Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ** **РАБОТЕ №3**

Динамические массивы (многомерные)

Тема

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Преподаватель |  |  |  |  |  | Черниговский А.С |
|  |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ19-16/2б |  |  |  | Шпаков С.Е. |
|  |  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2020

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc36295346)

[1 Цель и задачи 3](#_Toc36295347)

[1.1 Цель 3](#_Toc36295348)

[1.2 Задачи 3](#_Toc36295349)

[2 Описание варианта задания 3](#_Toc36295350)

[3 Ход выполнения 4](#_Toc36295351)

[4 Выводы 10](#_Toc36295352)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 11](#_Toc36295353)

# 1 Цель и задачи

## 1.1 Цель

Написать программу, выполняющую свои функции и выполняющую задачи ниже.

## 1.2 Задачи

На оценку 3 балла:

1. написать программу в соответствии с заданием;
2. отформатировать исходный код программы согласно стандарту оформления исходного кода;

На оценку 4 балла:

1. выполнить все предыдущие пункты;
2. для хранения строк использовать динамические массивы символов (размер массива определяется в процессе ввода);
3. организовать повтор программы по желанию пользователя;
4. добавить проверку входных аргументов на корректность.

На оценку 5 баллов:

1. выполнить все предыдущие пункты;
2. реализовать меню пользователя, состоящее минимум из 4-х пунктов (ввод элементов массива, обработка массива, вывод массива на экран, выход);
3. для корректной работы меню организовать промежуточное хранение результата;
4. разбить программу на функции;
5. организовать чтение данных и запись результата в файл формата txt.

# 2 Описание варианта задания

Вариант №24. Дан текст, который может содержать буквы английского алфавита. Напишите программу, предназначенную для шифрования и расшифровки текста, используя шифр ROT13. Перед шифрованием удалите из текста все знаки препинания и повторяющиеся пробелы, приведите символы к нижнему регистру.

# 3 Ход выполнения

Необходимо создать ввод текста с клавиатуры или из файла, где фильтруются все знаки, оставляя строчные буквы английского алфавита и одиночные пробелы.

После этого шифром ROT13 (Rotate) зашифровать текст. Добавить возможность сохранения зашифрованного текста в файл txt. Позволить пользователю повторять цикл по желанию. Вся эта программа реализована ниже (листинг 1).

Листинг 1 – Реализованная программа с комментариями

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define BACKSPACE 8

#define START\_RANGE 32

#define END\_RANGE 123

#define START\_UPPER 65

#define END\_UPPER 91

#define START\_LOWER 97

#define END\_LOWER 123

#define MIDDLE\_CHAR 110

#define LINE\_END 10

#define READ\_PATH "..\\text.txt"

#define WRITE\_PATH "..\\result.txt"

char\* input(int\* csize)

// Manual string input

{

// Allocating memory to the string of \0

char\* userString = (char\*)malloc(1 \* sizeof(char));

userString[0] = '\0';

char nextChar = 0;

int currentSize = 1;

bool spaced = false;

printf("Type in the text: \n");

while(nextChar != '\n')

{

// Getting next character of the input

nextChar = (char)getchar();

// delta to detect how string will change

int delta = 0;

int lengthDif = 0;

if (nextChar == BACKSPACE)

// in case of backspace event we need to clear one char => delta = -1

Продолжение Листинга 1

{

delta = -1;

lengthDif=1;

}

else if (nextChar == LINE\_END)

// Catching end line char and force-continuing (ending)

{

continue;

}

else

{

if (nextChar >= START\_RANGE && nextChar <= END\_RANGE)

// checking if char is in needed range

{

if (nextChar == START\_RANGE)

// START\_RANGE == 32 and represents space char

{

if (spaced)

// if we have space on previous char, we skip it

{

continue;

}

delta = 1;

lengthDif = 2;

spaced = true;

}

else if (nextChar >= START\_LOWER && nextChar < END\_LOWER)

{

// proceeding lowercase letters

delta = 1;

lengthDif = 2;

spaced = false;

}

else if (nextChar >= START\_UPPER && nextChar < END\_UPPER)

{

// in case of uppercase letter we change character to its lower case version

nextChar += 32;

delta = 1;

lengthDif = 2;

spaced = false;

}

else

continue;

}

}

// "Calculating" new size of the string

int newSize = currentSize + delta;

if (newSize == 0)

// catching possible backspace event

{

continue;

}

// creating temporary string

char\* tempStr = (char\*)malloc(newSize \* sizeof(char));

Продолжение Листинга 1

if (tempStr)

// if memory successfully allocated then..

{

for (int i = 0; i < newSize - lengthDif; ++i)

// filling temp string with previous symbols

{

tempStr[i] = userString[i];

}

if (nextChar != BACKSPACE)

// adding new char to the string if it wasn't backspace

{

tempStr[newSize - 2] = nextChar;

}

// Actually making it a string

tempStr[newSize - 1] = '\0';

// Freeing previous string

free(userString);

// and assigning temp to user string

userString = tempStr;

// updating size

currentSize = newSize;

}

else

{

printf("Unable to allocate memory");

break;

}

}

// Returning values

\*csize = currentSize;

return userString;

}

void encode(char\* src, int const ssize)

{

// If source string is not empty

if (src)

{

// Then we change every char apart from space

int size = ssize;

char nextChar = 0;

for (int i = 0; i < size-1; i++)

{

nextChar = src[i];

if (nextChar != 32 && nextChar < MIDDLE\_CHAR)

// If character is before middle element of alphabet then we add

{

nextChar += 13;

}

else if (nextChar != 32 && nextChar >= MIDDLE\_CHAR)

// And the opposite

{

nextChar -= 13;

}

// Replacing character

src[i] = nextChar;

Продолжение Листинга 1

}

}

}

char\* fileRead(int\* csize) {

// Basically the same as for user input

char \*userString = (char \*) malloc(1 \* sizeof(char));

userString[0] = '\0';

int currentSize = 1;

bool spaced = false;

int c;

// Creating a pointer and opening the file with Read mode

FILE \*file;

file = fopen(READ\_PATH, "r");

if (file) {

// While not the end of the file - check

while ((c = getc(file)) != EOF) {

// Everything below is almost as input func

int delta = 0;

int lengthDif = 0;

if (c >= START\_RANGE && c <= END\_RANGE) {

if (c == START\_RANGE)

{

if (spaced)

{

continue;

}

delta = 1;

lengthDif = 2;

spaced = true;

}

else if (c >= START\_LOWER && c < END\_LOWER)

{

delta = 1;

lengthDif = 2;

spaced = false;

}

else if (c >= START\_UPPER && c < END\_UPPER)

{

c += 32;

delta = 1;

lengthDif = 2;

spaced = false;

}

else

continue;

}

int newSize = currentSize + delta;

if (newSize == 0) {

continue;

}

char \*tempStr = (char \*) malloc(newSize \* sizeof(char));

if (tempStr) {

for (int i = 0; i < newSize - lengthDif; ++i) {

tempStr[i] = userString[i];

}

Продолжение Листинга 1

tempStr[newSize - 2] = (char)c;

tempStr[newSize - 1] = '\0';

free(userString);

userString = tempStr;

currentSize = newSize;

}

}

\*csize = currentSize;

// Closing the file

fclose(file);

return userString;

}

else

{

printf("\nError. Unable to open the file.\n");

}

}

void fileWrite(char\* src, int const ssize)

{

// Opening the file in Write mode

FILE\* file;

file = fopen(WRITE\_PATH, "w");

if (file)

{

// Printing every character in it.

for (int i = 0; i < ssize-1; i++)

{

fprintf(file, "%c", src[i]);

}

fclose(file);

}

}

int main()

{

char\* string = NULL;

int size = 0;

int choice;

while (1)

{

printf("What do you want to do?\n1.Input new string\n2.Proceed string\n3.Display string\n"

"4.Read text from the file\n5.Write text into the file\n6.Exit\n");

char checker = '\0';

while (scanf("%d%c", &choice, &checker, 1) != 2 || checker != '\n') {

printf("\nPlease enter a valid number\n");

while (getchar() != '\n');

}

switch (choice)

{

case 1:

if (string)

{

free(string);

}

string = input(&size);

Продолжение Листинга 1

break;

case 2:

if (string)

{

encode(string, size);

}

else

{

printf("\nThere is no string yet\n");

}

break;

case 3:

{

if (string) {

printf("\n>>> %s\n", string);

}

else

{

printf("\nError. There is no string yet.\n");

}

break;

}

case 4:

if (string)

{

free(string);

}

string = fileRead(&size);

break;

case 5:

if (string)

{

fileWrite(string, size);

}

else

{

printf("\nThere is nothing to write yet.\n");

}

break;

case 6:

free(string);

return 0;

default:

printf("\nThere is no such option\n");

break;

}

}

}

# 4 Выводы

Задача реализована на языке Си. Программа позволяет зашифровать текст с помощью ROT13, используя массивы знаков (строки), а также сохранить результат в текстовый файл txt.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2–07–2012 ; введ. 09.01.2014, - Красноярск : ИПК СФУ, 2014 – 60 с.