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 _____ Поршнев Р.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Поршнев Р.А.

Группа 1304

Тема работы: Работа со строками в языке Си

Исходные данные:

Строка(и), состоящая(ие) из латинских букв и цифр.

Содержание пояснительной записки:

Введение.

Основные теоретические положения.

Реализация 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 _____ Поршнев Р.А.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

В данной курсовой работе была реализована программа, которая предлагает пользователю выбрать тестируемую функцию, а после — запрашивает у пользователя строку (строки), которые нужно передать качестве аргумента (аргументов). Были реализованы следующие функции: нахождение произведения кодов символов строки, перевод числа из двоичной системы счисления в десятичную, проверка вхождения одной строки в другую, подсчёт количества гласных в строке. Вышеперечисленные функции были реализованы без использования стандартных и сторонний библиотек. Так же данная программа поддерживает уведомление о некорректности ситуации и защиту от переполнения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является разработка функций, которые выполняют заранее определённые задания, связанные со строками и символами. Для достижения поставленной цели требуется решить следующие задачи: изучить управляющих конструкций языка Си, изучить использования указателей и массивов в языке Си, изучить особенности работы с символами и строками в языке Си, реализовать функции по работе со строками и символами, создать защиту от переполнения и выхода за границу буфера, разработать механизм уведомления о некорректной ситуации, написать диалоговой программы. Все функции были реализованы без использования и сторонних библиотек.

Требуется реализовать следующие функции:

1. Функция считающая произведения всех кодов символов, символ с кодом 0 игнорируется в произведении. Необходимо предусмотреть защиту от переполнения.
2. Функция принимающая строку представляющая двоичное число и возвращающая строку представляющее это число в десятичной системе счисления.
3. Функция проверяющая наличие вхождения одной строки во вторую.
4. Функция считающая количество гласных букв в строке.

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. Строки и символы в языке Си

В языке Си существует такой тип данных, как char. В памяти он занимает 1 байт. В переменную данного типа данных можно записать лишь одно значение, которое будет интерпретироваться как ASCII-символ. Как такового строкового типа данных в языке Си нет, поэтому строка в данном языке программирования представляет собой последовательность символов в памяти, где последний символ — символ окончания строки, который обозначается “\0”. При наличии символа окончания строки она является корректной.

1.2. Используемые библиотеки

В данной курсовой работе использовались такие библиотеки как: *stdio.h*, *stdlib.h*, *string.h*, *limits.h*, *locale.h*.

Из библиотеки *stdio.h* использовались следующие функции:

- printf() — используется для вывода в стандартный поток вывода;
- scanf() — используется для ввода из стандартного потока ввода.

Из библиотеки *stdlib.h* использовались следующие функции:

- realloc — выполняет перераспределение блоков памяти;
- free — освобождает место в памяти.

Из библиотеки *string.h* использовались следующая функция:

- strlen(char *string) — возвращает длину строки string;
- strcmp(const char * string1, const char * string2) — функция сравнения символов строк string1 и string2;

Из библиотеки *limits.h* использовалась следующая константа:

- ULLONG_MAX — максимальное значение типа unsigned long long int.

Из библиотеки *locale.h* использовалась следующая функция:

- setlocale() — задаёт локализацию программы. Так же используется макрос LC_ALL, который указывает программе, что локализированы будут все функции.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Функция входа в программу

В функции main() устанавливается русская локализация текста и переменная stop типа int инициализируется значением 0, c1 и c2 типа char, string и substring типа char*. Далее выводится текст, в котором пользователю предлагается ввести цифру от 1 до 5 в зависимости того, результат работы какой функции он хочет получить. Затем запускается цикл со следующим условием: пока stop не равно 1. Далее происходит считывание вводимой цифры и символа перевода строки. Вводимая цифра записывается в переменную c1, в переменную c2 записывается символ перевода строки. Необходимость считывания символа перевода строки обусловлено тем, что при последующем вводе строки данный символ станет первым элементом строки, что не является верным.

Далее следует оператор switch(). Дальнейшие действия программы зависят от вводимого ранее значения x.

Если значение переменной c1 равно ‘1’, то выводится просьба ввести строку. Далее переменной string присваивается значение функции read(), которое является введённой строкой. Затем следует условие: если значение функции production_of_codes(string) не равно -1, то происходит вывод значения данной функции. В ином случае ничего не выводится. Данный блок заканчивается очисткой памяти оператором break.

```
     case ‘1’:
          printf("Введите строку: ");
          string = read();
          if (production_of_codes(string) != -1)
               printf("Произведение кодов символов данной строки:
%llu\n", production_of_codes(string));
          free(string);
          break;
```

Если значение переменной c1 равно ‘2’, то выводится просьба ввести строку. Далее переменной string присваивается значение функции read(), которое является введённой строкой. Если значение функции out_of_ull(char

*string) равно 0, то сравнивается значение функции from_binary_to_decimal(char *string) и строка “nothing”. Если строки не равны, то выводится представление числа в двоичной системе счисления. При значении функции out_of_ull(char *string) не равном 0, выводится сообщение о том, что введённое число выходит за рамки типа unsigned long long int. Данный блок заканчивается очисткой памяти оператором break.

```
     case ‘2’:
          printf("Введите число в двоичнной системе счисления: ");
          string = read();
          if (out_of_ull(string) == 0){
               if (strcmp(from_binary_to_decimal(string), "nothing") !=
0)
               printf("Число в десятичной системе счисления: %s\n",
from_binary_to_decimal(string));
          }
          else {
          printf("Введённое число выходит за рамки типа unsigned long
long int!\n");
          }
          free(string);
          break;
```

Если значение переменной c1 равно ‘3’, то выводится просьба ввести первую строку, которая будет являться подстрокой. Переменной substring присваивается значение функции read(). Далее выводится просьба ввести вторую строку. Затем следует условие: если значение функции is_substring_in_string(char *substring, char *string) не равно 0, то выводится сообщение о том, что первая строка является подстрокой второй строки. В ином случае выводится сообщение, что первая строка не является подстрокой второй. Данный блок заканчивается очисткой памяти оператором break.

```
     case ‘3’:
          printf("Введите строку, которая будет проверяться на вхождение
в другую строку: ");
          substring = read();
          printf("Введите вторую строку: ");
          string = read();
          if (is_substring_in_string(substring, string))
               printf("Первая строка входит во вторую");
          else
               printf("Первая строка не входит во вторую");
          free(string);
          free(sunstring)
          break;
```

Если значение переменной c1 равно ‘4’, то выводится просьба ввести строку. Затем переменной string присваивается значение функции read(). Далее выводится сообщение о количестве гласных в строке и данное значение. Количеством гласных в строке является значение функции count_vowels(char *string). Блок заканчивается оператором break.

```
      case 4:
            printf("Введите строку: ");
            string = read();
            printf("Количество гласных в строке: %d\n",
count_vowels(string));
            free(string);
            break;
```

Если значение переменной c1 равно ‘5’, то переменной stop присваивается 1, происходит выход из оператора switch(). Так происходит, потому что в начале функции main() выводилось сообщение о том, что если пользователь введёт число 5, то программа завершится. После выхода из оператора switch произойдёт остановка цикла

При всех остальных значениях x выводится сообщение о том, что в функции с таким номером не существует.

После оператора switch() следует освобождение памяти с помощью функции free(void * ptrmem), в которую в качестве аргумента передаётся строка string. Функция заканчивается тем, что функции main() возвращается значение 0.

2.2. Функция ввода строки

Функция ввода строки была названа read. Данная функция в качестве аргумента ничего не принимает, возвращает тип указатель на char. В начале объявляется переменная с типа char, в неё будет записан считываемый символ. Далее объявляется переменная string типа указатель на char, в ней будет храниться считанная строка. Переменной string присваивается значение нулевого указателя, то есть NULL. Далее происходит инициализация переменной size типа int значением 20, она понадобится для динамического выделения памяти блоками по мере необходимости. Затем происходит инициализация переменной n типа int значением 0, она будет хранить

количество введённых символов. Следующим шагом является выделение памяти под переменную temp с помощью realloc. Для данной переменной выделяется память 20 байт. Если память не выделилась, то есть, temp = NULL, то выводится сообщение о том, что память не выделилась и функции возвращается нулевой указатель. Если память выделилась, то переменной string присваивается temp.

```
char c;
char *string;
string = NULL;
int size = 20;
int n = 0;
temp = realloc(string, size * sizeof(char));
    if (temp == NULL){
        printf("Память не выделена!\n");
        return NULL;
    }
    else
        string = temp;
```

Далее запускается цикл, в котором сначала выполняется тело цикла, а затем проверяется условие продолжения цикла. В теле функции происходит считывание очередного символа, который записывается в переменную c. Переменной string[n] присваивается значение c. Затем следует условие: если n равно size, то добавляется ещё один блок памяти, размерность которого составляет 20 байт. Осуществляется это с помощью увеличения переменной size на 20. Далее следует перераспределение памяти и её расширение. Затем следует проверка на выделение памяти. Счётчик символов в строке n увеличивается на 1.

```
do{
        if (n == size - 1){
            size += 20;
            temp = realloc(string, (size + 20) * sizeof(char));
            if (temp == NULL){
                printf("Память не выделена!\n");
                return NULL;
            }
            else
                string = temp;
        }
        scanf("%c", &c);
        string[n] = c;
        n += 1;
    }while (c != '\n');
```

После выхода из цикла в ячейку string[n] записывается символ конца строки. Для того, чтобы удалить символ перевода строки, в ячейку string[n-1] записывается символ конца строки. Функции возвращается введённая строка string.

2.3. Функция нахождения произведения кодов символов строки

Функция ввода строки имеет название production_of_codes. В качестве аргумента данная функция принимает введённую строку. В начале функции объявляется переменная i типа int, переменная check типа int инициализируется значением 0. Переменная N типа unsigned long long int инициализируется значением ULLONG_MAX, переменная prod того же типа инициализируется значением 1. Далее объявляется переменная out_of_ull типа double.

Затем происходит посимвольный обход строки string с помощью цикла. Если код символа строки не равен 0, то переменной out_of_ull присваивается значение N/string[i]. Далее следует проверка: если произведение кодов символов, которое хранится в переменной prod, больше чем out_of_ull, то выводится сообщение о том, что произведение кодов символов строки больше максимального значения типа unsigned long long int, переменной check присваивается значение 1, цикл останавливается с помощью оператора break. Если условие выше не выполнилось, то переменная prod обновляется её умножением на код символа string[i].

```
      for(i = 0; i < strlen(string); i++){
              if (string[i] != 0){
                  out_of_ull = N / string[i];
                  if (prod > out_of_ull){
                      printf("Произведение кодов символов данной строки
больше максимального значения типа unsigned long long int!\n");
                      check = 1;
                      break;
                  }
                  prod = prod * string[i];
              }
          }
```

После завершения цикла следует условие: если значение переменной check равно 1, то функции возвращается значение -1. В ином случаем функции возвращается значение переменной prod. Стоит напомнить, что если значение переменной check равно -1, то произошло переполнение. В таком случае

отсутствует возможность вывести корректное значение произведения кодов символов строки.

2.4. Функция перевода числа из двоичной системы счисления в десятичную

Функция перевода числа из двоичной системы счисления в десятичную называется from_binary_to_decimal. В качестве аргумента данная функция принимает введённую строку string. В функции объявляются переменные i, number типа int, переменные типа int check и j инициализируются значением 0. Объявляется переменные ans и ans_rev типа указатель на char. Далее объявляется переменная c типа char. Переменной ans и ans_rev присваивается нулевой указатель. Далее происходит инициализация переменной decimal типа unsigned long long int значением 0. Затем происходит обход строки, представляющей собой число в двоичной системе счисления. Если символ строки не равен 1 и не равен 0, значит, данное число представлено не в двоичной системе счисления. Выводится соответствующее сообщение, переменной check присваивается значение 1, цикл останавливается с помощью оператора break. Если данное условие не выполнилось, то текущий символ переводится к целочисленному значению. Далее к переменной decimal прибавляется значение number*powered(strlen(string)-i-1). О функции с названием powered речь пойдёт в пункте 2.5. Таким образом, в переменную decimal будет записан результат перевода числа из двоичной системы счисления в десятичную.

```
    for(i = 0; i < strlen(string); i++){
            if ((string[i] != '1') && (string[i] != '0')){
                printf("Введённое число не в двоичной системе
счисления!\n");
                check = 1;
                break;
            }
            number = string[i] - '0';
            decimal = decimal + number * powered(strlen(string) - i -
1);
        }
```

После завершения цикла следует условие: если значение переменной check равно 1, значит, пользователь ввёл число не в двоичной системе

счисления и функции передаётся строка “nothing”. В ином случае переменной i присваивается 0. Далее запускается цикл, в котором каждая цифра числа decimal будет записываться в массив символов ans в обратном порядке. Во время каждой итерации от числа будет отделяться последняя цифра с помощью деления с остатком на 10 и записываться в переменную, число decimal будет нацело делиться на 10, в переменную будет записываться цифра number как символ. Данный символ будет заноситься в строку ans в ячейку с индексом i, предварительно под этот символ выделиться память в ans с помощью realloc. В конце каждой итерации счётчик i будет увеличиваться на 1. Цикл останавливается в том случае, если переменная decimal – однозначное число.

После выхода из цикла выделяется память под первую цифру числа decimal, значение которого было актуальным до цикла, который был описан ранее, и под знак конца строки.

```
i = 0;
while (decimal / 10 > 0){
      number = decimal % 10;
      decimal = decimal / 10;
      c = number + '0';
      ans = realloc(ans, (i + 1) * sizeof(char));
      ans[i] = c;
      i++;
}

ans = realloc(ans, (i + 2) * sizeof(char));
ans[i] = decimal + '0';
ans[i+1] = '\0';
```

В переменной ans хранятся цифры числа decimal до цикла, но в обратном порядке. Для решения данной проблемы запускается цикл, которой обходит строку ans начиная с конца и в ячейку с индексом j переменной ans_rev записывает символ ans[i]. Для записи символов в правильном порядке под ans_rev на каждой итерации выделяется память с помощью realloc.

```
for(i = strlen(ans) - 1; i > -1; i--){
            ans_rev = realloc(ans_rev, (j + 1) * sizeof(char));
            ans_rev[j] = ans[i];
            j += 1;
        }
```

После завершения обхода строки ans с помощью realloc динамически выделяется память под символ конца строки. После записи символа конца строки функции возвращается строка ans_rev.

2.5. Функция возведения числа в степень

Функция возведения числа в степень называется powered. В качестве аргумента функция принимает степень, в которую нужно возвести число. В данном случае будет производиться возведения 2 в некоторую степень, потому что производится перевод из двоичной системы счисления в десятичную. В функции инициализируется переменная i типа int. Переменная ans типа unsigned long long int инициализируется значением 1. Далее с помощью цикла начинается возведение 2 в степень n. Во время каждой итерации число умножается на 2 до тех пор, пока итератор не станет равным n. После завершения цикла функции возвращается значение ans.

```
int i;
unsigned long long int ans = 1;
for(i = 0; i < n; i++)
ans = ans * 2;
return ans;
```

2.6. Функция защиты от переполнения

Функция защиты от переполнения называется out_of_ull. В качестве аргумента данная функция принимает строку string, которая является представлением числа в двоичной системе счисления. Если длина строки больше 64, то функции возвращает 1. В ином случает функции возвращает 0. Сравнение длины строки с числом 64 обусловлено тем, что при переводе числа из двоичной системы счисления в десятичную придётся столкнутся с той проблемой, что максимальное целое число, которое может быть записано в языке Си, — это 2^{64}-1. Именно поэтому для представления числа в десятичной системе счисления используется тип unsigned long long int.

2.7. Функция проверки вхождения одной строки в другую

Функция по проверке на вхождение одной строки в другую имеет название is_substring_in_string. В качестве аргумента функции передаётся ранее

введённая подстрока и строка. В функции объявляются переменные i, j, k типа int.

Далее запускается обход строки. Условием остановки цикла является достижение итератора i значения strlen(string) – strlen(substring) + 1. В начало каждой итерации переменная check приравнивается к нулю. Затем запускается обод подстроки до тех пор, пока итератор не достигнет значения равному длине подстроки substring. Если не совпадает элемент строки и подстроки, то переменной check присваивается значение 1 и цикл останавливается с помощью оператора break. В ином случае следует увеличить на 1 итератор, связанный со строкой.

После обхода выхода из цикла следует следующее условие: если значение переменной check равно 0, то функции возвращается значение 1. Выполнение данного условия означает, что substring является подстрокой string. В ином случае функции возвращается 0.

```
int i, j, k, check;
    for(i = 0; i < strlen(string) - strlen(substring) + 1; i++){
        check = 0;
        j = i;
        for(k = 0; k < strlen(substring); k++){
            if (string[j] != substring[k]){
                check = 1;
                break;
            }
            j += 1;
        }
        if (check == 0)
            return 1;
    }
    return 0;
```

2.8. Функция подсчёта количества гласных в слове

Функция по подсчёту количества гласных в слове именуется как count_vowels. В качестве аргумента функция принимает введённую пользователем строку. В функции объявляется переменная i типа int и итерируется переменная count значением 0. Далее происходит обход строки через цикл, в котором во время каждой итерации к текущему символу строки применяется функция strchr(const char * string, int symbol), где string — это строка, состоящая из гласных букв латинского алфавита. Если значение данной

функции не равно 0, значит, текущий символ строки является гласной буквой. В таком случае переменная count увеличивается на 1. Функции возвращается значение переменной count.

```
int i, count = 0;
for(i = 0; i < strlen(string); i++)
      if (strchr("AEIOUYaeiouy", string[i]) != 0)
       count += 1;
return count;
```

3. ТЕСТИРОВАНИЕ

Входные данные	Выходные данные	Комментарии
1 a	Произведение кодов символов данной строки: 97	Ответ правильный, проверка нахождения произведения кодов символов
1 asfgsfdj	Произведение кодов символов данной строки: 14571721037340000	Ответ правильный, проверка нахождения произведения кодов символов
1 \t1	Произведение кодов символов данной строки: 441	Ответ правильный, проверка нахождения произведения кодов символов
1 }}}}}}}}}}	Произведение кодов символов данной строки больше максимального значения типа unsigned long long int!	Ответ правильный, проверка сообщения об ошибке
1 gsadfjashgj	Произведение кодов символов данной строки: 12931152000	Ответ правильный, проверка нахождения произведения кодов символов
2 10010010111011000010111	Число в десятичной системе счисления: 9631255	Ответ правильный, проверка перевода из двоичной системы счисления в десятичную
2 1000001101100111000	Число в десятичной системе счисления: 269112	Ответ правильный, проверка перевода из двоичной системы счисления в десятичную
2 11011100111001100011110	Число в десятичной системе счисления: 7238430	Ответ правильный, проверка перевода из двоичной системы счисления в десятичную
2 11111112890	Введено число не в двоичной системе счисления!	Ответ правильный, проверка сообщения об ошибке
2 000000000000000000001	Число в десятичной системе счисления: 1	Ответ правильный
3 C C++	Первая строка входит во вторую	Ответ правильный, проверка вхождения первой строки во вторую
3 Python C++javaPythobob	Первая строка не входит во вторую	Ответ правильный, проверка вхождения первой строки во вторую
3 Hello Hello	Первая строка входит во вторую	Ответ правильный, проверка вхождения первой строки во вторую
3 music musical	Первая строка входит во вторую	Ответ правильный, проверка вхождения первой строки во вторую
3 pop	Первая строка входит во вторую	Ответ правильный, проверка вхождения первой строки во

hyperpop		вторую
3 fgh asdfghjkl	Первая строка входит во вторую	Ответ правильный, проверка вхождения первой строки во вторую
4 AEIOUYaeiouy	Количество гласных в строке: 12	Ответ правильный, проверка подсчёта количества гласных в строке
4 qwertyuiop	Количество гласных в строке: 5	Ответ правильный, проверка подсчёта количества гласных в строке
4 As';dfksa;dlxmljsoilkm	Количество гласных в строке: 4	Ответ правильный, проверка подсчёта количества гласных в строке
4 wuyerowqiuroiio	Количество гласных в строке: 10	Ответ правильный, проверка подсчёта количества гласных в строке
4 zxcvbnmnbvcxz	Количество гласных в строке: 0	Ответ правильный, проверка подсчёта количества гласных в строке
5		Ответ правильный, программа завершилась
6	Функции с таким номером не существует!	Ответ правильный, данной функции не существует
-10	Функции с таким номером не существует!	Ответ правильный, данной функции не существует

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были изучены управляющие конструкции языка Си, работа с указателями и массивами, работа с символами и строками, а так же реализованы функции для работы со строками и символами, механизмы защиты от переполнения и выхода за буфер и механизмы уведомления о некорректности ввода, написана диалоговая программа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство “Невский диалект”, 2001. 352 с.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

char* read(){

    char c;
    char *string;
    char *temp;
    string = NULL;
    int size = 20;
    int n = 0;
    temp = realloc(string, size * sizeof(char));
    if (temp == NULL){
        printf("Память не выделена!\n");
        return NULL;
    }
    else
        string = temp;
    do{
        if (n == size - 1){
            size += 20;
            temp = realloc(string, (size + 20) * sizeof(char));
            if (temp == NULL){
                printf("Память не выделена!\n");
                return NULL;
            }
            else
                string = temp;
        }
        scanf("%c", &c);
        string[n] = c;
        n += 1;
    }while (c != '\n');
    string[n] = '\0';
    string[n - 1]= '\0';
    return string;
}


unsigned long long int production_of_codes(char *string){

    int i, check = 0;
    unsigned long long int N = ULLONG_MAX, prod = 1;
    double out_of_ull;
    for(i = 0; i < strlen(string); i++){
        out_of_ull = N / string[i];
        if (prod > out_of_ull){
            printf("Произведение кодов символов данной строки больше
максимального значения типа unsigned long long int!\n");
            check = 1;
            break;
```

```
        }
        if (string[i] != 0)
            prod = prod * string[i];
    }
    if (check == 1)
        return -1;
    else
        return prod;
}

unsigned long long int powered(int n){

    int i;
    unsigned long long int ans = 1;
    for(i = 0; i < n; i++)
        ans = ans * 2;
    return ans;
}
char* from_binary_to_decimal(char *string){

    int i, check = 0, number, j = 0;
    char *ans;
    char *ans_rev;
    char c;
    ans = NULL; ans_rev = NULL;
    unsigned long long int decimal = 0;
    for(i = 0; i < strlen(string); i++){
        if ((string[i] != '1') && (string[i] != '0')){
            printf("Введено число не в двоичной системе счисления!\n");
            check = 1;
            break;
        }
        number = string[i] - '0';
        decimal = decimal + number * powered(strlen(string) - i - 1);
    }
    if (check == 1)
        return "nothing";
    else{
        i = 0;
        while (decimal / 10 > 0){
            number = decimal % 10;
            decimal = decimal / 10;
            c = number + '0';
            ans = realloc(ans, (i + 1) * sizeof(char));
            ans[i] = c;
            i++;
        }
        ans = realloc(ans, (i + 2) * sizeof(char));
        ans[i] = decimal + '0';
        ans[i+1] = '\0';

        for(i = strlen(ans) - 1; i > -1; i--){
            ans_rev = realloc(ans_rev, (j + 1) * sizeof(char));
            ans_rev[j] = ans[i];
            j += 1;
        }
        ans_rev = realloc(ans_rev, (j + 1) * sizeof(char));
```

```
        ans_rev[j] = '\0';
        return ans_rev;
    }
}

int out_of_ull(char *string){

    if (strlen(string) > 64)
        return 1;
    return 0;
}


int is_substring_in_string(char *substring, char *string){

    int i, j, k, check;
    for(i = 0; i < strlen(string) - strlen(substring) + 1; i++){
        check = 0;
        j = i;
        for(k = 0; k < strlen(substring); k++){
            if (string[j] != substring[k]){
                check = 1;
                break;
            }
            j += 1;
        }
        if (check == 0)
            return 1;
    }
    return 0;
}

int count_vowels(char *string){

    int i, count = 0;
    for(i = 0; i < strlen(string); i++)
        if (strchr("AEIOUYaeiouy", string[i]) != 0)
            count += 1;
    return count;
}


int main(){

    setlocale(LC_ALL, "Russian");
    int stop = 0;
    char c1, c2;
    char *string;
    char *substring;
    printf("Введите [1], если хотите получить произведение кодов символов
строки\n"
           "Введите [2], если хотите получить число в десятичной системе
счисления, изначально введённое в двоичной\n"
           "Введите [3], если хотите проверить вхождение одной строки в
другую\n"
           "Введите [4], если хотите получить количество гласных в
введёной строке\n"
```

```
           "Введите [5], если хотите выйти из программы\n");
    while (stop != 1){
        scanf("%c%c", &c1, &c2);
        switch(c1){
            case '1':
                printf("Введите строку: ");
                string = read();
                if (production_of_codes(string) != -1)
                    printf("Произведение кодов символов данной строки:
%llu\n", production_of_codes(string));
                free(string);
                break;
            case '2':
                printf("Введите число в двоичнной системе счисления: ");
                string = read();
                if (out_of_ull(string) == 0){
                    if (strcmp(from_binary_to_decimal(string), "nothing")
!= 0)
                        printf("Число в десятичной системе счисления:
%s\n", from_binary_to_decimal(string));
                }
                else {
                    printf("Введённое число выходит за рамки типа unsigned
long long int!\n");
                }
                free(string);
                break;
            case '3':
                printf("Введите строку, которая будет проверяться на
вхождение в другую строку: ");
                substring = read();
                printf("Введите вторую строку: ");
                string = read();
                if (is_substring_in_string(substring, string))
                    printf("Первая строка входит во вторую\n");
                else
                    printf("Первая строка не входит во вторую\n");
                free(string);
                free(substring);
                break;
            case '4':
                printf("Введите строку: ");
                string = read();
                printf("Количество гласных в строке: %d\n",
count_vowels(string));
                free(string);
                break;
            case '5':
                stop = 1;
                break;
            default:
                printf("Функции с таким номером не существует!\n");
        }
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ А

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
roman@roman-VirtualBox:~$ gcc coursework.c -o workfile && ./workfile
Введите [1], если хотите получить произведение кодов символов строки
Введите [2], если хотите получить число в десятичной системе счисления, изначально введённое в двоичной
Введите [3], если хотите проверить вхождение одной строки в другую
Введите [4], если хотите получить количество гласных в введёной строке
Введите [5], если хотите выйти из программы
1
Введите строку: 12314
Произведение кодов символов данной строки: 318372600
2
Введите число в двоичнной системе счисления: 11111111111010001100000001
Число в десятичной системе счисления: 134193921
3
Введите строку, которая будет проверяться на вхождение в другую строку: bu
Введите вторую строку: haaaaaaaaaaaaaaaaaaaabu
Первая строка входит во вторую
4
Введите строку: wejfdajkdfhka
Количество гласных в строке: 3
5
roman@roman-VirtualBox:~$
```

```
roman@roman-VirtualBox:~$ gcc coursework.c -o workfile && ./workfile
Введите [1], если хотите получить произведение кодов символов строки
Введите [2], если хотите получить число в десятичной системе счисления, изначально введённое в двоичной
Введите [3], если хотите проверить вхождение одной строки в другую
Введите [4], если хотите получить количество гласных в введёной строке
Введите [5], если хотите выйти из программы
1
Введите строку: }}}
Произведение кодов символов данной строки: 1953125
2
Введите число в двоичнной системе счисления: 110101011111
Число в десятичной системе счисления: 3423
3
Введите строку, которая будет проверяться на вхождение в другую строку: ip
Введите вторую строку: dual
Первая строка не входит во вторую
4
Введите строку: bcvnxbvnxm
Количество гласных в строке: 0
5
```

```
roman@roman-VirtualBox:~$ gcc coursework.c -o workfile && ./workfile
Введите [1], если хотите получить произведение кодов символов строки
Введите [2], если хотите получить число в десятичной системе счисления, изначально введённое в двоичной
Введите [3], если хотите проверить вхождение одной строки в другую
Введите [4], если хотите получить количество гласных в введёной строке
Введите [5], если хотите выйти из программы
1
Введите строку: sjsha
Произведение кодов символов данной строки: 14141862800
2
Введите число в двоичнной системе счисления: 1111111111111111111111111111111111111111111101111111111111
Число в десятичной системе счисления: 144115188075847679
3
Введите строку, которая будет проверяться на вхождение в другую строку: huba
Введите вторую строку: buba
Первая строка не входит во вторую
4
Введите строку: rtwyep[df
Количество гласных в строке: 2
5
```

ПРИЛОЖЕНИЕ В

ПРИМЕРЫ ОБРАБОТКИ ОШИБОК

```
roman@roman-VirtualBox:~$ gcc coursework.c -o workfile && ./workfile
Введите [1], если хотите получить произведение кодов символов строки
Введите [2], если хотите получить число в десятичной системе счисления, изначально введённое в двоичной
Введите [3], если хотите проверить вхождение одной строки в другую
Введите [4], если хотите получить количество гласных в введёной строке
Введите [5], если хотите выйти из программы
1
Введите строку: }}}}}}}}}}
Произведение кодов символов данной строки больше максимального значения типа unsigned long long int!
2
Введите число в двоичнной системе счисления: 1111111111111111111111111111111111111111111111111111111111111111
Число в десятичной системе счисления: 18446744073709551615
2
Введите число в двоичнной системе счисления: 11111111111111111111111111111111111111111111111111111111111111111
Введённое число выходит за рамки типа unsigned long long int!
5
```