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: Гаврилов Д.А.
Группа: Б23-516
Преподаватель: Бабалова И.Ф.

Москва – 2023

1. Формулировка индивидуального задания

Необходимо спроектировать и реализовать на языке C **две** программы, позволяющие вычислять значения некоторой заданной функции.

Программа № 1 должна обеспечивать возможность вычисления значения функции при определённых значениях параметров, указанных пользователем. При этом, пользователь должен иметь возможность указать количество членов ряда, которое необходимо использовать при вычислениях.

Ключевым компонентом программы № 1 должна быть некоторая функция, на вход которой передаются значения параметров и количество членов ряда, необходимое для проведения вычислений. Возвращаемым значением для указанной функции должно быть вычисленное значение.

Программа № 2 должна обеспечивать возможность вычисления значения функции при определённых значениях параметров, указанных пользователем. При этом, пользователь должен иметь возможность указать точность, с которой должно быть вычислено значение функции.

Ключевым компонентом программы № 2 должна быть некоторая функция, на вход которой передаются значения параметров и точность, с которой необходимо вычислить результат. В качестве результата функция должна возвращать вычисленное значение и количество членов ряда, которое потребовалось для обеспечения заданной точности (данное значение необходимо вернуть через параметр).

При этом, в обеих программах должно осуществляться вычисление значения функции не только при помощи разложения в ряд, но и с использованием функций стандартной библиотеки.

Кроме того, необходимо научиться:

- оценивать область сходимости ряда — определять диапазон значений входных параметров, при которых ряд сходится;
- оценивать погрешность вычислений — определять количество верных цифр в записи результата.

Примечания:

1. Логически законченные части алгоритмов решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.
2. Программы должны осуществлять проверку корректности вводимых данных и, в случае ошибок, выдавать соответствующие сообщения, после чего продолжать работу.
3. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind (при тестировании и отладке программы ее необходимо запускать командой вида `valgrind ./lab2`, а при анализе производительности — `./lab2`).

Вариант №74

Задание

Вычислить значение функции в точке при помощи разложения в ряд:

$$\operatorname{arsh} x = \ln(x + \sqrt{x^2 + 1}) = x - \frac{x^3}{6} + \frac{3x^5}{40} - \dots = \sum_{n=0}^{\infty} \frac{(-1)^n (2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}$$

где $|x| < 1$.

Тип данных

Число с плавающей точкой двойной точности — double (спецификатор формата: `%lf`).

2. Описание использованных типов данных

При написании программы использовались типы данных int, short int, double и long long int.
Int - для работы с целыми числами.
Short int – для работы с целыми числами.
Long long int – для работы с целыми числами.
Double - для работы с числами с плавающей точкой.

3. Описание использованного алгоритма

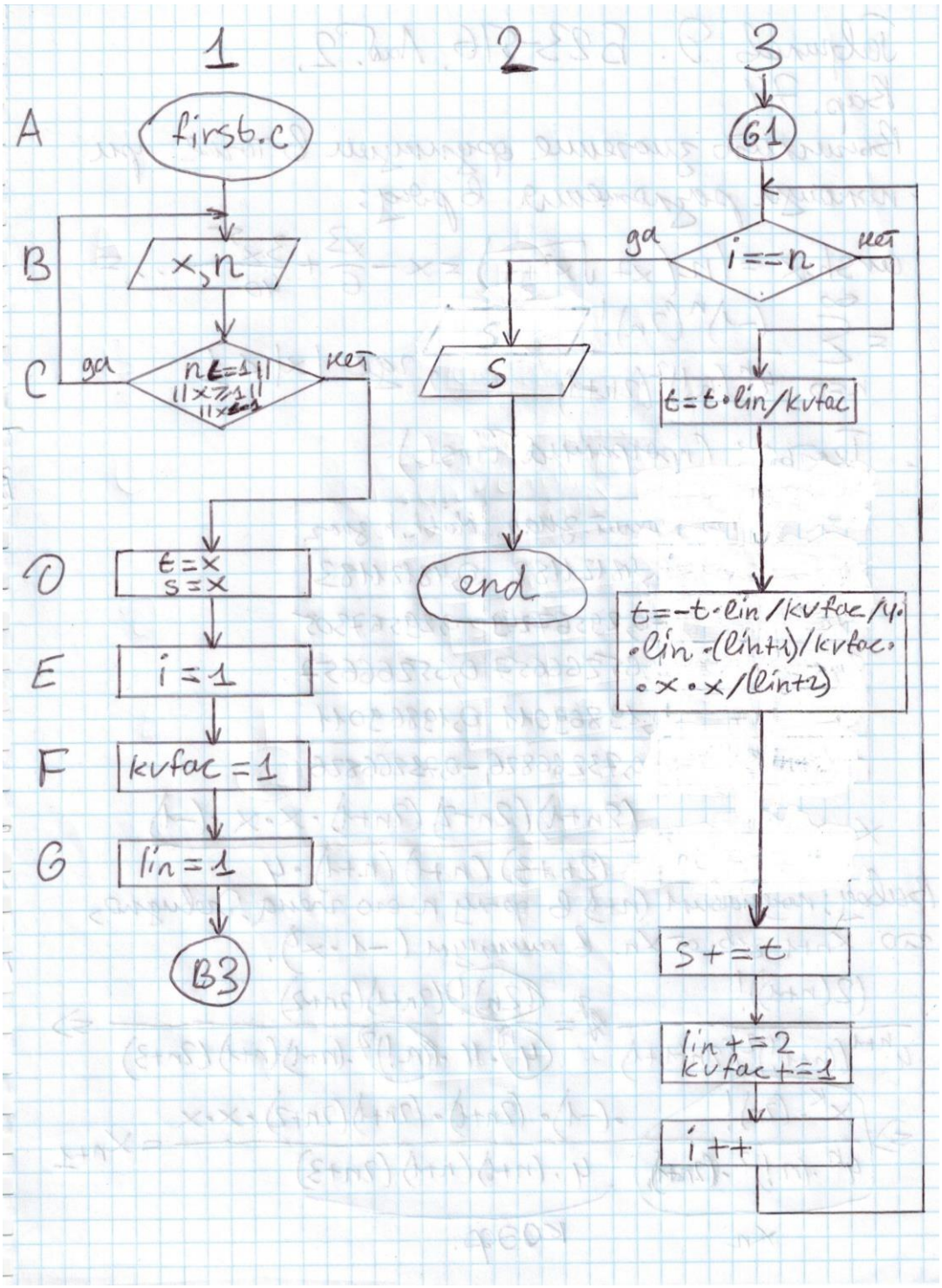

Рис. 1: Блок-схема алгоритма работы функции
`calcn(double x, int n)`

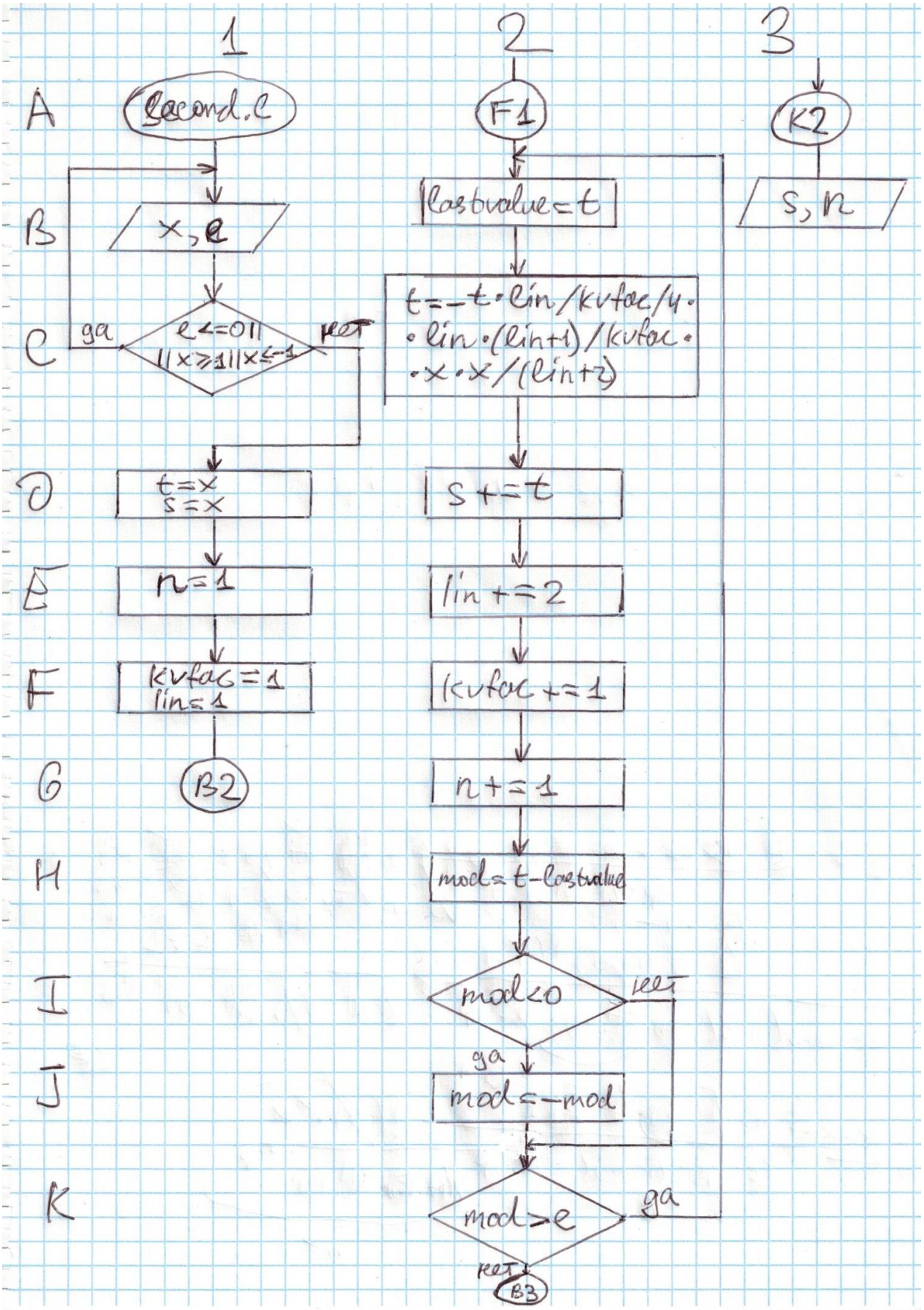

Рис. 2: Блок-схема алгоритма работы функции
`calcepls(double x, double e, int* n)`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы `reslab2`(файл: `lab2.c`)

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

double calcn(double x, int n) { // функция для n
    double s = x;  // x - первый член ряда, s - будущая сумма ряда
    double t = x;  // сохраняем x в переменную t
    int i = 1; // инициализируем счетчик количества членов ряда
    long long kvfac = 1;
    long long lin = 1;

    while(i != n){
        t = -t * lin / kvfac / 4 * lin * (lin + 1) /kvfac * x * x / (lin + 2);
         // выполняем преобразования с i-тым членом ряда теперь t - (i + 1)ый член ряда
        s += t; // суммируем
        lin += 2;
        kvfac += 1;
        i++;
    }

    return s; // возвращаем значение суммы
}

double calceps(double x, double e, int* n) { // функция для e
    double s = x;  // x - первый член ряда, s - будущая сумма ряда
    double t = x;  // сохраняем x в переменную t
    long long kvfac = 1;
    long long lin = 1;
    double lastvalue;

    do {
        lastvalue = t; // запоминаем значение предыдущего члена последовательности
        t = -t * lin / kvfac / 4 * lin * (lin + 1) /kvfac * x * x / (lin + 2);
        // выполняем преобразования с предыдущим членом, генерируем следующий
        s += t; // суммируем
        lin += 2;
        kvfac += 1;
        *n += 1; // обновляем количество членов ряда
    } while(fabs(t-lastvalue) > e);

    return s; // возвращаем значение суммы
}
```

```c

int main(){
   short flag, provd, cnt;
   do {
       printf("Enter <1> to count with n or enter <2> to count with e\n"); // выбор подсчёта ряда (n или e)
       provd = scanf("%hd", &flag);
       while (provd != 1 || ((flag != 1) && (flag != 2))){ // проверка для ввода 1/2
           scanf("%*[^\n]");
           printf("Please, enter correct value (1 or 2)\n");
           provd = scanf("%hd", &flag);
       }

       if (flag == 1){ // расчёт при заданных x и n
           double x;
           int n;
           printf("Enter x and n.\n");
           int r = scanf("%lf %d", &x, &n);
           while (r != 2 || fabs(x) >= 1 || n < 1){ // проверка входных данных
               scanf("%*[^\n]");
               if (r != 2) {
               printf("Incorrect input type.\n");
               }
               else if (fabs(x) >= 1) {
               printf("Please, enter x from -1 to 1.\n");
               }
               else if (n < 1) {
               printf("Please, enter correct count of row members.\n");
               }
               r = scanf("%lf %d", &x, &n);
           }
           printf("Calculated - %.15lf; Math.h value - %.15lf\n", calcn(x, n), log(x + sqrt(x * x + 1)));
       }

       else if (flag == 2){ // расчёт при заданных x и e
           double x;
           double e;
           int n = 1;
           printf("Enter x and e.\n");

           int r = scanf("%lf %lf", &x, &e);

           while (r != 2 || fabs(x) >= 1 || e <= 0){ // проверка входных данных
               scanf("%*[^\n]");
               if (r != 2) {
               printf("Incorrect input type.\n");
               }
               else if (fabs(x) >= 1) {
               printf("Please, enter x from -1 to 1.\n");
               }
               else if (e <= 0){
               printf("Please, enter correct epsilon value\n");
               }
               r = scanf("%lf %lf", &x, &e);
           }

           printf("Calculated - %.15lf; Math.h value - %.15lf\n", calceps(x, e, &n), log(x + sqrt(x * x + 1)));
           printf("Accuracy tooks %d row members\n", n);
       }

       printf("If you want to continue, please, enter <1> or another number to exit\n"); // продолжить/завершить работу
       scanf("%hd",&cnt);
       scanf("%*[^\n]");
   } while (cnt == 1);
   return 0;
}
```

5. Описание тестовых примеров проверки данных

Таблица 1: Тестовые примеры для calcn()

Значение `x,n`	Ожидаемое значение	Полученное значение
2,10	«Please, enter x from -1 to 1.»	«Please, enter x from -1 to 1»
0.5, -7	«Please, enter correct count of row members.»	«Please, enter correct count of row members.»
Dffr, 123	«Incorrect input type.»	«Incorrect input type.»
0.1, vbb	«Incorrect input type.»	«Incorrect input type.»
0.5, 30	0.4812118250596035	0.4812118250596035
0.2, 4	0.1986901103492414	0.1986900952380952

Таблица 2: Тестовые примеры для calceps()

Значение `x,e`	Ожидаемое значение	Полученное значение
4, 0.0001	«Please, enter x from -1 to 1.»	«Please, enter x from -1 to 1.»
0.3, -2	«Please, enter correct epsilon value.»	«Please, enter correct epsilon value.»
Trtr 0.0001	«Incorrect input type.»	«Incorrect input type.»
0.7 eoifjiov	«Incorrect input type.»	«Incorrect input type.»
0.3, 0.00001	0.2956730475634225	0.2956730449833467 «Accuracy tooks 6 row members»
0.9, 0.0000000000000001	0.8088669356527826	0.8088669356527829 «Accuracy tooks 139 row members»

6. Скриншоты к тестам

```
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
2 10
Please, enter x from -1 to 1.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
0.5 -7
Please, enter correct count of row members.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
Dffr 123
Incorrect input type.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
0.1 vbb
Incorrect input type.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
0.5 30
Calculated - 0.481211825059603; Math.h value - 0.481211825059603
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
1
Enter x and n.
0.2 4
Calculated - 0.198690095238095; Math.h value - 0.198690110349241
If you want to continue, please, enter <1> or another number to exit
0
[gavrilov.da@unix lab2]$
```

Рис. 3: Тестовые примеры для calcn()

```
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
4 0.0001
Please, enter x from -1 to 1.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.3 -2
Please, enter correct epsilon value
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
Trtr 0.0001
Incorrect input type.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.7 eoifjiov
Incorrect input type.
^C
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.3 0.00001
Calculated - 0.295673044983347; Math.h value - 0.295673047563422
Accuracy tooks 6 row members
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.9
0.0000000000000001
Calculated - 0.808866935652783; Math.h value - 0.808866935652783
Accuracy tooks 139 row members
If you want to continue, please, enter <1> or another number to exit
0
[gavrilov.da@unix lab2]$
```

Рис. 4: Тестовые примеры для calceps()

7. Проверка сходимости ряда

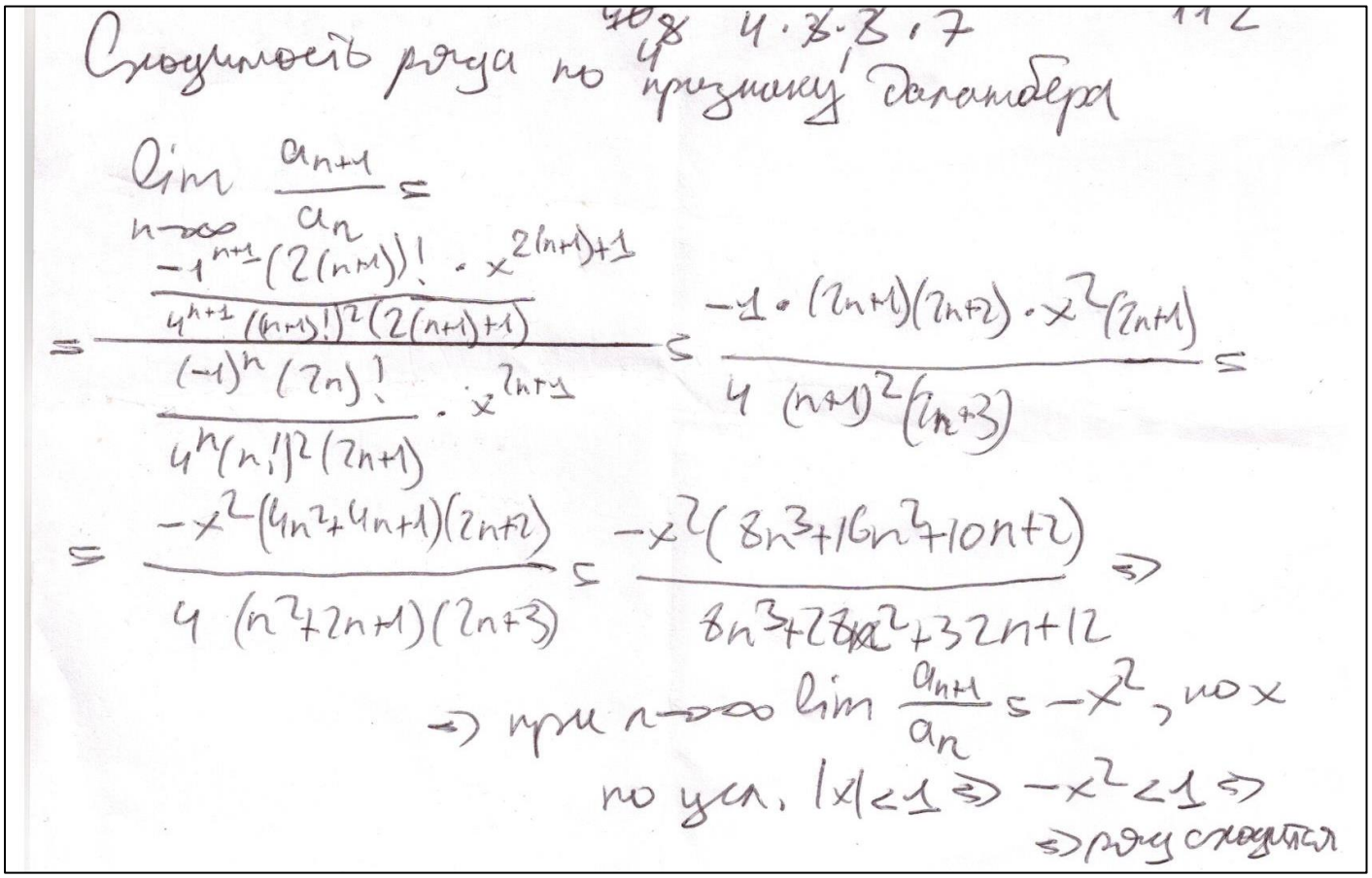

Рис. 5: Сходимость ряда

8. Подсчёт правильных цифр, дополнительные тесты

Для типа double максимальное количество совпадающих цифр числа, полученного разложением в ряд равно 16.

Определим оптимальную погрешность по формуле: $\varepsilon \leq 0.5 \cdot 10^{m-n+1}$

ε = 0.000000000000005

Приведу тесты с этой погрешностью:

```
[gavrilov.da@unix lab2]$ ./reslab2
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.01
0.00000000000000005
Calculated - 0.009999833340833; Math.h value - 0.009999833340833
Accuracy tooks 6 row members
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.2
0.00000000000000005
Calculated - 0.198690110349241; Math.h value - 0.198690110349241
Accuracy tooks 12 row members
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.5
0.00000000000000005
Calculated - 0.481211825059603; Math.h value - 0.481211825059603
Accuracy tooks 25 row members
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.8
0.00000000000000005
Calculated - 0.732668256045411; Math.h value - 0.732668256045411
Accuracy tooks 70 row members
If you want to continue, please, enter <1> or another number to exit
1
Enter <1> to count with n or enter <2> to count with e
2
Enter x and e.
0.99
0.00000000000000005
Calculated - 0.8742848121872 95; Math.h value - 0.8742848121872 95
Accuracy tooks 1305 row members
If you want to continue, please, enter <1> or another number to exit
0
[gavrilov.da@unix lab2]$
```

Рис. 6: Исследование с оптимальной погрешностью

Заметим, что чем больше x при фиксированной епсилон, тем больше итераций требуется для заданной точности. На примерах совпали все цифры. Программа calceps() способна достигать максимальную точность в 16 совпадающих знаков.

Найдем зависимость количества правильных цифр от погрешности e для фиксированного x:

Таблица 3:

x	e	Полученное значение	Значение math.h	Ответ
0.99	0.1	0.885752123830170	0.874284812187295	$0.8 \pm 0.5 * 10^{-1}$
0.99	0.001	0.874061493908388	0.874284812187295	$0.874 \pm 0.5 * 10^{-3}$
0.99	0.00001	0.874287227832333	0.874284812187295	$0.87428 \pm 0.5 * 10^{-5}$
0.99	0.000000001	0.874284812430832	0.874284812187295	$0.874284812 \pm 0.5 * 10^{-9}$
0.99	0.000000000000001	0.874284812187295	0.874284812187295	$0.874284812187295 \pm \pm 0.5 * 10^{-15}$
0.99	0.0000000000000001	0.874284812187295	0.874284812187295	$0.874284812187295 \pm \pm 0.5 * 10^{-15}$
-0.8	0.1	-0.729880380952381	-0.732668256045411	$-0.7 \pm 0.5 * 10^{-1}$
-0.8	0.0001	-0.732677079067873	-0.732668256045411	$-0.7326 \pm 0.5 * 10^{-4}$
-0.8	0.000000001	-0.732668256154602	-0.732668256045411	$-0.732668256 \pm 0.5 * 10^{-9}$
-0.8	0.000000000000001	-0.732668256045411	-0.732668256045411	$-0.732668256045411 \pm \pm 0.5 * 10^{-15}$
-0.8	0.0000000000000001	-0.732668256045411	-0.732668256045411	$-0.732668256045411 \pm \pm 0.5 * 10^{-15}$

Скриншоты: число 0.99 и разные e

e = 0.1

```
Enter x and e.
0.99
0.1
Calculated - 0.8857521238301705; Math.h value - 0.8742848121872949
Accuracy tooks 5 row members
If you want to continue, please, enter <1> or another number to exit
```

Совпало 2 цифры.

e = 0.001

```
Enter x and e.
0.99
0.001
Calculated - 0.8740614939083878; Math.h value - 0.8742848121872949
Accuracy tooks 42 row members
```

Совпало 4 цифры.

e = 0.00001

```
Enter x and e.
0.99
0.00001
Calculated - 0.8742872278323327; Math.h value - 0.8742848121872949
Accuracy tooks 165 row members
```

Совпали 6 цифр.

e = 0.000000001

```
Enter x and e.
0.99
0.000000001
Calculated - 0.8742848124308321; Math.h value - 0.8742848121872949
Accuracy tooks 535 row members
```

Совпали 10 цифр.

e = 0.000000000000001

```
Enter x and e.
0.99
0.000000000000001
Calculated - 0.874284812187295; Math.h value - 0.874284812187295
Accuracy tooks 1165 row members
```

Совпали 16 цифр.
При дальнейшем уменьшении модуля e лучшей точности мы не достигнем.

Например:
e = 0.0000000000000001

```
Enter x and e.
0.99
0.0000000000000001
Calculated - 0.874284812187295; Math.h value - 0.874284812187295
Accuracy tooks 1272 row members
```

Совпали всё те же 16 цифр.

Скриншоты: число -0.8 и разные e:

e = 0.1

```
Enter x and e.
-0.8
0.1
Calculated - -0.729880380952381; Math.h value - -0.732668256045411
Accuracy tooks 4 row members
```

Совпало 2 цифры.

e = 0.0001

```
Enter x and e.
-0.8
0.0001
Calculated - -0.732677079067873; Math.h value - -0.732668256045411
Accuracy tooks 13 row members
```

Совпало 5 цифр.

e = 0.000000001

```
Enter x and e.
-0.8
0.000000001
Calculated - -0.732668256154602; Math.h value - -0.732668256045411
Accuracy tooks 35 row members
```

Совпали 10 цифр.

e = 0.000000000000001

```
Enter x and e.
-0.8
0.000000000000001
Calculated - -0.732668256045411; Math.h value - -0.732668256045411
Accuracy tooks 64 row members
```

Совпали 16 цифр.
При дальнейшем уменьшении модуля e лучшей точности мы не достигнем.

Например:
e = 0.0000000000000001

```
Enter x and e.
-0.8
0.0000000000000001
Calculated - -0.732668256045411; Math.h value - -0.732668256045411
Accuracy tooks 69 row members
```

Совпали те же 16 цифр.

Найдем зависимость количества правильных цифр от n при фиксированном x:

Таблица 3:

x	n	Полученное значение	Значение math.h	Ответ
0.01	2	0.009999833333333	0.009999833340833	$0.0099998333 \pm 0.5*10^{-10}$
0.01	3	0.009999833340833	0.009999833340833	$0.009999833340833 \pm 0.5*10^{-15}$
0.9	20	0.808843728924502	0.808866935652783	$0.8088 \pm 0.5*10^{-4}$
0.9	50	0.808866925107077	0.808866935652783	$0.8088669 \pm 0.5*10^{-7}$
0.9	122	0.808866935652782	0.808866935652783	$0.80886693565278 \pm \pm 0.5*10^{-14}$
0.9	123	0.808866935652783	0.808866935652783	$0.808866935652783 \pm \pm 0.5*10^{-15}$
-0.5	3	-0.481510416666667	-0.481211825059603	$-0.481 \pm 0.5*10^{-3}$
-0.5	6	-0.481210060174675	-0.481211825059603	$-0.48121 \pm 0.5*10^{-5}$
-0.5	21	-0.481211825059604	-0.481211825059603	$-0.48121182505960 \pm \pm 0.5*10^{-14}$
-0.5	22	-0.481211825059603	-0.481211825059603	$-0.481211825059603 \pm \pm 0.5*10^{-15}$

Скриншоты: число x и разные n:

x = 0.01
n = 2

```
Enter x and n.
0.01
2
Calculated - 0.009999833333333; Math.h value - 0.009999833340833
```

Совпало 11 цифр.

n = 3 – это минимальное значение, при котором будет достигнута максимально возможная точность в 16 цифр

```
0.01
3
Calculated - 0.009999833340833; Math.h value - 0.009999833340833
```

x = 0.9

n = 20

```
Enter x and n.
0.9
20
Calculated - 0.808843728924502; Math.h value - 0.808866935652783
```

n = 50

```
0.9
50
Calculated - 0.808866925107077; Math.h value - 0.808866935652783
```

n = 122

```
Enter x and n.
0.9
122
Calculated - 0.808866935652782; Math.h value - 0.808866935652783
```

n = 123 – это минимальное значение, при котором будет достигнута максимально возможная точность в 16 цифр.

```
Enter x and n.
0.9
123
Calculated - 0.808866935652783; Math.h value - 0.808866935652783
```

x = -0.5

n = 3

```
Enter x and n.
-0.5
3
Calculated - -0.481510416666667; Math.h value - -0.481211825059603
```

n = 6

```
Enter x and n.
-0.5
6
Calculated - -0.481210060174675; Math.h value - -0.481211825059603
```

n = 21

```
Enter x and n.
-0.5
21
Calculated - -0.481211825059604; Math.h value - -0.481211825059603
```

n = 22 – это минимальное значение, при котором будет достигнута максимально возможная точность в 16 цифр.

```
Enter x and n.
-0.5
22
Calculated - -0.481211825059603; Math.h value - -0.481211825059603
```

9. Ошибки, возникшие в процессе работы

1. Ошибка в алгоритме разложения в ряд последовательности, выполнение лишних действий. Проблема решена после консультации с преподавателем и разработки более эффективного алгоритма.
2. Невозможность осуществления полноценной проверки ввода без char. Проблема решена с помощью упрощения алгоритма проверки, использования scanf().
3. Отсутствие работы функции очистки буфера fflush(stdin) на ОС Linux. Проблема решена с помощью использования другой функции scanf(«%*[^\n]»).

10. Выводы

В процессе выполнения этой лабораторной работы на примере программы, возвращающей значение функции путём разложения в ряд с заданной точностью (или заданным количеством членов) , были рассмотрены базовые принципы построения программ на языке С и обработки вещественных чисел:

1. Организация ввода / вывода, организация проверки ввода.
2. Создание диалога с пользователем, возможность выбора расчёта ряда (через n или e) и повторения программы без её перезапуска.
3. Выполнение арифметических операций над вещественными операндами.
4. Разработка функций, вычисляющих значение функции путём разложения в ряд с заданной точностью (заданным количеством членов ряда).
5. Использование указателей для изменения значения переменной по её адресу.