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: Горбатенко И. А.
Группа: С23-501
Преподаватель: Половнева Ю. А.

Москва — 2023

1. Формулировка индивидуального задания

Вариант №25. Необходимо спроектировать и реализовать на языке C две программы, позволяющие вычислять значения некоторой заданной функции. Первая программа должна принимать на вход значение аргумента `X` и количество членов ряда `N`, необходимое для проведения вычислений, вычислять значение функции с помощью суммы `N` членов ряда и выводить вычисленные значения. Вторая программа должна принимать на вход значение аргумента `X` и точность `EPS`, необходимую для проведения вычислений, вычислять значение суммы ряда с заданной точностью `EPS` и возвращать вычисленное значение, а также количество членов ряда, потребовавшееся для обеспечения заданной точности. При этом, в обеих программах должно осуществляться вычисление значения функции не только при помощи разложения в ряд, но и с использованием функций стандартной библиотеки.

$$e^x \cos(x) = \sum_{n=0}^{\infty} \frac{2^{\frac{n}{2}} \cos(\frac{\pi n}{4})}{n!} x^n$$

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовался встроенный тип данных `double`, предназначенный для работы с вещественными числами, а также встроенный тип данных `int`, предназначенный для работы с целыми числами.

3. Определение промежутков сходимости/расходимости

- При $|x| < \infty, \sum_{n=0}^{\infty} \frac{2^{\frac{n}{2}} \cos(\frac{\pi n}{4})}{n!} x^n$ ряд сходится.

Если ряд $A_n = \sum_{n=0}^{\infty} \frac{2^{\frac{n}{2}} \cos\frac{n\pi}{4}}{n!} x^n$ сходится, то выполняется:

$$\lim_{n \to \infty} \left| \frac{2^{n/2} \cos(\frac{\pi n}{4})}{n!} x^n \right| = 0 \quad <=> \quad \lim_{n \to \infty} \frac{2^{n/2} \left|\cos(\frac{\pi n}{4})\right|}{n!} |x|^n = 0$$

$$\frac{2^{(n+1)/2}}{(n+1)!} |x|^{n+1} : \frac{2^{n/2}}{n!} |x|^n = \frac{|x|\sqrt{2}}{n+1} < 0$$ Т.к. x = const, то $\lim \frac{2^{n/2}}{n!} |x|^n = 0$ и $0 < \left|\cos(\frac{\pi n}{4})\right| < 1$

Значит $\lim_{n \to \infty} \left| \frac{2^{n/2} \cos(\frac{\pi n}{4})}{n!} x^n \right| = 0$ т.е. ряд сходится при любом x

4. Упрощение данной формулы

Рассмотрим числитель n-го члена ряда:

1) Если ***n (mod 8) = 1*** или ***n (mod 8) = 7***

$2^{n/2} \cdot \frac{1}{\sqrt{2}}$ <=> $2^{(n-1)/2}$ <=> $2^{n//2}$

2) Если ***n (mod 8) = 3*** или ***n (mod 8) = 5***

$-2^{n/2} \cdot \frac{1}{\sqrt{2}}$ <=> $-2^{(n-1)/2}$ <=> $-2^{n//2}$

Так как n нечётно, (n-1)/2 <=> n//2

3) Если ***n (mod 8) = 2*** или ***n (mod 8) = 6***

$2^{n/2} \cdot 0$ <=> 0

4) Если ***n (mod 8) = 0***

$2^{n/2} \cdot 1$ <=> $2^{n//2}$

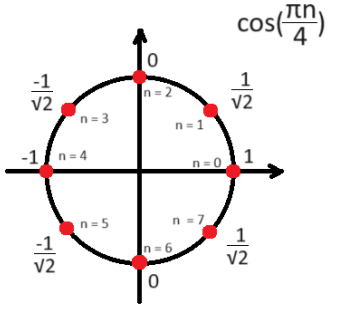
5) Если ***n (mod 8) = 4***

$2^{n/2} \cdot -1$ <=> $-2^{n//2}$

Так как n чётно n/2 <=> n//2

* где // - деление с округлением вниз

Числитель можно свести к $2^{n//2} \cdot k$ где k принимает значения {-1; 0; 1}, в зависимости от n



5. Описание использованного алгоритма

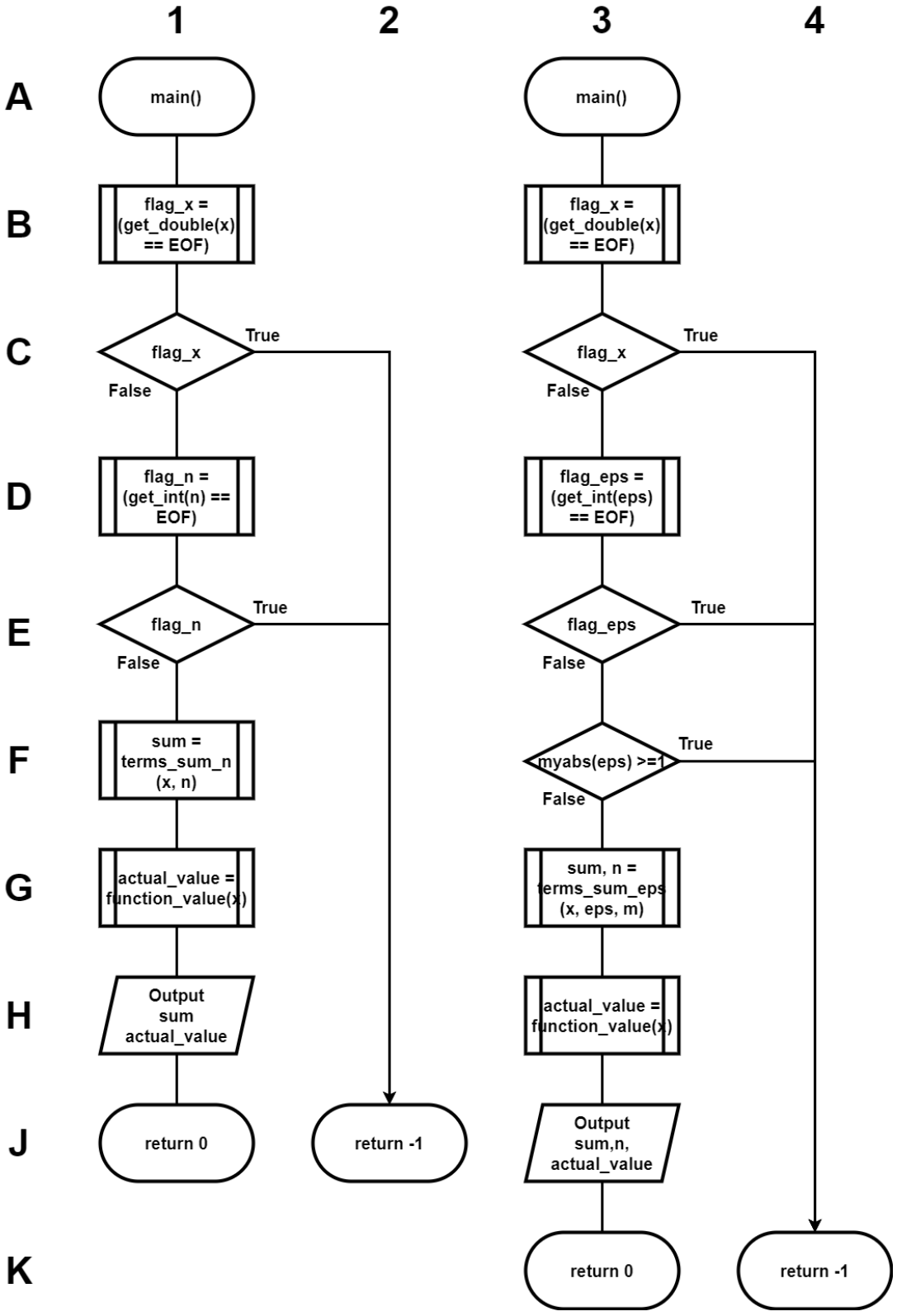


Рис. 1: Блок-схема алгоритма работы функции `main()` программы 1 (слева) и 2 (справа)

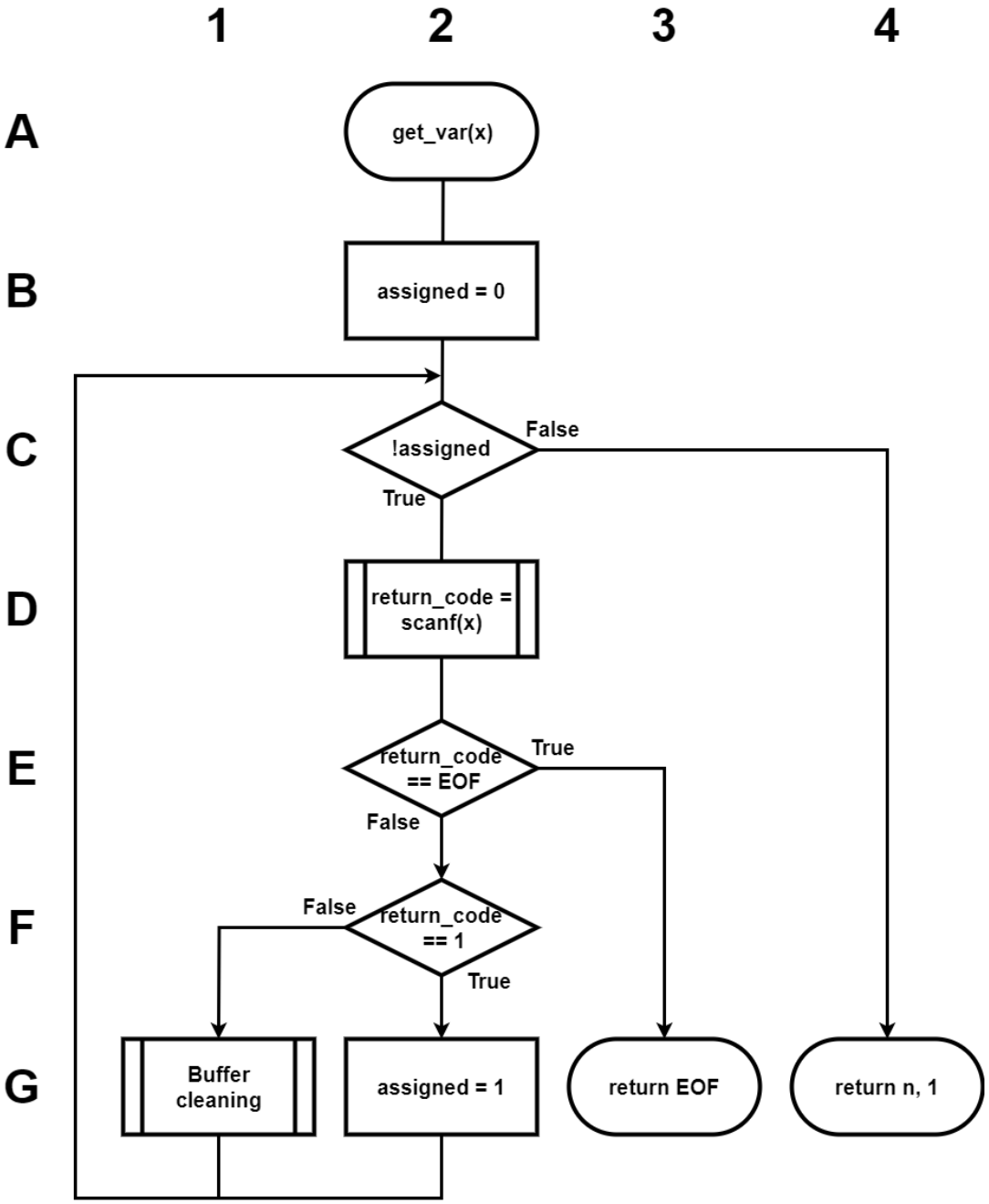


Рис. 2: Блок-схема алгоритма работы функции `get_var(x)`

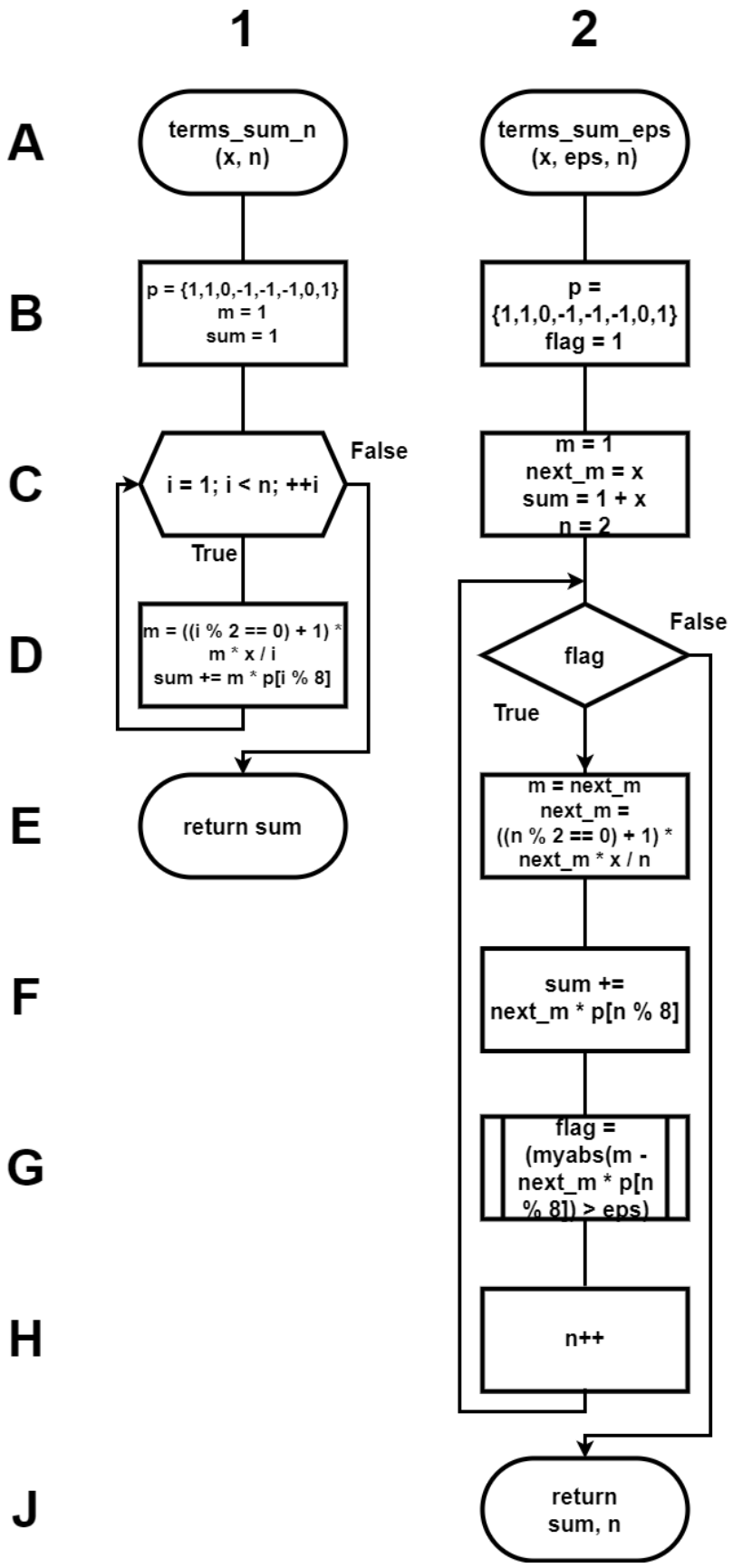


Рис. 3: Блок-схема алгоритма работы функции `terms_sum_n(x, n)` (слева) и функции `terms_sum_eps(x, eps, n)` (справа)

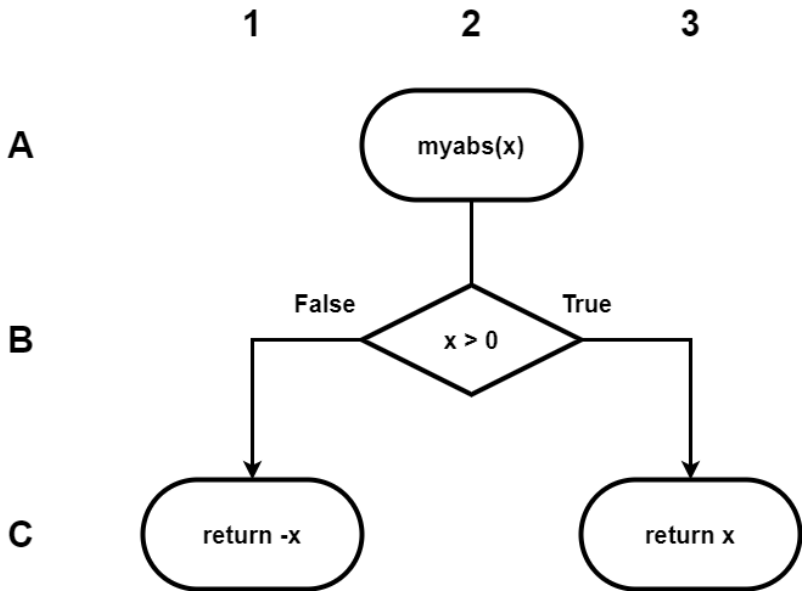


Рис. 4: Блок-схема алгоритма работы функции `myabs(x)`

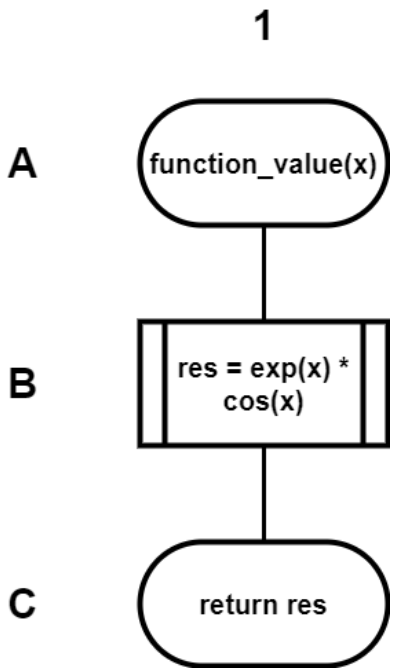


Рис. 5: Блок-схема алгоритма работы функции `function_value(x)`

6. Исходные коды разработанных программ

Листинг 1: Исходные коды программы `lab2_n` (файл: `lab2_n.c`)

```
#include ”mylib.c”

int
main() {
    int n = 0;
    double x = 0, sum = 0, actual_value = 0;

    printf(”Enter function parameter (X) and number of terms (N): ”);
    int flag = ((get_double(&x) == EOF) || (get_int(&n) == EOF));

    if (flag) {
        printf(”\nError: EOF!\n”);
        return -1;
    }

    if (n < 0) {
        printf(”Warning: You entered negative N. \
        Absolute value was taken.\n”);
        n = myabs(n);
    }

    sum = terms_sum_n(x, n);
    actual_value = function_value(x);

    printf(”Calculated value = %lf\n”, sum);
    printf(”Actual value = %lf\n”, actual_value);

    return 0;
}
```

Листинг 2: Исходные коды программы `lab2_eps` (файл: `lab2_eps.c`)

```
#include ”mylib.c”

int
main() {
    int n = 0;
    double x = 0, eps = 0, sum = 0, actual_value = 0;

    printf(”Enter function parameter (X) and precision (EPS): ”);
    int flag = ((get_double(&x) == EOF) || (get_double(&eps) == EOF));

    if (flag) {
        printf(”\nError: EOF!\n”);
        return -1;
    }

    if (myabs(eps) >= 1) {
        printf(”Error: EPS is greater than 1!\n”);
        return -1;
    }

    if (eps < 0) {
        printf(”Warning: You entered negative EPS. \
        Absolute value was taken.\n”);
        eps = myabs(eps);
    }
```

```
26
27      sum = terms_sum_eps(x, eps, &n);
28      actual_value = function_value(x);
29
30      printf(”Calculated value = %lf\n”, sum);
31      printf(”Terms summed = %d\n”, n);
32      printf(”Actual value = %lf\n”, actual_value);
33
34      return 0;
35  }
```

Листинг 3: Исходные коды программы `mylib` (файл: `mylib.c`)

```
 1  #include <stdio.h>
 2  #include <math.h>
 3
 4
 5  #define myabs(x) ((x) > 0 ? (x) : -(x))
 6
 7  double terms_sum_n(double x, int n);
 8  double terms_sum_eps(double x, double eps, int* n);
 9
10  double function_value(double x);
11  int get_double(double* x);
12  int get_int(int* x);
13
14
15  double
16  terms_sum_n(double x, int n) {
17      const int p[8] = {1, 1, 0, -1, -1, -1, 0, 1};
18      double m = 1, sum = 1;
19
20      for (int i = 1; i < n; ++i) {
21          m = ((i % 2 == 0) + 1) * m * x / i;
22          sum += m * p[i % 8];
23      }
24
25      return sum;
26  }
27
28
29  double
30  terms_sum_eps(double x, double eps, int* n) {
31      const int p[8] = {1, 1, 0, -1, -1, -1, 0, 1};
32
33      int flag = 1;
34      double m = 1, next_m = x, sum = 1 + x;
35      *n = 2;
36
37      while (flag) {
38          m = next_m;
39          next_m = ((*n % 2 == 0) + 1) * next_m * x / *n;
40
41          sum += next_m * p[(*n) % 8];
42          flag = (myabs(m - next_m * p[(*n) % 8]) > eps);
43          (*n)++;
44      }
45
46      return sum;
47  }
48
```

```
49
50 double
51 function_value(double x) {
52     return exp(x) * cos(x);
53 }
54
55
56 int
57 get_double(double* x) {
58     int assigned = 0;
59     while (!assigned) {
60         int input = scanf("%15lf", x);
61         scanf("%*[^ \t\n]");
62
63         if (input == EOF) {
64             return EOF;
65         } else if (input == 1) {
66             assigned = 1;
67         } else if (input == 0) {
68             printf("Error: incorrect input. Try again: ");
69             scanf("%*[^\n]");
70         }
71     }
72     return 1;
73 }
74
75
76 int
77 get_int(int* x) {
78     int assigned = 0;
79     while (!assigned) {
80         int input = scanf("%9d", x);
81         scanf("%*[^ \t\n]");
82
83         if (input == EOF) {
84             return EOF;
85         } else if (input == 1) {
86             assigned = 1;
87         } else if (input == 0) {
88             printf("Error: incorrect input. Try again: ");
89             scanf("%*[^\n]");
90         }
91     }
92     return 1;
93 }
```

7. Описание тестовых примеров

	Значение X	Значение N	Вычисленное значение	Точное значение
1	0.21183	122	1.208311824481737	1.208311824481737
2	0.845	151	1.545151700465733	1.545151700465733
3	-0.548	12	0.493451923507169	0.493451923413293
4	0.1458748	8	1.144762419011370	1.144762419094054
5	12	6	-12313.399999999999636	137341.275041607499588
6	1	192	1.468693939915885	1.468693939915885
7	-1	85	0.198766110346413	0.198766110346413
8	0.158851545421	1218151	1.157405910390012	1.157405910390012
9	0.985854	11184511	1.479820536555969	1.479820536555969
10	-0.5819785	974512	0.466801595537202	0.466801595537202
11	21156	124551	NaN	inf
12	0.5816945	125	1.494824567259316	1.494824567259316
13	49	121518	5733341891351591649 28.000000000000000	5733341580270095564 80.000000000000000
14	0.8741742666	754545	1.537920939341499	1.537920939341499
15	0	74	1	1

Рис. 6: Тестовые примеры для программы `lab2_n`

	Значение X	Значение EPS	Вычисленное значение N	Вычисленное значение	Точное значение
1	0.21183	0.0000000001	11	1.208311824481769	1.208311824481737
2	0.845	0.45494	5	1.558911008229167	1.545151700465733
3	-0.548	0.4843641212	4	0.506855530666667	0.493451923413293
4	0.1458748	0.0001	6	1.144762416780258	1.144762419094054
5	12	0.25612	25	92413.690707934903912	137341.275041607499588
6	1	0.151848	7	1.466666666666667	1.468693939915885
7	-1	11515	-	-	-
8	0.158851545421	-0.1515	4	1.157515401998139	1.157405910390012
9	0.985854	11818.15151	-	-	-
10	-0.5819785	0.0000011184	11	0.466801593650353	0.466801595537202
11	21156	0.145483997	111	NaN	inf
12	0.5816945	0.0000000000001	17	1.494824567259316	1.494824567259316
13	49	0.00001001	199	5733341891351591649 28.000000000000000	5733341580270095564 80.000000000000000
14	0.8741742666	0.000597845	9	1.537908006004943	1.537920939341499
15	0	0.000080856	3	1	1

Рис. 7: Тестовые примеры для программы `lab2_eps`

8. Скриншоты

```
[gorbatenko.ia@unix lab2]$ gcc lab2_eps.c -o lab2_eps -Wall -Wextra -Werror -ftrapv -lm -std=c99
[gorbatenko.ia@unix lab2]$ gcc lab2_n.c -o lab2_n -Wall -Wextra -Werror -ftrapv -lm -std=c99
[gorbatenko.ia@unix lab2]$ ./lab2_eps
Enter function parameter (X) and precision (EPS): 0.21183 0.0000000001
Calculated value = 1.208311824481769
Terms summed = 11
Actual value = 1.208311824481737
[gorbatenko.ia@unix lab2]$ ./lab2_n
Enter function parameter (X) and number of terms (N): 0.21183 11
Calculated value = 1.208311824481769
Actual value = 1.208311824481737
[gorbatenko.ia@unix lab2]$ ./lab2_eps
Enter function parameter (X) and precision (EPS): -0.5819785 0.0000011184
Calculated value = 0.466801593650353
Terms summed = 11
Actual value = 0.466801595537202
[gorbatenko.ia@unix lab2]$ ./lab2_n
Enter function parameter (X) and number of terms (N): -0.5819785 11
Calculated value = 0.466801593650353
Actual value = 0.466801595537202
[gorbatenko.ia@unix lab2]$ ./lab2_eps
Enter function parameter (X) and precision (EPS): 0.2545
155
Error: EPS is greater than 1!
[gorbatenko.ia@unix lab2]$ ./lab2_n
Enter function parameter (X) and number of terms (N): 12 -6
Warning: You entered negative N.         Absolute value was taken.
Calculated value = -12313.399999999999636
Actual value = 137341.275041607499588
[gorbatenko.ia@unix lab2]$ date
Thu Oct 19 03:15:00 AM MSK 2023
[gorbatenko.ia@unix lab2]$
```

Рис. 8: Сборка и запуск программы `lab2_n` и `lab2_eps`

9. Выводы

В ходе выполнения данной работы на примере программ `lab2_n` и `lab2_eps`, вычисляющих значения заданной функции, были рассмотрены базовые принципы построения программ на языке C и обработки вещественных чисел:

1. Объявление и использование переменных, а также указателей на переменные.
2. Использование циклов `while` и ветвлений `if`.
3. Организация ввода/вывода.
4. Реализация пользовательской функции ввода
5. Работа с функциями из встроенной библиотеки <math.h>
6. Разработка функций.
7. Реализация математических функций.
8. Выделение компонентов программ в отдельные файлы.