Министерство образования и науки РФ
Государственное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

РАБОТА С МАССИВАМИ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С

Лабораторная работа № 10

по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)

Проверил: к. ф.-м. н., доцент _____ Сулимова В.В.
(подпись)

Тула 2011

Цель работы

Цель работы заключается в том, чтобы изучить структурированных типов данных — одномерный массив и двумерный массив. Также требуется написать программу, обрабатывающую массивы.

Задание

1. Задана последовательность из N вещественных чисел. Определить, сколько чисел меньше K, равно K и больше K.

2. Умножить j-й столбец в матрице на k.

Теоретическая справка

1. ЗАДАЧА ОЦЕНКИ ЭЛЕМЕНТОВ ПОСЛЕДОВАТЕЛЬНОСТИ

1.1. Схема алгоритма

На рисунке 1.1 представлена схема получения параметров расчета, заполнения массива различными способами, оценки его элементов и вывода результатов оценки.

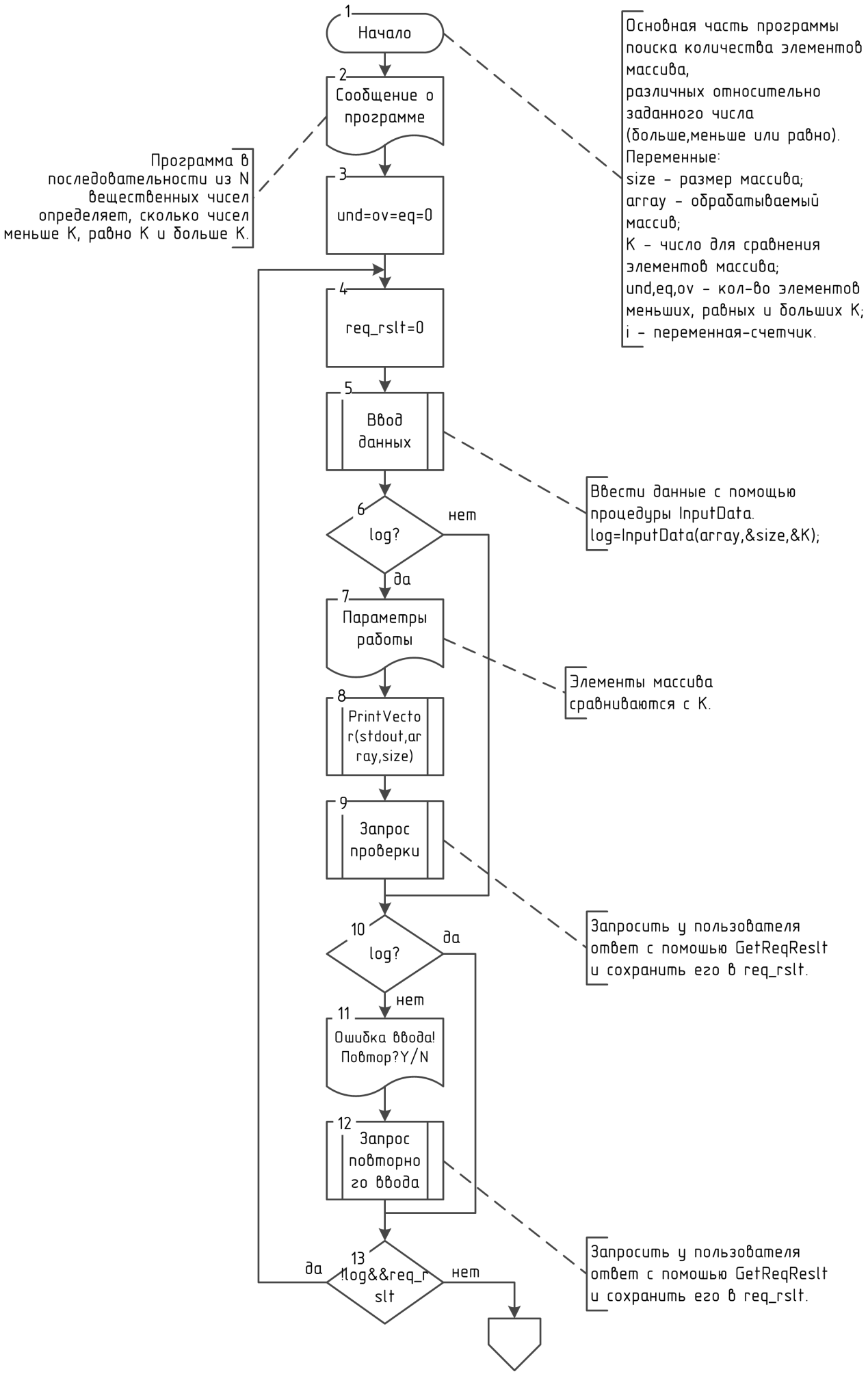

Рисунок 1.1 — Схема обобщенного алгоритма оценки элементов массива(начало)

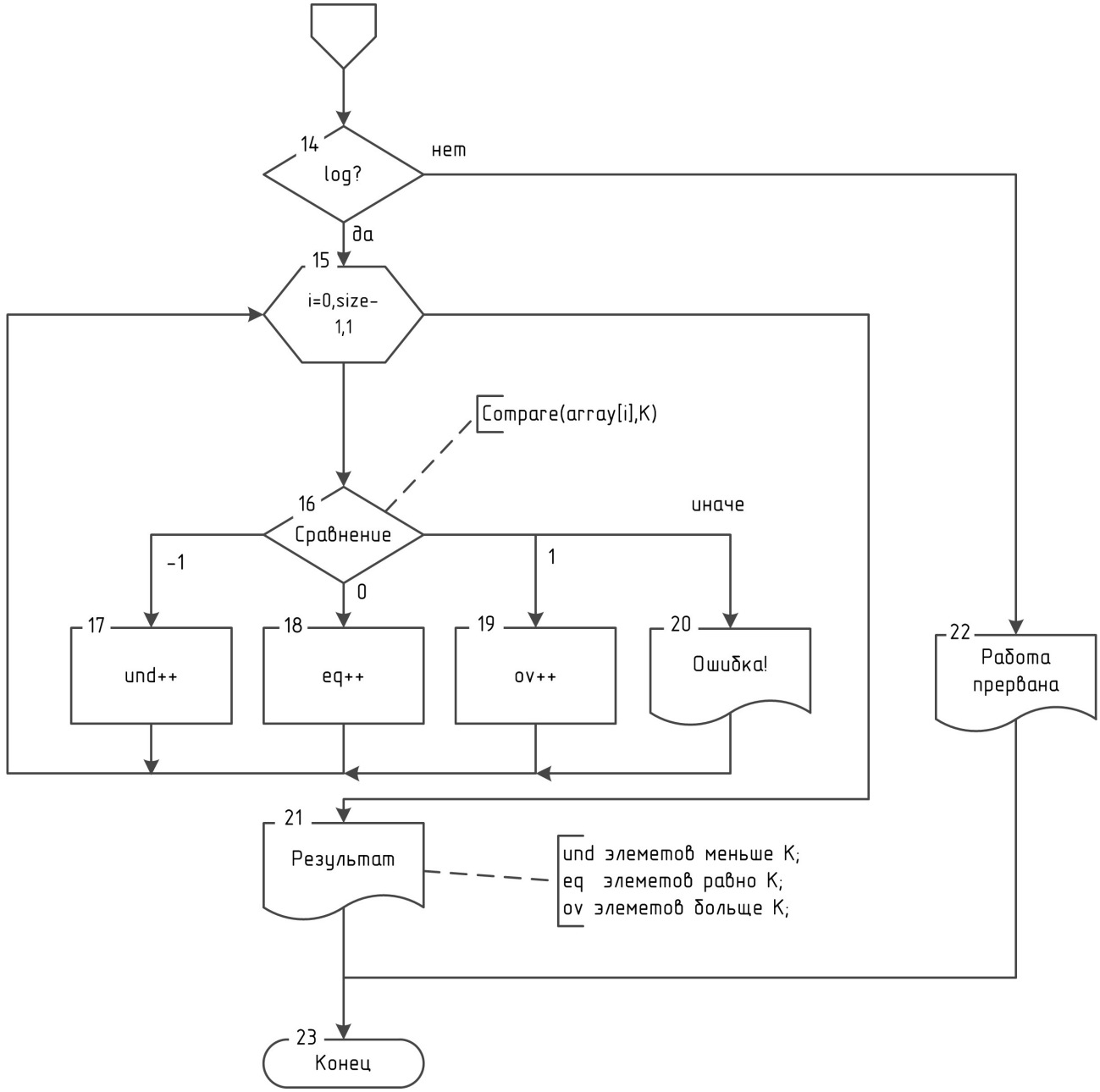

Рисунок 1.2 – Схема обобщенного алгоритма оценки элементов массива(продолжение)

Схему алгоритма получения ответа от пользователя на поставленный вопрос, можно увидеть на рисунке 1.3.

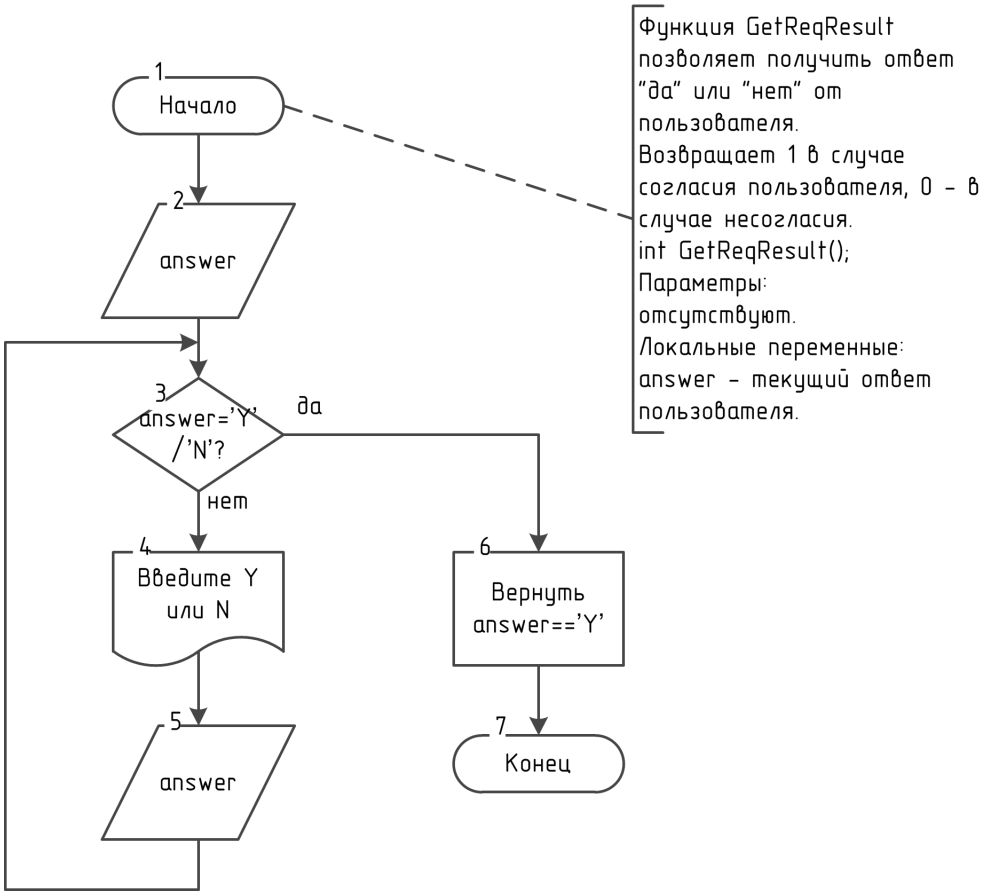

Рисунок 1.3 - Схема алгоритма получения ответа от пользователя

На рисунке 1.4 представлена схема алгоритма предоставления пользователю различных вариантов ответа.

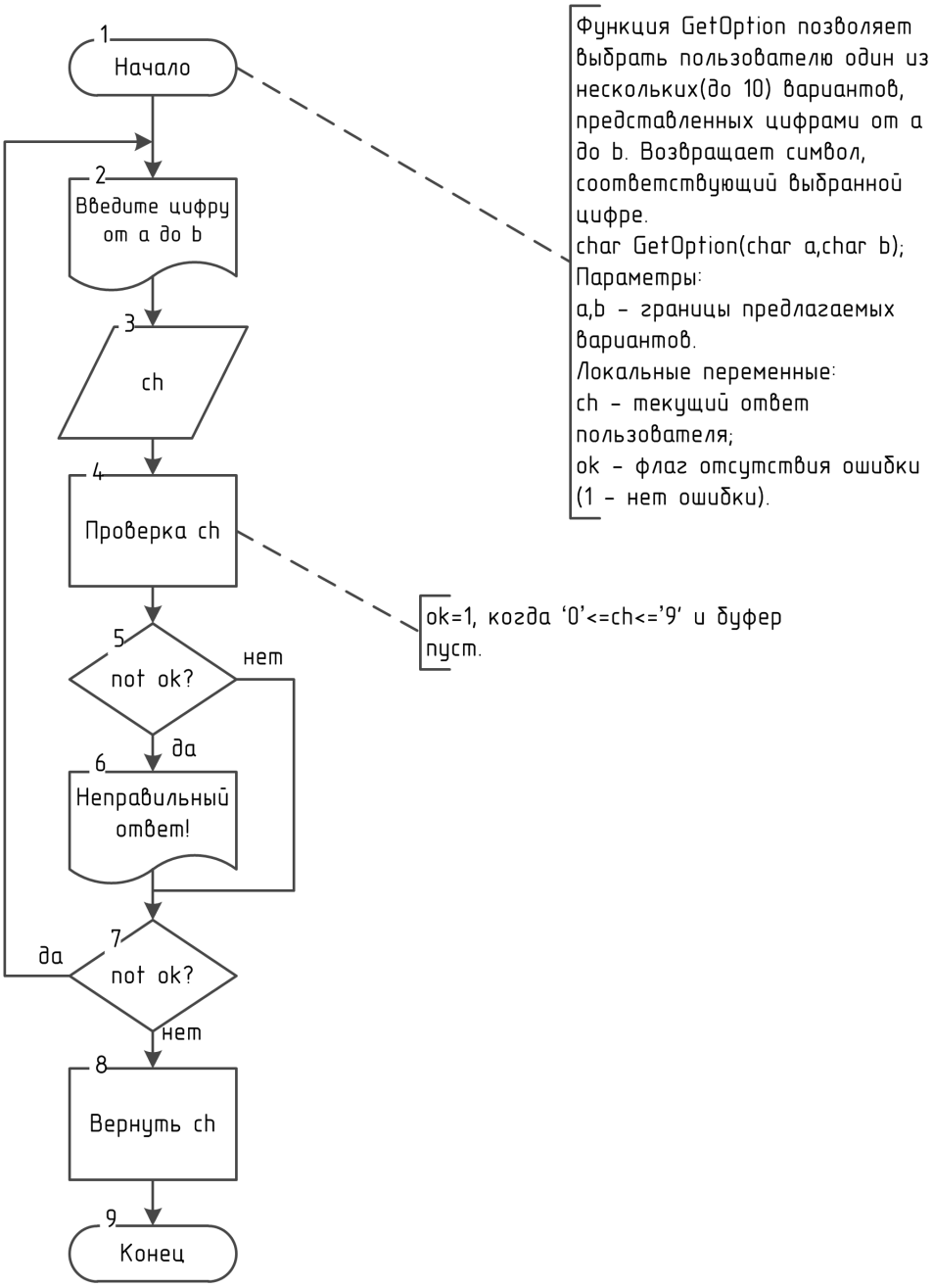

Рисунок 1.4 - Схема алгоритма предоставления пользователю различных вариантов ответа

На рисунке 1.5 можно увидеть схему алгоритма сравнения 2-х элементов.

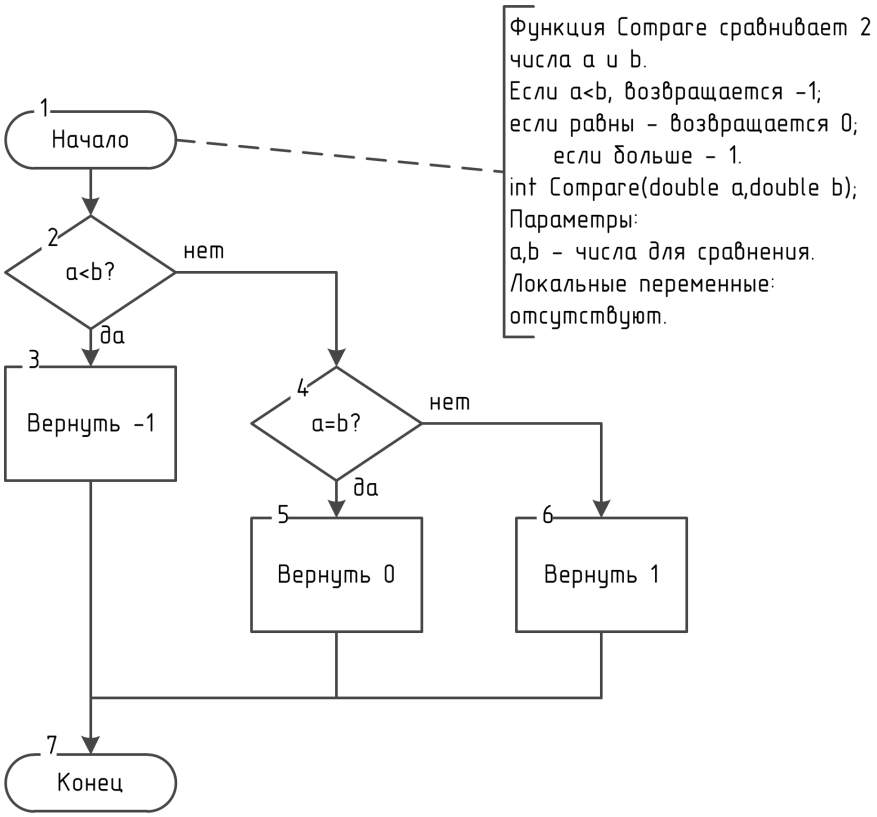

Рисунок 1.5 - Схема алгоритма сравнения 2-х элементов.

На рисунке 1.6 можно увидеть общую схему ввода данных для программы оценки элементов одномерного массива.

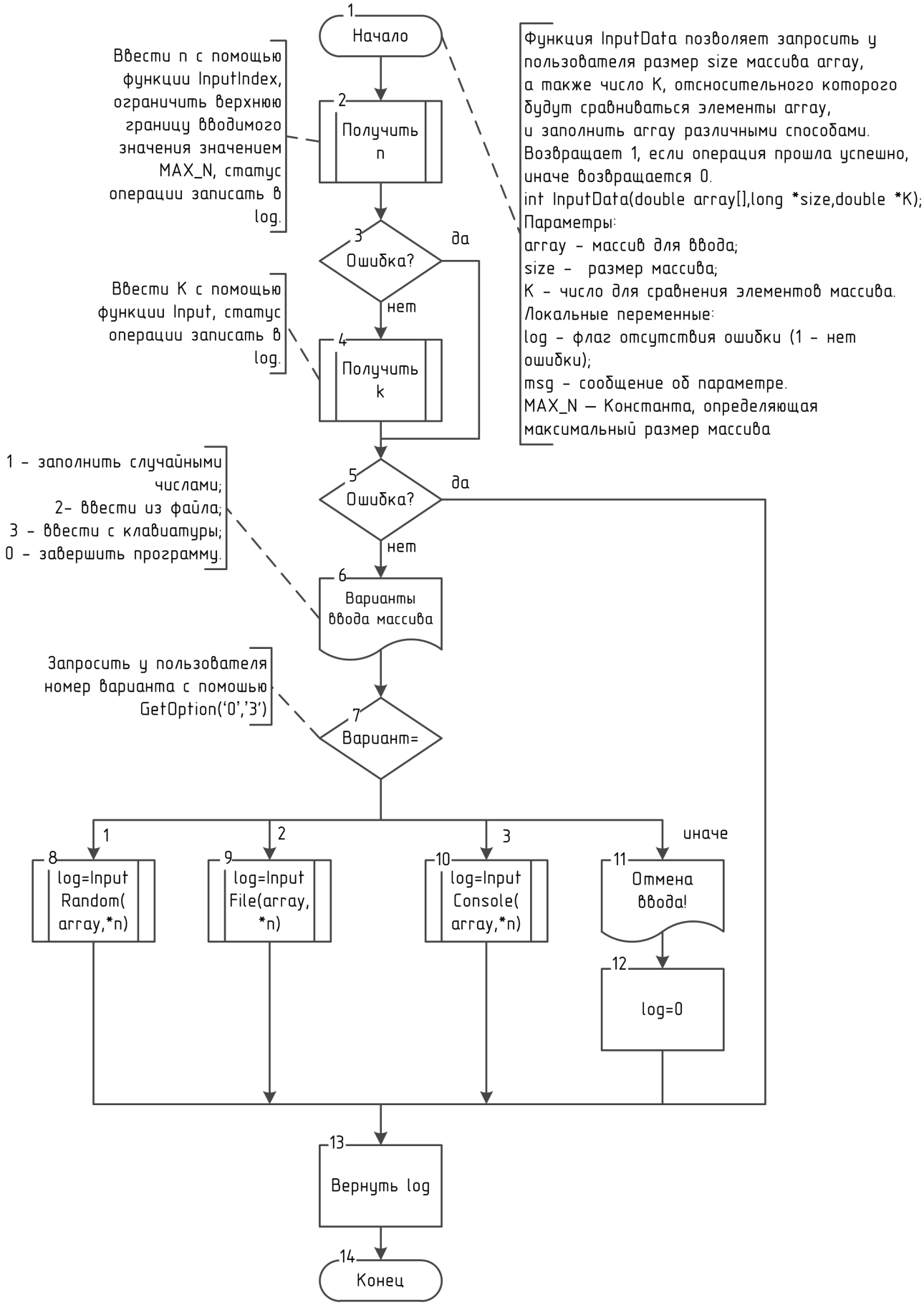

Рисунок 1.6 — Схема общего алгоритма ввода данных для программы оценки элементов одномерного массива

На рисунке 1.7 представлена схема заполнения массива случайными числами.

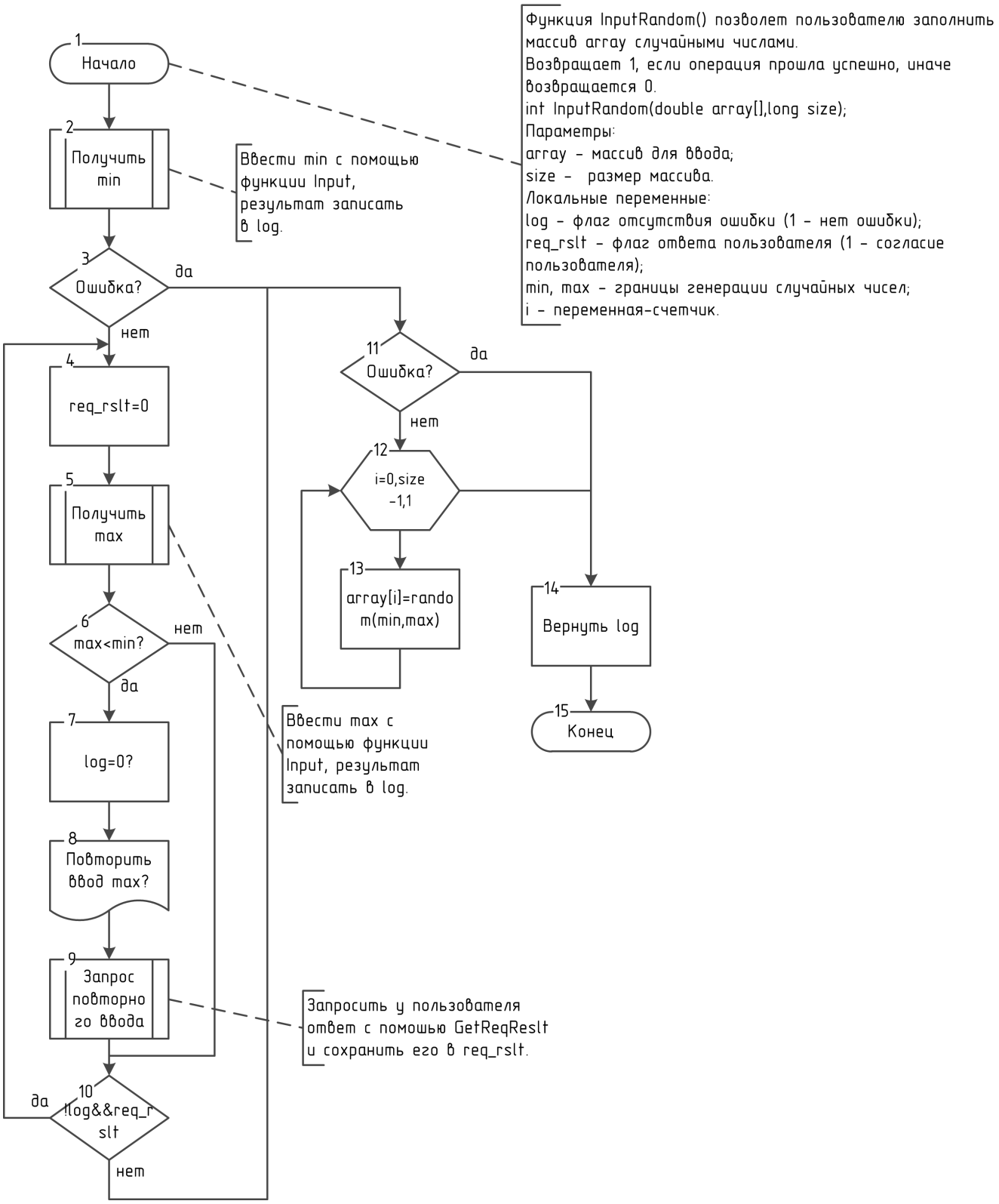

Рисунок 1.7 - Схема алгоритма заполнения одномерного массива случайными числами

На рисунке 1.8 представлена схема ввода массива с клавиатуры.

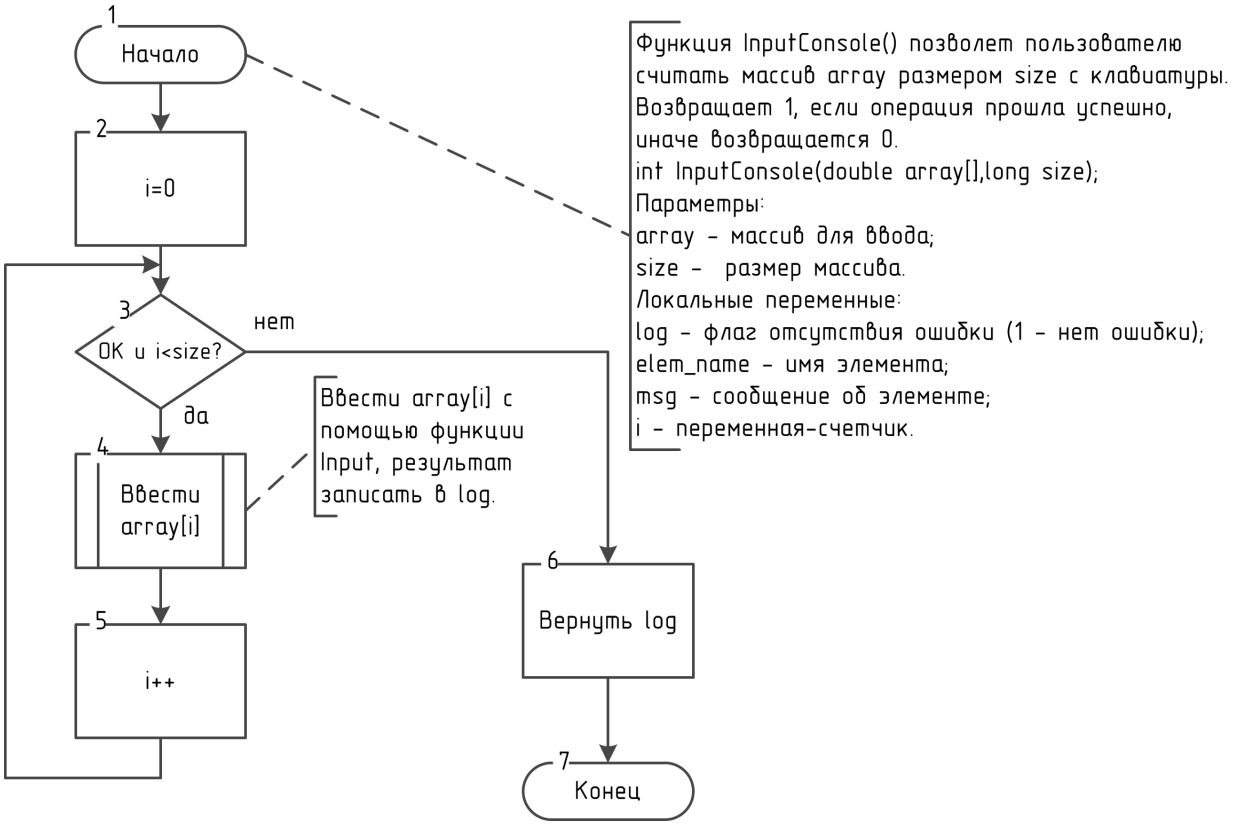

Рисунок 1.8 - Схема алгоритма ввода одномерного массива с клавиатуры

На рисунке 1.9 представлена блок-схема алгоритма заполнения массива данными из файла.

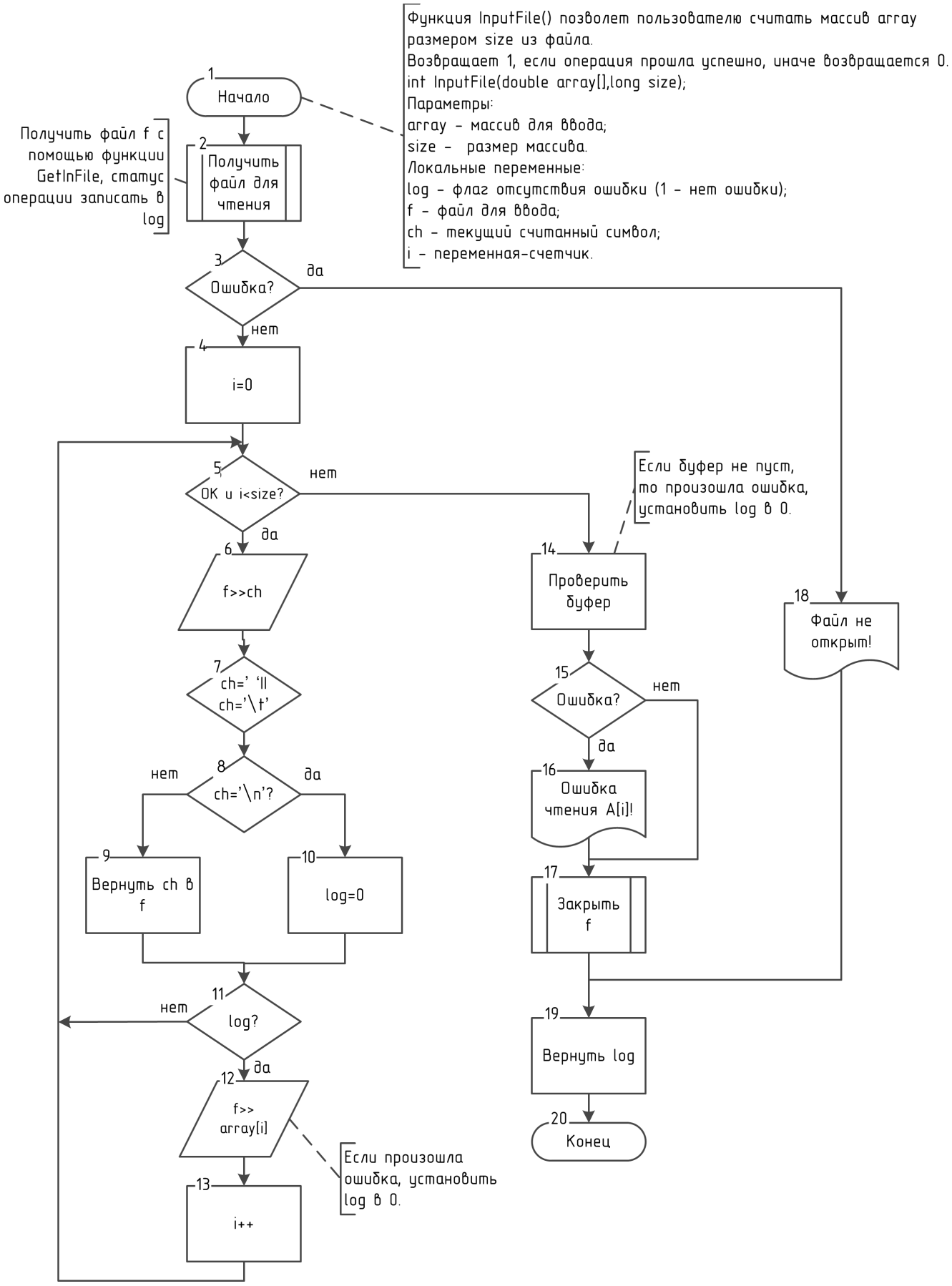

Рисунок 1.9 - Схема алгоритма ввода одномерного массива из файла

На рисунке 1.10 представлена схема алгоритма получения файла-источника, имя которого вводит пользователь.

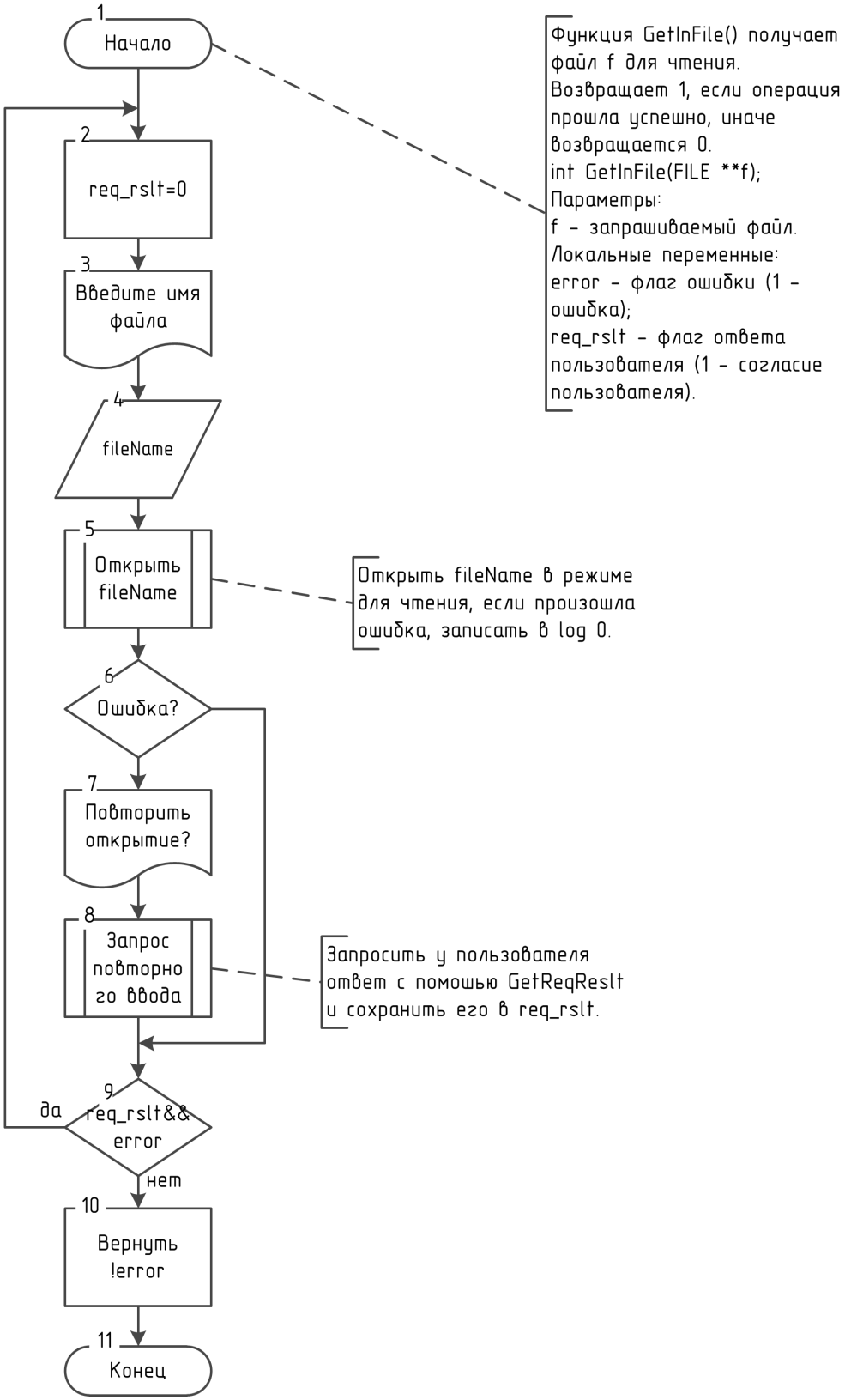

Рисунок 1.10 - Схема алгоритма получения файла-источника

На рисунке 1.11 представлена схема получения от пользователя некоторого вещественного числа.

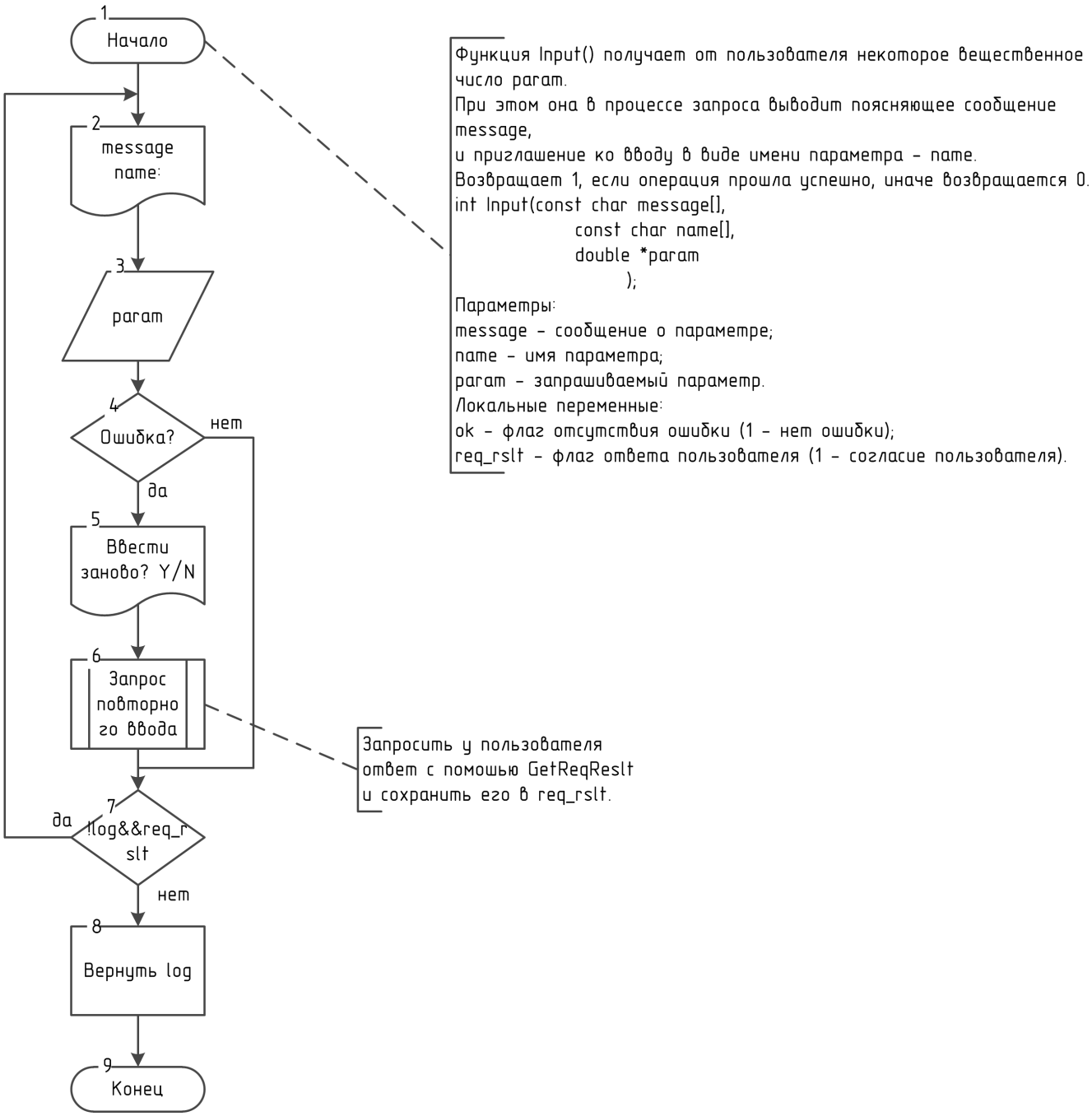

Рисунок 1.11 - Схема алгоритма получения от пользователя вещественного числа

На рисунке 1.12 можно увидеть схему алгоритма получения от пользователя произвольного индекса массива.

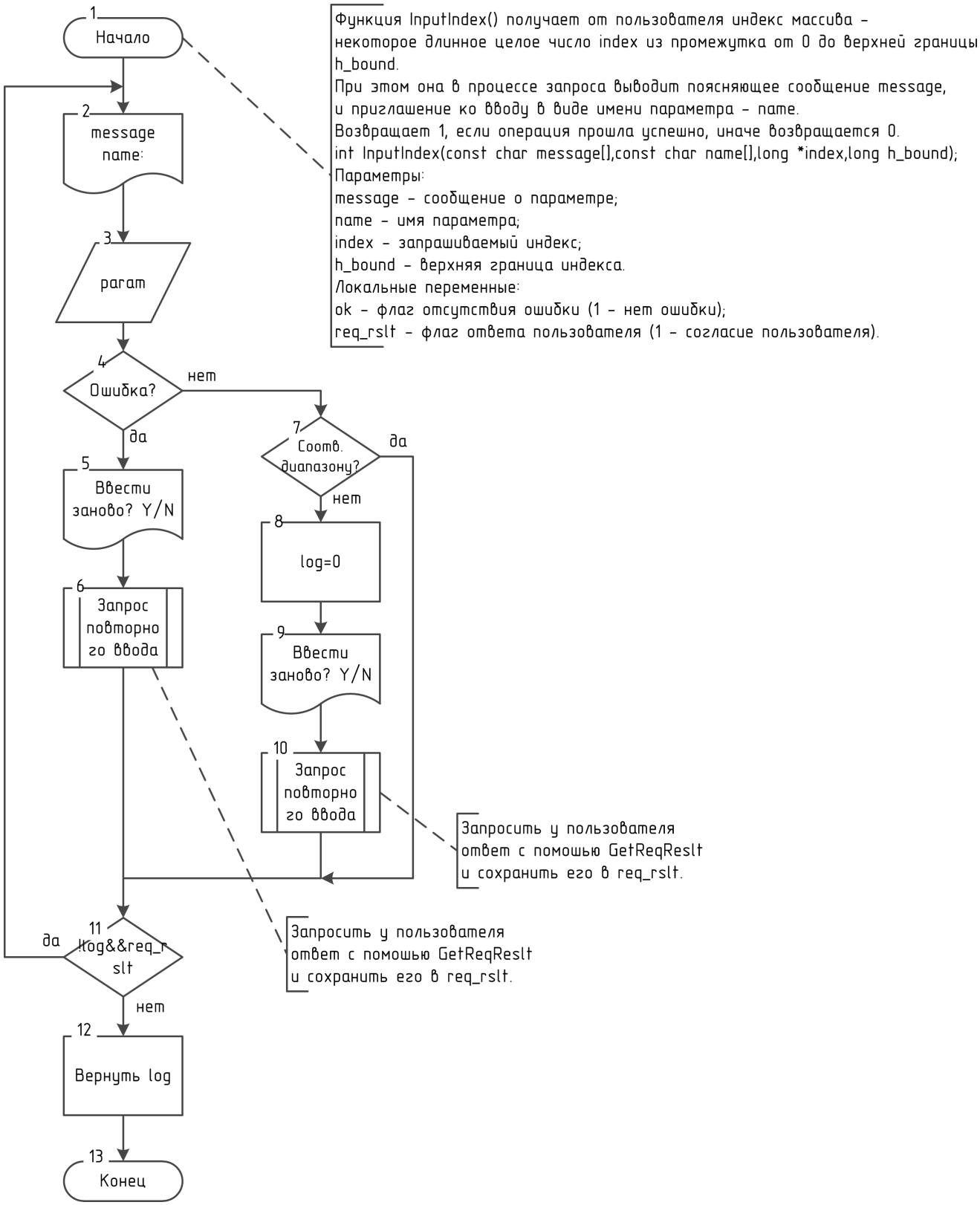

Рисунок 1.12 - Схема алгоритма получения от пользователя номера элемента массива

На рисунке 1.13 изображена схема алгоритма печати одномерного массива в файл.

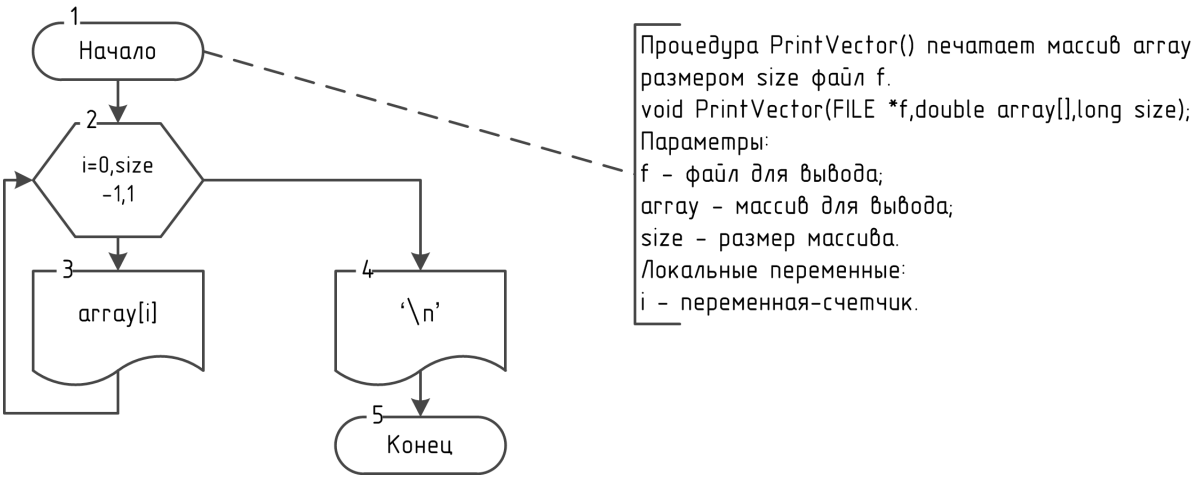

Рисунок 1.13 - Схема алгоритма печати одномерного массива в файл

1.3. Инструкция пользователю

Данная программа находит, сколько в указанной последовательности элементов меньше, равно или больше заданного числа.

Передайте программе сначала длину последовательности, затем число для сравнения. После этого можно сгенерировать последовательность с помощью случайных чисел, считать из файла или с клавиатуры. Следует учитывать, что в файле должна быть единственная строка, и в ней должно содержаться указанное ранее количество вещественных чисел, разделенных пробелами или табуляциями.

После того, как будут получены все данные, программа проанализирует последовательность и выведет требуемую информацию. Для завершения работы программы нажмите клавишу Enter.

1.4. Инструкция программисту

Для решения задачи оценки элементов вектора были предприняты следующие действия.

Объявлен макрос MAX_SIZE, который содержит максимальный размер массива.

Подключены заголовочные файлы <ctype.h> - для функции toupper() и <stdio.h> для функций ввода-вывода.

Объявлены следующие структуры данных:

Таблица 1.1 - Структуры данных, используемые в в основной части программы оценки элементов вектора

имя	**тип**	**предназначение**
size	long	размер массива;
array	double*	обрабатываемый массив;
K	double	число для сравнения элементов массива;
und,eq,ov	long	кол-во элементов меньших, равных и больших K;
i	long	переменная-счетчик.

Также программа была разбита на следующие подпрограммы:

1. Функция Compare сравнивает 2 числа a и b.

Если a<b, возвращается -1; если равны - возвращается 0; если больше – 1.

int Compare(double a,double b);

Параметры функции представлены в таблице 1.2:

Таблица 1.2 - Структуры данных, используемые в в основной части программы оценки элементов вектора

имя	**тип**	**предназначение**
a,b	double	числа для сравнения.

Локальные переменные:

отсутствуют.

2. Функция GetInFile() получает файл f для чтения.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int GetInFile(FILE **f);

Параметры функции представлены в таблице 1.3:

Таблица 1.3 - Параметры функции сравнения 2-х чисел

имя	**тип**	**предназначение**

f	FILE *	запрашиваемый файл.

Локальные переменные функции представлены в таблице 1.4:

Таблица 1.4 - Локальные переменные функции сравнения 2-х чисел

имя	**тип**	**предназначение**
error	int	флаг ошибки (1 - ошибка);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

3. Процедура PrintVector() печатает массив array размером size файл f.

void PrintVector(FILE *f,double array[],long size);

Параметры функции представлены в таблице 1.5:

Таблица 1.5 - Параметры функции печати одномерного массива

имя	**тип**	**предназначение**
f	FILE *	файл для вывода;
array	double*	массив для вывода;
size	long	размер массива.

Локальные переменные функции представлены в таблице 1.6:

Таблица 1.6 - Локальные переменные функции печати одномерного массива

имя	**тип**	**предназначение**
i	long	переменная-счетчик.

4. Функция Input() получает от пользователя некоторое вещественное число param.

При этом она в процессе запроса выводит поясняющее сообщение message, и приглашение ко вводу в виде имени параметра – name.

```
int Input(const char message[],
          const char name[],
          double *param
            );
```

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

Параметры функции представлены в таблице 1.7:

Таблица 1.7 - Параметры функции ввода вещественного числа

имя	**тип**	**предназначение**
message	char*	сообщение о параметре;
name	char*	имя параметра;
param	double*	запрашиваемый параметр.

Локальные переменные функции представлены в таблице 1.8:

Таблица 1.8 - Локальные переменные функции ввода вещественного числа

имя	тип	предназначение
ok	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

5. Функция InputIndex() получает от пользователя индекс массива - некоторое длинное целое число index из промежутка от 0 до верхней границы h_bound. При этом она в процессе запроса выводит поясняющее сообщение message, и приглашение ко вводу в виде имени параметра - name.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputIndex(const char message[],const char name[],long *index,long h_bound);

Параметры функции представлены в таблице 1.9:

Таблица 1.9 - Параметры функции ввода индекса элемента массива

имя	тип	предназначение
message	char*	сообщение о параметре;
name	char*	имя параметра;
index	long*	запрашиваемый индекс;
h_bound	long*	верхняя граница индекса.

Локальные переменные функции представлены в таблице 1.10:

Таблица 1.10 - Локальные переменные функции ввода индекса элемента массива

имя	тип	предназначение
ok	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

6. Функция InputConsole() позволет пользователю считать массив array размером size с клавиатуры.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputConsole(double array[],long size);

Параметры функции представлены в таблице 1.11:

Таблица 1.11 - Параметры функции ввода одномерного массива с клавиатуры

имя	тип	предназначение
array	double*	массив для ввода;
size	long	размер массива.

Локальные переменные функции представлены в таблице 1.12:

Таблица 1.12 - Локальные переменные функции ввода одномерного массива с клавиатуры

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
elem_name	char[20]	имя элемента;
msg	char[150]	сообщение об элементе;
i	long	переменная-счетчик.

7. Функция InputFile() позволет пользователю считать массив array размером size из файла.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputFile(double array[],long size);

Параметры функции представлены в таблице 1.13:

Таблица 1.13 - Параметры функции ввода одномерного массива из файла

имя	тип	предназначение
array	double*	массив для ввода;
size	long	размер массива.

Локальные переменные функции представлены в таблице 1.14:

Таблица 1.14 - Локальные переменные функции ввода одномерного массива из файла

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
f	FILE *	файл для ввода;
ch	char	текущий считанный символ;
i	long	переменная-счетчик.

8. Функция InputRandom() позволет пользователю заполнить массив array случайными числами.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputRandom(double array[],long size);

Параметры функции представлены в таблице 1.15:

Таблица 1.15 - Параметры функции заполнения массива случайными числами

имя	тип	предназначение
array	double*	массив для ввода;
size	long	размер массива.

Локальные переменные функции представлены в таблице 1.16:

Таблица 1.16 - Локальные переменные функции заполнения массива случайными числами

имя	тип	предназначение

log	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя);
min, max	double	границы генерации случайных чисел;
i	long	переменная-счетчик.

9. Функция InputData позволяет запросить у пользователя размер size массива array, а также число K, отсносительного которого будут сравниваться элементы array, и заполнить array различными способами.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputData(double array[],long *size,double *K);

Параметры функции представлены в таблице 1.17:

Таблица 1.17 - Параметры функции ввода параметров программы оценки элементов вектора

имя	**тип**	**предназначение**
array	double*	массив для ввода;
size	long*	размер массива.
K	double*	число для сравнения элементов массива.

Локальные переменные функции представлены в таблице 1.18:

Таблица 1.18 - Локальные переменные функции ввода параметров программы оценки элементов вектора

имя	**тип**	**предназначение**
log	int	флаг отсутствия ошибки (1 - нет ошибки);
msg	char[255]	сообщение об параметре.

10.Функция clearline используется для очистки строки файла.

Заголовок функции:

int clearline(FILE *f);

Функция считывает до конца файла или строки файла f символы в цикле while и возвращает их количество.

Параметры функции представлены в таблице 1.19, локальные переменные — в таблице 1.20.

Таблица 1.19 - Параметры функции получения сочетания

имя	**тип**	**предназначение**
f	FILE*	Файл, в котором очищается строка.

Таблица 1.20 - Локальные переменные функции получения сочетания

имя	тип	предназначение
count	long	Счетчик считанных символов.

11. Функция GetReqResult используется для получения ответов пользователя на запросы программы.

Заголовок функции:

int GetReqResult();

Функция запрашивает у пользователя символы 'y','Y','n' или 'N' до тех пор, пока он не введет какой-либо из них. Соответственно возвращается либо символ 'Y', либо 'N'.

Локальные переменные функции представлены в таблице 1.21.

Таблица 1.21 - Локальные переменные функции получения ответа от пользователя

имя	тип	предназначение
answer	char	Ответ пользователя.

12. Функция GetOption позволяет выбирать пользователю из нескольких (до 10) вариантов ответа.

Заголовок функции:

int GetOption(char a,char b);

Функция запрашивает у пользователя символы из промежутка от a до b до тех пор, пока он не введет какой-либо из них. Соответственно возвращается значение введенного символа.

Параметры функции представлены в таблице 1.22, локальные переменные — в таблице 1.23.

Таблица 1.22 - Параметры функции предоставления различных вариантов ответа

имя	тип	предназначение
a,b	char	Различные варианты представлены цифрами от a до b.

Таблица 1.23 - Локальные переменные функции предоставления различных вариантов ответа

имя	тип	предназначение
ch	char	Ответ пользователя.
ok	int	Флаг — равен 1, если ответ пользователя допустим, иначе равен 0.

1.5. Текст программы

Ниже представлен текст программы на языке ANSI C, компилируемый Borland C 3.1 и GCC 4.5.2 ,для оценки элементов вектора относительного заданного числа.

```
#include <stdio.h>
#include <ctype.h>
#include <time.h>
#define MAX_N 5L

/************************************/
/*
Функция Compare сравнивает 2 числа a и b.
Если a<b, возвращается -1; если равны - возвращается 0;
    если больше - 1.
Параметры:
a,b - числа для сравнения.
Локальные переменные:
отсутствуют.
*/
int Compare(double a,double b);


/*
Функция clearline очищает строку в файле f. Возвращает количество
непробельных символов в считанной строке.
Параметры:
f - Файл для очистки строки.
Локальные переменные:
count - количество непробельных символов;
ch - текущий считанный символ.
*/
int clearline(FILE *f);


/*
Функция GetReqResult позволяет получить ответ "да" или "нет" от пользователя.
Возвращает 1 в случае согласия пользователя, 0 - в случае несогласия.
Параметры:
отсутствуют.
Локальные переменные:
answer - текущий ответ пользователя.
*/
```

```
int GetReqResult();


/*
Функция GetOption позволяет выбрать пользователю один из нескольких(до 10)
вариантов,
представленных цифрами от a до b. Возвращает символ, соответствующий выбранной
цифре.
Параметры:
a,b - границы предлагаемых вариантов.
Локальные переменные:
ch - текущий ответ пользователя;
ok - флаг отсутствия ошибки (1 - нет ошибки).
*/
char GetOption(char a,char b);


/*
Функция fexists() проверяет существование файла с именем fname.
Возвращает 1, если такой файл существует, или 0, если нет.
Параметры:
fname - имя файла для проверки.
Локальные переменные:
f - временный файл для проверки.
*/
int fexists(char  *fname);


/*
Функция GetInFile() получает файл f для чтения.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
f - запрашиваемый файл.
Локальные переменные:
error - флаг ошибки (1 - ошибка);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int GetInFile(FILE **f);


/*
Функция GetOutFile() получает файл f для записи.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
```

```
f - запрашиваемый файл.
Локальные переменные:
error - флаг ошибки (1 - ошибка);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int GetOutFile(FILE **f);


/*
Процедура PrintVector() печатает массив array размером size файл f.
Параметры:
f - файл для вывода;
array - массив для вывода;
size - размер массива.
Локальные переменные:
i - переменная-счетчик.
*/
void PrintVector(FILE *f,double array[],long size);


/*
Основная часть программы поиска количества элементов массива,
различных относительно заданного числа (больше,меньше или равно).
Переменные:
size - размер массива;
array - обрабатываемый массив;
K - число для сравнения элементов массива;
und,eq,ov - кол-во элементов меньших, равных и больших K;
i - переменная-счетчик.
*/
int main();


/*
Функция Input() получает от пользователя некоторое вещественное число param.
При этом она в процессе запроса выводит поясняющее сообщение message,
и приглашение ко вводу в виде имени параметра - name.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
message - сообщение о параметре;
name - имя параметра;
param - запрашиваемый параметр.
Локальные переменные:
ok - флаг отсутствия ошибки (1 - нет ошибки);
```

```
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int Input(const char message[],
              const char name[],
              double *param
                    );


/*
Функция InputIndex() получает от пользователя индекс массива -
некоторое длинное целое число index из промежутка от 0 до верхней границы h_bound.
При этом она в процессе запроса выводит поясняющее сообщение message,
и приглашение ко вводу в виде имени параметра - name.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
message - сообщение о параметре;
name - имя параметра;
index - запрашиваемый индекс;
h_bound - верхняя граница индекса.
Локальные переменные:
ok - флаг отсутствия ошибки (1 - нет ошибки);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int InputIndex(const char message[],const char name[],long *index,long h_bound);


/*
Функция InputConsole() позволет пользователю считать массив array размером size с
клавиатуры.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
size -  размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
elem_name - имя элемента;
msg - сообщение об элементе;
i - переменная-счетчик.
*/
int InputConsole(double array[],long size);


/*
```

```
Функция InputFile() позволет пользователю считать массив array размером size из
файла.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
size -  размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
f - файл для ввода;
ch - текущий считанный символ;
i - переменная-счетчик.
*/
int InputFile(double array[],long size);


/*
Функция InputRandom() позволет пользователю заполнить массив array случайными
числами.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
size -  размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
req_rslt - флаг ответа пользователя (1 - согласие пользователя);
min, max - границы генерации случайных чисел;
i - переменная-счетчик.
*/
int InputRandom(double array[],long size);


/*
Функция InputData позволяет запросить у пользователя размер size массива array,
а также число K, отсносительного которого будут сравниваться элементы array,
и заполнить array различными способами.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
size -  размер массива;
K - число для сравнения элементов массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
msg - сообщение об параметре.
*/
```

```
int InputData(double array[],long *size,double *K);

/************************************/
int Compare(double a,double b){
    if (a<b)
        return -1;
    else if (a==b)
        return 0;
    else
        return 1;
}
int clearline(FILE *f){
  int count=0;char ch;
      while (!feof(f)&&((ch=getc(f))!='\n'))
        if(ch!=' '&&ch!='\t')
            count++;
      return count;
}
int GetReqResult(){
  char answer;
    answer=getchar();
    clearline(stdin);
    while (toupper(answer)!='Y'
           &&toupper(answer)!='N'){
        printf("Неправильный ответ! Допустимо:\n"\
                                "Y - да; N - нет.\n");
        answer=getchar();
        clearline(stdin);
    }

    return toupper(answer)=='Y';
}
char GetOption(char a,char b){
  char ch; int ok;
    do{
        printf("Введите цифру от %c до %c.\n",a,b);
            ch=getchar();
            ok=(a<=ch)&&(ch<=b);
            ok=!clearline(stdin)&&ok;
            if (!ok)
            printf("Неправильный ответ!\n");
    }while (!ok);
      return ch;
}
```

```
int fexists(char  *fname){
  FILE *f;
    f=fopen(fname,"r");
    if (f!=NULL)
        fclose(f);
    return f!=NULL;
}
int GetInFile(FILE **f){
  int error,req_rslt;
  char fileName[255]="\0";
    do{
        req_rslt=0;
        printf("Введите имя файла-источника.\n");
        //считать строку (gets() deprecated!)
        printf("%s: ","Файл");scanf("%255[^\n]",fileName);
        clearline(stdin);
        *f=fopen(fileName,"r");
        error=*f==NULL;
        if (error) {
            printf("Неправильное имя файла! \n");
            printf("Хотите повторить ввод? (Y/N)\n");
            req_rslt=GetReqResult();
        };
    } while (req_rslt&&error);
    return !error;
  };
int GetOutFile(FILE * * f){
  char fileName [255]="\0";
  int error=0,req_rslt;
    do{
        req_rslt=0;
        printf("Введите имя файла-результата.\n");
        //считать строку (gets() deprecated!)
        printf("%s: ","Файл");scanf("%255[^\n]",fileName);
        clearline(stdin);
        if (fexists(fileName)){
            error=1;
            printf("ВНИМАНИЕ! Указанное имя файла занято!\n");
            printf("ПЕРЕЗАПИСАТЬ ФАЙЛ? (Y/N)\n");
            error=!GetReqResult();
        }
        if (!error){
            *f=fopen(fileName,"w");
            error=*f==NULL;
```

```
        }
        if (error) {
            printf("Неправильное имя файла! \n");
            printf("Хотите повторить ввод? (Y/N)\n");
            req_rslt=GetReqResult();
        };
    } while (req_rslt&&error);
    return !error;
};
void PrintVector(FILE *f,double array[],long size){
  int i;
    for (i=0;i<size;i++){

        fprintf(f,"%10.4lf",array[i]);
    }
    fprintf(f,"\n");

}
int main(){
  long size;
  double array[MAX_N];
  double K;
  long eq,ov,und,i;
  int log,req_rslt;
    und=ov=eq=0;
    printf("Программа находит количество элементов массива размером m,\n"
           "больших, равных и меньших K.\n");
    do{
        req_rslt=0;
        log=InputData(array,&size,&K);
        if (log){
            printf("Числа:\n");
            PrintVector(stdout,array,size);
            printf("будут сравниваться относительно %lf. OK (Y/N)?:\n",K);
            log=GetReqResult();
        }
        if(!log){
            printf("Ввод не завершен или отменен!\nПовторить(Y/N)?\n");
            req_rslt=GetReqResult();
        }
    }while(!log&&req_rslt);

    if (log){
        for (i=0;i<size;i++){
```

```
            switch(Compare(array[i],K)){
                case -1: und++;
                break;
                case 0: eq++;
                break;
                case 1: ov++;
                break;
            }
        }
        printf("Количестов элементов массива,\n"
               "меньших %lf: %ld;\n"
               "равных %lf: %ld;\n"
               "больших %lf: %ld;\n",
                        K,und,
                        K,eq,
                        K,ov);
    } else printf("Программа прервана...");
    printf("Нажмите <ENTER>...");
    clearline(stdin);
    return 0;
}
int Input(const char message[], //paramName - имя запрашиваемого параметра;
              const char name[], //paramCond - дополнительная информация о
параметре;
              double *param
                    ){
  int log,req_rslt;
    do {
        printf("%s",message);
        printf("%s: ",name);
        log=scanf("%lf",param);
        log=!clearline(stdin)&&log;
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
    } while (!log&&req_rslt); //пока пользователь не отказался или число
некорректное
    return log;

}
int InputIndex(const char message[],const char name[],long *index,long h_bound){
  int log,req_rslt;
```

```
    do{
        req_rslt=0;
        printf("%s",message);
        printf("%s: ",name);
        log=scanf("%ld",index);
        log=!clearline(stdin)&&log;
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
        if (log&&(*index<1||*index>h_bound)){
            log=0;
            printf("Ошибка! %s не принадлежит [1..%ld]!\n"
                   "Повторить ввод(Y/N)?\n",name,h_bound);
            req_rslt=GetReqResult();
        }
    } while(!log&&req_rslt);
    return log;
}
int InputConsole(double array[],long size){
  int log=1; char elem_name[20]; char msg[50]; long i;

    for(i=0;log&&i<size;i++){
        sprintf(elem_name,"A[%ld]",i+1);
        sprintf(msg,"Введите %s - вещественное число.\n",elem_name);
        log=Input(msg,elem_name,array+i);

        }
    return log;
}
int InputFile(double array[],long size){
  int log=1; FILE *f; long i; char ch;
    log=GetInFile(&f);
    if (log){
            for (i=0;(i<size)&&log;i++){
                while((ch=getc(f))==' '||ch=='\t');
                if (ch!='\n')
                    ungetc(ch,f);
                else
                    log=0;
                if (log){
                    log=fscanf(f,"%lf",array+i);
                }
```

```
            }
            if (log)
                log=!clearline(f)&&log;
            if (!log){
                //i--;j--;
                printf("Ошибка при чтении из файла элемента [%ld].\n",i);

            }
            if (f!=stdin)
                fclose(f);
    } else printf("Неправильное имя файла! \n");

    return log;
}
int InputRandom(double array[],long size){
  int log=1,req_rslt=0; double min,max; long i;
    srand (time(NULL));

    log=Input("Введите min - минимальное из случайных чисел.\n","min",&min);
    if (log)
        do{
            req_rslt=0;
            log=Input("Введите max - минимальное из случайных
чисел.\n","max",&max);
            if (log&&max<min){
               log=0;
                printf("Верхняя граница (max) меньше нижней (min)!\n");
                printf("Хотите повторить ввод? (Y/N)\n");
                req_rslt=GetReqResult();
            }
        } while (!log&&req_rslt);
    if (log){
        for (i=0;i<size;i++)
                array[i]=(double)((double)rand()/RAND_MAX *(max-min) + min);

    }
    return log;
}

int InputData(double array[],long *size,double *K){
  int log=1; char msg[50];
    sprintf(msg,"Введите m - размер массива. 1<=m<=%ld.\n",MAX_N);
    log=InputIndex(msg,"m",size,MAX_N);
    if (log)
```

```
        log=Input("Введите K - вещественное число.\n","K",K);
    if (log){
        printf("Заполните матрицу:\n"\
            "1 - заполнить случайными числами;\n"\
            "2 - ввести из файла;\n"\
            "3 - ввести с клавиатуры;\n"\
            "0 - завершить программу.\n");
        switch(GetOption('0','3')){
            case '1':
                log=InputRandom(array,*size);
            break;
            case '2':
                log=InputFile(array,*size);
            break;
            case '3':
                log=InputConsole(array,*size);
            break;
            default:
                printf("Ввод отменён!\n");
                log=0;
        }
    }

    return log;
}
```

1.6. Тестовый пример

Ниже на рисунке 1.14 представлен результат работы программы для массива, заполненного случайными числами (в данном случае {6.4516 138.0810 -53.3158 17.5481 -56.3463) , относительно K=3.

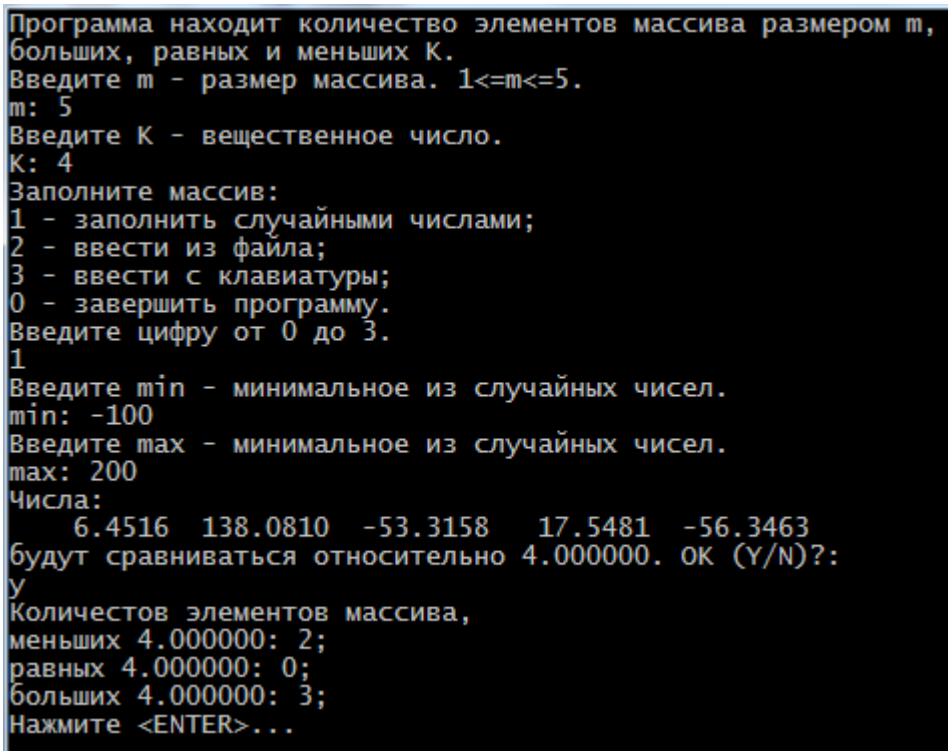

Рисунок 1.14 — Результат работы программы оценки элементов вектора

2. ЗАДАЧА УМНОЖЕНИЯ СТОЛБЦА МАТРИЦЫ НА ЧИСЛО

2.1. Схема алгоритма

На рисунке 2.1 изображена схема получения данных, умножения столбца матрицы на число, и вывода обновленной матрицы в файл или на экран.

Схему алгоритма получения ответа от пользователя на поставленный вопрос и схему алгоритма предоставления пользователю различных вариантов ответа можно увидеть на рисунках 1.3 и 1.4 соответственно, так как алгоритмы, осуществляющие решение данных задач, идентичны рассмотренным ранее.

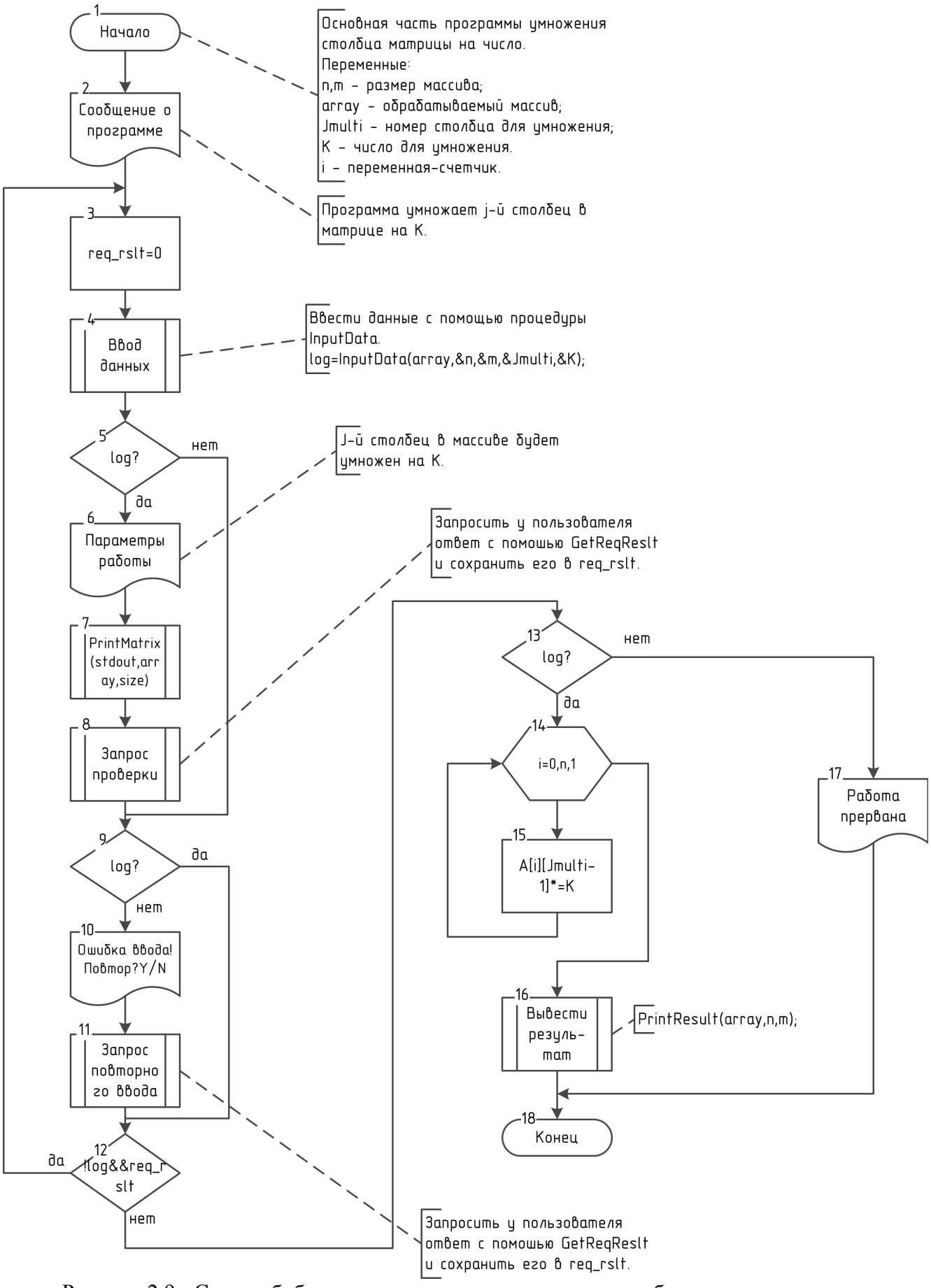

Рисунок 2.9 - Схема обобщенного алгоритма умножения столбца матрицы на число

Схема алгоритма заполнения двумерного массива с клавиатуры представлена на рисунке 2.2.

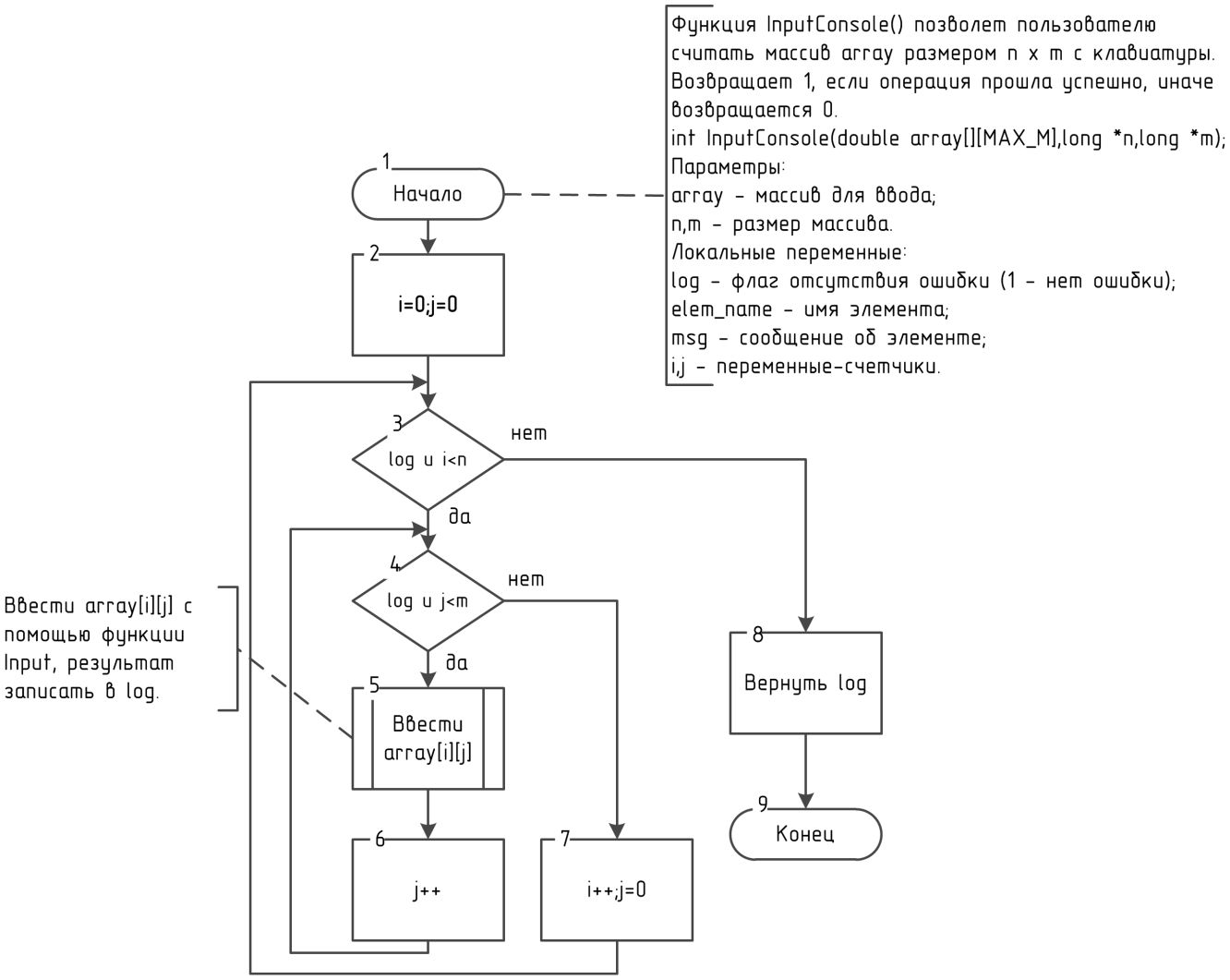

Рисунок 2.2 - Схема алгоритма заполнения двумерного массива с клавиатуры

Алгоритм ввода произвольного вещественного числа идентичен рассмотренному в предыдущем разделе, поэтому его можно увидеть на рисунке 1.11.

На рисунке 2.3 представлена схема заполнения двумерного массива из файла.

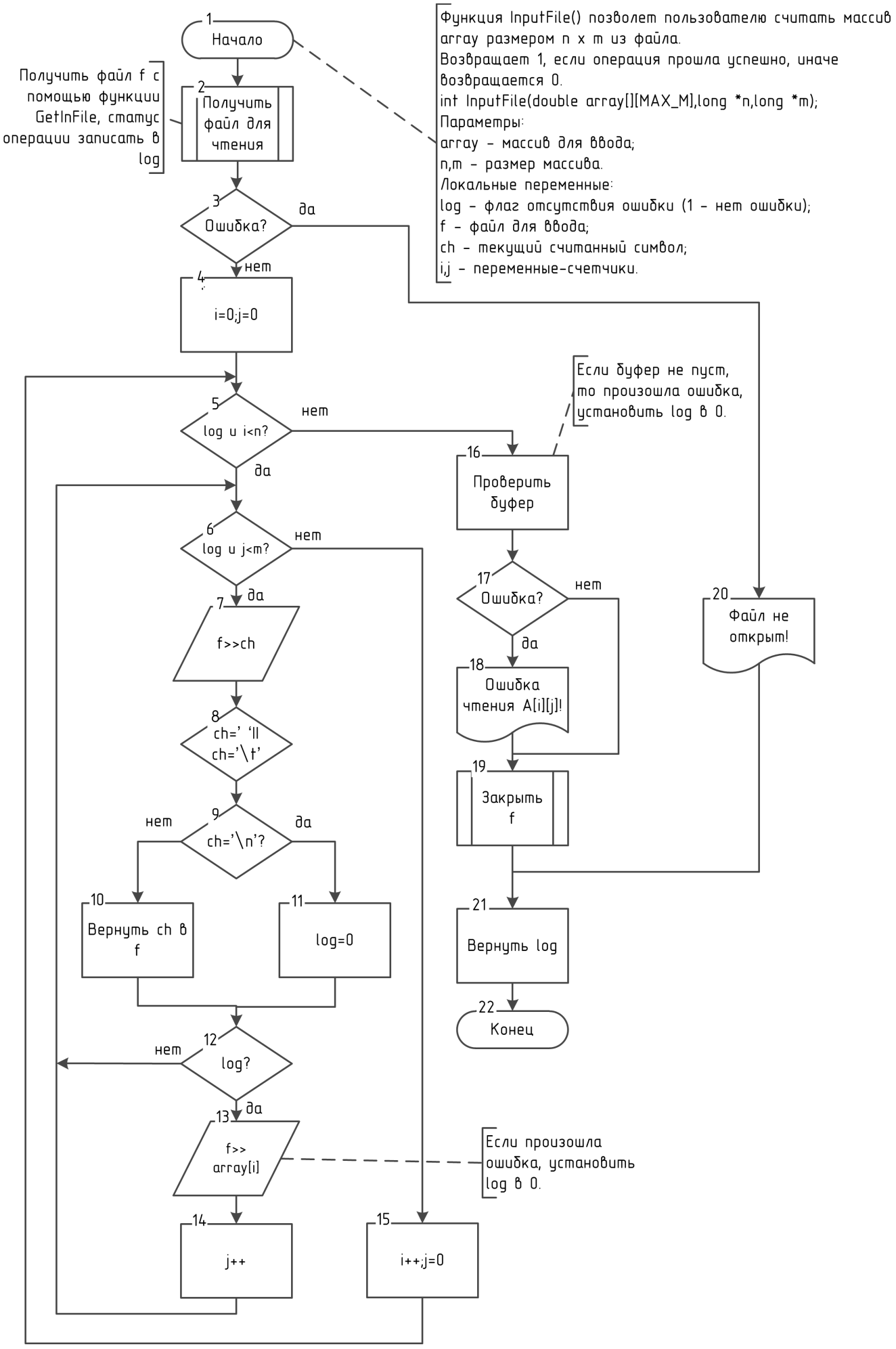

Рисунок 2.3 - Схема алгоритма заполнения двумерного массива из файла

На рисунке 2.4 представлена схема заполнения двумерного массива из случайными числами.

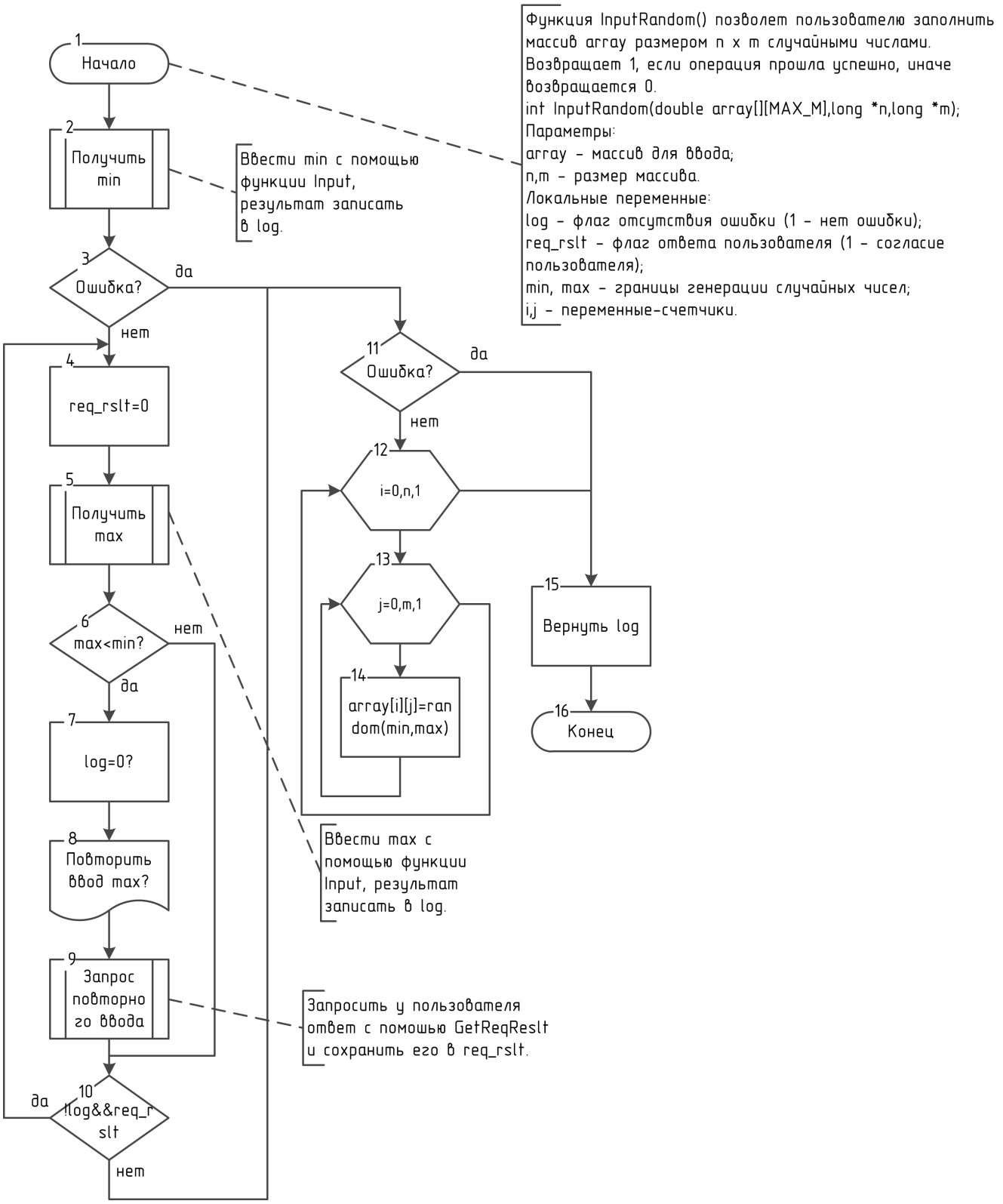

Рисунок 2.4 - Схема алгоритма заполнения двумерного массива случайными числами

На рисунке 2.5 представлена обобщенная схема ввода данных для программы умножения столбца матрицы на число.

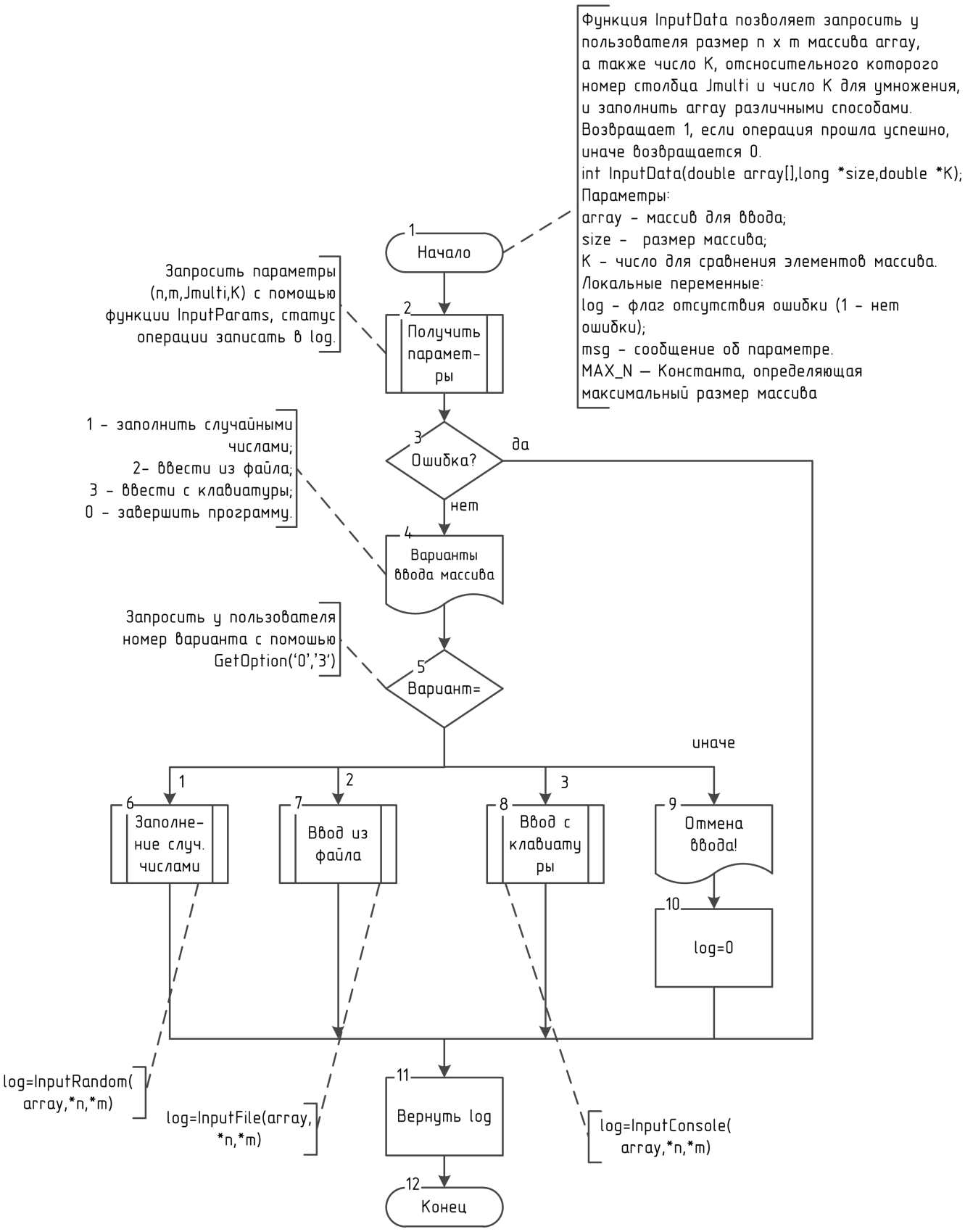

Рисунок 2.5 - Схема общего алгоритма ввода данных для программы умножения столбца матрицы на число

На рисунке 2.5 представлена обобщенная схема ввода скалярных (не являющихся массивами) данных для программы умножения столбца матрицы на число.

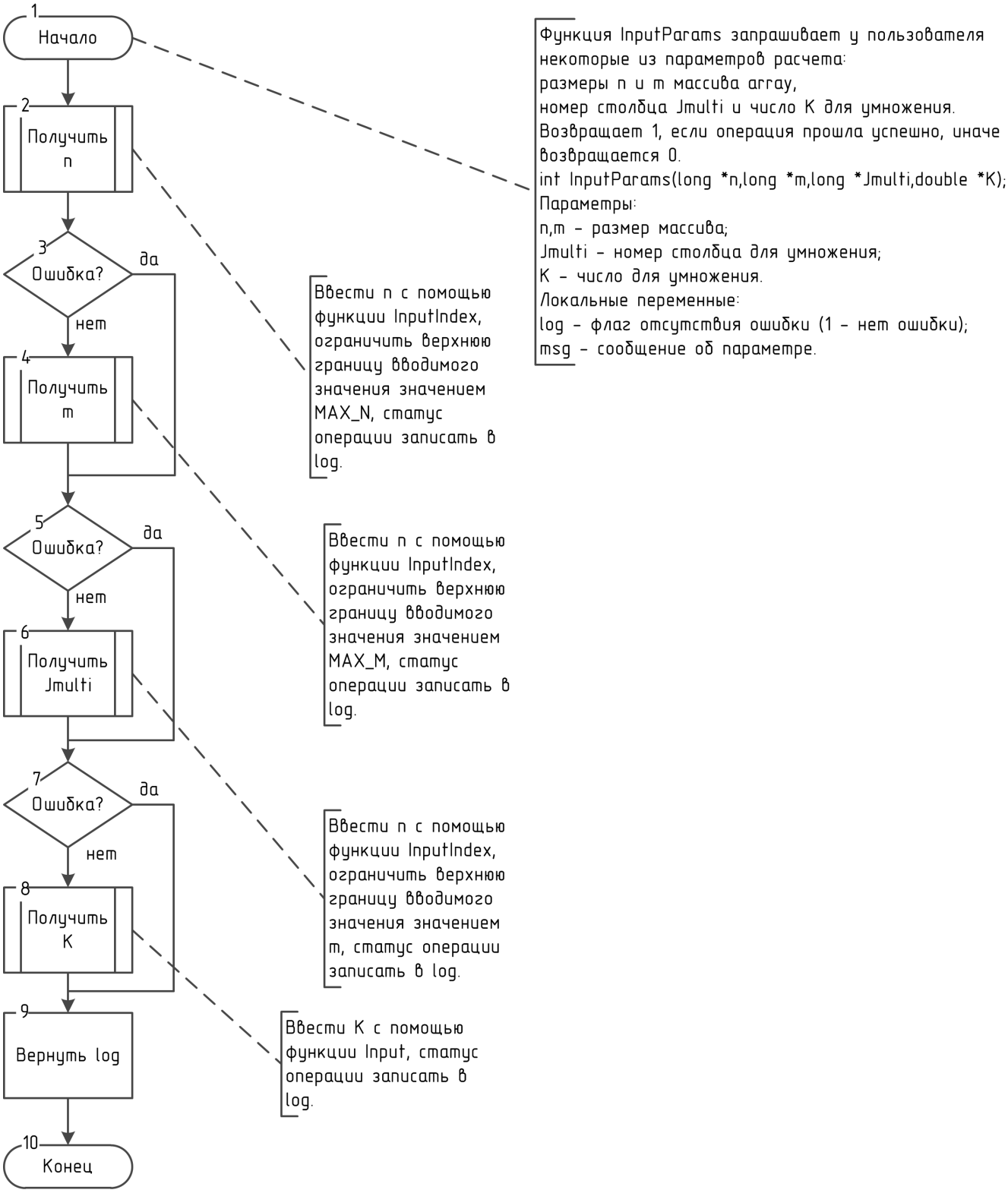

Рисунок 2.6 - Схема алгоритма ввода скалярных данных для программы умножения столбца матрицы на число

Схема получения индекса элемента массива была рассмотрена ранее, и её можно увидеть на рисунке 1.12.

На рисунке 2.7 представлена общая схема вывода матрицы в файл или на экран по желанию пользователя.

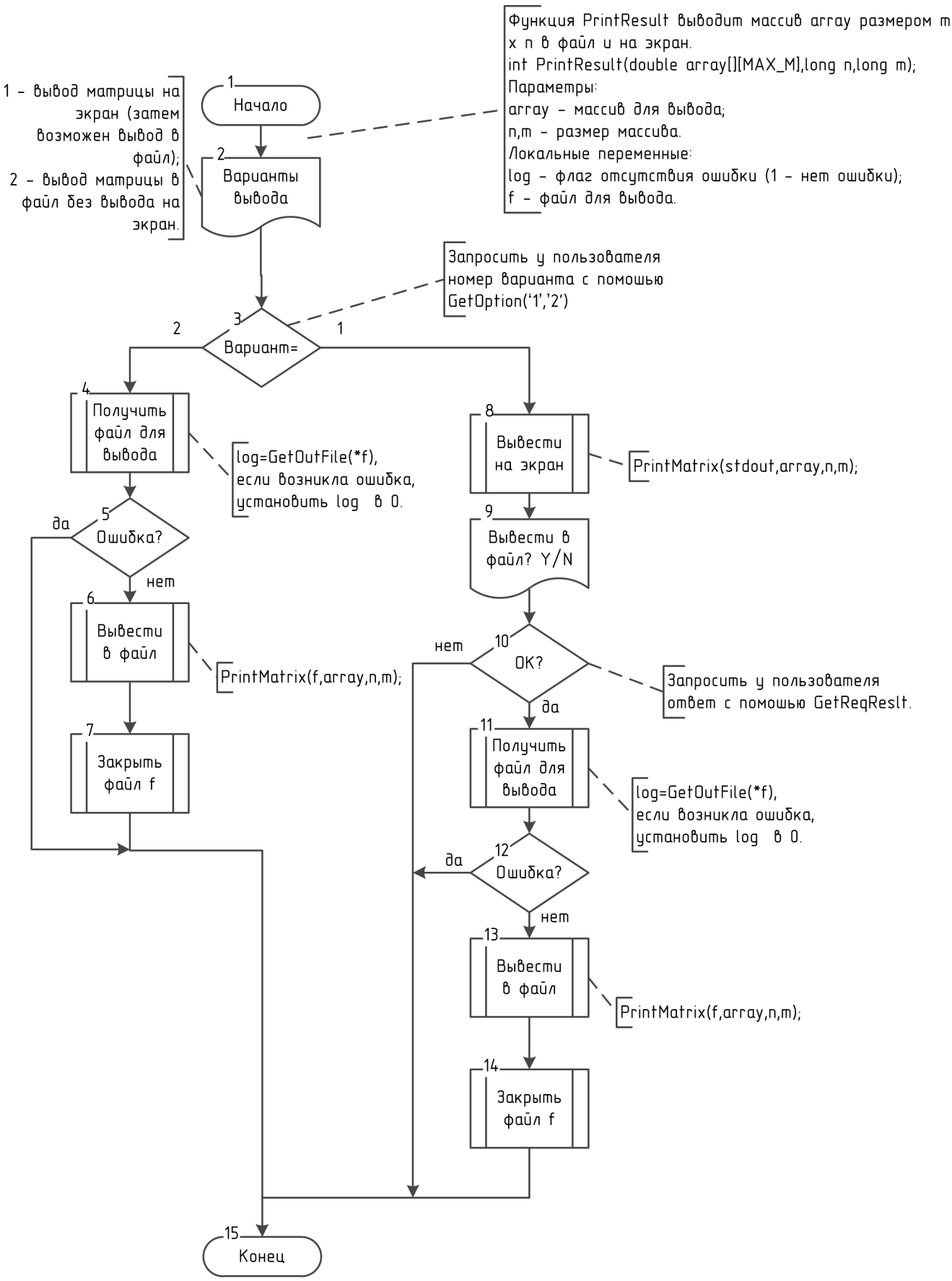

Рисунок 2.7 - Схема алгоритма вывода результатов работы программы умножения столбца матрицы на число

На рисунке 2.8 представлена схема вывода матрицы в файл.

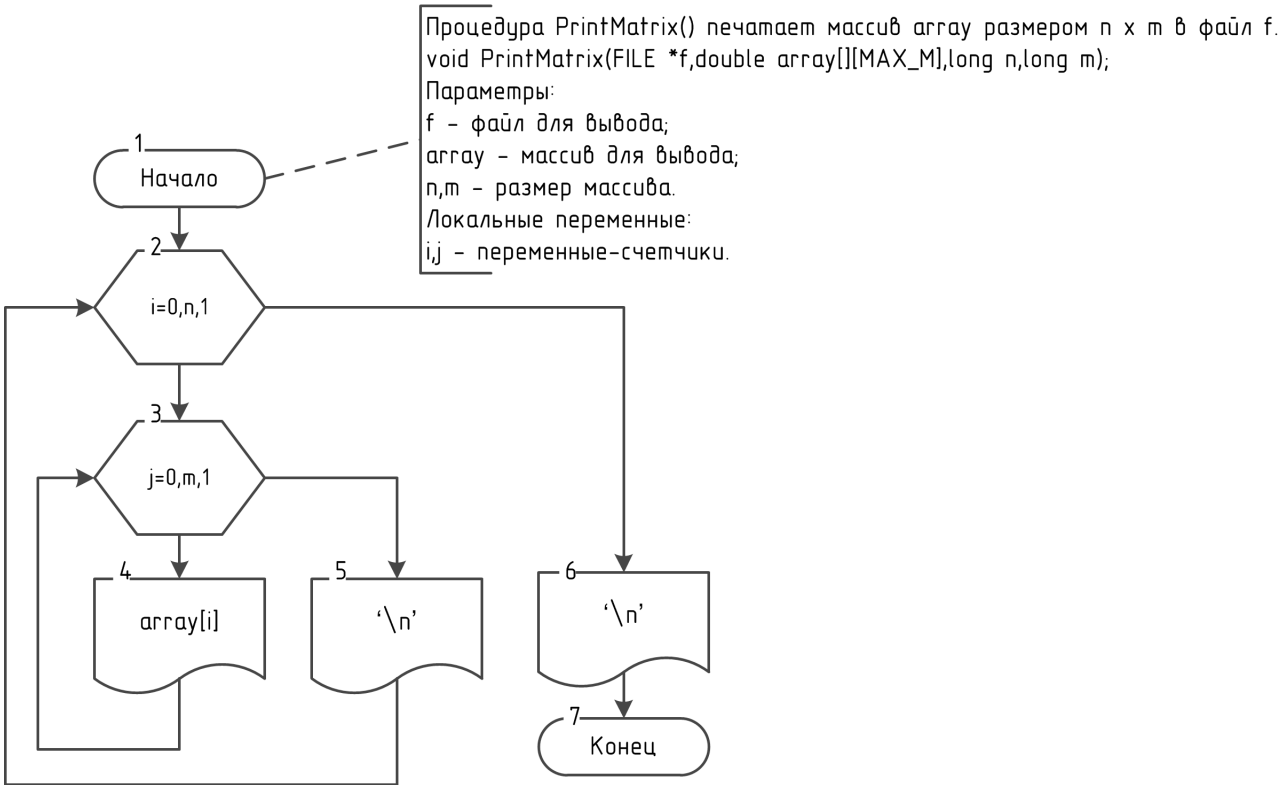

Рисунок 2.8 - Схема алгоритма вывода двумерного массива в файл

На рисунке 2.9 нарисована схема получения от пользователя файла-результата, в который будет выводится полученная матрица.

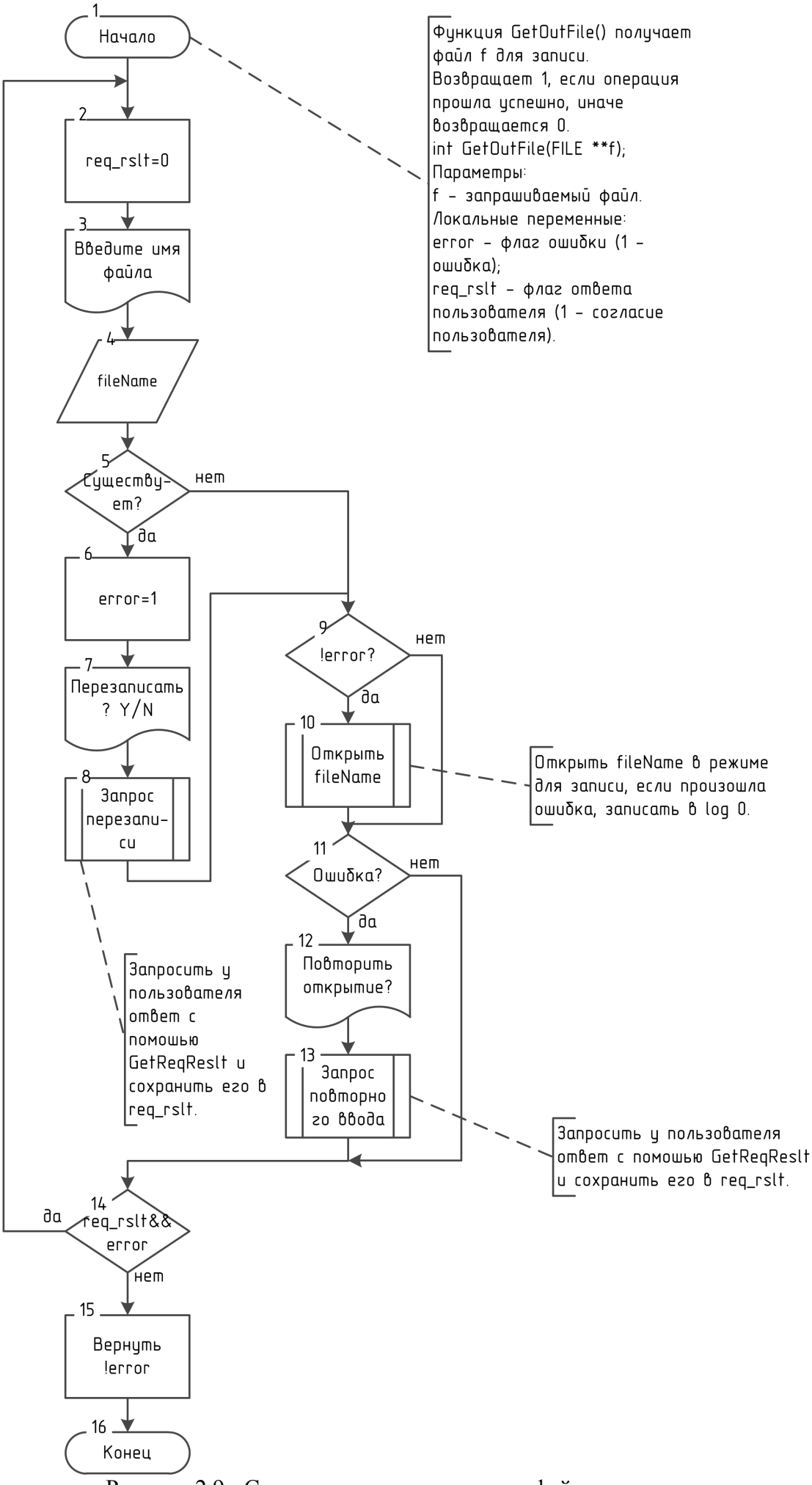

Рисунок 2.9 - Схема алгоритма получения файла-результата

2.2. Инструкция пользователю

Данная программа умножает произвольный столбец произвольной матрицы на произвольное число.

Для работы передайте программе размеры массива, затем номер столбца и число для умножения. После этого можно заполнить матрицу случайными числами, из файла или вручную с клавиатуры. При вводе матрицы из файла следует учитывать, что количество строк файла должно соответствовать указанному, количество элементов в строке также должно быть равно указанному значению. Элементы разделяются пробелами или\и табуляциями.

После умножения указанного столбца на указанное число матрицу можно вывести в файл или на экран по вашему желанию. При выборе файла следует учитывать потерю содержащийся в нем информации — файл будет перезаписан.

2.3. Инструкция программисту

Для решения задачи умножения столбца матрицы на число были предприняты следующие действия..

Объявлен макрос MAX_N и MAX_M, которые содержиат максимальное количество строк и столбцов массива соответственно.

Подключены заголовочные файлы <ctype.h> - для функции toupper() и <stdio.h> для функций ввода-вывода.

Объявлены следующие структуры данных:

Таблица 2.1 - Структуры данных, используемые в в основной части программы умножения столбца матрицы на число

имя	**тип**	**предназначение**
size	long	размер массива;
array	double*[MAX_M]	обрабатываемый массив;
K	double	число для сравнения элементов массива;
und,eq,ov	long	кол-во элементов меньших, равных и больших K;
i	long	переменная-счетчик.

Также программа была разбита на следующие подпрограммы:

1. Функция GetInFile() получает файл f для чтения.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int GetInFile(FILE **f);

Параметры функции представлены в таблице 2.2:

Таблица 2.2 - Параметры функции получения файла для чтения

имя	**тип**	**предназначение**
f	FILE *	запрашиваемый файл.

Локальные переменные функции представлены в таблице 2.3:

Таблица 2.3 - Локальные переменные функции получения файла для чтения

имя	**тип**	**предназначение**
error	int	флаг ошибки (1 - ошибка);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

2. Функция GetOutFile() получает файл f для записи.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int GetOutFile(FILE **f);

Параметры функции представлены в таблице 2.4:

Таблица 2.4 - Параметры функции получения файла для записи

имя	тип	предназначение
f	FILE *	запрашиваемый файл.

Локальные переменные функции представлены в таблице 2.5:

Таблица 2.5 - Локальные переменные функции получения файла для записи

имя	тип	предназначение
error	int	флаг ошибки (1 - ошибка);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

3. Функция PrintResult выводит массив array размером m x n в файл и на экран.

int PrintResult(double array[][MAX_M],long n,long m);

Параметры функции представлены в таблице 2.6:

Таблица 2.6 - Параметры функции вывода матрицы в файл или на экран

имя	тип	предназначение
array	double*[MAX_M]	массив для вывода;
n,m	long	размер массива.

Локальные переменные функции представлены в таблице 2.7:

Таблица 2.7 - Локальные переменные функции вывода матрицы в файл или на экран

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
f	FILE *	файл для вывода;

4. Процедура PrintMatrix() печатает массив array размером n x m в файл f.

void PrintMatrix(FILE *f,double array[][MAX_M],long n,long m);

Параметры функции представлены в таблице 2.8:

Таблица 2.8 - Параметры функции вывода матрицы в файл

имя	тип	предназначение
f	FILE *	файл для вывода;
array	double*	массив для вывода;
n,m	long	размер массива.

Локальные переменные функции представлены в таблице 2.9:

Таблица 2.9 - Локальные переменные функции вывода матрицы в файл

имя	тип	предназначение
i ,j	long	переменные-счетчики.

5. Функция Input() получает от пользователя некоторое вещественное число param.

При этом она в процессе запроса выводит поясняющее сообщение message, и приглашение ко вводу в виде имени параметра – name.

```
int Input(const char message[],
        const char name[],
        double *param
            );
```

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

Параметры функции представлены в таблице 2.10:

Таблица 2.10 - Параметры функции получения вещественного числа

имя	**тип**	**предназначение**
message	char*	сообщение о параметре;
name	char*	имя параметра;
param	double*	запрашиваемый параметр.

Локальные переменные функции представлены в таблице 2.11:

Таблица 2.11 - Локальные переменные функции получения вещественного числа

имя	**тип**	**предназначение**
ok	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

6. Функция InputIndex() получает от пользователя индекс массива - некоторое длинное целое число index из промежутка от 0 до верхней границы h_bound.

При этом она в процессе запроса выводит поясняющее сообщение message, и приглашение ко вводу в виде имени параметра – name.

```
int InputIndex(const char message[],const char name[],long *index,long h_bound);
```

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

Параметры функции представлены в таблице 2.12:

Таблица 2.12 - Параметры функции получения индекса элемента массива

имя	**тип**	**предназначение**
message	char*	сообщение о параметре;
name	char*	имя параметра;

index	long*	запрашиваемый индекс;
h_bound	long*	верхняя граница индекса.

Локальные переменные функции представлены в таблице 2.13:

Таблица 2.13 - Локальные переменные функции получения индекса элемента массива

имя	**тип**	**предназначение**
ok	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя).

7. Функция InputConsole() позволет пользователю считать массив array размером n x m с клавиатуры.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputConsole(double array[][MAX_M],long *n,long *m);

Параметры функции представлены в таблице 2.14:

Таблица 2.14 - Параметры функции заполнения матрицы с клавиатуры

имя	**тип**	**предназначение**
array	double*[MAX_M]	массив для ввода;
n,m	long	размер массива.

Локальные переменные функции представлены в таблице 2.15:

Таблица 2.15 - Локальные переменные функции заполнения матрицы с клавиатуры

имя	**тип**	**предназначение**
log	int	флаг отсутствия ошибки (1 - нет ошибки);
elem_name	char[20]	имя элемента;
msg	char[150]	сообщение об элементе;
i,j	long	переменные-счетчики.

8. Функция InputFile() позволет пользователю считать массив array размером n x m из файла.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputFile(double array[][MAX_M],long *n,long *m);

Параметры функции представлены в таблице 2.16:

Таблица 2.16 - Параметры функции заполнения матрицы из файла

имя	**тип**	**предназначение**
array	double*[MAX_M]	массив для ввода;
n,m	long	размер массива.

Локальные переменные функции представлены в таблице 2.17:

Таблица 2.17 - Локальные переменные функции заполнения матрицы из файла

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
f	FILE *	файл для ввода;
ch	char	текущий считанный символ;
i ,j	long	переменные-счетчики.

9. Функция InputRandom() позволет пользователю заполнить массив array случайными числами.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputRandom(double array[][MAX_M],long *n,long *m);

Параметры функции представлены в таблице 2.18:

Таблица 2.18 - Параметры функции заполнения матрицы случайными числами

имя	тип	предназначение
array	double*[MAX_M]	массив для ввода;
n,m	long	размер массива.

Локальные переменные функции представлены в таблице 2.19:

Таблица 2.19 - Локальные переменные функциизаполнения матрицы случайными числами

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
req_rslt	int	флаг ответа пользователя (1 - согласие пользователя);
min, max	double	границы генерации случайных чисел;
i ,j	long	переменные-счетчики.

10. Функция InputParams запрашивает у пользователя некоторые из параметров расчета: размеры n и m массива array, номер столбца Jmulti и число K для умножения.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputParams(long *n,long *m,long *Jmulti,double *K);

Параметры функции представлены в таблице 2.20:

Таблица 2.20 - Параметры функции получения параметров работы

имя	тип	предназначение
n,m	long	размер массива;
Jmulti	long	номер столбца для умножения;
K	double	число для умножения.

Локальные переменные функции представлены в таблице 2.21:

Таблица 2.21 - Локальные переменные функции получения параметров работы

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
msg	char[150]	сообщение о параметре;

11. Функция InputData позволяет запросить у пользователя размеры n и m массива array, номер столбца Jmulti и число K для умножения, и заполнить array различными способами.

Возвращает 1, если операция прошла успешно, иначе возвращается 0.

int InputData(double array[][MAX_M],long *n,long *m,long *Jmulti,double *K);

Параметры функции представлены в таблице 2.22:

Таблица 2.22 - Параметры функции ввода данных программы

имя	тип	предназначение
array	double*[MAX_M]	массив для ввода;
size	long*	размер массива.
K	double*	число для сравнения элементов массива.

Локальные переменные функции представлены в таблице 2.23:

Таблица 2.23 - Локальные переменные функции ввода данных программы

имя	тип	предназначение
log	int	флаг отсутствия ошибки (1 - нет ошибки);
msg	char[255]	сообщение об параметре.

11.Функция clearline используется для очистки строки файла.

Заголовок функции:

int clearline(FILE *f);

Функция считывает до конца файла или строки файла f символы в цикле while и возвращает их количество.

Параметры функции представлены в таблице 2.24, локальные переменные — в таблице 2.25.

Таблица 2.24 - Параметры функции получения сочетания

имя	тип	предназначение
f	FILE*	Файл, в котором очищается строка.

Таблица 2.25 - Локальные переменные функции получения сочетания

имя	тип	предназначение

count	long	Счетчик считанных символов.

12. Функция GetReqResult используется для получения ответов пользователя на запросы программы.

Заголовок функции:

int GetReqResult();

Функция запрашивает у пользователя символы 'y','Y','n' или 'N' до тех пор, пока он не введет какой-либо из них. Соответственно возвращается либо символ 'Y', либо 'N'.

Локальные переменные функции представлены в таблице 2.26.

Таблица 2.26 - Локальные переменные функции получения сочетания

имя	**тип**	**предназначение**
answer	char	Ответ пользователя.

13. Функция GetOption позволяет выбирать пользователю из нескольких (до 10) вариантов ответа.

Заголовок функции:

int GetOption(char a,char b);

Функция запрашивает у пользователя символы из промежутка от a до b до тех пор, пока он не введет какой-либо из них. Соответственно возвращается значение введенного символа.

Параметры функции представлены в таблице 2.27, локальные переменные — в таблице 2.28.

Таблица 2.27 - Параметры функции получения сочетания

имя	**тип**	**предназначение**
a,b	char	Различные варианты представлены цифрами от a до b.

Таблица 2.28 - Локальные переменные функции получения сочетания

имя	**тип**	**предназначение**
ch	char	Ответ пользователя.
ok	int	Флаг — равен 1, если ответ пользователя допустим, иначе равен 0.

2.4. Текст программы

Ниже представлен текст программы, написанной на языке Borland C 3.1, которая умноженает столбец матрицы на число.

```
#include <stdio.h>
#include <ctype.h>
#include <time.h>
#define MAX_N 5L
#define MAX_M 5L
/**************************/
/*
Функция clearline очищает строку в файле f. Возвращает количество
непробельных символов в считанной строке.
Параметры:
f - Файл для очистки строки.
Локальные переменные:
count - количество непробельных символов;
ch - текущий считанный символ.
*/
int clearline(FILE *f);


/*
Функция GetReqResult позволяет получить ответ "да" или "нет" от пользователя.
Возвращает 1 в случае согласия пользователя, 0 - в случае несогласия.
Параметры:
отсутствуют.
Локальные переменные:
answer - текущий ответ пользователя.
*/
int GetReqResult();


/*
Функция GetOption позволяет выбрать пользователю один из нескольких(до 10)
вариантов,
представленных цифрами от a до b. Возвращает символ, соответствующий выбранной
цифре.
Параметры:
a,b - границы предлагаемых вариантов.
Локальные переменные:
ch - текущий ответ пользователя;
ok - флаг отсутствия ошибки (1 - нет ошибки).
*/
```

```
char GetOption(char a,char b);


/*
Функция fexists() проверяет существование файла с именем fname.
Возвращает 1, если такой файл существует, или 0, если нет.
Параметры:
fname - имя файла для проверки.
Локальные переменные:
f - временный файл для проверки.
*/
int fexists(char  *fname);


/*
Функция GetInFile() получает файл f для чтения.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
f - запрашиваемый файл.
Локальные переменные:
error - флаг ошибки (1 - ошибка);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int GetInFile(FILE **f);


/*
Функция GetOutFile() получает файл f для записи.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
f - запрашиваемый файл.
Локальные переменные:
error - флаг ошибки (1 - ошибка);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int GetOutFile(FILE **f);


/*
Функция PrintResult выводит массив array размером m x n в файл и на экран.
Параметры:
array - массив для вывода;
n,m - размер массива.
Локальные переменные:
```

```
log - флаг отсутствия ошибки (1 - нет ошибки);
f - файл для вывода.
*/
int PrintResult(double array[][MAX_M],long n,long m);


/*
Процедура PrintMatrix() печатает массив array размером n x m в файл f.
Параметры:
f - файл для вывода;
array - массив для вывода;
n,m - размер массива.
Локальные переменные:
i,j - переменные-счетчики.
*/
void PrintMatrix(FILE *f,double array[][MAX_M],long n,long m);


/*
Основная часть программы умножения столбца матрицы на число.
Переменные:
n,m - размер массива;
array - обрабатываемый массив;
Jmulti - номер столбца для умножения;
K - число для умножения.
i - переменная-счетчик.
*/
int main();


/*
Функция Input() получает от пользователя некоторое вещественное число param.
При этом она в процессе запроса выводит поясняющее сообщение message,
и приглашение ко вводу в виде имени параметра - name.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
message - сообщение о параметре;
name - имя параметра;
param - запрашиваемый параметр.
Локальные переменные:
ok - флаг отсутствия ошибки (1 - нет ошибки);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int Input(const char message[],
```

```
                const char name[],
                double *param
                      );


/*
Функция InputIndex() получает от пользователя индекс массива -
некоторое длинное целое число index из промежутка от 0 до верхней границы h_bound.
При этом она в процессе запроса выводит поясняющее сообщение message,
и приглашение ко вводу в виде имени параметра - name.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
message - сообщение о параметре;
name - имя параметра;
index - запрашиваемый индекс;
h_bound - верхняя граница индекса.
Локальные переменные:
ok - флаг отсутствия ошибки (1 - нет ошибки);
req_rslt - флаг ответа пользователя (1 - согласие пользователя).
*/
int InputIndex(const char message[],const char name[],long *index,long h_bound);


/*
Функция InputConsole() позволет пользователю считать массив array размером size с
клавиатуры.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
n,m - размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
elem_name - имя элемента;
msg - сообщение об элементе;
i,j - переменные-счетчики.
*/
int InputConsole(double array[][MAX_M],long *n,long *m);


/*
Функция InputFile() позволет пользователю считать массив array размером size из
файла.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
```

```
array - массив для ввода;
n,m - размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
f - файл для ввода;
ch - текущий считанный символ;
i,j - переменные-счетчики.
*/
int InputFile(double array[][MAX_M],long *n,long *m);


/*
Функция InputRandom() позволет пользователю заполнить массив array случайными
числами.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
n,m - размер массива.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
req_rslt - флаг ответа пользователя (1 - согласие пользователя);
min, max - границы генерации случайных чисел;
i,j - переменные-счетчики.
*/
int InputRandom(double array[][MAX_M],long *n,long *m);


/*
Функция InputData позволяет запросить у пользователя размеры n и m массива array,
номер столбца Jmulti и число K для умножения.
а также число K, отсносительного которого будут сравниваться элементы array,
и заполнить array различными способами.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
array - массив для ввода;
n,m - размер массива;
Jmulti - номер столбца для умножения;
K - число для умножения.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки).
*/
int InputData(double array[][MAX_M],long *n,long *m,long *Jmulti,double *K);
```

```
/*
Функция InputParams запрашивает у пользователя некоторые из параметров расчета:
размеры n и m массива array,
номер столбца Jmulti и число K для умножения.
Возвращает 1, если операция прошла успешно, иначе возвращается 0.
Параметры:
n,m - размер массива;
Jmulti - номер столбца для умножения;
K - число для умножения.
Локальные переменные:
log - флаг отсутствия ошибки (1 - нет ошибки);
msg - сообщение об параметре.
*/
int InputParams(long *n,long *m,long *Jmulti,double *K);
/**************************/

int clearline(FILE *f){
  int count=0;char ch;
      while (!feof(f)&&((ch=getc(f))!='\n'))
        if(ch!=' '&&ch!='\t')
            count++;
      return count;
}
int GetReqResult(){
  char answer;
    answer=getchar();
    clearline(stdin);
    while (toupper(answer)!='Y'
           &&toupper(answer)!='N'){
        printf("Неправильный ответ! Допустимо:\n"\
                                "Y - да; N - нет.\n");
        answer=getchar();
        clearline(stdin);
    }

    return toupper(answer)=='Y';
}
char GetOption(char a,char b){
  char ch; int ok;
    do{
        printf("Введите цифру от %c до %c.\n",a,b);
            ch=getchar();
            ok=(a<=ch)&&(ch<=b);
            ok=!clearline(stdin)&&ok;
```

```
            if (!ok)
            printf("Неправильный ответ!\n");
    }while (!ok);
      return ch;
}
int fexists(char  *fname){
  FILE *f;
    f=fopen(fname,"r");
    if (f!=NULL)
        fclose(f);
    return f!=NULL;
}
int GetInFile(FILE **f){
  int error,req_rslt;
  char fileName[255]="\0";
    do{
        req_rslt=0;
        printf("Введите имя файла-источника.\n");
        //считать строку (gets() deprecated!)
        printf("%s: ","Файл");scanf("%255[^\n]",fileName);
        clearline(stdin);
        *f=fopen(fileName,"r");
        error=*f==NULL;
        if (error) {
            printf( "Неправильное имя файла! \n");
            printf("Хотите повторить ввод? (Y/N)\n");
            req_rslt=GetReqResult();
        };
    } while (req_rslt&&error);
    return !error;
  };
int GetOutFile(FILE * * f){
  char fileName [255]="\0";
  int error=0,req_rslt;
    do{
        req_rslt=0;
        printf("Введите имя файла-результата.\n");
        //считать строку (gets() deprecated!)
        printf("%s: ","Файл");scanf("%255[^\n]",fileName);
        clearline(stdin);
        if (fexists(fileName)){
            error=1;
            printf("ВНИМАНИЕ! Указанное имя файла занято!\n");
            printf("ПЕРЕЗАПИСАТЬ ФАЙЛ? (Y/N)\n");
```

```
            error=!GetReqResult();
        }
        if (!error){
            *f=fopen(fileName,"w");
            error=*f==NULL;
        }
        if (error) {
            printf("Неправильное имя файла! \n");
            printf("Хотите повторить ввод? (Y/N)\n");
            req_rslt=GetReqResult();
        };
    } while (req_rslt&&error);
    return !error;
  };
int Input(const char message[], //paramName - имя запрашиваемого параметра;
              const char name[], //paramCond - дополнительная информация о
параметре;
              double *param
                    ){
  int log,req_rslt;
    do {
        printf("%s",message);
        printf("%s: ",name);
        log=scanf("%lf",param);
        log=!clearline(stdin)&&log;
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
    } while (!log&&req_rslt); //пока пользователь не отказался или число
некорректное
    return log;

}

int InputIndex(const char message[],const char name[],long *index,long h_bound){
  int log,req_rslt;
    do{
        req_rslt=0;
        printf("%s",message);
        printf("%s: ",name);
        log=scanf("%ld",index);
        log=!clearline(stdin)&&log;
```

```
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
        if (log&&(*index<1||*index>h_bound)){
            log=0;
            printf("Ошибка! %s не принадлежит [1..%ld]!\n"
                   "Повторить ввод(Y/N)?\n",name,h_bound);
            req_rslt=GetReqResult();
        }
    } while(!log&&req_rslt);
    return log;
}
int InputParams(long *n,long *m,long *Jmulti,
                double *K){
  int log; char msg[100];
    sprintf(msg,"Введите n - количество строк массива. 1<=n<=%ld.\n",MAX_N);
    log=InputIndex(msg,"n",n,MAX_N);
    if (log){
        sprintf(msg,"Введите m - количество столбцов массива. 1<=m<=
%ld.\n",MAX_M);
        log=InputIndex(msg,"m",m,MAX_M);
    }

    if (log){
        sprintf(msg,"Введите J - номер столбца для умножения. 1<=J<=%ld.\n",*m);
        log=InputIndex(msg,"J",Jmulti,*m);
    }

    if (log)
        log=Input("Введите K - вещественное число.\n","K",K);
    return log;
}
int InputConsole(double array[][MAX_M],long *n,long *m){
  int log=1; char elem_name[20]; long i,j;

    for(i=0;log&&i<*n;i++)
        for(j=0;j<*m&&log;j++){
            sprintf(elem_name,"A[%ld][%ld]",i+1,j+1);
            log=Input("",elem_name,array[i]+j);

        }
    return log;
```

```
}
int InputFile(double array[][MAX_M],long *n,long *m){
  int log=1; FILE *f; long i,j; char ch;
    log=GetInFile(&f);
    if (log){
            for (i=0;(i<=*n-1)&&log;i++){
                for (j=0;(j<=*m-1)&&log;j++){
                    while((ch=getc(f))==' '||ch=='\t');
                    if (ch!='\n')
                        ungetc(ch,f);
                    else
                        log=0;
                    if (log){
                        log=fscanf(f,"%lf",array[i]+j);
                    }
                }
                if (log){
                    log=!clearline(f);
                }
            };
            if (!log){
                //i--;j--;
                printf("Ошибка при чтении из файла элемента %ld,%ld.\n",i,j);

            }
            if (f!=stdin)
                fclose(f);
    } else printf("Неправильное имя файла! \n");

    return log;
}
int InputRandom(double array[][MAX_M],long *n,long *m){
  int log=1,req_rslt=0; double min,max; long i,j;
    srand (time(NULL));

    log=Input("Введите min - минимальное из случайных чисел.\n","min",&min);
    if (log)
        do{
            req_rslt=0;
            log=Input("Введите max - минимальное из случайных
чисел.\n","max",&max);
            if (log&&max<min){
               log=0;
                printf("Верхняя граница (max) меньше нижней (min)!\n");
```

```
                printf( "Хотите повторить ввод? (Y/N)\n");
                req_rslt=GetReqResult();
            }
        } while (!log&&req_rslt);
    if (log){
        for (i=0;i<=*n-1;i++)
            for (j=0;j<=*m-1;j++)
                array[i][j]=(double)((double)rand()/RAND_MAX *(max-min) + min);

    }
    return log;
}
void PrintMatrix(FILE *f,double array[][MAX_M],long n,long m){
  int i,j;
    for (i=0;i<n;i++){
        for(j=0;j<m;j++){
            fprintf(f,"%10.4lf",array[i][j]);
        }
        fprintf(f,"\n");
    }
}
int main(){
  long n,m,Jmulti;
  long i;
  double array[MAX_N][MAX_M];
  double K;
  printf("Программа умножает J-ый столбец массива nxm,\n"
         "на число K.\n");
  int log,req_rslt;
    do{
        log=InputData(array,&n,&m,&Jmulti,&K);
        if (log){
            printf("%ld-й столбец массива:\n",Jmulti);
            PrintMatrix(stdout,array,n,m);
            printf("будет умножен на %lf. OK(Y/N)?\n",K);
            log=GetReqResult();
        }
        if(!log){
            printf("Ввод не завершен или отменен!\nПовторить(Y/N)?\n");
            req_rslt=GetReqResult();
        }
    }while(!log&&req_rslt);
        if (log){
        Jmulti--;
```

```
        for (i=0;i<n;i++){
            array[i][Jmulti]*=K;
        }
        PrintResult(array,n,m);
    } else printf("Программа прервана...\n");
    printf("Нажмите <ENTER>...");
    clearline(stdin);
    return 0;
}
int PrintResult(double array[][MAX_M],long n,long m){
  int log=1; FILE *f;
    printf("Работа выполнена!\n");
    printf("1 - вывод матрицы на экран (затем возможен вывод в файл);\n"\
                                        "2 - вывод матрицы в файл без вывода на
экран.\n");
    switch(GetOption('1','2')){
        case '1':
            PrintMatrix(stdout,array,n,m);
            printf("Хотите вывести матрицу в файл? Y/N\n");
            if (GetReqResult()){
                if ((log=GetOutFile(&f))){
                    PrintMatrix(f,array,n,m);fclose(f);
                }
        }
        break;
        case '2':
            if ((log=GetOutFile(&f)))
                PrintMatrix(f,array,n,m);
        break;
    }
    return log;
}
int InputData(double array[][MAX_M],long *n,long *m,long *Jmulti,
                double *K){
  int log=1;
    log=InputParams(n,m,Jmulti,K);
    if (log){
        printf("Заполните матрицу:\n"\
                                        "1 - заполнить случайными числами;\n"\
                                        "2 - ввести из файла;\n"\
                                        "3 - ввести с клавиатуры;\n"\
                                        "0 - завершить программу.\n");
        switch(GetOption('0','3')){
            case '1':
```

```
                log=InputRandom(array,n,m);
            break;
            case '2':
                log=InputFile(array,n,m);
            break;
            case '3':
                log=InputConsole(array,n,m);
            break;
            default:
                printf( "Ввод отменён!\n");
                log=0;
        }
    }

    return log;
}
```

2.5. Тестовый пример

На рисунке 2.10 представлен пример работы программы для массива 3x4, заполняемого случайными числами (конкретные значения можно увидеть на рисунке 2.10), у которого третий столбец умножается на 3..

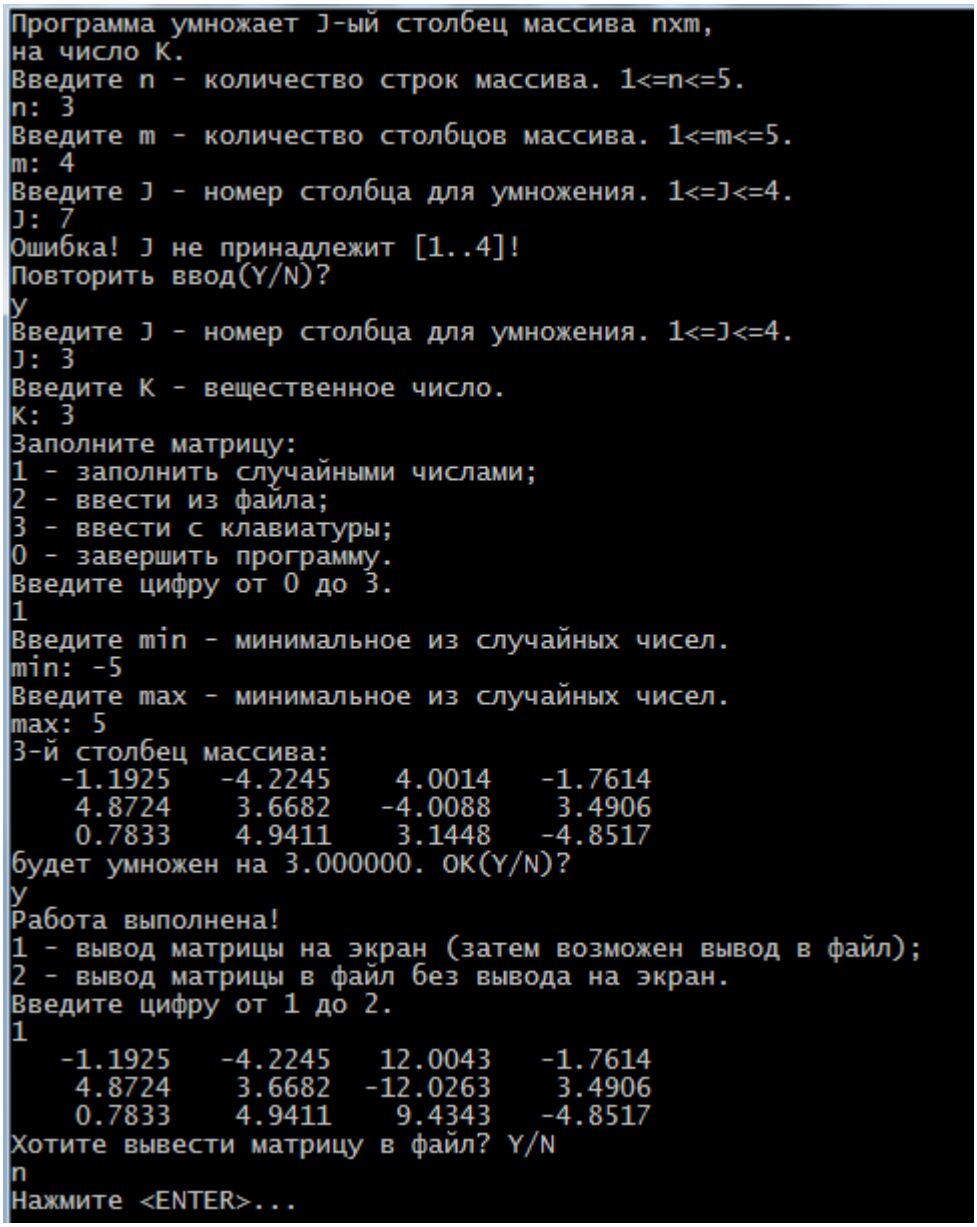

Рисунок 2.8— Результат работы программы умножения столбца матрицы

Вывод

В ходе выполнения данной лабораторной работы я научился использовать массивы различных размерностей при написании программ на языке Си. Поддержка индексированных структур данных и простой и ясный синтаксис языка позволяют с успехом решать научные, инженерные, экономические и многие другие задачи в тех областях, гле требуется быстрая и качественная обработка больших объёмов данных.