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). Вывести массив на экран. Составить блок-схему.

Таблица 1 – Формулировка задания 3.2

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 5 | 1. Заменить первый отрицательный элемент массива нулем. 2. Вставить число К после всех элементов, кратных своему номеру. 3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле Ai = (i-1)\*Di, а если нечетный, то по формуле Ai = Di\*i\*2. | [-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 1 -­ 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 replaceFirstNegativeWithZero(array, n)



Рисунок 11 – Блок-схема функции newSize(array, n)



Рисунок 12 – Блок-схема функции insertKIntoArray(array, m, k)



Рисунок 13 - Блок-схема функции createArrayAFromD(array, A, n)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <time.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\* array, const size\_t n);

*/\*\**

*\* @brief Проверяет введенное число на неотрицательность*

*\* @return Возвращает ошибку, если число отрицательно, иначе возвращает введенное значение*

*\*/*

int positiveInput(void);

*/\*\**

*\* @brief Заполняет массив случайными числами*

*\* @param array массив*

*\* @param n количество элементов в массиве*

*\* @param a начало диапазона*

*\* @param b конец диапазона*

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

*\*/*

void fillArrayRandom(int\* array, const size\_t n, const int a, const int b);

*/\*\**

*\* @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 Заменяет первый отрицательный элемент нулем*

*\* @param array исходный массив*

*\* @param n число элементов в массиве*

*\* @return Возвращает измененный массив*

*\*/*

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

*/\*\**

*\* @brief Формирует новый размер массива*

*\* @param array исходный массив*

*\* @param n исходный размер массива*

*\* @return Возвращает размер массива*

*\*/*

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

*/\*\**

*\* @brief Вставляет число k после каждого элемента, кратного своему индексу*

*\* @param array массив*

*\* @param m число элементов массива*

*\* @param k число, вводимое пользователем*

*\* @return Возвращает измененный массив*

*\*/*

int\* insertKIntoArray(int\* array, size\_t m, const int k);

*/\*\**

*\* @brief Создает массив A из D по правилам*

*\* @param array иссходный массив, он же массив D*

*\* @param A новый массив*

*\* @param n число элементов в массиве*

*\* @return Возвращает новый массив*

*\*/*

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

*/\*\**

*\* @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));

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

        const int a = input();

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

        const int b = input();

        fillArrayRandom(array, n, a, b);

        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);

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

    replaceFirstNegativeWithZero(clonnedArray, n);

    printf("Array after replacing first negative element with zero:\n");

    printArray(clonnedArray, n);

    free(clonnedArray);

    printf("Enter k value:");

    const int k = input();

    size\_t m = newSize(array, n);

    int\* arrayK = insertKIntoArray(array, m, k);

    printf("Array after inserting K after multiples of index:\n");

    printArray(arrayK, m);

    free(array);

    int\* A = getArray(m);

    createArrayAFromD(arrayK, A, m);

    free(arrayK);

    printf("Array A formed from source:\n");

    printArray(A, m);

    free(A);

    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 fillArrayRandom(int\* array, const size\_t n, const int a, const int b)

{

    if (a > b)

    {

        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() % (b-a+1)) - b;

    }

}

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

{

    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)

{

    printf("{ ");

    for (size\_t i = 0; i < n; i++)

    {

        printf("%d", array[i]);

        if (i < n - 1)

        {

            printf(", ");

        }

    }

    printf(" }\n");

}

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

{

    for (size\_t i = 0; i < n; i++)

    {

        if (array[i] < 0)

        {

            array[i] = 0;

            break;

        }

    }

}

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

{

    size\_t count = 0;

    if (array[0] == 0)

        {

            count++;

        }

    for (size\_t i = 1; i < n; i++)

    {

        if (array[i] % i == 0)

        {

            count++;

        }

    }

    size\_t newSize = n + count;

    return newSize;

}

int\* insertKIntoArray(int\* array, size\_t m, const int k)

{

    int\* newArray = getArray(m);

    size\_t j = 0;

    for (size\_t i = 0; i < m; i++)

    {

        newArray[j++] = array[i];

        if (i == 0 && array[i] == 0)

        {

            newArray[j++] = k;

        }

        if (i > 0 && array[i] % i == 0)

        {

            newArray[j++] = k;

        }

    }

    return newArray;

}

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

{

    for (size\_t i = 0; i < n; i++)

    {

        if (i % 2 == 0)

        {

            A[i] = (i) \* array[i];

        } else

        {

            A[i] = array[i] \* (i + 1) \* 2;

        }

    }

}

1. Результаты выполнения программы

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

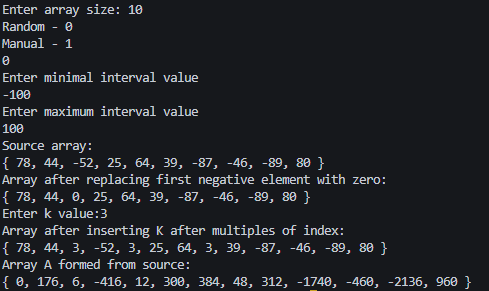


Рисунок 14 – Результаты выполнения программы



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

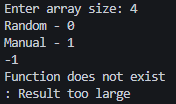


Рисунок 18 - Результаты выполнения программы, если число выбора отрицательно

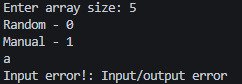


Рисунок 19 – Результаты выполнения программы, если введенный выбор – буква

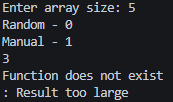
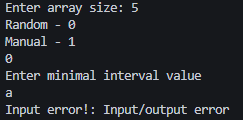


Рисунок 20 – Результат выполнения программы, если выбор больше диапазона

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

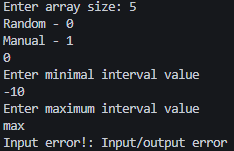


Рисунок 22 – Результат выполнения программы, если введенный максимум интервала – буква

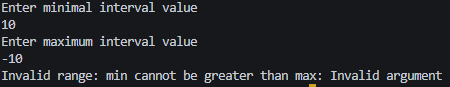


Рисунок 23 – Результат выполнения программы, если минимум интервала больше максимума

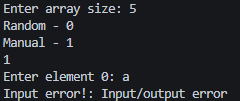


Рисунок 24 – Результат выполнения программы, если введенный элемент массива – буква

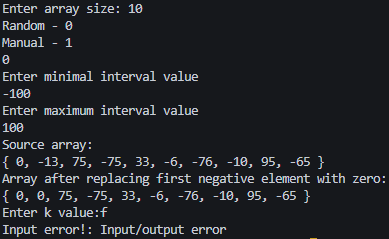


Рисунок 25 – Результат выполнения программы, если введенное значение k - буква

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

