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 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  – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |
| 4 | 1. Найти сумму четных элементов. 2. Подсчитать количество элементов массива, значения которых состоят из двух цифр. 3. Заменить последний отрицательный элемент массива на модуль первого элемента массива. |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схема функций is\_positive (Рисунок 2), summ\_of\_even (Рисунок 3), count\_of\_two\_char (Рисунок 4), replace\_list (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) представлены ниже.

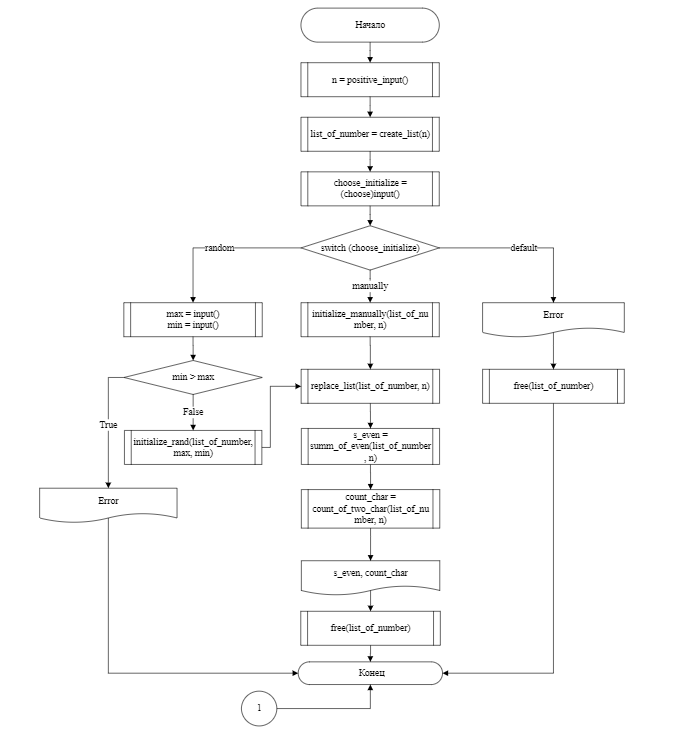


Рисунок 1 ­ Блок-схема основного алгоритма

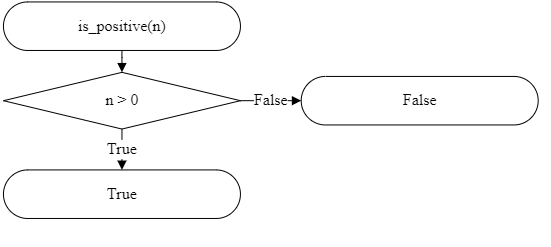


Рисунок  – Блок-схемы используемой функции is\_positive

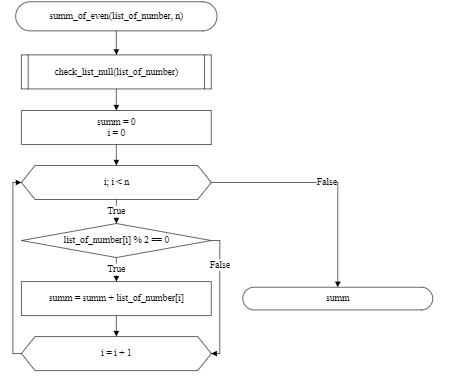


Рисунок  –  Блок-схема используемой функции summ\_of\_even

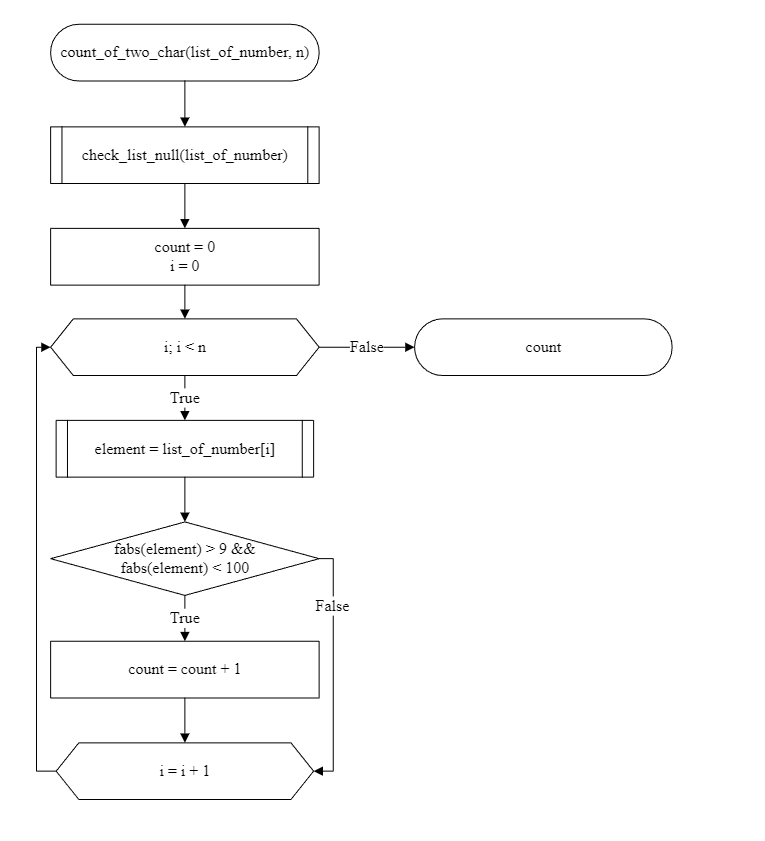


Рисунок 4 - Блок-схема используемой функции count\_of\_two\_char

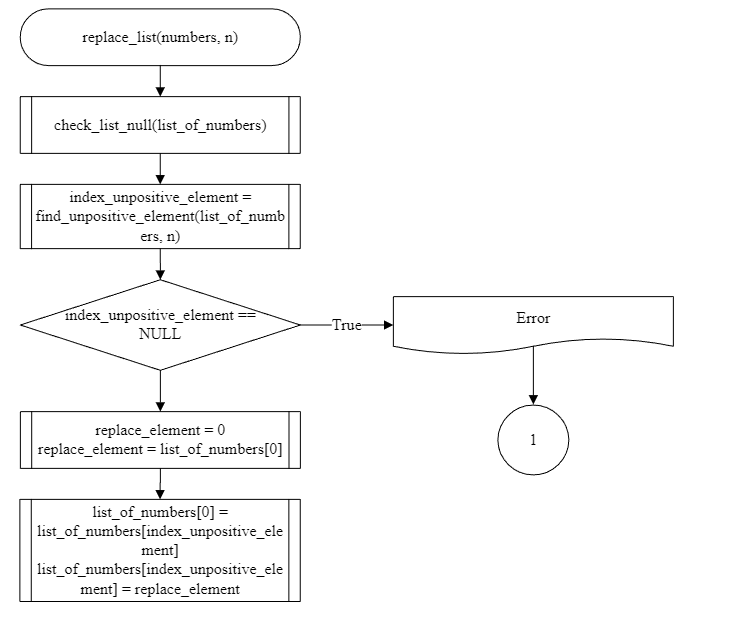


Рисунок 5 - Блок-схема используемой функции replace\_list

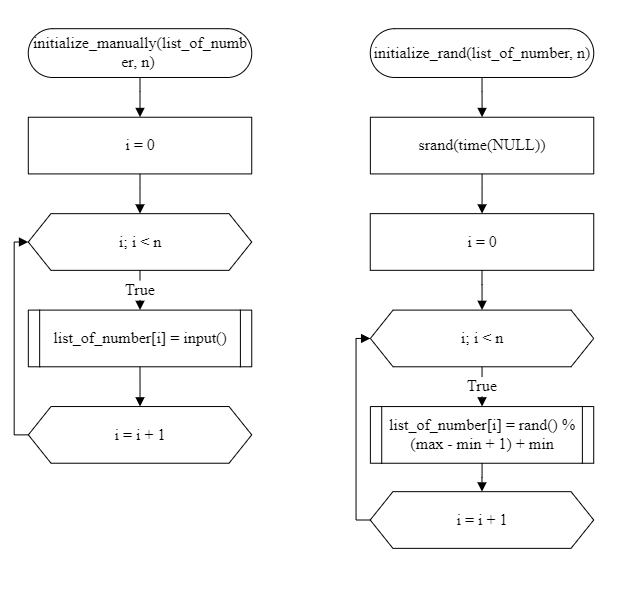
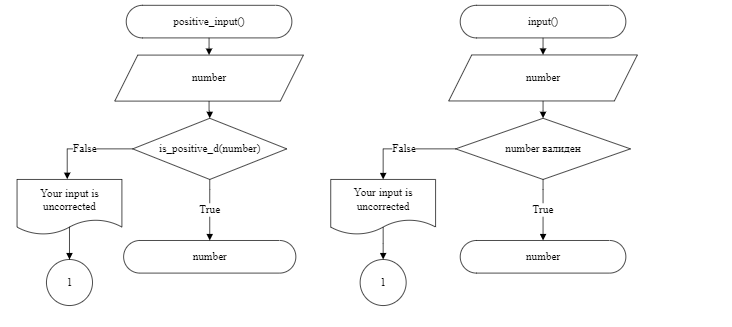


Рисунок 6 - Блок-схема используемых функий initialize\_manually и initialize\_rand

Рисунок 7 - используемые функции ввода positive\_input и input

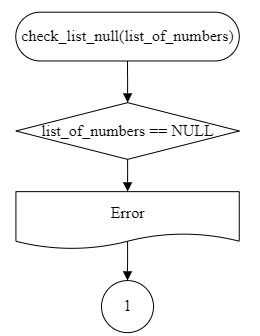
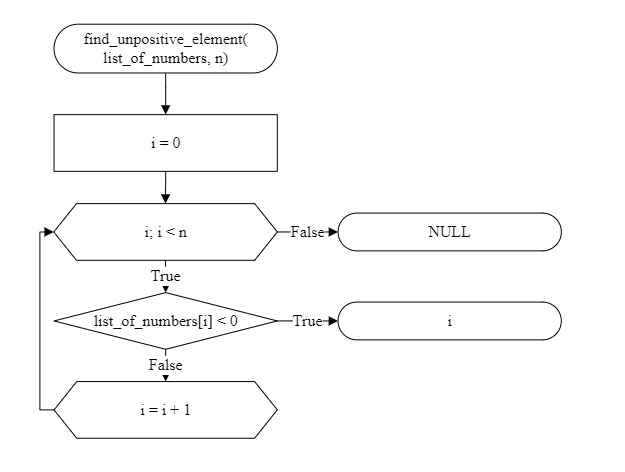


Рисунок 8 - Блок-схема используемой функции check\_list\_null

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

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#include <stdbool.h>

#include <time.h>

/\*\*

@brief choose перечисляет констаты для того, чтобы пользователь выбрал, как заполнять список

@param manually вручную

@param random случайные числа

\*/

typedef enum {

manually = 1,   
random

}choose;

/\*\*

@brief is\_positive проверяет, является ли число положительным

@param n проверяемое число

@return true, если число положительное и false, если отрицательное

\*/

bool is\_positive(const int n);

/\*\*

@brief summ\_of\_number считает сумму четных элементов массива

@param list\_of\_number массив чисел

@param n размер передаваемого массива

@return сумму целый чисел массива

*\*/*

*int summ\_of\_even(const int* list\_of\_number, const size\_t n);

/\*\*

@brief summ\_of\_two\_char считает количество элементов массива, состоящих из двух цифр

@param list\_of\_number массив целых чисел

@param n размер передаваемого массива

@return количество чисел, состоящих из двух цифр

*\*/*

*int count\_of\_two\_char(const int* list\_of\_number, const size\_t n);

/\*\*

@brief replace\_list заменяет последний отрицательный элемент массива на модуль первого элемента массива

@param list\_of\_number массив целых чисел

@param n размер передаваемого массива

*\*/*

*void replace\_list(int* list\_of\_number, const size\_t n);

/\*\*

@brief initialize\_manually заполняет массив целыми числами, которые ввёл пользователь

@param list\_of\_number массив

@param n длина списка

*\*/*

*void initialize\_manually(int* list\_of\_number,const int n);

/\*\*

@brief initialize\_rand заполняет массив случайными целыми числами

@param list\_of\_number массив целых чисел

@param n длина списка

@param max максимальное число списка

@param min минимальное число списка

*\*/*

*void initialize\_rand(int* list\_of\_number,const size\_t n, const int max, const int min);

/\*\*

@brief input считывает значение, вводимое пользователем

@return возвращает значение, вводимое пользователем

\*/

int input(void);

/\*\*

@brief positive\_input считывает число, введённое пользователем и проверяет положительно ли оно

@return возвращает число, введённое пользователем

\*/

size\_t positive\_input(void);

/\*\*

@brief create\_list создаёт массив

@param n длина массива

@return возвращает указатель на первый элемент массива

*\*/*

*int* create\_list(const int n);

/\*\*

@brief check\_list\_not\_null проверяет, выделилась ли память для массива

@param list\_numbers массив

@return возвращает true, если память для массива не выделилась и false, если выделилась

*\*/*

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

/\*\*

@brief find\_unpositive\_element ищет первый отрицательный элемент в массиве

@param list\_of\_numbers массив целых чисел

@param n размер передаваемого массива

@return возвращает индекс первого отрицательного элемента, если его нет, возвращает NULL

*\*/*

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

/\*\*

@brief main выводит результат всех действий пользователю на экран

@return возвращает 0 в случае успеха

\*/

int main(void) {

puts("Enter the n:");  
  
const size\_t n = positive\_input();  
  
int\* list\_of\_number = create\_list(n);  
  
choose choose\_initialize = (choose)input();  
  
switch (choose\_initialize) {  
  
case manually:  
 initialize\_manually(list\_of\_number, n);  
 break;  
case random:  
 puts("Enter the max element of list");  
 const int max = input();  
 puts("Enter the min element of list");  
 const int min = input();  
 if (min > max)  
 {  
 puts("Error");  
 return 1;  
 }  
 initialize\_rand(list\_of\_number, n, max, min);  
 break;  
default:  
 puts("Error");  
 free(list\_of\_number);  
 return 1;  
}  
  
replace\_list(list\_of\_number, n);  
  
printf("summ of even numbers = %d\nThe count of numbers consisting of two digits = %d\n", summ\_of\_even(list\_of\_number, n), count\_of\_two\_char(list\_of\_number, n));  
  
free(list\_of\_number);  
  
return 0;

}

int summ\_of\_even(const int\* list\_of\_number, const size\_t n) { check\_list\_null(list\_of\_number);

int summ = 0;  
  
for (size\_t i = 0; i < n; i++)  
{  
 if (list\_of\_number[i] % 2 == 0)  
 {  
 summ += list\_of\_number[i];  
 }  
}  
  
return summ;

}

int count\_of\_two\_char(const int\* list\_of\_number, const size\_t n) {

check\_list\_null(list\_of\_number);  
  
int count = 0;  
  
for (size\_t i = 0; i < n; i++) {  
  
 int element = list\_of\_number[i];  
  
 if (fabs(element) > 9 && fabs(element) < 100) {  
  
 count += 1;  
 }  
}  
  
return count;

}

void replace\_list(int\* list\_of\_numbers, const size\_t n) { check\_list\_null(list\_of\_numbers);

size\_t index\_unpositive\_element = find\_unpositive\_element(list\_of\_numbers, n);  
  
if(index\_unpositive\_element == NULL){  
 puts("Error");  
 exit(EXIT\_FAILURE);  
}  
  
int replace\_element = 0;  
  
replace\_element = list\_of\_numbers[0];  
  
list\_of\_numbers[0] = list\_of\_numbers[index\_unpositive\_element];  
  
list\_of\_numbers[index\_unpositive\_element] = replace\_element;

}

void initialize\_manually(int\* list\_of\_number,const int n) {

for (size\_t i = 0; i < n; i++)  
  
{  
 puts("\nEnter the your number:");  
 list\_of\_number[i] = input();  
}

}

void initialize\_rand(int\* list\_of\_number, const size\_t n, const int max, const int min) {

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

}

int input(void) { int number = 0;

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

}

bool is\_positive(const int n) {

return n > 0;

}

size\_t positive\_input(void) { int number = 0;

if (scanf\_s("%zu", &number) != 1 || !is\_positive(number))  
{  
 puts("Your input is uncorrected");  
  
 exit(EXIT\_FAILURE);  
}  
  
return (size\_t)number;

}

int\* create\_list(const int n) { int\* list\_of\_number = (int\*)malloc(sizeof(int) \* n);

check\_list\_null(list\_of\_number);  
  
return list\_of\_number;

}

void check\_list\_null(const int\* list\_numbers) { if (list\_numbers == NULL) { puts("Error");

exit(EXIT\_FAILURE);  
}

}

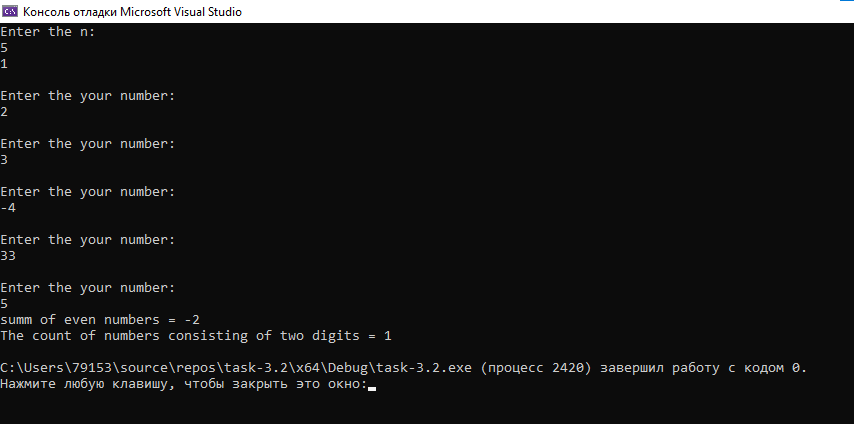
size\_t find\_unpositive\_element(const int\* list\_of\_numbers, const size\_t n) { for (size\_t i = 0; i < n; i++) { if (list\_of\_numbers[i] < 0) { return i; } }

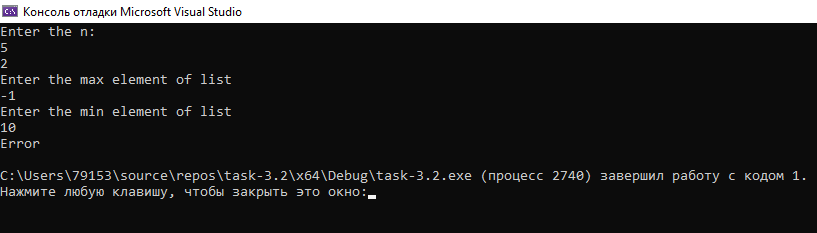
return NULL;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 10, Рисунок 11).

Рисунок 10 - результаты выполнения программы c ручным вводом чисел

Рисунок 11 - результаты выполнения программы с заполнением списка случайными числами

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

Ниже представлено доказательство того, что задание 4-1 было принято (Рисунок 12)



Рисунок 12 – Approve задания 4-1