

Задание 6

Пусть дано: номер выбранной функции (например, в моём укороченном примере функция одна и это синус), точка x , точность eps , количество членов ряда N (всё вводит пользователь).

Расчет оценки значения выбранной функции выполняется либо до достижения заданной точности, либо до исчерпания введенного числа слагаемых.

Ход написания программы.

1. Ищем в Google формулу ☺

Находим вариант с последним слагаемым в виде расчётной формулы для любого слагаемого.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots = \sum_{n=1}^{\infty} \frac{x^n}{n!}, |x| < \infty,$$
$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + \frac{(-1)^{n+1} x^{2n-1}}{(2n-1)!} - \dots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^{2n-1}}{(2n-1)!}, |x| < \infty,$$
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + \frac{(-1)^{n+1} x^{2n}}{(2n)!} - \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}, |x| < \infty,$$

2. Далее проделываем действия по аналогии с пары по нормам векторов. Преобразуем формулу в код:

- Заводим цикл от 1 до N (у нас по условию задачи количество слагаемых ограничивается этим значением, а не бесконечностью);
- Аккуратно переносим формулу (в точности как она написана) с использованием математических функций из библиотеки `math`.

```
for (i = 1; i <= N; i++) {  
    f += pow(-1, i + 1) * pow(x, 2 * i - 1) / (float)Fact(2 * i - 1);  
    if (fabs(etalon - f) < eps) break;  
}
```

- Функции факториала в библиотеке нет, но мы реализовывали эту функцию в 4 строчки на одном из занятий (тема рекурсия). Добавляем данную функцию в нашу программу.

- Проверяем всё это пока в `main()`.

3. Оформляем код для вычисления синуса в отдельную функцию.

```
double Sin(double x, double etalon, double eps, int N) {  
    int i = 0;  
    double f = 0;  
    for (i = 1; i <= N; i++) {  
        f += pow(-1, i + 1) * pow(x, 2 * i - 1) / (float)Fact(2 * i - 1);  
        if (fabs(etalon - f) < eps) break;  
    }  
    return f;  
}
```

4. Проверяем, что это всё работает.

```
int main() {  
    float x, eps;  
    int N;
```

```

N = 100;
x = 1.5;           // пусть ввод для таких функций в радианах, а не в градусах
eps = 0.000001;

double etalon = sin(x);
double res = Sin(x, etalon, eps, N);

printf("True value: %.12lf\nf(x) = %.12lf\n", etalon, res);

system("pause");
return 0;
}

```

5. Обращаемся к постановке задачи

По результатам расчета программа должна вывести следующую информацию:

- эталонное значение (полученное с использованием встроенных математических функций языка программирования C),
- вычисленную оценку значения функции,
- разницу между оценкой и эталонным значением,
- количество слагаемых, которое было вычислено.

То есть необходимо возвращать из функции не только значение, но и погрешность с количеством слагаемых.

Тут возникает необходимость обратиться к теме Структуры. Создадим структуру для хранения такого ответа:

```

struct Answer {
    double f;
    double error;
    int count;
};

```

По факту мы определили свой тип данных. Тип данных Answer. Чтобы обратиться к полям структуры используется символ точка.

```

struct Answer Sin(double x, double etalon, double eps, int N) { // возвращаем тип данных
    struct Answer
    {
        int i = 0;
        struct Answer result;
        result.f = 0;           // работаем с полями ответа как с отдельными переменными
        for (i = 1; i <= N; i++) {
            result.f += pow(-1, i + 1) * pow(x, 2 * i - 1) / (float)Fact(2 * i - 1);
            if (fabs(etalon - result.f) < eps) break;
        }
        result.error = fabs(etalon - result.f); // вычисляем и записываем ошибку
        result.count = i;                     // сохраняем количество слагаемых, которое
        использовалось при расчёте
        return result;
    }
}

```

Также вносим изменения в main().

6. Окончательно имеем:

```

#include <stdio.h>
#include <stdlib.h>

```

```

#include <math.h>

#define PI 3.1415926535897931

int Fact(int x) {
    if (x < 0) return 0;
    if (x == 1) return 1;
    return x * Fact(x - 1);
}

struct Answer Sin(double x, double etalon, double eps, int N) {
    int i = 0;
    struct Answer result;
    result.f = 0;
    for (i = 1; i <= N; i++) {
        result.f += pow(-1, i + 1) * pow(x, 2 * i - 1) / (float)Fact(2 * i - 1);
        if (fabs(etalon - result.f) < eps) break;
    }
    result.error = fabs(etalon - result.f);
    result.count = i;
    return result;
}

int main() {
    float x, eps;
    int N;

    N = 100;
    x = 1.5;           // пусть ввод для таких функций в радианах, а не в градусах
    eps = 0.000001;

    double etalon = sin(x);
    struct Answer res = Sin(x, etalon, eps, N);

    printf("True value: %.12lf\nf(x) = %.12lf\nError: %.12lf\nCount of elements: %d\n",
    etalon, res.f, res.error, res.count);

    system("pause");
    return 0;
}

```