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 №4.2

По дисциплине «Языки программирования»

ВАРИАНТ 4

Выполнил: ст. гр. ТКИ-141

Бышовец Михаил Александрович

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.

Таблица 1 – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |
| 4 | 1. Заменить первый отрицательный элемент массива на первый положительный. 2. Удалить из него все элементы кратные 7 и принадлежащие промежутку [a, b]. 3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле Ai = Di2+i, а если нечетный, то по формуле Ai = i\*Di |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схема функций is\_positive (Рисунок 2), create\_list (Рисунок 3), is\_zero (Рисунок 4), replace (Рисунок 5), initialize\_manually и initialize\_rand (Рисунок 6), функций ввода input и positive\_input (Рисунок 7), функции check\_list\_null (Рисунок 8) и функции, find\_unpositive\_element (Рисунок 9), find\_positive\_element (Рисунок 10), len\_divis\_seven (Рисунок 11), list\_divis\_seven (Рисунок 12), create\_list (Рисунок 13) представлены ниже.

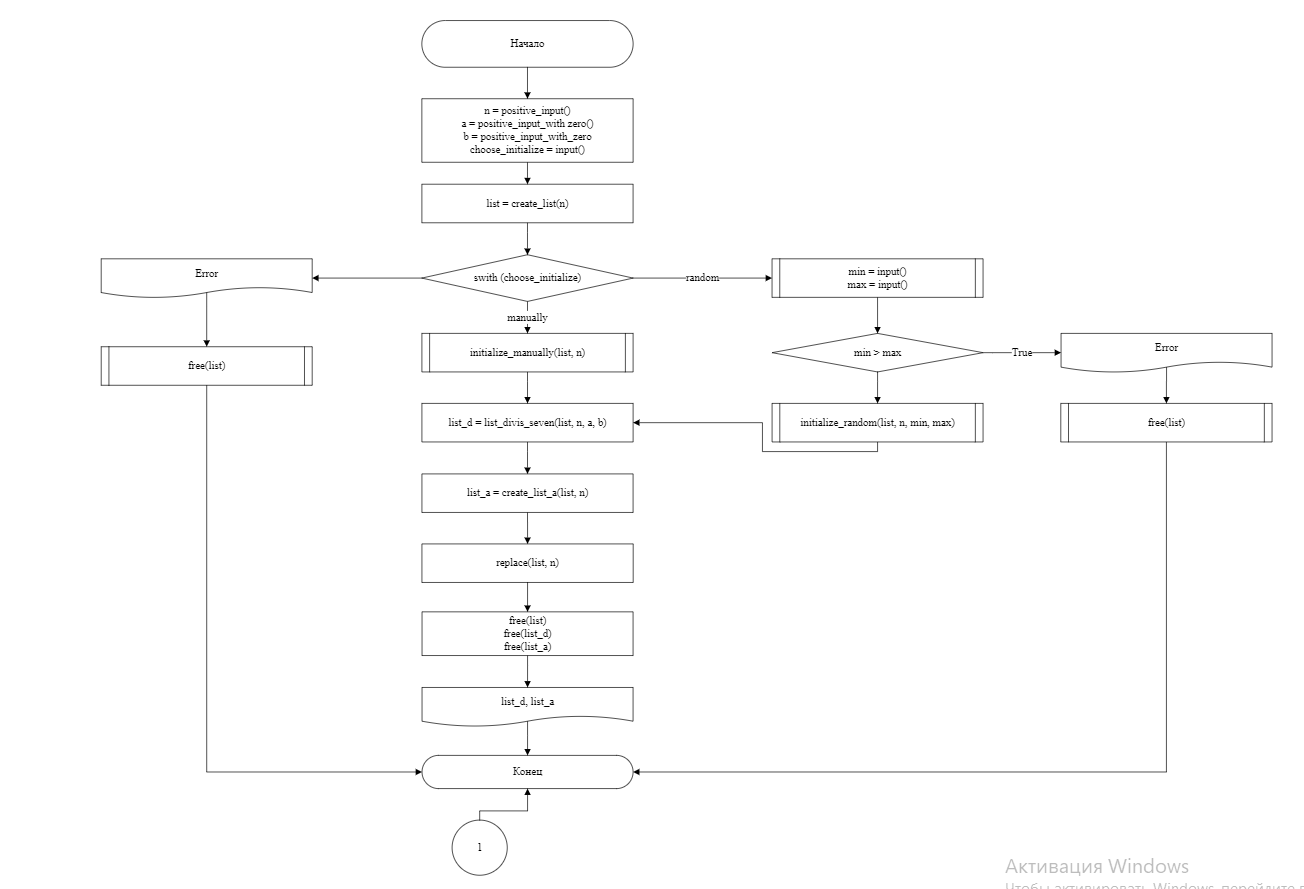


Рисунок 1 ­ Блок-схема основного алгоритма

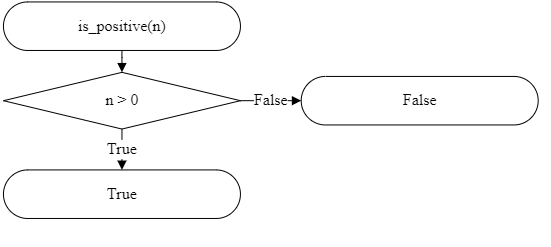


Рисунок 2 – Блок-схемы используемой функции is\_positive

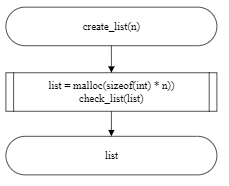


Рисунок 3 –  Блок-схема используемой функции create\_list

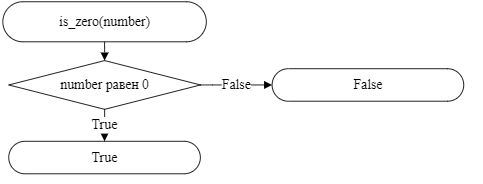


Рисунок 4 - Блок-схема используемой функции count\_of\_two\_char

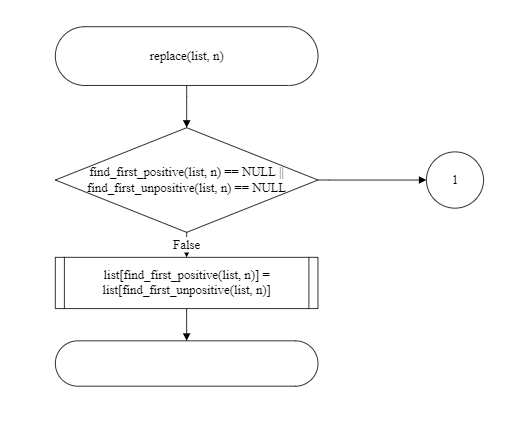


Рисунок 5 - Блок-схема используемой функции replace

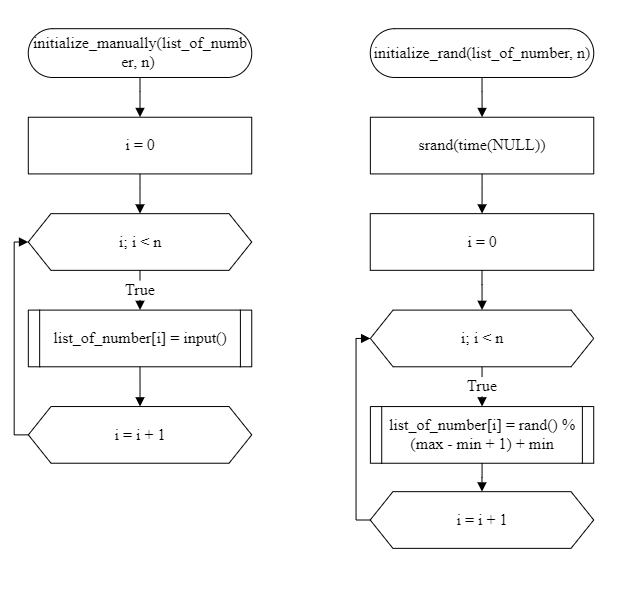


Рисунок 6 - Блок-схема используемых функий initialize\_manually и initialize\_rand

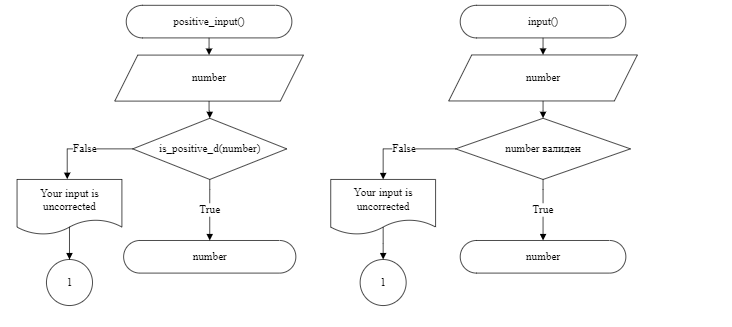


Рисунок 7 - используемые функции ввода positive\_input и input

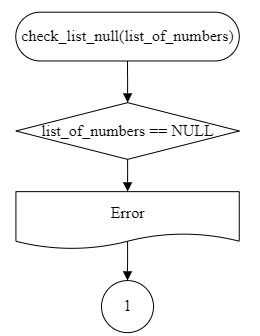
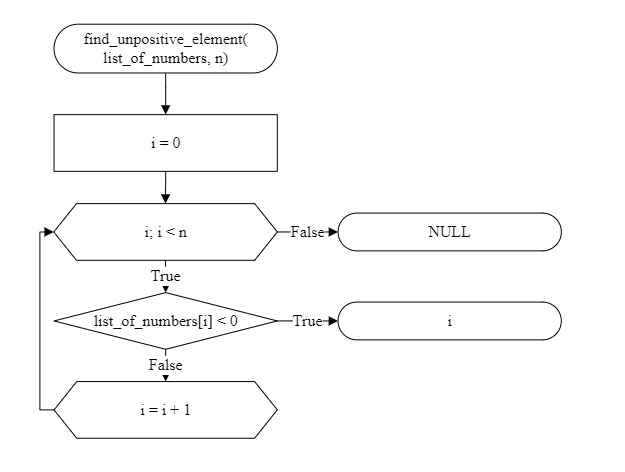


Рисунок 8 - Блок-схема используемой функции check\_list\_null

Рисунок 9 - Блок-схема используемой функции find\_unpositive\_element

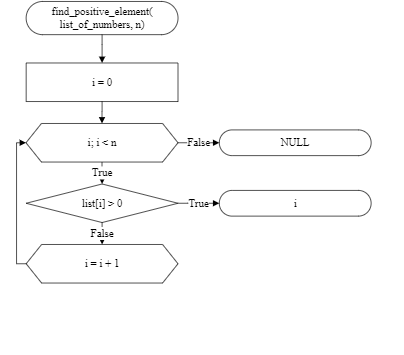


Рисунок 10 - Блок-схема используемой функции find\_positive\_element

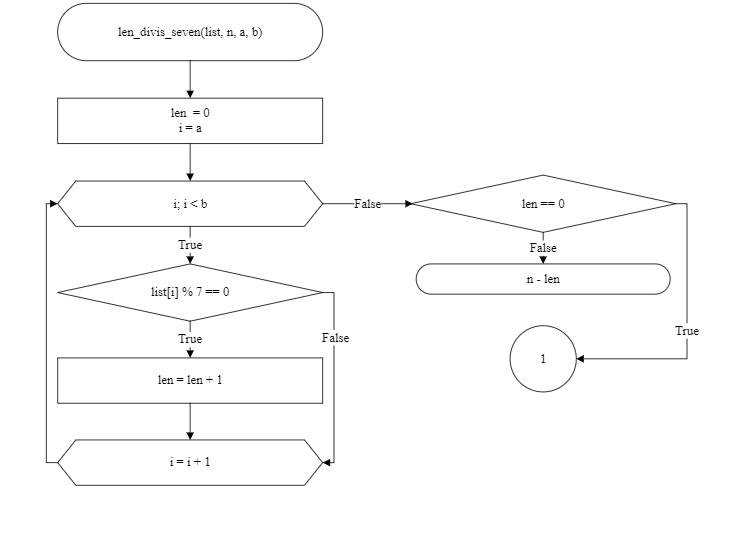


Рисунок 11 - Блок-схема используемой функции len\_divis\_seven

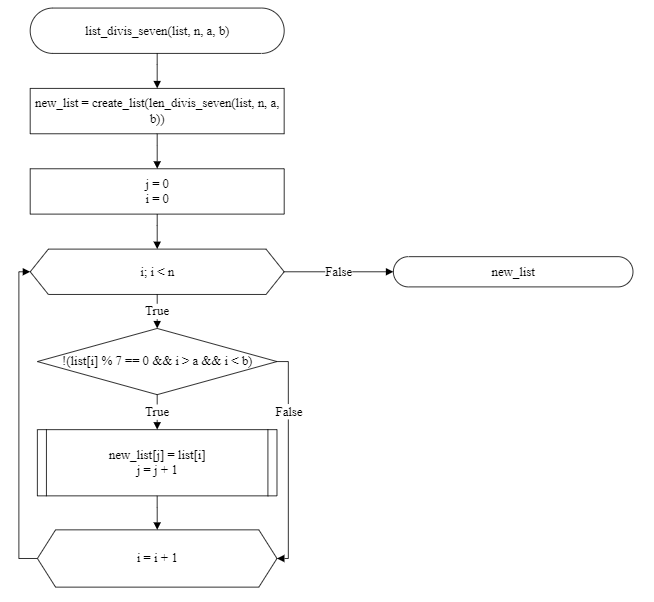
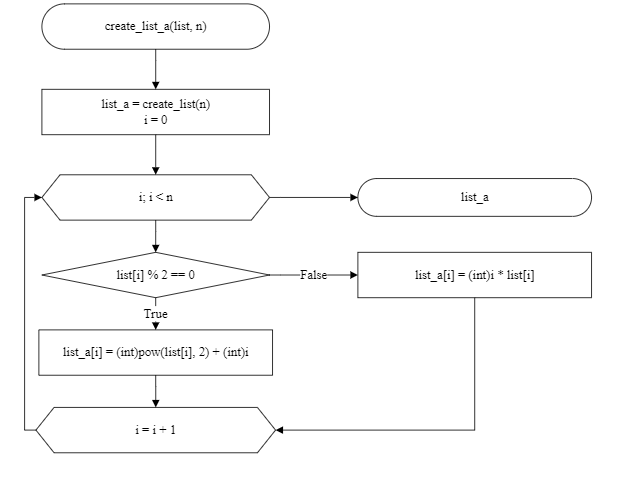


Рисунок 12 - Блок-схема используемой функции list\_divis\_seven

Рисунок 13 - Блок-схема используемой функции create\_list\_a

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

#include <time.h>

/\*\*

@brief input считывает значение, введенное пользователем

@return возвращает значение, введённо пользователем

\*/ int input(void);

/\*\*

@brief is\_positive проверяет больше ли число 0 или нет

@param number число

@return возвращает true, если число больше 0 и false, если число меньше 0

\*/ bool is\_positive(const int number);

/\*\*

@brief positive\_input считывает значение введённое пользователем и проверяет его на положительность

@return возвращает число, введённое пользователем

\*/ size\_t positive\_input(void);

/\*\*

@brief initialize\_manually заполняет массив числами, введёнными пользователем

@param list массив

@param n длина массива

*/ void initialize\_manually(int* list, const size\_t n);

/\*\*

@brief initialize\_random заполняет массив случайными числами

@param list массив

@param n длина массива

@param min минимальный элемент массива

@param max максимальный элемент массива

*/ void initialize\_random(int* list, const size\_t n, const int min, const int max);

/\*\*

@brief positive\_input\_with\_zero считывает значение типа int введённое пользователем

@return возвращает значение введённое пользователем

\*/ int positive\_input\_with\_zero(void);

/\*\*

@brief check\_list проверяет выделилась ли память для массива

@param list массив

*/*

*void check\_list\_null(const int* list);

/\*\*

@brief create\_list создаёт массив

@param n длина массива

@return возвращает указатель на первый элемент массива

*/ int* create\_list(const size\_t n);

/\*\*

@brief list\_divis\_seven удаляет из массива все элементы, которые деляется на 7

@param list массив

@param n длина массива

@return возвращает указатель на первый элемент массива, из которого удалены все элементы массива, делящиеся на 7

*/ int* list\_divis\_seven(const int\* list, const size\_t n, const size\_t a, const size\_t b);

/\*\*

@brief create\_list\_a создаёт новый массив

@param list массив

@param n длина массива

@return возвращает указатель на первый элемент нового массива

*/ int* create\_list\_a(const int\* list, const size\_t n);

/\*\*

@brief is\_zero проверяет является ли число 0

@param number число

@return возвращает true, если число является 0 и false, если не является

\*/ bool is\_zero(const int number);

/\*\*

@brief len\_divis\_seven считает длину массива без чисел, кратных 7

@param list массив

@param n длина массива

@return возвращает длину нового массива

*/ size\_t len\_divis\_seven(const int* list, const size\_t n, const size\_t a, const size\_t b);

/\*\*

@brief find\_unpositive\_element ищет первый отрицательный элемент

@param list массив

@param n длина массива

@return возвращает индекс первого отрицательного элемента

*/ size\_t find\_unpositive\_element(const int* list, const size\_t n);

/\*\*

@brief find\_positive\_element ищет первый положительный элемент в массиве

@param list массив

@param n длина массива

@return возвращает индекс первого положительного элемента

*/ size\_t find\_positive\_element(const int* list, const size\_t n);

/\*\*

@brief replace меняет значение элемента

@param list массив

@param n длина массива

*/ void replace(const int* list, const size\_t n);

typedef enum { manually = 1, random }choose;

int main(void) {

puts("Enter the len of list:");

const size\_t n = positive\_input;

const size\_t a = positive\_input\_with\_zero;

const size\_t b = positive\_input\_with\_zero;

const choose choose\_initialize = (choose)input();

int\* list = create\_list(n);

switch (choose\_initialize){

case manually:

{

initialize\_manually(list, n); break;

}

case random:

{

int min = input();

int max = input();

if (min > max)

{

free(list);

return 1;

}

initialize\_random(list, n, min, max);

break;

}

default:

{

puts("Error");

free(list); return 1;

}

}

replace(list, n);  
int\* list\_d = list\_divis\_seven(list, n, a, b);  
  
int\* list\_a = create\_list\_a(list, n);  
  
printf("list = { ");  
for (size\_t i = 0; i < n; i++)  
{  
 printf("%d", list\_d[i]);  
}  
  
printf(" }\n");  
printf("list\_a = { ");  
for (size\_t i = 0; i < n; i++)  
{  
 printf("%d", list\_a[i]);  
}  
printf(" }\n");  
  
free(list);  
free(list\_d);  
free(list\_a);  
return 0;

}

int input(void) { int number = 0;

if (scanf\_s("%d", &number) != 1)  
{  
 puts("Error");  
 exit(EXIT\_FAILURE);  
}  
  
return number;

}

bool is\_positive(const int number) { return number > 0; }

size\_t positive\_input(void) { int number = input();

if (!is\_positive(number))  
{  
 puts("Error");  
 exit(EXIT\_FAILURE);  
}  
  
return (size\_t)number;

}

void initialize\_manually(int\* list, const size\_t n) { for (size\_t i = 0; i < n; i++) { list[i] = input(); } }

void initialize\_random(int\* list, const size\_t n, const int min, const int max) { srand((unsigned int)time((time\_t\*)NULL));

for (size\_t i = 0; i < n; i++)  
{  
 list[i] = rand() % (max - min + 1) + min;  
}

}

bool is\_zero(const int number) { return number == 0; }

int positive\_input\_with\_zero(void) { int number = input();

if (!(is\_zero(number) || is\_positive(number)))  
{  
 puts("Error");  
 exit(EXIT\_FAILURE);  
}  
  
return number;

}

void check\_list\_null(const int\* list) { if (list == NULL) { puts("Error"); exit(EXIT\_FAILURE); } }

int\* create\_list(const size\_t n) { int\* list = (int\*)malloc(sizeof(int) \* n); check\_list(list); return list; }

int\* list\_divis\_seven(const int\* list, const size\_t n, const size\_t a, const size\_t b)

{

int\* new\_list = create\_list(len\_divis\_seven(list, n, a, b));

size\_t j = 0;

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

{

if (!(list[i] % 7 == 0 && i > a && i < b))

{

new\_list[j] = list[i]; j++;

}

}

return new\_list;

}

size\_t len\_divis\_seven(const int\* list, const size\_t n, const size\_t a, const size\_t b) { size\_t len = 0;

for (size\_t i = a; i <+ b; i++)  
{  
 if (list[i] % 7 == 0)  
 {  
 len++;  
 }  
}  
  
return n - len;

}

int\* create\_list\_a(const int\* list, const size\_t n) { int\* list\_a = create\_list(n);

for (size\_t i = 0; i < n; i++)  
{  
 if (list[i] % 2 == 0)  
 {  
 list\_a[i] = (int)pow(list[i], 2) + (int)i;  
 }  
 else {  
 list\_a[i] = (int)i \* list[i];  
 }  
}  
  
return list\_a;

}

size\_t find\_positive\_element(const int\* list, const size\_t n)

{

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

{

if (is\_positive(list[i]))

{

return i;

}

}

return NULL;

}

size\_t find\_unpositive\_element(const int\* list, const size\_t n)

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

{

if (!is\_positive(list[i]) && !is\_zero(list[i]))

{

return i;

}

}

return NULL;

}

void replace(int\* list, const size\_t n) {

if (find\_positive\_element(list, n) == NULL || find\_unpositive\_element(list, n) == NULL)

{

exit(EXIT\_FAILURE);

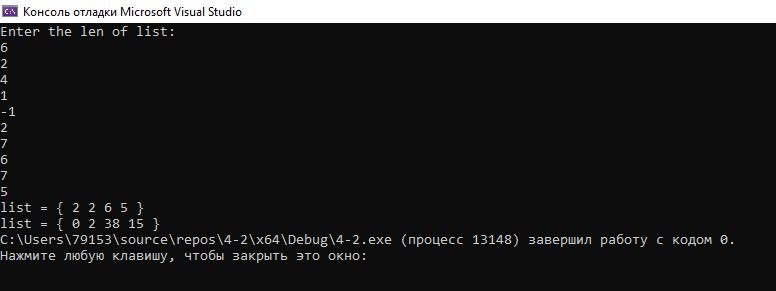
}

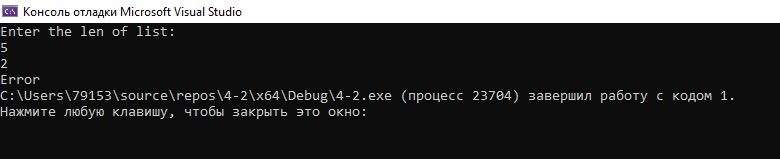
list[find\_first\_positive(list, n)] = list[find\_first\_unpositive(list, n)];

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 14, Рисунок 15).

Рисунок 14 - Результаты выполнения программы c ручным вводом чисел

Рисунок 15 - Ошибка, выведенная программа при неверном вводе значений

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