Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №2
«Вычисление значений числовых рядов и функций с заданной точностью»

Студент: Курочкин Д. И.
Группа: Б24-513
Преподаватель: Комаров Т. И.

Москва — 2024

1. Формулировка индивидуального задания

Вариант №27. Вычислить значение функции $y = \sqrt[x]{a}$ в точке по итерационной формуле:

$$y_{i+1} = \frac{1}{n}((n-1)y_i + \frac{a}{y_i^{n-1}})$$

где $y_0 = a$.

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовался встроенный тип данных число с плавающей точкой повышенной точности `long double` (спецификатор формата: `%Lf`).

3. Описание использованного алгоритма

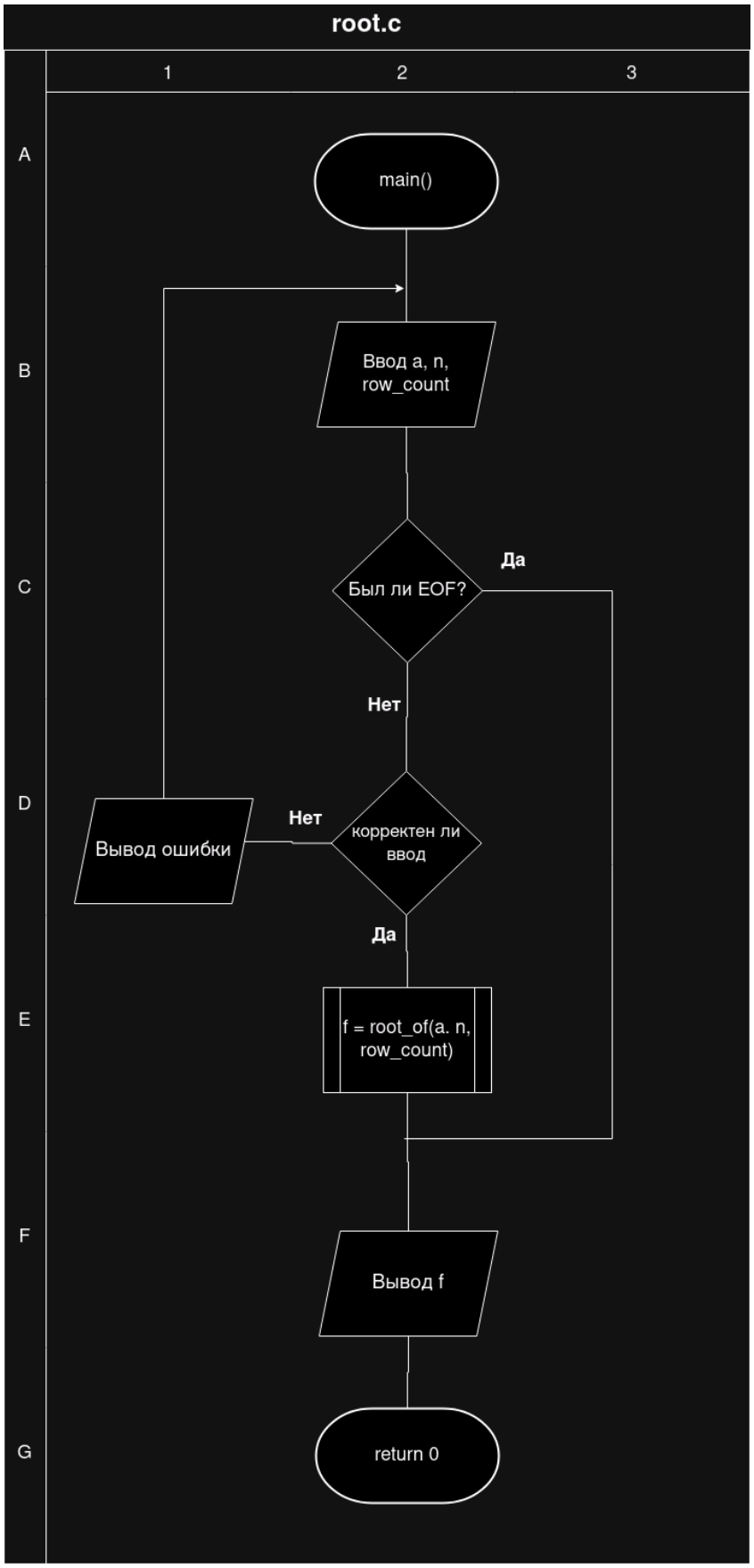


Рис. 1: Блок-схема алгоритма работы функции `main()` для программы `root`

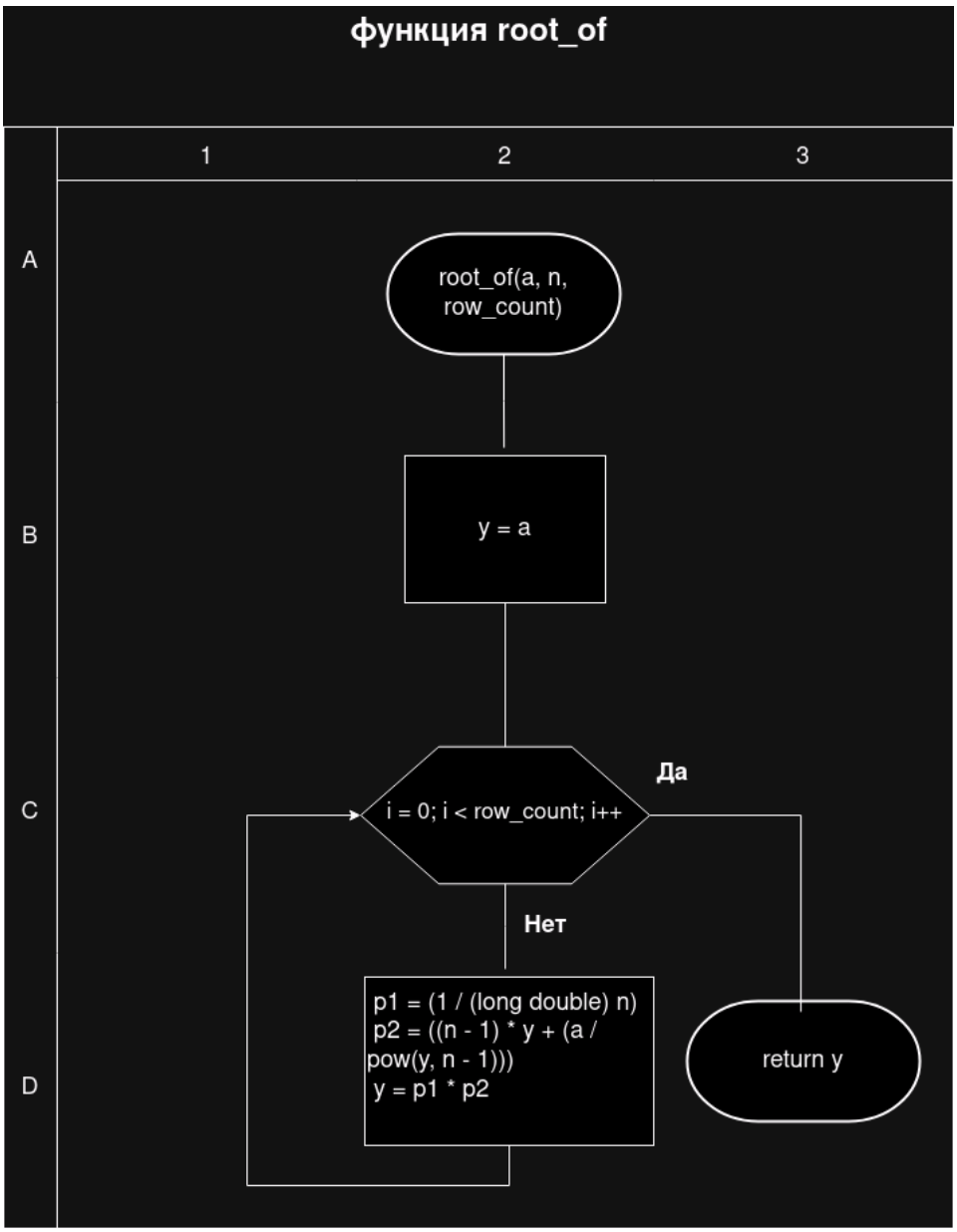


Рис. 2: Блок-схема алгоритма работы функции `root_of()` для программы `root`

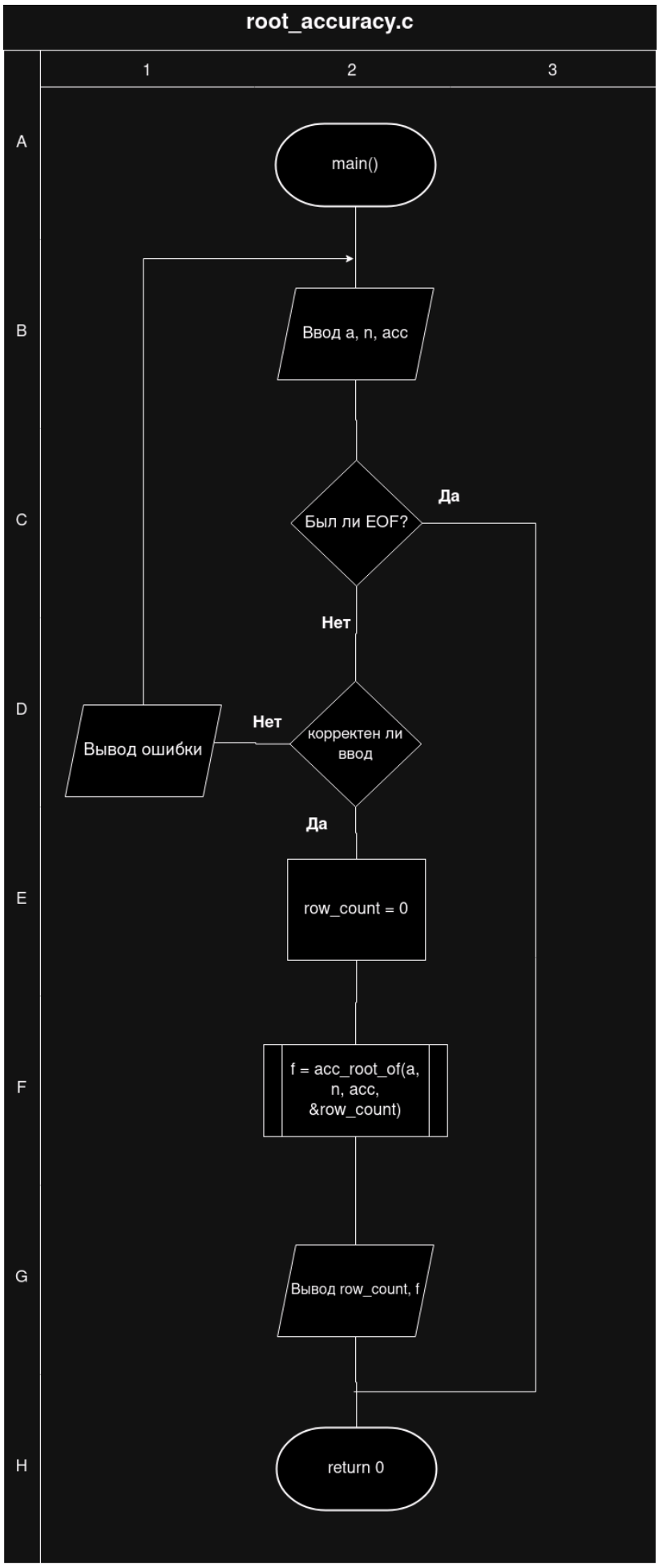
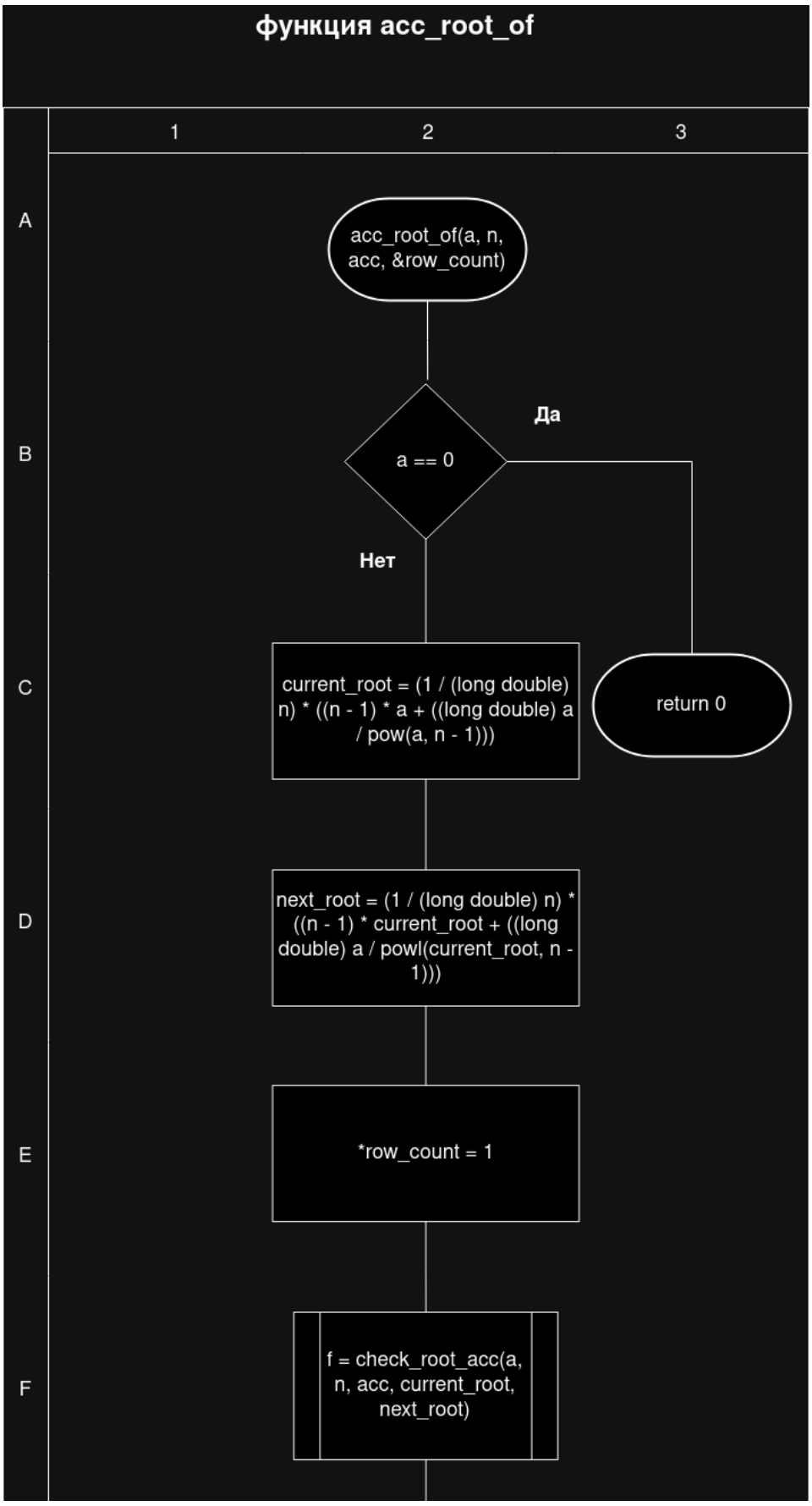


Рис. 3: Блок-схема алгоритма работы функции `main()` для программы `root_accuracy`


функция acc_root_of
1
2
3
A
B
C
D
E
F
acc_root_of(a, n, acc, &row_count)
a == 0
Да
Нет
current_root = (1 / (long double) n) * ((n - 1) * a + ((long double) a / pow(a, n - 1)))
return 0
next_root = (1 / (long double) n) * ((n - 1) * current_root + ((long double) a / powl(current_root, n - 1)))
*row_count = 1
f = check_root_acc(a, n, acc, current_root, next_root)

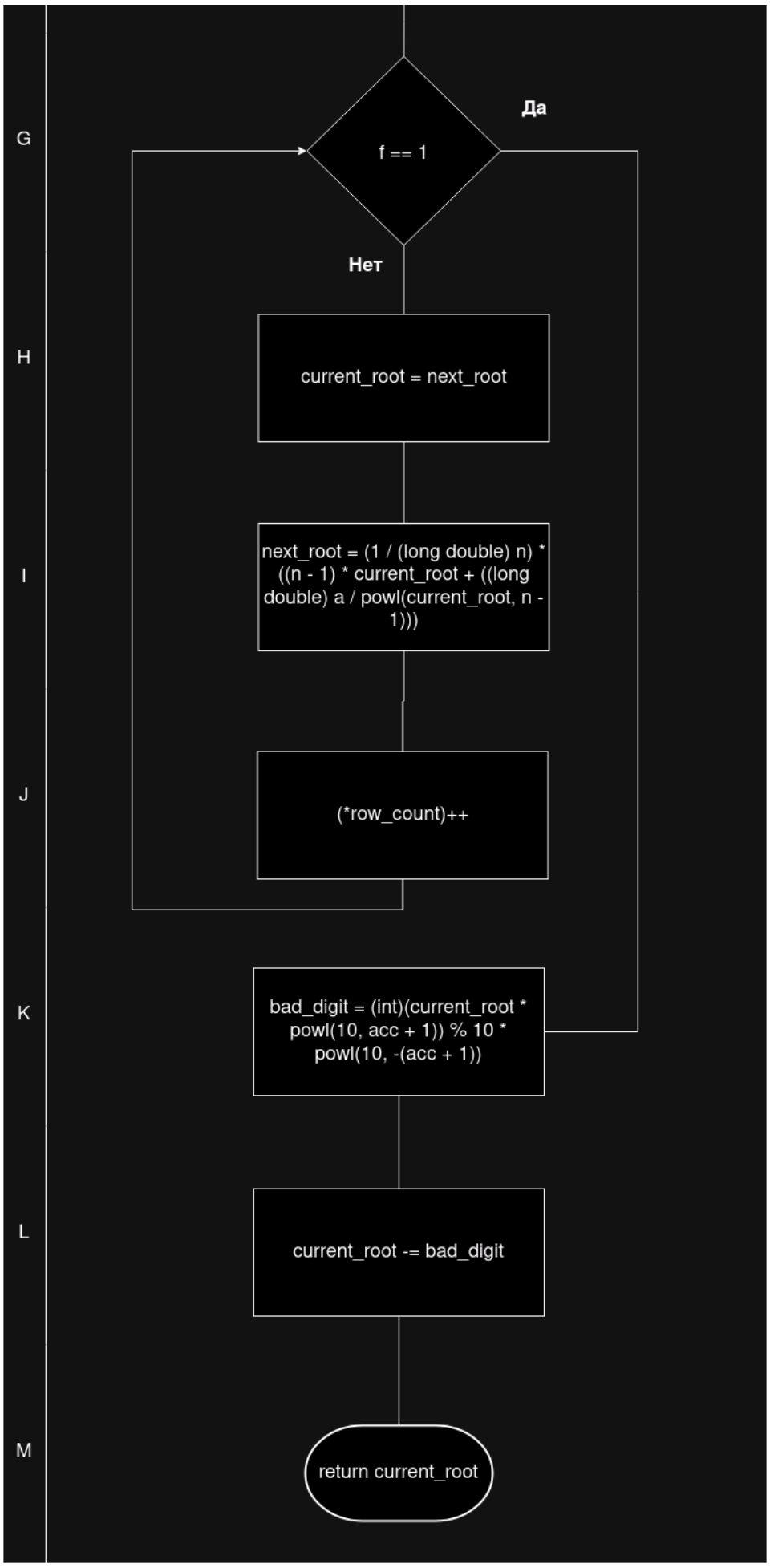


Рис. 4: Блок-схема алгоритма работы функции `acc_root_of()` для программы `root_accuracy`

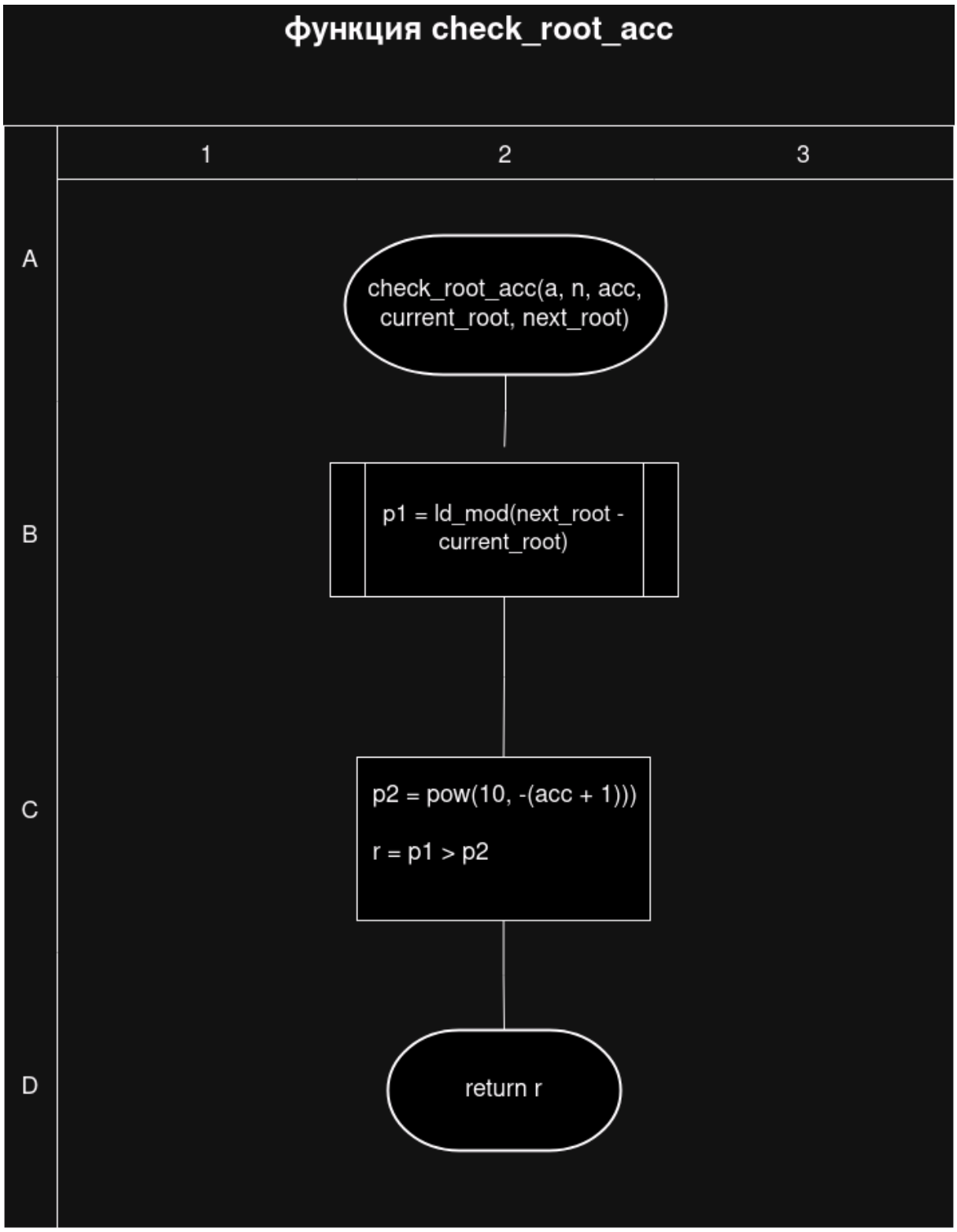


Рис. 5: Блок-схема алгоритма работы функции `check_root_acc()` для программы `root_accuracy`

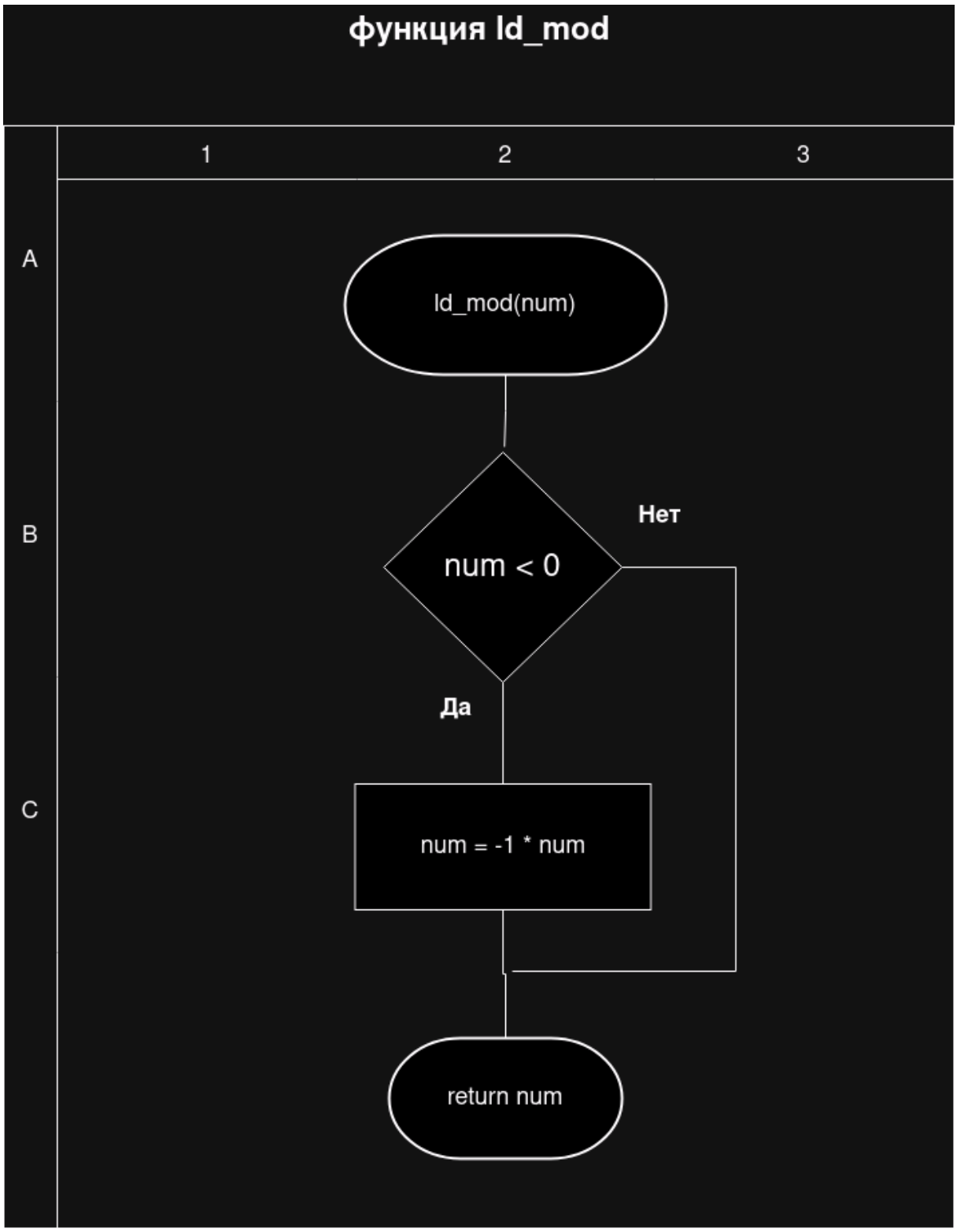


Рис. 6: Блок-схема алгоритма работы функции `ld_mod()` для программы `root_accuracy`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы `root` (файл: `root.c`)

```
#include <stdio.h>
#include <math.h>


int get_input(int *number) {

    int input_status = scanf("%d", number);

    if (input_status == EOF) {
        printf("\nEOF!\n");
        return 3;
    }

    while (getchar() != '\n');

    if (input_status != 1) {

        printf("Error: invalid input!\n");
        return 1;
    }

    if (*number < -1) {
        printf("Error: incorrect value!\n");
        return 2;
    }

    return 0;
}


long double root_of(int a, int n, int row_count) {

    long double y = a;

    for (int i = 0; i < row_count; i++) {

        y = (1 / (long double) n) * ((n - 1) * y + (a / pow(y, n - 1)));
    }

    return y;
}


int main() {

    int a = -2;
    int n = -2;
    int row_count = -2;

    int status = 0;
    for (int i = 0; i < 3; i++) {
        printf("Enter %d argument: ", i + 1);

        if (i == 0) status = get_input(&a);
        if (i == 1) status = get_input(&n);
        if (i == 2) status = get_input(&row_count);
```

```
57
58          if (status == 3) return 0;
59          if (status != 0) {
60              i--;
61              continue;
62          }
63      }
64
65      printf("%Lf\n", root_of(a, n, row_count));
66      printf("compare: %lf\n", pow(a, 1.0 / n));
67
68      return 0;
69  }
```

Листинг 2: Исходные коды программы root_accuracy (файл: root_accuracy.c)

```c
#include <stdio.h>
#include <math.h>


int get_input(int *number) {

    int input_status = scanf("%d", number);

    if (input_status == EOF) {
        printf("\nEOF!\n");
        return 3;
    }

    while (getchar() != '\n');

    if (input_status != 1) {

        printf("Error: invalid input!\n");
        return 1;
    }

    if (*number < -1) {
        printf("Error: incorrect value!\n");
        return 2;
    }

    return 0;
}


long double ld_mod(long double num) {

    return (num < 0) ? -num : num;
}


int check_root_acc(int a, int n, int acc, long double current_root, long double
    next_root) {

    return ld_mod(next_root - current_root) > pow(10, -(acc + 1));
}


long double acc_root_of(int a, int n, int acc, int *row_count) {

    if (a == 0) {
        return 0;
    }

    long double current_root = (1 / (long double) n) * ((n - 1) * a + ((long
        double) a / pow(a, n - 1)));
    long double next_root = (1 / (long double) n) * ((n - 1) * current_root + ((
        long double) a / powl(current_root, n - 1)));

    *row_count = 1;

    while (check_root_acc(a, n, acc, current_root, next_root)) {

        current_root = next_root;
```

```
57          next_root = (1 / (long double) n) * ((n - 1) * current_root + ((long
                double) a / powl(current_root, n - 1)));
58
59          (*row_count)++;
60      }
61
62      long double bad_digit = (int)(current_root * powl(10, acc + 1)) % 10 * powl
            (10, -(acc + 1));
63      current_root -= bad_digit;
64
65      return current_root;
66  }
67
68
69  int main() {
70
71      int a = -2;
72      int n = -2;
73      int acc = -2;
74
75      int status = 0;
76      for (int i = 0; i < 3; i++) {
77          printf("Enter %d argument: ", i + 1);
78
79          if (i == 0) status = get_input(&a);
80          if (i == 1) status = get_input(&n);
81          if (i == 2) status = get_input(&acc);
82
83          if (status == 3) return 0;
84          if (status != 0) {
85              i--;
86              continue;
87          }
88      }
89
90      int row_count = 0;
91
92      long double result = acc_root_of(a, n, acc, &row_count);
93
94      printf("Number of rows: %d\n", row_count);
95      printf("Result: %.*Lf\n", acc, result);
96      printf("compare: %.*lf\n", acc, pow(a, 1.0 / n));
97
98      return 0;
99  }
```

5. Описание тестовых примеров

Таблица 1: Тестовые примеры для программы `root`

Значение `a`	Значение `n`	Значение `row_count`	Ожидаемый вывод	Полученный вывод
20	2	5	4.472140 compare: 4.472136	4.472140 compare: 4.472136
9	2	4	3.000092 compare: 3.000000	3.000092 compare: 3.000000
4	4	4	1.524275 compare: 1.414214	1.524275 compare: 1.414214

Таблица 2: Тестовые примеры для программы `root_accuracy`

Значение `a`	Значение `n`	Значение `acc`	Ожидаемый вывод	Полученный вывод
30	2	5	Number of rows: 6 Result: 5.47722 compare: 5.47723	Number of rows: 6 Result: 5.47722 compare: 5.47723
9	2	4	Number of rows: 5 Result: 3.0000 compare: 3.0000	Number of rows: 5 Result: 3.0000 compare: 3.0000
4	4	4	Number of rows: 7 Result: 1.4142 compare: 1.4142	Number of rows: 7 Result: 1.4142 compare: 1.4142

6. Скриншоты

```
kurochkin.di@unix:~/inf/lab2$
kurochkin.di@unix:~/inf/lab2$ gcc -lm -o out root.c
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 20
Enter 2 argument: 2
Enter 3 argument: 5
4.472140
compare: 4.472136
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 9
Enter 2 argument: 2
Enter 3 argument: 4
3.000092
compare: 3.000000
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 4
Enter 2 argument: 4
Enter 3 argument: 4
1.524275
compare: 1.414214
```

Рис. 7: Сборка и запуск программы `root`

```
kurochkin.di@unix:~/inf/lab2$ gcc -lm -o out root_accuracy.c
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 30
Enter 2 argument: 2
Enter 3 argument: 5
Number of rows: 6
Result: 5.47722
compare: 5.47723
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 9
Enter 2 argument: 2
Enter 3 argument: 4
Number of rows: 5
Result: 3.0000
compare: 3.0000
kurochkin.di@unix:~/inf/lab2$ ./out
Enter 1 argument: 4
Enter 2 argument: 4
Enter 3 argument: 4
Number of rows: 7
Result: 1.4142
compare: 1.4142
```

Рис. 8: Сборка и запуск программы `root_accuracy`

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей вычисление значения функции $y = \sqrt[n]{a}$ в точке по данной итерационной формуле,
были рассмотрены базовые принципы построения программ на языке C и обработки чисел с плавающей точкой повышенной точности:

1. Использование функций стандартной библиотеки.
2. Возвращение нескольких значений из функции с помощью параметра.
3. Осуществление проверки корректности вводимых данных и, в случае ошибок, выдача соответствующих сообщений.
4. Оценка области сходимости ряда.
5. Оценка погрешности вычислений.