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

По дисциплине «Языки программирования»

ВАРИАНТ 5

Выполнил: ст. гр. ТКИ-142

Скрипников Егор Сергеевич

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 |
| 5 | 1. Заменить последние k элементов массива на противоположные по знаку. 2. Вывести индексы тех элементов, значения которых кратны 3. 3. Определить, есть ли пара соседних элементов с суммой, равной заданному числу. | [-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 getArray(n)



Рисунок 4 – Блок-схема функции copy(array, n)



Рисунок 5 – Блок-схема функции positiveInput()



Рисунок 6 – Блок-схема функции checkArray(array)



Рисунок 7 – Блок-схема функции fillArrayRandom(array, n, a, b)



Рисунок 8 – Блок-схема функции fillArrayManual (array, n)



Рисунок 9 – Блок-схема функции printArray(array, n)



Рисунок 10 – Блок-схема функции checkK (k, n)



Рисунок 11 – Блок-схема функции invertLastKElements

(arrayInverted, n, k)



Рисунок 12 – Блок-схема функции printIndicesDivisibleBy3

(array, n)



Рисунок 13 - Блок-схема функции hasAdjacentPairWithSum

(array, n, targetSum)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdbool.h>

*/\*\**

*\* @brief Считывает введеное значение*

*\* @return Возвращает введенное значение*

*\*/*

int input(void);

*/\*\**

*\* @brief Создает массив*

*\* @param n размер массива*

*\* @return Возвращает массив с n элементами*

*\*/*

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

*/\*\**

*\* @brief Копирует исходный массив*

*\* @param array исходный массив*

*\* @param n число элементов массива*

*\* @return Возвращает скопированный массив*

*\*/*

int\* copy(const int\* const array, const size\_t n);

*/\*\**

*\* @brief Проверяет введенное число на неотрицательность*

*\* @param value вводимое число*

*\* @return Возвращает ошибку, если число отрицательно*

*\*/*

int positiveInput(void);

*/\*\**

*\* @brief Проверяет значение k*

*\* @param k количество элементов с конца массива, которые нужно инвертировать*

*\* @param n количество элементов в массиве*

*\* @return возвращает ошибку если введено число больше количества элементов в массиве, либо если введено число меньше 0*

*\*/*

void checkK(const size\_t k,const size\_t n);

*/\*\**

*\* @brief Заполняет массив случайными числами*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @param min начало диапазона*

*\* @param max конец диапазона*

*\* @return возвращает массив, заполненный случайными числами*

*\*/*

void fillArrayRandom(int\* array, const size\_t n, const int min, const int max);

*/\*\**

*\* @brief Заполнение массива при помощи ввода элементов с клавиатуры*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @return возвращает массив, заполненный числами, введеными вручную*

*\*/*

void fillArrayManual(int\* array, const size\_t n);

*/\*\**

*\* @brief Выводит исходный массив*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @return Возвращает вывод исходного массива*

*\*/*

void printArray(const int\* array, const size\_t n);

*/\*\**

*\* @brief Меняет знак на противоположный у k элементов с конца массива*

*\* @param arrayInverted массив с инвертированными k элементами в массиве*

*\* @param n количество элементов в массиве*

*\* @param k количество элементов с конца массива, которые нужно инвертировать*

*\* @return Возвращает массив, с инвертированными элементами*

*\*/*

void invertLastKElements(int\* arrayInverted, const size\_t n, const size\_t k);

*/\*\**

*\* @brief Выводит индексы элементов массива, кратных 3*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @return Возвращает индексы элементов кратных 3*

*\*/*

void printIndicesDivisibleBy3(const int\* array, const size\_t n);

*/\*\**

*\* @brief Ищет пару соседних элементов с искомой суммой в массиве*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @param targetSum Искомая сумма*

*\* @return Вощвращает элементы с искомой суммой, а также их индексы*

*\*/*

bool hasAdjacentPairWithSum(const int\* array, const size\_t n,const int targetSum);

*/\*\**

*\* @brief Проверяет массив*

*\* @param array массив*

*\* @return возвращает ошибку, в случае нулевого массива*

*\*/*

void checkArray(const int\* array);

*/\*\**

*\* @param random заполнение массива случайными числами*

*\* @param manual заполнение массива пользователем*

*\*/*

enum request

{

    random, manual

};

*/\*\**

*\* @brief Точка входа в программу*

*\* @return Возвращает 0 в случае успеха*

*\*/*

int main(void)

{

    printf("Enter array size: ");

    size\_t n = positiveInput();

    int\* array = getArray(n);

    printf("Random - %d\n", random);

    printf("Manual - %d\n", manual);

    int choice = input();

    switch (choice)

    {

    case random:

        srand(time(0)); *// Инициализация генератора случайных чисел*

        printf("Enter minimal interval value \n");

        const int min = input();

        printf("Enter maximum interval value \n");

        const int max = input();

        fillArrayRandom(array, n, min, max);

        break;

    case manual:

        fillArrayManual(array, n);

        break;

    default:

        errno = ERANGE;

        perror("Function does not exist\n");

        exit(EXIT\_FAILURE);

        break;

    }

    printf("Source array: \n");

    printArray(array, n);

    printf("Enter amount of last elements to be inverted: ");

    size\_t k = positiveInput();

    checkK(k, n);

    int\* arrayInverted = copy(array, n);

    invertLastKElements(arrayInverted, n, k);

    printf("Array after inverting last %zu elements: ", k);

    printArray(arrayInverted,  n);

    free(arrayInverted);

    printf("Indices divisible by 3: ");

    printIndicesDivisibleBy3(array, n);

    printf("Enter the required sum: ");

    int targetSum = input();

    if (hasAdjacentPairWithSum(array, n, targetSum))

    {

        printf("Pair with required sum: %d found\n", targetSum);

    }

    else

    {

        printf("Pair with required sum: %d is not found.\n", targetSum);

    }

    free(array);

    return 0;

}

int input(void)

{

    int value = 0;

    int result = scanf("%d", &value);

    if (result != 1)

    {

        errno = EIO;

        perror("Input error!");

        exit(EXIT\_FAILURE);

    }

    return value;

}

int\* getArray(const size\_t n)

{

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

    checkArray(array);

    return array;

}

int\* copy(const int\* array, const size\_t n)

{

    checkArray(array);

    int\* copiedArray = getArray(n);

    for (size\_t i = 0; i < n; ++i)

    {

        copiedArray[i] = array[i];

    }

    return copiedArray;

}

int positiveInput(void)

{

    int value = input();

    if (value <= 0)

    {

        errno = EINVAL;

        perror("Value must be higher than zero");

        exit(EXIT\_FAILURE);

    }

    return value;

}

void checkArray(const int\* array)

{

    if (array == NULL)

    {

        errno = ENOMEM;

        perror("Memory allocation failed");

        exit(EXIT\_FAILURE);

    }

}

void checkK(size\_t k, const size\_t n)

{

    if (k > n)

    {

        errno = ERANGE;

        perror("Value cannot be higher than amount of elements");

        exit(EXIT\_FAILURE);

    }

}

void fillArrayRandom(int\* array, const size\_t n, const int min, const int max)

{

    checkArray(array);

    if (min > max)

    {

        errno = EINVAL;

        perror("Invalid range: min cannot be greater than max");

        exit(EXIT\_FAILURE);

    }

    for (size\_t i = 0; i < n; i++)

    {

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

    }

}

void fillArrayManual(int\* array, const size\_t n)

{

    checkArray(array);

    for (size\_t i = 0; i < n; i++)

    {

        printf("Enter element %zu: ", i);

        array[i] = input();

    }

}

void printArray(const int\* array, const size\_t n)

{

    checkArray(array);

    for (size\_t i = 0; i < n; i++)

    {

        printf("%d ", array[i]);

    }

    printf("\n");

}

void invertLastKElements(int\* arrayInverted, const size\_t n, const size\_t k)

{

    checkArray(arrayInverted);

    for (size\_t i = n - k; i < n; i++)

    {

        arrayInverted[i] = -arrayInverted[i];

    }

}

void printIndicesDivisibleBy3(const int\* array, const size\_t n)

{

    checkArray(array);

    for (size\_t i = 0; i < n; i++)

    {

        if (array[i] % 3 == 0)

        {

            printf("%d ", i);

        }

    }

    printf("\n");

}

bool hasAdjacentPairWithSum(const int\* array, const size\_t n, const int targetSum)

{

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

    {

        if (array[i] + array[i + 1] == targetSum)

        {

            return true;

        }

    }

    return false;

}

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)

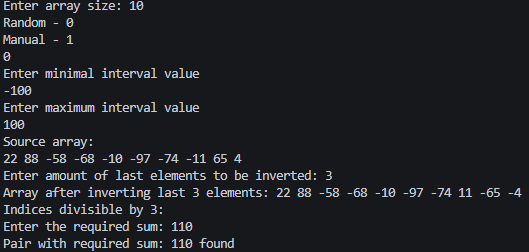


Рисунок 14 – Результаты выполнения программы



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 – нуль

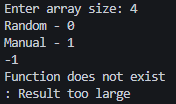


Рисунок 18 - Результаты выполнения программы, если число выбора отрицательно

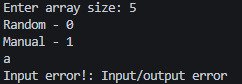


Рисунок 19 – Результаты выполнения программы, если введенный выбор – буква

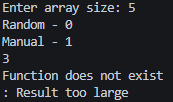
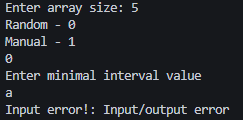


Рисунок 20 – Результат выполнения программы, если выбор больше диапазона

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

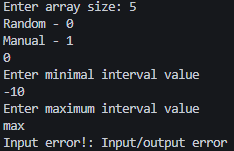


Рисунок 22 – Результат выполнения программы, если введенный максимум интервала – буква

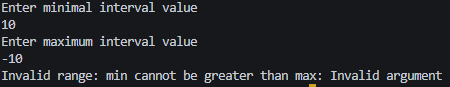


Рисунок 23 – Результат выполнения программы, если минимум интервала больше максимума

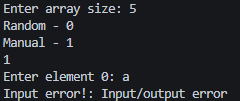


Рисунок 24 – Результат выполнения программы, если введенный элемент массива – буква

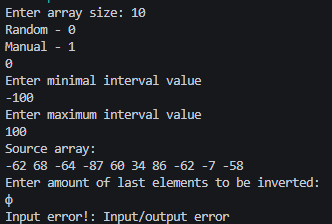


Рисунок 25 – Результат выполнения программы, если введенное значение k – буква

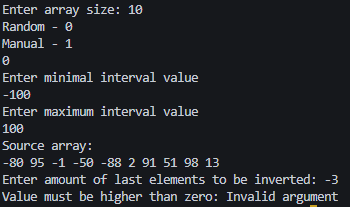


Рисунок 26 – Результат выполнения программы, если введенное значение k отрицательно

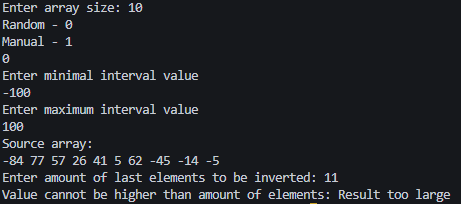


Рисунок 27 – Результат выполнения программы, если введенное значение k больше n

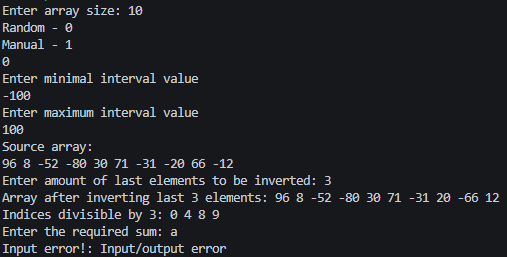


Рисунок 28 – Результат выполнения программы, если введенное значение искомой суммы – буква

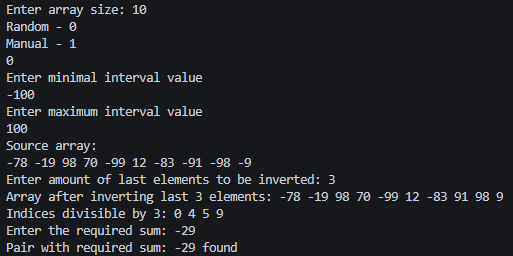


Рисунок 29 – Результат выполнения программы, когда искомая сумма найдена

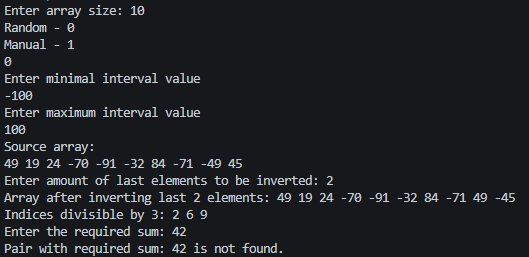


Рисунок 30 – Результат выполнения программы, если искомая сумма не найдена

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

