**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Лабораторная работа №5

По дисциплине «Программирование»

 Рекурсия.

Группа: АВТ-041

Студент: Дударь М.А.

Преподаватель: Балакин В.В.

НОВОСИБИРСК 2021

1. **Постановка задачи:**

Для словаря из слов одинаковой длины, например «барак, баран, банан, дурак, бутон, бутан, питон, барин», определить цепочку слов так, чтобы каждое следующее слово отличалось не более чем на 2 буквы.

1. **Алгоритм программы:**
2. Считывание строки, состоящей из слов одинаковой длины.
3. Считывание слова той же дины, что и слова в строке, по которой будет строиться цепочка.
4. Построение цепочки слов, в которой следующее слово отличается не более чем на 2 буквы от предыдущего.
5. **Принцип работы:**

Функция считывания строки, с проверкой длин входящих в нее слов (при mode = 0), и ключевого слова, той же длины (при mode = 1).

char \*\*readSameWords(int \*height, int \*length, int mode) {  
 int flag = 1;  
 char buf[100];  
 char c = ' ';  
 char \*\*str = (char \*\*) malloc(sizeof(char \*));  
 if (mode) flag = 0;  
 do {  
 scanf("%s", buf);  
 scanf("%c", &c);  
 if (flag) {  
 flag = 0;  
 \*length = (int) strlen(buf);  
 }  
 if (strlen(buf) != \*length) {  
 printf("Words has different length!\n");  
 exit(-1);  
 }  
 if (mode) {  
 str = (char \*\*) realloc(str, sizeof(char \*));  
 \*str = (char \*) malloc(strlen(buf) \* sizeof(char));  
 strcpy(\*str, buf);  
 return str;  
 }  
 str = (char \*\*) realloc(str, (\*height + 1) \* sizeof(char \*));  
 str[\*height] = (char \*) malloc(sizeof(char));  
 str[\*height] = (char \*) realloc(str[\*height], strlen(buf) \* sizeof(char));  
 strcpy(str[\*height], buf);  
 (\*height)++;  
 } while (c != '\n');  
 return str;  
}

Рекурсивная функция определения количества отличающихся символов в двух словах.

void sameCheck(char \*a, char \*b, int \*n, int \*count, int length) {  
 if (\*n != length) {  
 if (a[\*n] != b[\*n])(\*count)++;  
 (\*n)++;  
 sameCheck(a, b, n, count, length);  
 }  
}

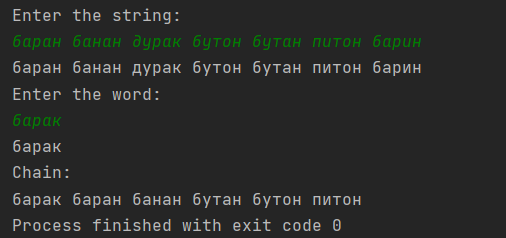
Функция построения цепочки, начинающейся с ключевого слова, в которой следующее слово отличается не более чем на 2 буквы от предыдущего.

char \*\*chainBuild(char \*\*str, char \*word, const int \*length, int \*height, int \*size) {  
 int n = 0, diff = 0, flag = 0;  
 char \*\*chain = (char \*\*) malloc(sizeof(char \*));  
 chain[0] = (char \*) malloc(\*length \* sizeof(char));  
 chain[0] = word;  
 while (1) {  
 for (int i = 0; i < \*height; ++i) {  
 n = 0;  
 diff = 0;  
 sameCheck(chain[\*size], str[i], &n, &diff, \*length);  
 if (diff <= 2) {  
 (\*size)++;  
 chain = (char \*\*) realloc(chain, (\*size + 1) \* sizeof(char \*));  
 chain[\*size] = (char \*) malloc(\*length \* sizeof(char));  
 chain[\*size] = str[i];  
 for (int j = i; j < \*height; ++j) str[j] = str[j + 1];  
 (\*height)--;  
 str = (char \*\*) realloc(str, \*height \* sizeof(char \*));  
 flag = 0;  
 break;  
 } else flag = 1;  
 }  
 if (!\*height) break;  
 if (flag) break;  
 }  
 return chain;  
}

1. **Вывод:**

В ходе выполнения задания была произведена работа с рекурсией, массивами указателей на языке C. Организовано рекурсивное определение количества разных символов в двух словах, написан алгоритм построения цепочки слов, в которой следующее слово отличается не более чем на 2 буквы от предыдущего.

Приложение 1. Вывод программы.



Приложение 2. Текст программы.

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
char \*\*readSameWords(int \*height, int \*length, int mode) {  
 int flag = 1;  
 char buf[100];  
 char c = ' ';  
 char \*\*str = (char \*\*) malloc(sizeof(char \*));  
 if (mode) flag = 0;  
 do {  
 scanf("%s", buf);  
 scanf("%c", &c);  
 if (flag) {  
 flag = 0;  
 \*length = (int) strlen(buf);  
 }  
 if (strlen(buf) != \*length) {  
 printf("Words has different length!\n");  
 exit(-1);  
 }  
 if (mode) {  
 str = (char \*\*) realloc(str, sizeof(char \*));  
 \*str = (char \*) malloc(strlen(buf) \* sizeof(char));  
 strcpy(\*str, buf);  
 return str;  
 }  
 str = (char \*\*) realloc(str, (\*height + 1) \* sizeof(char \*));  
 str[\*height] = (char \*) malloc(sizeof(char));  
 str[\*height] = (char \*) realloc(str[\*height], strlen(buf) \* sizeof(char));  
 strcpy(str[\*height], buf);  
 (\*height)++;  
 } while (c != '\n');  
 return str;  
}  
  
void sameCheck(char \*a, char \*b, int \*n, int \*count, int length) {  
 if (\*n != length) {  
 if (a[\*n] != b[\*n])(\*count)++;  
 (\*n)++;  
 sameCheck(a, b, n, count, length);  
 }  
}  
  
char \*\*chainBuild(char \*\*str, char \*word, const int \*length, int \*height, int \*size) {  
 int n = 0, diff = 0, flag = 0;  
 char \*\*chain = (char \*\*) malloc(sizeof(char \*));  
 chain[0] = (char \*) malloc(\*length \* sizeof(char));  
 chain[0] = word;  
 while (1) {  
 for (int i = 0; i < \*height; ++i) {  
 n = 0;  
 diff = 0;  
 sameCheck(chain[\*size], str[i], &n, &diff, \*length);  
 if (diff <= 2) {  
 (\*size)++;  
 chain = (char \*\*) realloc(chain, (\*size + 1) \* sizeof(char \*));  
 chain[\*size] = (char \*) malloc(\*length \* sizeof(char));  
 chain[\*size] = str[i];  
 for (int j = i; j < \*height; ++j) str[j] = str[j + 1];  
 (\*height)--;  
 str = (char \*\*) realloc(str, \*height \* sizeof(char \*));  
 flag = 0;  
 break;  
 } else flag = 1;  
 }  
 if (!\*height) break;  
 if (flag) break;  
 }  
 return chain;  
}  
  
int main() {  
 int height = 0, length = 0, size = 0;  
 char \*\*str, \*\*chain;  
 char \*word;  
 printf("Enter the string:\n");  
 str = readSameWords(&height, &length, 0);  
 for (int i = 0; i < height; ++i) {  
 printf("%s ", str[i]);  
 }  
 printf("\nEnter the word:\n");  
 word = \*readSameWords(&height, &length, 1);  
 printf("%s\n", word);  
 chain = chainBuild(str, word, &length, &height, &size);  
 printf("Chain:\n");  
 for (int i = 0; i <= size; ++i) {  
 printf("%s ", chain[i]);  
 }  
 return 0;  
}