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 9

Выполнил: ст. гр. ТКИ - 141

Сивунов Пётр Сергеевич

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

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

Москва 2024

4.1.1 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  – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 9 | 1. Найти сумму элементов, значения которых по модулю меньше 10. 2. Вывести индексы тех элементов, значения которых больше значения последующего элемента.   Умножить все элементы массива, кратные 3, на третий элемент массива. | [-40;40] |

4.1.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций представлены ниже (Рисунок 2).

Изображение выглядит как текст, снимок экрана, дизайн, черно-белый

Автоматически созданное описание

Рисунок  ­ Блок-схема основного алгоритма

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок  – Блок-схема функций ввода

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Рисунок 3 – Блок-схема функций ввода массива

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок 4 – Блок-схема функций для задачи

4.1.3 Текст программы на языке C

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "array\_functions.h"

/\*\*

\* @brief Точка входа

\* @return 0

\*/

int main(void)

{

setlocale(LC\_ALL, "Rus");

const size\_t size = get\_positive\_int("Введите количество элементов массива");

int\* arr = try\_allocate\_memory(size);

puts("Выберите способ заполнения массива:");

printf("%d. Заполнить случайными числами\n", RANDOM\_FILL);

printf("%d. Заполнить с клавиатуры\n", MANUAL\_FILL);

const FillMethod fill\_method = int\_input(NULL);

switch (fill\_method)

{

case RANDOM\_FILL:

{

fill\_array\_random(arr, size);

break;

}

case MANUAL\_FILL:

{

fill\_array\_manual(arr, size);

break;

}

default:

{

puts("Неверный выбор!");

free(arr);

return 1;

}

}

printf("Исходный массив:\n");

print\_array(arr, size);

int sum = sum\_of\_elements\_less\_than\_10(arr, size);

printf("Сумма элементов, значения которых по модулю меньше 10: %d\n", sum);

print\_indices\_greater\_than\_next(arr, size);

int \*arr\_copy = copy\_array(arr, size);

multiply\_by\_third\_element(arr\_copy, size);

puts("Массив после умножения элементов, кратных 3, на третий элемент");

print\_array(arr\_copy, size);

free(arr\_copy);

free(arr);

return 0;

}

int int\_input(const char\* prompt)

{

if (prompt)

{

puts(prompt);

}

int value = 0;

if (scanf\_s("%d", &value) != 1)

{

abort();

}

return value;

}

void print\_array(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

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

{

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

}

putchar('\n');

}

void fill\_array\_random(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

const int bottom\_limit = int\_input("Нижняя граница");

const int top\_limit = int\_input("Верхняя граница");

if (top\_limit < bottom\_limit)

{

puts("Верхняя граница не может быть меньше нижней");

abort();

}

srand(time(0));

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

{

arr[i] = rand() % (top\_limit - bottom\_limit + 1) + bottom\_limit;

}

}

void fill\_array\_manual(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

printf("Введите %llu целых чисел:\n", size);

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

{

printf("Элемент %llu: ", i + 1);

arr[i] = int\_input(NULL);

}

}

int sum\_of\_elements\_less\_than\_10(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

int sum = 0;

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

{

if (abs(arr[i]) < 10)

{

sum += arr[i];

}

}

return sum;

}

void print\_indices\_greater\_than\_next(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

puts("Индексы элементов, которые больше следующего");

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

{

if (arr[i] > arr[i + 1])

{

printf("%llu ", i);

}

}

printf("\n");

}

void multiply\_by\_third\_element(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

if (size < 3)

{

puts("Недостаточно элементов для умножения на третий элемент");

return;

}

const int third\_element = arr[2];

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

{

if (arr[i] % 3 == 0)

{

arr[i] \*= third\_element;

}

}

}

int\* try\_allocate\_memory(const size\_t size)

{

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

return array\_is\_not\_null(array);

}

int\* copy\_array(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

int\* arr\_copy = try\_allocate\_memory(size);

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

{

arr\_copy[i] = arr[i];

}

return arr\_copy;

}

int get\_positive\_int(const char\* prompt)

{

int value = 0;

if (prompt)

{

value = int\_input(prompt);

}

if (value < 0)

{

abort();

}

return value;

}

const int\* array\_is\_not\_null(const int\* arr)

{

if (arr == NULL)

{

abort();

}

return arr;

}

4.1.4 Результаты выполнения программы

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5 – Результаты выполнения программы

4.2.1 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 2 – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 9 | 1. Заменить минимальный по модулю положительный элемент массива нулем. 2. Удалить из него все элементы, первая и последняя цифра которых четная. 3. Из элементов массива D сформировать массив A той же размерности по правилу: элементы с 3-го по 12-й находятся по формуле Ai = -Di2, остальные по формуле Ai = Di-1 | [-40;40] |

4.2.2 БЛОК-СХЕМА АЛГОРИТМА

Изображение выглядит как текст, снимок экрана, черно-белый, линия

Автоматически созданное описание

Рисунок 6 – Блок схема функции main

Изображение выглядит как текст, снимок экрана, Графика, графический дизайн

Автоматически созданное описание

Рисунок 7 - Блок схема вспомогательных функций

Изображение выглядит как текст, снимок экрана, Графика, Шрифт

Автоматически созданное описание

Рисунок 8 - Блок схема функций для задания

4.2.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "array\_functions.h"

/\*\*

\* @brief Точка входа

\* @return 0

\*/

int main(void)

{

setlocale(LC\_ALL, "Rus");

const size\_t size = int\_input("Введите количество элементов массива");

int\* arr = try\_allocate\_memory(size);

puts("Выберите способ заполнения массива:");

printf("%d. Заполнить случайными числами\n", RANDOM\_FILL);

printf("%d. Заполнить с клавиатуры\n", MANUAL\_FILL);

const FillMethod fill\_method = int\_input(NULL);

switch (fill\_method)

{

case RANDOM\_FILL:

{

fill\_array\_random(arr, size);

break;

}

case MANUAL\_FILL:

{

fill\_array\_manual(arr, size);

break;

}

default:

{

puts("Неверный выбор!");

free(arr);

return 1;

}

}

puts("Исходный массив:");

print\_array(arr, size);

int\* arr\_copy1 = copy\_array(arr, size);

replace\_min\_positive\_with\_zero(arr\_copy1, size);

puts("Массив после замены минимального положительного элемента на 0:");

print\_array(arr\_copy1, size);

free(arr\_copy1);

const size\_t new\_size = count\_elements\_to\_save(arr, size);

int\* filtered\_array = remove\_even\_last(arr, size, new\_size);

puts("Массив после удаления элементов с четной первой и последней цифрой:");

print\_array(filtered\_array, new\_size);

free(filtered\_array);

int\* array\_a = form\_array\_based\_on\_rule(arr, size);

printf("Массив A, сформированный по правилам:\n");

print\_array(array\_a, size);

free(arr);

free(array\_a);

return 0;

}

size\_t find\_first\_positive\_index(const int\* arr, const size\_t size)

{

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

{

if (arr[i] > 0)

{

return i;

}

}

return size+1;

}

size\_t find\_min\_positive\_index(const int\* arr, const size\_t size)

{

size\_t min\_positive\_index = find\_first\_positive\_index(arr, size);

if (min\_positive\_index == size+1)

{

puts("Нет положительных чисел");

abort();

}

for (size\_t i = min\_positive\_index + 1; i < size; i++)

{

if (arr[i] > 0 && arr[i] < arr[min\_positive\_index])

{

min\_positive\_index = i;

}

}

return min\_positive\_index;

}

void replace\_min\_positive\_with\_zero(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

const size\_t min\_positive\_index = find\_min\_positive\_index(arr, size);

arr[min\_positive\_index] = 0;

}

int get\_first\_digit(const int n)

{

int first\_digit = abs(n);

while (first\_digit >= 10)

{

first\_digit /= 10;

}

return first\_digit;

}

size\_t count\_elements\_to\_save(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

size\_t count\_to\_save = 0;

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

{

const int first\_digit = get\_first\_digit(arr[i]);

const int last\_digit = abs(arr[i]) % 10;

if (!(first\_digit % 2 == 0 && last\_digit % 2 == 0))

{

count\_to\_save++;

}

}

return count\_to\_save;

}

int\* remove\_even\_last(const int\* arr, const size\_t size, const size\_t new\_size)

{

arr = array\_is\_not\_null(arr);

int\* filtered\_arr = try\_allocate\_memory(new\_size);

size\_t filtered\_size = 0;

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

{

const int first\_digit = get\_first\_digit(arr[i]);

const int last\_digit = abs(arr[i]) % 10;

if (!(first\_digit % 2 == 0 && last\_digit % 2 == 0))

{

filtered\_arr[filtered\_size++] = arr[i];

}

}

return filtered\_arr;

}

int\* form\_array\_based\_on\_rule(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

int\* array\_A = try\_allocate\_memory(size);

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

{

if (i >= 2 && i <= 11)

{

array\_A[i] = -(arr[i] \* arr[i]);

}

else

{

array\_A[i] = arr[i] - 1;

}

}

return array\_A;

}

int int\_input(const char\* prompt)

{

if (prompt)

{

puts(prompt);

}

int value = 0;

if (scanf\_s("%d", &value) != 1)

{

abort();

}

return value;

}

void print\_array(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

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

{

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

}

putchar('\n');

}

void fill\_array\_random(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

const int bottom\_limit = int\_input("Нижняя граница");

const int top\_limit = int\_input("Верхняя граница");

if (top\_limit < bottom\_limit)

{

puts("Верхняя граница не может быть меньше нижней");

abort();

}

srand(time(0));

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

{

arr[i] = rand() % (top\_limit - bottom\_limit + 1) + bottom\_limit;

}

}

void fill\_array\_manual(int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

printf("Введите %llu целых чисел:\n", size);

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

{

printf("Элемент %llu: ", i + 1);

arr[i] = int\_input(NULL);

}

}

int\* try\_allocate\_memory(const size\_t size)

{

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

return array\_is\_not\_null(array);

}

int\* copy\_array(const int\* arr, const size\_t size)

{

arr = array\_is\_not\_null(arr);

int\* arr\_copy = try\_allocate\_memory(size);

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

{

arr\_copy[i] = arr[i];

}

return arr\_copy;

}

const int\* array\_is\_not\_null(const int\* arr)

{

if (arr == NULL)

{

abort();

}

return arr;

}

4.2.4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Результаты выполнения программы

4.3.1 ФОРМУЛИРОВКА ЗАДАНИЯ

Создать многомерный массив nˣm 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.

Таблица 3 – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |
| 9 | 1. Заменить минимальный элемент каждого столбца нулем. 2. Вставить после каждой строки, содержащей максимальный по модулю элемент, последнюю строку. |

4.3.2 БЛОК-СХЕМА АЛГОРИТМА

Изображение выглядит как текст, снимок экрана, черно-белый, черный

Автоматически созданное описание

Рисунок 10 – Блок схема main

Изображение выглядит как текст, снимок экрана, Графика, графический дизайн

Автоматически созданное описание

Рисунок 11 – Блок схемы функций ввода

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

Рисунок 12 – Блок схемы функций управления памятью

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Рисунок 13 – Блок схемы функций задачи

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок 14 – Блок схемы функций задачи

4.3.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

#include <float.h>

#include "array\_functions.h"

int main(void)

{

setlocale(LC\_ALL, "Rus");

const size\_t n = (size\_t)int\_input("Введите количество строк (n): ");

const size\_t m = (size\_t)int\_input("Введите количество столбцов (m): ");

int\*\* arr = try\_allocate\_2d\_array(n, m);

puts("Выберите способ заполнения массива:");

printf("%d. Заполнить случайными числами\n", RANDOM\_FILL);

printf("%d. Заполнить с клавиатуры\n", MANUAL\_FILL);

const FillMethod fill\_method = int\_input(NULL);

switch (fill\_method)

{

case RANDOM\_FILL:

fill\_2d\_array\_random(arr, n, m);

break;

case MANUAL\_FILL:

fill\_2d\_array\_manual(arr, n, m);

break;

default:

printf("Неверный выбор!\n");

free\_2d\_array(arr, n);

return 1;

}

printf("Исходный массив:\n");

print\_2d\_array(arr, n, m);

// // 1. Заменить минимальный элемент каждого столбца нулем.

int\*\* arr\_copy = copy\_2d\_array(arr, n, m);

replace\_min\_in\_columns\_with\_zero(arr\_copy, n, m);

printf("Массив после замены минимальных элементов каждого столбца на 0:\n");

print\_2d\_array(arr\_copy, n, m);

free\_2d\_array(arr\_copy, n);

// 2. Вставить после каждой строки с максимальным элементом последнюю строку.

const size\_t new\_n = get\_rows\_with\_max(arr, n, m);

int\*\* new\_arr = insert\_last\_row\_after\_max(arr, n, new\_n, m);

printf("Массив после вставки последней строки:\n");

print\_2d\_array(new\_arr, new\_n, m);

free\_2d\_array(new\_arr, new\_n);

free\_2d\_array(arr, n);

return 0;

}

#include "array\_functions.h"

int int\_input(const char\* prompt)

{

if (prompt)

{

puts(prompt);

}

int value = 0;

if (scanf\_s("%d", &value) != 1)

{

abort();

}

return value;

}

int\*\* allocate\_2d\_array(const size\_t n, const size\_t m)

{

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

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

{

arr[i] = (int\*)malloc(m \* sizeof(int));

if (arr[i] == NULL)

{

abort();

}

}

return arr;

}

int\*\* try\_allocate\_2d\_array(const size\_t n, const size\_t m)

{

int\*\* arr = allocate\_2d\_array(n, m);

return array\_is\_not\_null(arr);

}

void fill\_2d\_array\_random(int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

const int bottom\_limit = int\_input("Нижняя граница");

const int top\_limit = int\_input("Верхняя граница");

if (top\_limit < bottom\_limit)

{

puts("Верхняя граница не может быть меньше нижней");

abort();

}

srand(time(0));

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

{

for (size\_t j = 0; j < m; j++)

{

arr[i][j] = rand() % (top\_limit - bottom\_limit + 1) + bottom\_limit;

}

}

}

void fill\_2d\_array\_manual(int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

printf("Введите элементы массива размером %llu x %llu:\n", n, m);

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

{

for (size\_t j = 0; j < m; j++)

{

printf("Элемент [%llu][%llu]: ", i + 1, j + 1);

arr[i][j] = int\_input(NULL);

}

}

}

void print\_2d\_array(const int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

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

{

for (size\_t j = 0; j < m; j++)

{

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

}

printf("\n");

}

}

void replace\_min\_in\_columns\_with\_zero(int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

for (size\_t j = 0; j < m; j++)

{

int min\_val = get\_column\_min(arr, n, j);

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

{

if (arr[i][j] == min\_val)

{

arr[i][j] = 0;

}

}

}

}

size\_t get\_rows\_with\_max(const int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

const int max\_val = get\_2d\_array\_max(arr, n, m);

size\_t rows\_with\_max = 0;

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

{

for (size\_t j = 0; j < m; j++)

{

if (arr[i][j] == max\_val)

{

rows\_with\_max++;

break;

}

}

}

return rows\_with\_max + n;

}

int\*\* insert\_last\_row\_after\_max(int\*\* arr, const size\_t old\_n, const size\_t new\_n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

const int max\_val = get\_2d\_array\_max(arr, old\_n, m);

const int\* last\_row = arr[old\_n - 1];

int\*\* new\_arr = try\_allocate\_2d\_array(new\_n, m);

size\_t curr\_new\_ind = 0;

for (size\_t i = 0; i < old\_n; i++)

{

for (size\_t j = 0; j < m; j++) {

new\_arr[curr\_new\_ind][j] = arr[i][j];

}

curr\_new\_ind++;

for (size\_t j = 0; j < m; j++)

{

if (arr[i][j] == max\_val)

{

for (size\_t j = 0; j < m; j++) {

new\_arr[curr\_new\_ind][j] = last\_row[j];

}

curr\_new\_ind++;

break;

}

}

}

return new\_arr;

}

void free\_2d\_array(int\*\* arr, const size\_t n)

{

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

{

free(arr[i]);

}

free(arr);

}

int\*\* copy\_2d\_array(const int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

int\*\* new\_arr = try\_allocate\_2d\_array(n, m);

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

{

for (size\_t j = 0; j < m; j++)

{

new\_arr[i][j] = arr[i][j];

}

}

return new\_arr;

}

int get\_column\_min(const int\*\* arr, const size\_t n, const size\_t column\_index)

{

arr = array\_is\_not\_null(arr);

int min\_val = arr[0][column\_index];

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

{

if (arr[i][column\_index] < min\_val)

{

min\_val = arr[i][column\_index];

}

}

return min\_val;

}

int get\_2d\_array\_max(const int\*\* arr, const size\_t n, const size\_t m)

{

arr = array\_is\_not\_null(arr);

int max\_val = arr[0][0];

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

{

for (size\_t j = 0; j < m; j++)

{

if (arr[i][j] > max\_val)

{

max\_val = arr[i][j];

}

}

}

return max\_val;

}

const int\*\* array\_is\_not\_null(const int\*\* arr)

{

if (arr == NULL)

{

abort();

}

return arr;

}

4.3.4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 15 – Результаты выполнения программы

**Approve скриншоты**

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание