменные — int size_of_result = 200, char* result (и сразу при помощи malloc выделяем память размером size_of_result), char* tmp (так же выделяем память размером 50), int i = 0(отвечает за индекс элемента в начальной формуле, которая поступалась на вход функции), int j. Далее задаем цилк пока i < длины начальной строки. Внутри цикла мы изначально проверяем, если i-ый элемент — заглавная буква, а i+1-ый — маленькая. ***Если это так то:***

инициализируем переменную flag_one ранвую 1, приравниваем к j значение i+2, инициализируем массив number_str и выделяем для него память в размере 4 байт. Далее мы проверяется является ли j-ый элемент начальной строки числом. Если это так то добавляем к str_num j-ый элемент, а flag_one приравнием к нулю. Далее проверяем чему равен flag_one. *Если он равен нулю* — инициализируем переменную number_int и приравнивем к ней числовое значение str_num. Далее создаем цикл for(int n = 0; n < number_int; n++), внутри него сначал проверяем на переполнение памяти, если что добавляем, и добавляем к массиву tmp значение i-го и i+1-го элемента начальной строки. После чего склеиваем наш result с переменной tmp. *Если же flag_one = 1 —* добавляем память в случае необходимости, приравнием к tmp i-ый и i+1-ый элемент и склеиваем reuslt и tmp.

Если это не так, то:

Фактические делаем все то же самое что и в случае, если это так. Единственное, когда добавляем элементы в tmp — добавляем только i-ый элемент.

В конце нашего цикла проверяем хватает ли памяти, если нет — добавляем; а после — прибавляем к i 1. После цикла возвращаем result.

2.5.Функция сравнения сокращенной и раскрытой формулы

```
void third_task_if_substances_are_equal(char* cut_sub, char* uncover_sub)
{
   char* cut_uncover_sub = second_task_uncover(uncover_sub);

   if (is_string_correct(cut_sub, 1) && is_string_correct(uncover_sub, 2)) {


      if (strcmp(cut_sub, cut_uncover_sub) == 0) {
         puts("[+]The substances are equal!\n");
      } else {
         puts("[-]The substances are NOT equal!\n");
      }
   }else {
      puts("[ERROR] Wrong input!\n");
   }
   free(cut_sub);
   free(cut_uncover_sub);
   free(uncover_sub);


}
```

Данная функция принимаем два значений — не сокращенная формула и сокращенная формула. Далее мы при помощи нашей прошлой функции (2.4.) инициализируем строку cut_uncover_sub, в которой будет хранится возвращаемое значение функции (2.4.) от аргумента «сокращенная формула». Далее мы при помощи функции (2.1.) проверяем правильно ли записаны формулы. Если это так, то проверяем, равна ли строка cut_uncover_sub строке

«не сокращенная формула». Если это так, выводим «[+]The substances are equal!», в противном случае выводим «[-]The substances are NOT equal!». Если же хотя бы одна из формул записана неправильно — выводим ошибку([ERROR] Wrong input!). В конце функции освобождаем память под каждый массив.

2.6. Функция main

```
int main()
{

   char* substance_cut;
   char* substance_uncover;
   char* result ;



   char answer;
   printf("Chose your function:\n");
   printf("type [1] if you want to CUT your substance\n");
   printf("type [2] if you want to UNCOVER your substance\n");
   printf("type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are
Equal\n");
   scanf(" ");
   scanf("%c", &answer);
   switch(answer){
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      case '1':
         printf("Type the substance you want to CUT\n");
         substance_cut = read_substance();
         if (substance_cut == NULL) {
            printf("[ERROR]Memory Cannot be Allocated!\n");
            break;
         }
         printf("Your substance is  %s!\n", substance_cut);

         result = first_task_cut(substance_cut);
```

```
            if (result != NULL){
                printf("\nYour Cutten Substance is: ");
                puts(result);
            }else{
                printf("[ERROR]The Substance is INCORRECT!\n");
                break;
            }
            free(substance_cut);
            free(result);
            break;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        case '2':
            printf("Type the substance you want to UNCOVER\n");
            substance_uncover = read_substance();
            if (substance_uncover == NULL) {
                printf("[ERROR]Memory Cannot be Allocated!\n");
                break;
            }
            printf("Your substance is  %s!\n", substance_uncover);

            result = second_task_uncover(substance_uncover);
            if (result != NULL){
                printf("\nYour Uncovered Substance is: ");
                puts(result);
            }else{
                printf("[ERROR]The Substance is INCORRECT!\n");
                break;
            }

            free(substance_uncover);
            free(result);
            break;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        case '3':
```

```
            printf("Type the substance you want number 1\n");
            substance_cut = read_substance();
            if (substance_cut == NULL) {
                printf("[ERROR]Memory Cannot be Allocated!\n");
                break;
            }
            printf("Type the substance you want number 2\n");
            substance_uncover = read_substance();
            if (substance_uncover == NULL) {
                printf("Memory Cannot be Allocated!\n");
                break;
            }
            printf("[1]Your substance number one is  %s!\n", substance_cut);
            printf("[2]Your substance number two is  %s!\n", substance_uncover);

            third_task_if_substances_are_equal(substance_cut, substance_uncover);

            break;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        default:
            printf("[ERROR]Wrong answer!\n");
            break;

    }

    return 0;
}
```

Данная функция отвечает за основные действия со стороны пользователя. Сначал мы инициализируем три строки — substance_cut, substance_uncover, result. Далле инициализируем answer типа char. Далее мы выводим на экран инструкцию, что должен ввести пользователь и заносим в переменную answer ответ пользователя. Далее мы смотрим на сам ответ:

Если answer — 1: Выводим на экран просьбу напечатать строку, которую хотим сократить, посел чего считываем строку, если она равна NULL, выводится что нельзя выделить память и завершаем цикл. Если такого нет, то выводим на экран строку введенную пользователем. Далее на эту строку применяем функцию (2.3.) и записывем резульат в result. Если result не равен NULL, выводим нашу сокращенную строку, в противном случае — выводим ошибку, что введенная пользователем формула некорректна. Далее освобождаем память под result и substance_cut. Для answer — 2 все то же самое, только мы используем функию (2.4.). Для answer — 3 мы сначали списываем две строки (все так же с проверкой если строки != NULL), далее выводим пользователю две строки и проверяем их при помощи функции (2.5.), после чего делаем break. Для значения по умолчанию мы просто выводим ошибку «[ERROR]Wrong answer!», и далее далем break. В конце функция возвращает 0.

ЗАКЛЮЧЕНИЕ

Для успешного достижения поставленной цели — написания программы для считывания и редактирования строки , соответствующей заданию курсовой работы, были выполнены соответствующие задачи:

1. Изучен теоретический материал по теме курсовой работы.
2. Разработан программный код.
3. Реализован программный код.
4. Проведено тестирование программы.

Исходный код программы представлен в приложении А, результаты тестирования - в приложении Б.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

2. Основы программирования на языках Си и C++ [Электронный ресурс URL: http://cplusplus.com

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#INCLUDE <STDIO.H>
#INCLUDE <STDLIB.H>
#INCLUDE <STRING.H>
#INCLUDE <CTYPE.H>

INT IS_STRING_CORRECT(CHAR* SUBSTANCE, INT OPTION)
{
   IF (OPTION == 1) {
      INT I = 0;
      INT FLAG = 1;
      WHILE (SUBSTANCE[I] != '\0') {
         IF (!((ISUPPER(SUBSTANCE[I])) && !(ISUPPER(SUBSTANCE[I + 1])) ||
             (!(ISUPPER(SUBSTANCE[I])) && (ISUPPER(SUBSTANCE[I - 1]))
&& (I != 0)) ||
             (ISUPPER(SUBSTANCE[I]) && ISUPPER(SUBSTANCE[I+1])))) {
            FLAG = 0;
            BREAK;
         }
         I++;
      }

      IF (FLAG == 1) {
         RETURN 1;
      } ELSE {
         RETURN 0;
      }
```

```
   }ELSE IF (OPTION == 2){         // 2ND OPTION
      INT I = 0;
      INT FLAG = 1;
      WHILE (SUBSTANCE[I] != '\0') {
         IF (!(((ISUPPER(SUBSTANCE[I])) && !(ISUPPER(SUBSTANCE[I + 1])) ||
               (!(ISUPPER(SUBSTANCE[I])) && (ISUPPER(SUBSTANCE[I - 1]))
&& (I != 0)) ||
               (ISUPPER(SUBSTANCE[I]) && ISUPPER(SUBSTANCE[I+1]))) ||
(ISDIGIT(SUBSTANCE[I]) )
               && !(ISDIGIT(SUBSTANCE[0])) )  ) {
            FLAG = 0;
            BREAK;
         }
         I++;
      }
      IF (FLAG == 1) {
         RETURN 1;
      } ELSE {
         RETURN 0;
      }




   }

}

CHAR* READ_SUBSTANCE()
{
   INT SIZE = 30;
```

```
CHAR *SUB = MALLOC(SIZEOF(CHAR)*SIZE);
CHAR* TEMP;
INT I = 0;
CHAR SYMBOL;
SCANF(" ");

DO{
   SYMBOL = GETCHAR();
   IF (SYMBOL == '\N'){
      //SUB[I] = '\0';
      BREAK;

   }

   SUB[I] = SYMBOL;
   I++;

   IF (I == SIZE){
      SIZE += 5;
      TEMP = REALLOC(SUB, SIZEOF(CHAR)*SIZE);
      IF (TEMP != NULL){
         SUB = TEMP;
      }ELSE{
         FREE(SUB);
         RETURN NULL;
      }


   }
```

```
   }WHILE (1);
   SUB[I] = '\0';

   RETURN  SUB;
}




CHAR* FIRST_TASK_CUT(CHAR* SUB)
{
   IF (IS_STRING_CORRECT(SUB, 1) == 1) {
      CHAR **NEW_SUB = NULL;
      INT I = 0;
      INT J = 0;
      WHILE (1) {
         NEW_SUB = REALLOC(NEW_SUB, SIZEOF(CHAR *) * (I + 1));
         NEW_SUB[I] = NULL;
         NEW_SUB[I] = REALLOC(NEW_SUB[I], SIZEOF(CHAR));
         NEW_SUB[I][0] = SUB[J];
         J++;
         IF (SUB[J] == '\0') {
            I++;
            NEW_SUB = REALLOC(NEW_SUB, SIZEOF(CHAR*) * (I + 1));
            NEW_SUB[I] = MALLOC(SIZEOF(CHAR)*3);
            SPRINTF(NEW_SUB[I], "%S", "!!!");
            BREAK;
         } ELSE IF ( !( ISUPPER(SUB[J]) ) ) {
            NEW_SUB[I] = REALLOC(NEW_SUB[I], SIZEOF(CHAR) * 3);
```

```
      NEW_SUB[I][1] = SUB[J];
      NEW_SUB[I][2] = '\0';
      J++;
   } ELSE {
      NEW_SUB[I] = REALLOC(NEW_SUB[I], SIZEOF(CHAR) * 2);
      NEW_SUB[I][1] = '\0';
   }
   I++;


}


//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


CHAR *RESULT = MALLOC(SIZEOF(CHAR) * 300);
CHAR *TMP = MALLOC(SIZEOF(CHAR));
INT N = 0;
WHILE (STRCMP(NEW_SUB[N], "!!!") != 0) {

   INT COUNTER = 1;
   INT M = N + 1;
   WHILE (STRCMP(NEW_SUB[N], NEW_SUB[M]) == 0) {
      COUNTER++;
      M++;
   }

   IF (COUNTER != 1) {
      SPRINTF(TMP, "%S%D", NEW_SUB[N], COUNTER);
```

```
            STRCAT(RESULT, TMP);
            N = M;
            CONTINUE;
         } ELSE {
            SPRINTF(TMP, "%S", NEW_SUB[N]);
            STRCAT(RESULT, TMP);
         }


         N++;
      }


      RETURN RESULT;
   }ELSE{
      RETURN NULL;
   }
}




CHAR* SECOND_TASK_UNCOVER(CHAR* SUB)
{
   IF (IS_STRING_CORRECT(SUB, 2) == 1) {
      INT SIZE_OF_RESULT = 200;
      CHAR* RESULT = MALLOC(SIZEOF(CHAR)*SIZE_OF_RESULT);
      CHAR* TMP = MALLOC(SIZEOF(CHAR)*50);
      INT I = 0;
      INT J;
```

```
    WHILE (I < STRLEN(SUB) ) {


      IF (ISUPPER(SUB[I]) && ISLOWER(SUB[I+1])){
        INT FLAG_ONE = 1;
        J = I+2;
        CHAR* NUMBER_STR = MALLOC(SIZEOF(CHAR)*4);

        IF (ISDIGIT(SUB[J])){
          STRCAT(NUMBER_STR, &SUB[J]);
          FLAG_ONE = 0;
        }

        IF (FLAG_ONE == 0) {
          INT NUMBER_INT = ATOI(NUMBER_STR);
          FOR (INT N = 0; N < NUMBER_INT; N++) {//||||||||
            IF (STRLEN(RESULT) >= SIZE_OF_RESULT){
              SIZE_OF_RESULT += 200;
              RESULT = REALLOC(RESULT,
SIZEOF(CHAR)*SIZE_OF_RESULT);
            }
            SPRINTF(TMP, "%C%C", SUB[I], SUB[I+1]);
            STRCAT(RESULT, TMP);
          }
        }ELSE{//||||||||||||||||||||||||||||||||||||||||||||||
          IF (STRLEN(RESULT) >= SIZE_OF_RESULT){
            SIZE_OF_RESULT += 200;
            RESULT = REALLOC(RESULT,
SIZEOF(CHAR)*SIZE_OF_RESULT);
          }
```

```
            SPRINTF(TMP, "%C%C", SUB[I], SUB[I+1]);
            STRCAT(RESULT, TMP);

          }//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        }ELSE IF (ISUPPER(SUB[I]) && !(ISLOWER(SUB[I+1]))){
          INT FLAG_TWO = 1;
          J = I+1;
          CHAR* NUMBER_STR = MALLOC(SIZEOF(CHAR)*4);

          IF (ISDIGIT(SUB[J])){
            STRCAT(NUMBER_STR, &SUB[J]);
            FLAG_TWO = 0;
          }

          IF (FLAG_TWO == 0){
            INT NUMBER_INT = ATOI(NUMBER_STR);
            FOR (INT N = 0; N < NUMBER_INT; N++){//******

              IF (STRLEN(RESULT) >= SIZE_OF_RESULT){
                SIZE_OF_RESULT += 200;
                RESULT = REALLOC(RESULT,
SIZEOF(CHAR)*SIZE_OF_RESULT);
              }

              SPRINTF(TMP, "%C", SUB[I]);
              STRCAT(RESULT, TMP);
            }//****************************************
```

```
        }ELSE IF (!(ISDIGIT(SUB[I]))){//^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*

            IF (STRLEN(RESULT) >= SIZE_OF_RESULT){
               SIZE_OF_RESULT += 200;
               RESULT = REALLOC(RESULT,
SIZEOF(CHAR)*SIZE_OF_RESULT);
            }

            SPRINTF(TMP, "%C", SUB[I]);
            STRCAT(RESULT, TMP);
          }

        }//^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*

        IF (STRLEN(RESULT) >= SIZE_OF_RESULT){
           SIZE_OF_RESULT += 200;
           RESULT = REALLOC(RESULT, SIZEOF(CHAR)*SIZE_OF_RESULT);
        }
        I++;


      }

      //* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


      RETURN RESULT;
    }ELSE{
      RETURN NULL;
    }
```

```
}



VOID THIRD_TASK_IF_SUBSTANCES_ARE_EQUAL(CHAR* CUT_SUB,
CHAR* UNCOVER_SUB)
{
   CHAR* CUT_UNCOVER_SUB =
SECOND_TASK_UNCOVER(UNCOVER_SUB);

   IF (IS_STRING_CORRECT(CUT_SUB, 1) &&
IS_STRING_CORRECT(UNCOVER_SUB, 2)) {


      IF (STRCMP(CUT_SUB, CUT_UNCOVER_SUB) == 0) {
         PUTS("[+]THE SUBSTANCES ARE EQUAL!\N");
      } ELSE {
         PUTS("[-]THE SUBSTANCES ARE NOT EQUAL!\N");
      }
   }ELSE {
      PUTS("[ERROR] WRONG INPUT!\N");
   }
   FREE(CUT_SUB);
   FREE(CUT_UNCOVER_SUB);
   FREE(UNCOVER_SUB);



}
```

```
INT MAIN()
{

   CHAR* SUBSTANCE_CUT;
   CHAR* SUBSTANCE_UNCOVER;
   CHAR* RESULT ;



   CHAR ANSWER;
   PRINTF("CHOSE YOUR FUNCTION:\N");
   PRINTF("TYPE [1] IF YOU WANT TO CUT YOUR SUBSTANCE\N");
   PRINTF("TYPE [2] IF YOU WANT TO UNCOVER YOUR SUBSTANCE\N");
   PRINTF("TYPE [3] IF YOU WANT TO CHECK IF YOUR CUT SUBSTANCE &
UNCOVER SUBSTANCE ARE EQUAL\N");
   SCANF(" ");
   SCANF("%C", &ANSWER);
   SWITCH(ANSWER){
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      CASE '1':
         PRINTF("TYPE THE SUBSTANCE YOU WANT TO CUT\N");
         SUBSTANCE_CUT = READ_SUBSTANCE();
         IF (SUBSTANCE_CUT == NULL) {
            PRINTF("[ERROR]MEMORY CANNOT BE ALLOCATED!\N");
            BREAK;
         }
         PRINTF("YOUR SUBSTANCE IS  %S!\N", SUBSTANCE_CUT);
```

```
            RESULT = FIRST_TASK_CUT(SUBSTANCE_CUT);

            IF (RESULT != NULL){
               PRINTF("\NYOUR CUTTEN SUBSTANCE IS: ");
               PUTS(RESULT);
            }ELSE{
               PRINTF("[ERROR]THE SUBSTANCE IS INCORRECT!\N");
               BREAK;
            }
            FREE(SUBSTANCE_CUT);
            FREE(RESULT);
            BREAK;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         CASE '2':
            PRINTF("TYPE THE SUBSTANCE YOU WANT TO UNCOVER\N");
            SUBSTANCE_UNCOVER = READ_SUBSTANCE();
            IF (SUBSTANCE_UNCOVER == NULL) {
               PRINTF("[ERROR]MEMORY CANNOT BE ALLOCATED!\N");
               BREAK;
            }
            PRINTF("YOUR SUBSTANCE IS  %S!\N", SUBSTANCE_UNCOVER);

            RESULT = SECOND_TASK_UNCOVER(SUBSTANCE_UNCOVER);
            IF (RESULT != NULL){
               PRINTF("\NYOUR UNCOVERED SUBSTANCE IS: ");
               PUTS(RESULT);
            }ELSE{
               PRINTF("[ERROR]THE SUBSTANCE IS INCORRECT!\N");
               BREAK;
            }
```

```
            FREE(SUBSTANCE_UNCOVER);
            FREE(RESULT);
            BREAK;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        CASE '3':
            PRINTF("TYPE THE SUBSTANCE YOU WANT NUMBER 1\N");
            SUBSTANCE_CUT = READ_SUBSTANCE();
            IF (SUBSTANCE_CUT == NULL) {
               PRINTF("[ERROR]MEMORY CANNOT BE ALLOCATED!\N");
               BREAK;
            }
            PRINTF("TYPE THE SUBSTANCE YOU WANT NUMBER 2\N");
            SUBSTANCE_UNCOVER = READ_SUBSTANCE();
            IF (SUBSTANCE_UNCOVER == NULL) {
               PRINTF("MEMORY CANNOT BE ALLOCATED!\N");
               BREAK;
            }
            PRINTF("[1]YOUR SUBSTANCE NUMBER ONE IS  %S!\N",
SUBSTANCE_CUT);
            PRINTF("[2]YOUR SUBSTANCE NUMBER TWO IS  %S!\N",
SUBSTANCE_UNCOVER);

            THIRD_TASK_IF_SUBSTANCES_ARE_EQUAL(SUBSTANCE_CUT,
SUBSTANCE_UNCOVER);

            BREAK;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        DEFAULT:
            PRINTF("[ERROR]WRONG ANSWER!\N");
```

```
            BREAK;

        }

        RETURN 0;
    }
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

```
/Users/pavlov/CLionProjects/kursovaya_prg_1sem/cmake-build-debug/kursovaya_prg_1sem
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
1
Type the substance you want to CUT
aHgHg
Your substance is  aHgHg!
[ERROR]The Substance is INCORRECT!
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
1
Type the substance you want to CUT
2HgHg
Your substance is  2HgHg!
[ERROR]The Substance is INCORRECT!
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
1
Type the substance you want to CUT
Hg2
Your substance is  Hg2!
[ERROR]The Substance is INCORRECT!
```

```
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
1
Type the substance you want to CUT
HgHgHgPPPPPPPPPPPPPPPPPPPPPPPPaPaPaPaDsDsDsSSSSSSS
Your substance is  HgHgHgPPPPPPPPPPPPPPPPPPPPPPPPaPaPaPaDsDsDsSSSSSSS!

Your Cutten Substance is: Hg3P23Pa4Ds3S7
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
2
Type the substance you want to UNCOVER
Hg2Pa3P100
Your substance is  Hg2Pa3P100!

Your Uncovered Substance is: HgHgPaPaPaPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
3
Type the substance you want number 1
PaPa
Type the substance you want number 2
Pa2
[1]Your substance number one is  PaPa!
[2]Your substance number two is  Pa2!
[+]The substances are equal!
```

```
Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
3
Type the substance you want number 1
PaPa
Type the substance you want number 2
Ma2
[1]Your substance number one is  PaPa!
[2]Your substance number two is  Ma2!
[-]The substances are NOT equal!

Chose any function:
type [1] if you want to CUT your substance
type [2] if you want to UNCOVER your substance
type [3] if you want to CHECK If Your Cut Substance & Uncover Substance Are Equal
type [4] if you want to finish the programm
4

Process finished with exit code 0
```