|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКИ**

Студент Камалов Антон Павлович

Группа РК6-25Б

Модуль № 2

Вариант T11

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А.П.**

*подпись, дата фамилия, и.о.*

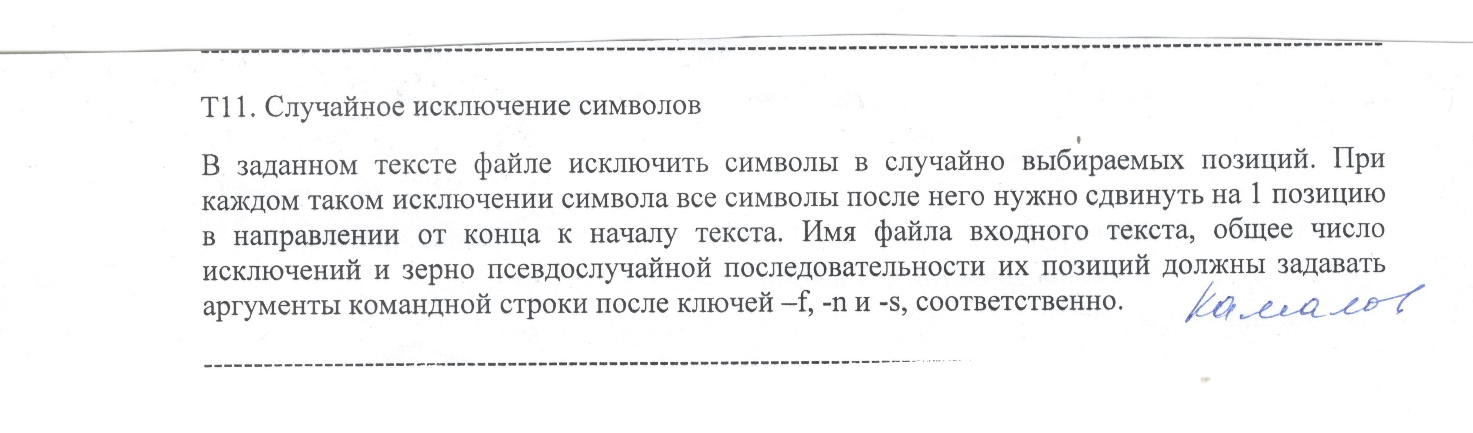
Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

**Задание**

****

**Описание программы**

**Входные данные:** имя файла входного текста, общее число исключений, зерно псевдослучайной последовательности.

**Выходные данные:** изменённый текст из входного файла.

**Общий алгоритм решения:**

1. Проверяем количество и корректность аргументов командной строки. Если аргументы заданы некорректно, программа выводит сообщение об ошибке и завершает работу.
2. Будем представлять текст как двусвязный список. Открываем входной файл и его содержимое посимвольно считываем с помощью функции fgetc(). Каждый символ добавляется в список с помощью функции insert(), которая добавляет новый узел в конец списка. Если список пустой, новый узел становится головой списка. В противном случае, происходит поиск последнего узла в списке и устанавливается связь с новым узлом.
3. Выводим исходный текст с помощью функции printList().
4. Вызывается функция exclude(), которая выполняет исключение символов из текста на указанное количество и с заданным зерном.

Сначала функция вычисляет длину текста, а затем в цикле генерирует случайную позицию и выполняет сдвиг узлов с помощью функции shift(). После каждого исключения длина текста уменьшается. Функция возвращает измененный список.

1. Функция shift() выполняет сдвиг узлов в списке. Сначала находит узел с указанной позицией, затем выполняет сдвиг. Если указанный узел имеет следующий узел, то текущий узел заменяется следующим, обновляются связи у соседних узлов и освобождается память удаленного узла. Если указанный узел является последним, то текущий узел заменяется предыдущим, обновляется связь у предыдущего узла и освобождается память удаленного узла. Функция возвращает измененный список.
2. Выводим измененный текст с помощью функции printList().
3. Освобождаем выделенную память с помощью функции freeList().

**Список литературы**

1. Керниган Б., Ритчи Д. Язык программирования C: пер. с англ / под ред. В.Л. Бродового. М.: Издательство Вильямс, 2019. 288 с. [Brian Kernighan, Dennis Ritchie. The C Programming Language. 2nd ed. Prentice Hall, 1988. 288 p.].
2. Федорук Е.В., Шварц Д.Т. Основы программирования. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=OP2/OP\_T.cou (дата обращения 28.05.2023).

**Приложение 1**

Код программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

char data;

struct Node\* next;

struct Node\* prev;

} Node;

Node\* createNode(char data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

if (newNode == NULL) {

printf("Memory allocation error.\n");

exit(-1);

}

newNode->data = data;

newNode->next = NULL;

newNode->prev = NULL;

return newNode;

}

void insert(Node\*\* head, char data) {

Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

current->next->prev = current;

}

}

Node\* shift(Node\* text, int position) {

Node\* current = text;

Node\* temp;

int count = 0;

while (count < position) {

current = current->next;

count++;

}

if (current->next){

temp = current;

current = current->next;

current->prev = temp->prev;

if (temp->prev != NULL) temp->prev->next = current;

else current->prev = NULL;

free(temp);

} else {

temp = current;

current = current->prev;

if (current) current->next = temp->next;

free(temp);

}

if (position == 0) return current;

else return text;

}

Node\* exclude(Node\* text, int num, unsigned int seed) {

srand(seed);

int length = 0;

Node\* head = text;

while (head != NULL) {

length++;

head = head->next;

}

if (num > length){

printf("ERROR! Length of text less than count of exclusion!\n");

exit(-1);

}

for (int i = 0; i < num; i++) {

int position = rand() % length;

text = shift(text, position);

length--;

}

return text;

}

void printList(Node\* text) {

Node\* current = text;

while (current != NULL) {

printf("%c", current->data);

current = current->next;

}

printf("\n");

}

void freeList(Node\* text) {

Node \*temp;

while (text){

temp = text;

text = text->next;

free(temp);

}

}

int main(int argc, char\* argv[]) {

if (argc != 7) {

printf("INCORRECT INPUT: ./a.out -f [input\_file] -n [num\_exclusions] -s [seed]\n");

return -1;

}

char\* file\_name;

int num\_exclude;

unsigned int seed;

for (int i = 1; i < argc; i++) {

if (strcmp(argv[i], "-f") == 0) {

file\_name = argv[++i];

} else if (strcmp(argv[i], "-n") == 0) {

num\_exclude = atoi(argv[++i]);

} else if (strcmp(argv[i], "-s") == 0) {

seed = atoi(argv[++i]);

}

}

if (num\_exclude < 0){

exit(-2);

}

FILE\* in = fopen(file\_name, "r");

if (!in) {

printf("Error opening input file.\n");

return -1;

}

Node\* text = NULL;

char ch;

while ((ch = fgetc(in)) != EOF) {

insert(&text, ch);

}

fclose(in);

printf("Input text: ");

printList(text);

text = exclude(text, num\_exclude, seed);

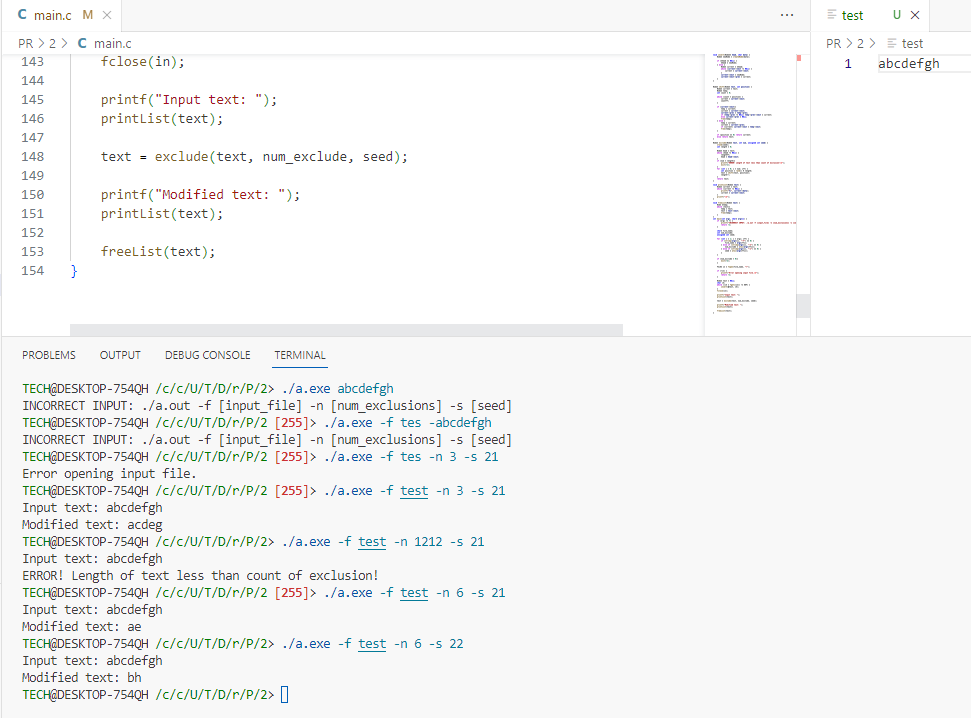
printf("Modified text: ");

printList(text);

freeList(text);

}

**Приложение 2**

Результат работы программы