# Лабораторная работа №3

# Рекуррентные вычисления, вложенные циклы

## Цель работы.

Практика в организации итерационных и арифметических циклов, организация рекуррентных вычислений.

## 1. Теоретические сведения.

### 1.1. Разложение функции в степенной ряд

Действительная функция f(x) называется аналитической в точке , если в некоторой окрестности x-<R этой точки функция разлагается в степенной ряд (ряд Тейлора):

 (1)

При =0 получаем ряд Маклорена:

 (2)

Разность  (3)

называется остаточным членом и представляет собой ошибку при замене функции f(x) полиномом Тейлора.

Для ряда Маклорена

 где 0<<1. (4)

Таким образом, вычисление значения функции можно свести к вычислению суммы числового ряда

а1+а2+ . . . +an+ . . . . (5)

Известно, что числовой ряд называется сходящимся, если существует предел последовательности его частных сумм:

, (6)

где Sn= а1+а2+ . . . +an+ . . . .

Число S называется суммой ряда.

Из формулы (13) получаем S=Sn+Rn ,

где Rn - остаток ряда, причем R0 при n.

Для нахождения суммы S сходящегося ряда (5) с заданной точностью  нужно выбрать число слагаемых n столь большим, чтобы имело место неравенство

Rn<.

Тогда частная сумма Sn приближенно может быть принята за точную сумму S ряда (5).

Приближенно n выбрать так, чтобы имело место неравенство

Sn+1-Sn< или an<.

### 1.2. Сложность алгоритма

**Сложность алгоритма** — это количественная характеристика, которая показывает сколько времени или сколько памяти требуется для выполнения алгоритма. Но при выполнении алгоритма на компьютере время его выполнения будет зависеть от характеристик компьютера. Поэтому принято измерять сложность алгоритма в количестве операций, которые нужно совершить для выполнения алгоритма. При этом можно вычислять минимальное количество операций, среднее количество операций, максимальное количество операций. Для практических задач важным показателем является максимальное количество операций - худший случай. Для оценки сложности используется обозначение О(), которое называется “нотация О большое” или О-нотация, которое показывает как сложность задачи, т.е. количество выполненных операций, будет расти с ростом входных данных. Примеры классов сложности алгоритмов приведены в таблице 1.

*Таблица 1 - Классы сложности алгоритмов*

|  |  |
| --- | --- |
| Обозначение | Классы сложности |
| О(с) | Константная сложность, т.е. количество операций не будет меняться при увеличении размера входных данных. Идеальный вариант -  О(1), т.е. задача решается за одну операцию, например, получение элемента по индексу |
| O( log n) | Логарифмическая сложность, т.е. количество операций с ростом размера входных данных увеличивается логарифмически. Примером такой задачи является бинарный поиск элемента в массиве. |
| O(n) | Линейная сложность, т.е. при увеличении размера входных данных количество операций будет зависеть от размера данных. Например, если выполняется поиск элемента в массиве перебором, в худшем случае нужно будет перебрать все элементы массива, соответственно, чем больше элементов, тем больше операций нужно будет выполнить. |
| O(nc) | Полиномиальная сложность, т.е.  в задаче используются вложенные циклы. Если циклов два, то сложность алгоритма считается равной О(n2), если циклов три, то О(n3) и т. д.  Примером таких задач являются простые сортировки массивов. |
| O(2n) | Экспоненциальная сложность, количество операций при увеличении количества данных растет экспоненциально. Примером является рекурсивное вычисление чисел Фибоначчи |
| O(n!) | Факториальная сложность, количество операций растет пропорционально факториалу от размера входных данных. Примером является формирование всех перестановок элементов массива. |

Понятие сложности алгоритма нужно для разработки эффективных алгоритмов, т.е. чем меньше количество операций нужно выполнить при увеличении количества входных данных, тем более эффективным является алгоритм.

## 2. Выполнение задания

Задача сводится к замене функции степенным рядом и нахождению суммы некоторого количества слагаемых при различных параметрах суммирования х . Каждое слагаемое суммы зависит от параметра х и номера n, определяющего место этого слагаемого в сумме.

Обычно формула общего члена суммы принадлежит одному из следующих трех типов:

а) ; ; ;

б); ; ;

в); ; .

В случае а) для вычисления члена суммы аn целесообразно использовать **рекуррентные соотношения**, т. е. выражать последующий член суммы через предыдущий: an+1=(x, n)an. Это позволит существенно сократить объем вычислительной работы. Кроме того, вычисление члена суммы по общей формуле в ряде случаев невозможно (например, из-за наличия n!).

В случае б) применение рекуррентных соотношений нецелесообразно. Вычисления будут наиболее эффективными, если каждый член суммы вычислять по общей формуле an=(x, n).

В случае в) член суммы целесообразно представить в виде двух сомножителей, один из которых вычисляется по рекуррентному соотношению, а другой непосредственно an=(x, n)\*сn(x,n), где сn=cn-1(x,n).

## 3. Постановка задачи.

Для х изменяющегося от a до b с шагом (b-a)/k, где (k=10), вычислить функцию f(x), используя ее разложение в степенной ряд в двух случаях:

а) для заданного n;

б) для заданной точности  (=0.0001).

Для сравнения найти точное значение функции.

## 3. Методические указания.

1. Алгоритм решения задачи сводится к трем циклам, причем два из них вложены в третий. Внутренние циклы суммируют слагаемые при фиксированном параметре x, один (арифметический для заданного n), другой (итерационный для заданной точности . При организации этих циклов следует обратить внимание на **правильный выбор формулы** для вычисления элемента ряда an и правильное присвоение начальных значений переменным цикла.
2. Внешний цикл организует изменение параметра х.
3. Результаты расчетов отпечатать в следующем виде:

Вычисление функции

X=...... SN=...... SE=..... Y=......

X=...... SN=...... SE=..... Y=......

..........

X=...... SN=...... SE=..... Y=......

Здесь X- значение параметра;

SN- значение суммы для заданного n;

SE- значение суммы для заданной точности;

Y-точное значение функции.

3. Если степень можно вычислить рекуррентно, то функция Math.Pow() не используется.

## 4. Варианты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | функция | Диапазон  Изменения аргумента | n | сумма |
| 1 |  |  | 10 |  |
| 2 |  |  | 40 |  |
| 3 |  |  | 10 |  |
| 4 |  |  | 10 |  |
| 5 |  |  | 15 |  |
| 6 |  |  | 25 |  |
| 7 |  |  | 10 |  |
| 8 |  |  | 40 |  |
| 9 |  |  | 30 |  |
| 10 |  |  | 20 |  |
| 11 |  |  | 10 |  |
| 12 |  |  | 35 |  |
| 13 |  |  | 10 |  |
| 14 |  |  | 20 |  |
| 15 |  |  | 30 |  |
| 16 |  |  | 40 |  |
| 17 |  |  | 10 |  |
| 18 |  |  | 50 |  |
| 19 |  |  | 20 |  |
| 20 |  |  | 30 |  |
| 21 |  |  | 40 |  |
| 22 |  |  | 35 |  |
| 23 |  |  | 15 |  |
| 24 |  |  | 40 |  |
| 25 |  |  | 20 |  |
| 26 |  |  | 25 |  |
| 27 |  |  | 10 |  |
| 28 |  |  | 40 |  |
| 29 |  |  | 3 |  |
| 30 |  |  | 20 |  |

## 5. Содержание отчета:

1. Постановка задачи (общая и конкретного варианта).
2. Анализ задачи (пояснить к какому типу относится член ряда, вычисление рекуррентного соотношения).
3. Алгоритм программы в виде блок-схемы.
4. Текст программы.
5. Результаты работы программы (10 точек, для каждой 3 результата: y, SN, SE) и объяснение результатов.

## 6. Примерные вопросы для защиты лабораторной работы

1. Что такое сложность программы?
2. Как оценить сложность программы?
3. Зачем использовать рекуррентное соотношение?
4. Как вычислить рекуррентное соотношение?
5. За счет чего уменьшается сложность при использовании рекуррентного соотношения?
6. Какие циклы лучше использовать для вычисления числового ряда?
7. Как вычислить сумму числового ряда с заданной точностью?
8. Как определить нужно ли использовать рекуррентное соотношение при вычислении числового ряда?

## 7. Критерии оценки выполнения программы

|  |  |  |
| --- | --- | --- |
| № | Критерий | Баллы |
| 1 | Решение задачи без использования рекуррентного соотношения[[1]](#footnote-1)(выбор типа общего члена суммы, блок-схема, программа, результаты). | 4 |
| 2. | Решение задачи с рекуррентным соотношением[[2]](#footnote-2) | 6 |
| 3 | Даны полные ответы на теоретические вопросы (п.6) | 1 |
| 4 | Оформление программы с учетом стайл-гайда | 1 |

1. общий член суммы относится к типу 1 или типу 3 [↑](#footnote-ref-1)
2. общий член суммы относится к типу 1 или типу 3, для задачи с общим членом суммы типа 2 рекуррентное соотношение не требуется [↑](#footnote-ref-2)