ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ЛАБОРАТОРНОЙ РАБОТЕ №4.1

По дисциплине «Языки программирования»

ВАРИАНТ 14

Выполнил: ст. гр. ТКИ-142

Кузнецов Денис Константинович

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

1. Формулировка задания

Создать одномерный массив из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Составить блок-схему.

Таблица 1 – Формулировка задания 4.1

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 14 | 1. Найти сумму элементов, значения которых состоят из одной цифры. 2. Заменить элементы массива между минимальным и максимальным на те же элементы в обратном порядке 3. Найти номер последней пары соседних элементов с одинаковыми знаками, произведение которых меньше заданного числа. | [-100;100] |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы остальных функций представлены ниже (Рисунок 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13).





Рисунок  -­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции createArr(n)



Рисунок 4 – Блок-схема функции copyArr(arr, n)



Рисунок 5 – Блок-схема функции posInput()



Рисунок 6 – Блок-схема функции checkArray(array)



Рисунок 7 – Блок-схема функции fillArrRandomly(arr, n)



Рисунок 8 – Блок-схема функции fillArrManually (arr, n)



Рисунок 9 – Блок-схема функции printArr(arr, n)



Рисунок 11 – Блок-схема функции oneDigitSum (arr, n)



Рисунок 12 – Блок-схема функции lessThanX

(arr, n, x)



Рисунок 13 - Блок-схема функции reverseBetweenMinAndMax(arr,n,x)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <stdbool.h>

#include <time.h>

/\*\*

\* @brief функция, считывающая вводимое значение

\* @return введенное значение

\*/

int input(void);

/\*\*

\* @brief функция, проверяющая ввод на положительность

\* @param value вводимое число

\* @return Возвращает: число, если число положительно; ошибку, если число отрицательно

\*/

int posInput(void);

/\*\*

\*@param manual реализация способа заполнения массива числами, которые вводит пользователь

\* @param random реализация способа заполнения массива рандомными(случайными) числами

\*/

enum opetation {

fillManually,

fillRandomly

};

/\*\*

\* @brief функция, которая создаёт массив размером n

\* @param n размер массива

\* @return созданный массив

\*/

int\* createArr(const size\_t n);

/\*\*

\* @brief функция, проверяющая корректность адреса указателя массива

\* @param arr массив

\* @return Возвращает ошибку, если потерялась память

\*/

void checkArr(const int\* arr);

/\*\*

\* @brief функция, которая выводит массив пользователю

\* @param arr указатель на наш массив

\* @param n размер массива

\* @return наш массив

\*/

void printArr(const int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая позволяет пользователю заполнить массив самостоятельно

\* @param arr массив

\* @param n количество элементов в массиве

\* @return массив, заполненный пользователем самостоятельно

\*/

void fillArrManually(int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая позволяет пользователю заполнить массив с помощью случайных чисел

\* @param arr массив

\* @param n количество элементов в массиве

\* @return массив, заполненный случайными числами

\*/

void fillArrRandomly(int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая суммирует все числа, состоящие из 1 цифры

\* @param arr массив

\* @param n количество элементов в массиве

\* @return сумму чисел из 1 цифры

\*/

void oneDigitSum(const int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая копирует элементы из одного массива в новый

\* @param arr исходный массив

\* @param n размер массива

\* @return новый копированный массив, который повторяет исходный

\*/

int\* copyArr(const int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая заменяет элементы между минимальным и максимальным значением

\* @param arr исходный массив

\* @param n размер массива

\* @return измененный масств

\*/

void reverseBetweenMinAndMax(int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая ищет номер последней пары соседних элементов с одинаковыми знаками, произведение которых меньше заданного числа.

\* @param arr исходный массив

\* @param n размер массива

\* @param x число, меньше которого должно быть произведение пары

\* @return номер последней пары

\*/

void lessThanX(const int\* arr, size\_t n, const int x);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

printf("Введите размер массива: ");

size\_t n = posInput();

printf("Введите число x: ");

int x = input();

int\* arr = createArr(n);

printf("Как вы хотите заполнить массив? \n%d - Самостоятельно \n%d - Рандомными числами\n", fillManually, fillRandomly);

int operation = input();

switch(operation) {

case fillManually:

fillArrManually(arr, n);

break;

case fillRandomly:

fillArrRandomly(arr, n);

break;

default:

free(arr);

arr = NULL;

errno = ERANGE;

perror("Неверно введена функция");

exit(EXIT\_FAILURE);

break;

}

printf("Исходный массив: ");

printArr(arr, n);

printf("\nСумма чисел, состоящих из 1 цифры: ");

oneDigitSum(arr, n);

int\* copiedArr = copyArr(arr, n);

printf("\nМассив с перевернутыми элементами между минимумом и максимумом: ");

reverseBetweenMinAndMax(copiedArr, n);

printArr(copiedArr, n);

free(copiedArr);

printf("\nНомер последней пары соседних элементов с одинаковыми знаками, произведение которых меньше заданного числа: ");

lessThanX(arr, n, x);

free(arr);

return 0;

}

int input(void) {

int value = 0;

int res = scanf("%d", &value);

if (res != 1) {

errno = EIO;

perror("Ошибка ввода (число должно быть целым)");

exit(EXIT\_FAILURE);

}

return value;

}

int posInput(void) {

int value = input();

if (value <= 0) {

errno = EINVAL;

perror("Ошибка ввода (число должно быть положительным)");

exit(EXIT\_FAILURE);

}

return value;

}

int\* createArr(const size\_t n) {

int\* arr = (int\*)malloc(n \* sizeof(int));

checkArr(arr);

return arr;

}

void checkArr(const int\* arr) {

if (arr == NULL) {

errno = ENOMEM;

perror("Ошибка памяти, адрес указателя на массив оказался нулевым");

exit(EXIT\_FAILURE);

}

}

void printArr(const int\* arr, const size\_t n) {

checkArr(arr);

for (size\_t i = 0; i < n; i++) {

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

if (i < n - 1) {

printf(", ");

}

}

}

void fillArrManually(int\* arr, const size\_t n) {

checkArr(arr);

for (size\_t i = 0; i < n; i++) {

printf("Введите элемент %zu: ", i + 1);

arr[i] = input();

}

}

void fillArrRandomly(int\* arr, const size\_t n) {

checkArr(arr);

printf("Введите минимальное значение интервала \n");

int min = input();

printf("Введите максимальное значение интервала \n");

int max = input();

if (min > max) {

errno = EINVAL;

perror("У вас неправильно заданы границы интервала: конец должен быть меньше начала");

exit(EXIT\_FAILURE);

}

srand(time(NULL));

for (size\_t i = 0; i < n; i++) {

arr[i] = (rand() % (max - min + 1)) + min;

}

}

void oneDigitSum(const int\* arr, const size\_t n) {

int sum = 0;

for (size\_t i = 0; i < n; i++) {

if (arr[i] >= 0 && arr[i] < 10) {

sum += arr[i];

}

}

printf("%d\n", sum);

}

int\* copyArr(const int\* arr, const size\_t n) {

checkArr(arr);

int\* copiedArr = createArr(n);

for (size\_t i = 0; i < n; i++) {

copiedArr[i] = arr[i];

}

return copiedArr;

}

void reverseBetweenMinAndMax(int\* arr, const size\_t n) {

checkArr(arr);

size\_t minI = 0;

size\_t maxI = 0;

for (size\_t i = 1; i < n; i++) {

if (arr[i] < arr[minI]) {

minI = i;

}

if (arr[i] > arr[maxI]) {

maxI = i;

}

}

if (minI == maxI) {

printf("Минимальный и максимальный элементы массива совпадают, по этому развернуть их невозможно.\n");

return;

}

if (minI > maxI) {

size\_t temp = minI;

minI = maxI;

maxI = temp;

}

while (minI + 1 < maxI) {

size\_t temp = arr[minI + 1];

arr[minI + 1] = arr[maxI - 1];

arr[maxI - 1] = temp;

minI++;

maxI--;

}

}

void lessThanX(const int\* arr, size\_t n, const int x) {

checkArr(arr);

int lastPairIndex = -1;

for (size\_t i = 0; i < n - 1; i++) {

if (arr[i] \* arr[i + 1] > 0 && arr[i] \* arr[i + 1] < x) {

lastPairIndex = i;

}

}

if (lastPairIndex != -1) {

printf("%d\n", lastPairIndex+1);

} else {

printf("Пары не найдено.\n");

}

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30)

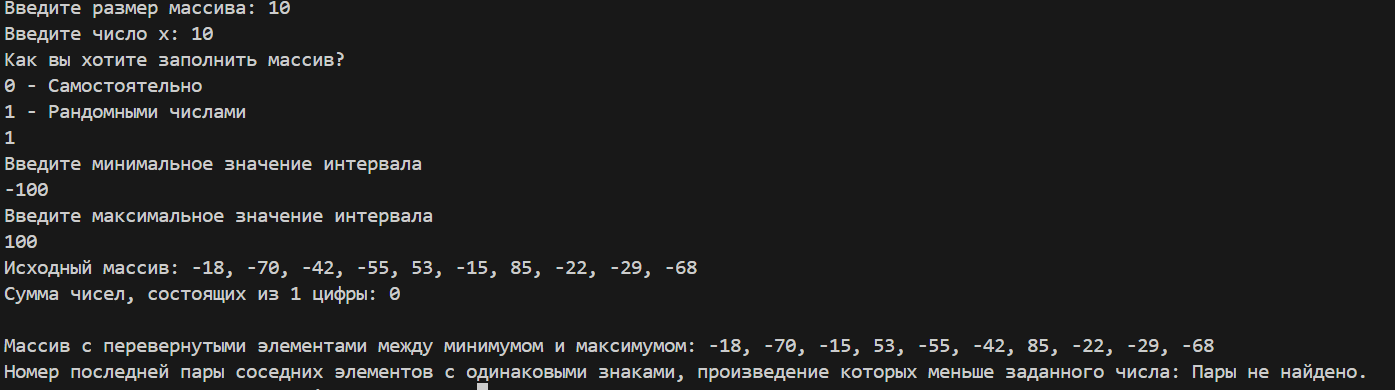


Рисунок 14 – Результаты выполнения программы



Рисунок 15 – Результаты выполнения программы при вводе отрицательного числа в величину размера массива.



Рисунок 16 – Результаты выполнения программы если размер массива – буква



Рисунок 17 – Результаты выполнения программы если размер массива является нолём

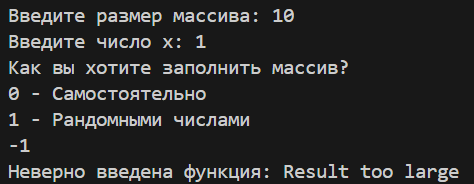


Рисунок 18 - Результаты выполнения программы, если число выбора отрицательно

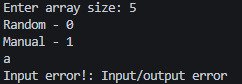


Рисунок 19 – Результаты выполнения программы, если введенный выбор – буква

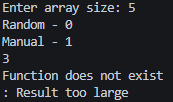
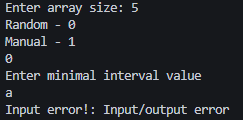


Рисунок 20 – Результат выполнения программы, если выбор больше диапазона

  
Рисунок 21 – Результат выполнения программы, если минимальное значения интервала – буква

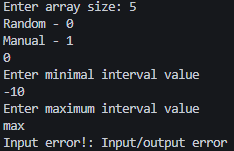


Рисунок 22 – Результат выполнения программы, если введенный максимум интервала – буква

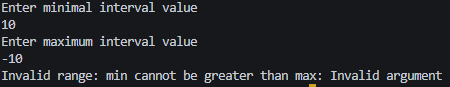


Рисунок 23 – Результат выполнения программы, если минимум интервала больше максимума

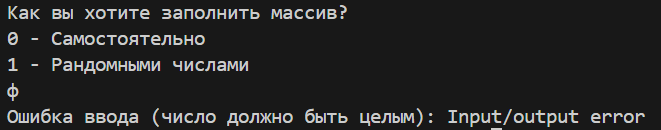


Рисунок 24 – Результат выполнения программы, если введенный элемент массива – буква

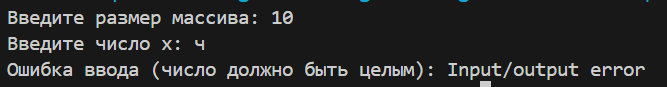


Рисунок 25 – Результат выполнения программы, если введенное значение x – буква

1. Отметка о выполнении задания в веб-хостинге системы контроля версий

