ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ  
Ордена Трудового Красного Знамени федеральное государственное   
бюджетное учреждение высшего образования

Московский технический университет связи и информатики

МТУСИ

Кафедра информационной безопасности

Лабораторная работа

Вариант 2

Выполнил студент группы БИН1703 ОТФ-2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бен Режеб Софиэн

Старший преподаватель кафедры ИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Барков

**Практикум №7**  
**Динамическое выделение памяти**

**Цель практикума:** Овладеть навыками выделения динамической памяти в языке программирования C.

***Задание 1***

Написать функцию, которая создаёт в динамической памяти целое число, осуществляет ввод, выводит на экран адрес переменной и её значение, увеличивает значение на единицу и снова выводит адрес и значение на экран. Перед выходом из функции необходимо освободить ранее выделенную память.

Прототип функции void incrementHeapVariable().

Формат вывода: адрес значение адрес значение.

***Задание 2***

Написать функцию, которая создаёт в динамической памяти из N элементов (число N вводится пользователем с клавиатуры), выводиn n на экран и с помощью разработанных функций (практикум №4, задания 2-4) осуществляет ввод данных, вывод массива на экран, сортировку и повторный вывод отсортированного массива на экран. Перед выходом из функции необходимо освободить выделенную ранее память.

Прототип функции void sortHeapArray(int \*arr, int n).

***Задание 3***

Написать функцию, удаления подстроки в строке. Функция принимает исходную строку и подстроку для поиска. С помощью функции find найти адрес начала подстроки и затем удалить эту подстроку. Функция выделяет необходимое количество памяти в области динамической памяти, копирует необходимые части строки и возвращает указатель на вновь выделенную память. Обращение к символам строки после нулевого символа недопустимо.

Прототип функции char \*delete(char \*str, const char \*substr).

Функция не должна использовать функции консольного ввода-вывода.

***Задание 4***

Написать функцию копирования строк. Функция принимает исходную строку, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое исходной строки и возвращает указатель на вновь выделенную память

Прототип функции char \*strcpy(const char \*src).

Функция не должна использовать функции консольного ввода-вывода.

***Задание 5***

Написать функцию конкатенации строк. Функция принимает исходные строки, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое строк и возвращает указатель на вновь выделенную память

Прототип функции char \*strcat(const char \*first, const char \*second).

Функция не должна использовать функции консольного ввода-вывода.

***Задание 6***

Написать функцию вставки подстроки в строку. Функция принимает исходнную строку, строку для вставки, индекс, определяющий место вставки, выделяет необходимое количество памяти в области динамической памяти, копирует реузльтат и возвращает указатель на вновь выделеную память.

Прототип функции char \*insert(const char \*src, const char \*str, int index).

Функция не должна использовать функции консольного ввода-вывода.

Листинг 1 — Исходный файл AuxiliaryFunc.h

#ifndef AuxiliaryFunc\_h

#define AuxiliaryFunc\_h

#include <stdlib.h>

#include <stdio.h>

void writeArray(int \*arr, int n);

void readArray(int \*arr, int n);

void ShellSort(int \*arr,int n);

int mine\_strlen(const char\* c\_str);

char \*find( char \*c\_str, char \*substr);

#endif /\* AuxiliaryFunc\_h \*/

Листинг 2 — Исходный файл AuxiliaryFunc.c

#include <stdlib.h>

#include <stdio.h>

#include "AuxiliaryFunc.h"

void writeArray(int \*arr, int n)

{

int i;

for (i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

}

int my\_strlen(const char \*str)

{

int count = 0;

int i = 0;

while (str[i] != 0)

{

count++;

i++;

}

return count;

}

void readArray(int \*arr, int n)

{

int i;

for (i = 0; i < n; ++i)

scanf("%d", arr + i);

}

void ShellSort(int \*arr,int n)

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++)

{

tmp = arr[i];

for (j = i; j >= step; j -= step)

{

if (tmp < arr[j - step])

arr[j] = arr[j - step];

else

break;

}

arr[j] = tmp;

}

}

int mine\_strlen(const char\* c\_str)

{

int count=0;

while(\*c\_str++) ++count;

return count;

}

char \*find( char \*c\_str, char \*substr) {

int lenс\_Str = my\_strlen(c\_str);

int lenSub = my\_strlen(substr);

for (int i = 0; i <= lenс\_Str - lenSub; ++i) {

int j;

for (j = 0; c\_str[i + j] == substr[j]; ++j);

if (j - lenSub == 1 && i == lenс\_Str - lenSub) return &c\_str[i] ;

}

return 0;

}

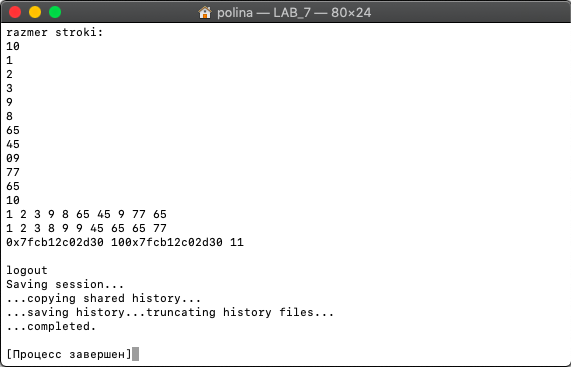


Рисунок 1