

Основы алгоритмизации

1.Подготовка и решение задач

1.1.Этапы подготовки и решения задач на ЭВМ

2. Основы алгоритмизации

2.1.Алгоритмические структуры

1.2.1. Графический способ представления алгоритмов

1.2.2. Базовые алгоритмические структуры

1.2.3. Итерационные циклы

1.2.4. Вложенные циклы

2.2.Разработка алгоритмов решения задач

2.2.1.Разработка алгоритмов линейной структуры

2.2.2.Разработка алгоритмов разветвляющейся структуры

2.2.3.Разработка алгоритмов циклической структуры



1.Подготовка и решение задач. Основы алгоритмизации

1.1.Этапы подготовки и решения задач на ЭВМ

Задачи, решаемые на ЭВМ, делятся на расчетные и информационные. Технология решения включает 5 этапов.

1 этап. Постановка задачи

Постановка задачи - это неформализованное, содержательное описание проблемы и возможных способов действий. При постановке задачи необходимо:

1. Сформулировать цель решения задачи;
2. Определить критерий эффективности;
3. Описать форму и содержание вх. и вых. информации;
4. Определить допустимую продолжительность решения;
5. Определить ожидаемую продолжительность;
6. Определить связь с др. задачами;
7. Определить соответствие требованиям, предъявляемым к системе.

Для разработки постановки задачи привлекаются различные специалисты.

2 этап. Построение математической модели

М.М.- это формализованное математическое описание процесса обработки информации. При построении М.М. необходимо:

- определить ограничения на действующие факторы;
- определить и обозначить исходные данные и результаты;
- произвести запись математических соотношений, связывающих исходные данные с результатом.

3 этап. Разработка схемы алгоритма

Определение. Алгоритм - это точное предписание, определяющее процесс перехода от исходных данных к исходному результату.

При разработке схемы алгоритма необходимо:

- определить порядок ввода исх. данных;
- построить логическую последовательность процесса обработки информации;
- определить порядок вывода результатов решения задачи.

4 этап. Составление программы

Программа - это запись алгоритма на языке, понятном ЭВМ.

При разработке программы необходимо:

- произвести идентификацию исходных данных, массивов, подпрограмм, стандартных процедур и функций;
- описать порядок ввода исходных данных, обработки информации, вывода результатов решения задачи;

5 этап. Решение задачи

На данном этапе необходимо:

- произвести отладку программы;
- выполнить программу и получить результат решения;
- произвести анализ и оценку полученных результатов;
- оформить эксплуатационную документацию

Примечание: В летных отрядах анализ и оценка результатов решения задачи производится летным или диспетчерским составом.

1.2.Алгоритмические структуры

1.2.1. Графический способ представления алгоритмов

Наиболее удобный способ описания алгоритмов - с помощью схем.

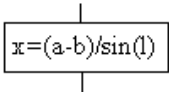
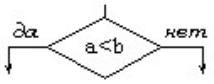
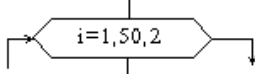
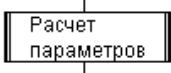
Схема алгоритма - это наглядное графическое представление алгоритма, дополненное элементами словесной записи.

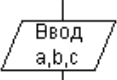
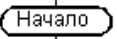
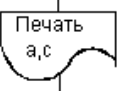
Каждый пункт алгоритма отображается на схеме не-которой геометрической фигурой (блоками), причем , различным по типу выполняе-мых действий, соответствуют различные геометрические фигуры, изображаемые по ГОСТу. Графические символы на схемах соединяются линиями потока информации.

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным.

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Такое графическое представление называется схемой алгоритма или **блок-схемой**. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде **блочного символа**. Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий. В таблице приведены наиболее часто употребляемые символы.

Название блока	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде

		
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		

1.2.2. Базовые алгоритмические структуры

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных **базовых** (т.е. основных) **элементов**. Естественно, что при таком подходе к алгоритмам изучение основных принципов их конструирования должно начинаться с изучения этих базовых элементов. Для их описания будем использовать язык схем алгоритмов и Описание алгоритма.

Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: **последовательность, разветвление, цикл.**

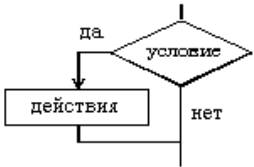
Характерной особенностью базовых структур является наличие в них **одного входа и одного выхода.**

1. Базовая структура "последовательность". Образуется последовательностью действий, следующих одно за другим:

Описание алгоритма	Язык блок-схем
действие 1 действие 2 действие n	

2. Базовая структура "разветвление". Обеспечивает в зависимости от результата проверки условия (**да** или **нет**) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к **общему выходу**, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран. Структура **разветвление** существует в четырех основных вариантах:

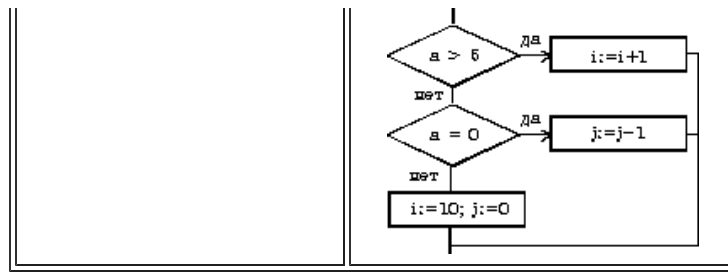
- если—то;
- если—то—иначе;
- выбор;
- выбор—иначе.

Описание алгоритма	Язык блок-схем
1. если—то	
если условие то действия все	
2. если—то—иначе	

если условие то действия 1 иначе действия 2 все	
3. выбор	
выбор при условии 1: действия 1 при условии 2: действия 2 при условии N: действия N все	
4. выбор—иначе	
выбор при условии 1: действия 1 при условии 2: действия 2 при условии N: действия N иначе действия N+1 все	

Примеры структуры **разветвление**

Описание алгоритма	Язык блок-схем
если $x > 0$ то $y := \sin(x)$ все	
если $a > b$ то $a := 2*a; b := 1$ иначе $b := 2*b$ все	
выбор при $n = 1$: $y := \sin(x)$ при $n = 2$: $y := \cos(x)$ при $n = 3$: $y := 0$ все	
выбор при $a > 5$: $i := i+1$ при $a = 0$: $j := j+1$ иначе $i := 10; j := 0$ все	



3. Базовая структура "цикл". Обеспечивает многократное выполнение некоторой совокупности действий, которая называется **телом цикла**. Основные разновидности циклов представлены в таблице:

Описание алгоритма	Язык блок-схем
Цикл типа пока (с предусловием). Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока.	
нц пока условие тело цикла (последовательность действий) кц	
Цикл типа для (с параметром). Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.	
нц для i от i1 до i2 тело цикла (последовательность действий) кц	

Примеры структуры **ЦИКЛ**

Описание алгоритма	Язык блок-схем
нц пока i <= 5 S := S+A[i] i := i+1 кц	
нц для i от 1 до 5 X[i] := i*i*i Y[i] := X[i]/2 кц	

1.2.3. Итерационные циклы

Особенностью итерационного цикла является то, что число повторений операторов тела цикла заранее неизвестно. Для его организации используется цикл типа **пока**. Выход из итерационного цикла осуществляется в случае выполнения заданного условия.

На каждом шаге вычислений происходит **последовательное приближение к искомому результату и проверка условия достижения последнего**.

Пример. Составить алгоритм вычисления бесконечной суммы

$$S = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots (-1)^{i-1} \frac{x^i}{i} + \dots$$

с заданной точностью ε (для данной знакопеременной бесконечной суммы требуемая точность будет достигнута, когда очередное слагаемое станет по абсолютной величине меньше ε).

Вычисление сумм — типичная циклическая задача. Особенностью же нашей конкретной задачи является то, что число слагаемых (а, следовательно, и число повторений тела цикла) заранее неизвестно. Поэтому выполнение цикла должно завершиться в момент достижения требуемой точности.

При составлении алгоритма нужно учесть, что знаки слагаемых чередуются и степень числа x в числителях слагаемых возрастает.

Решая эту задачу "в лоб" путем вычисления на каждом i -ом шаге частичной суммы

$$S := S + ((-1)^{(i-1)}) * (x^i) / i ,$$

мы получим очень неэффективный алгоритм, требующий выполнения большого числа операций. Гораздо лучше организовать вычисления следующим образом: если обозначить числитель какого-либо слагаемого буквой p , то у следующего слагаемого числитель будет равен $-p*x$ (знак минус обеспечивает чередование знаков слагаемых), а само слагаемое m будет равно p/i , где i — номер слагаемого.

Сравните эти два подхода по числу операций.

Запись алгоритма на алгоритмическом языке	Блок-схема алгоритма
<pre> алг Сумма (арг вещ x, Eps, рез вещ S) дано 0 < x < 1 надо S = x - x**2/2 + x**3/3 - ... нач цел i, вещ m, p ввод x, Eps S := 0; i := 1 начальные значения m := 1; p := -1 нц пока abs(m) > Eps p := -p*x p - числитель очередного слагаемого m := p/i m - очередное слагаемое S := S + m S - частичная сумма i := i + 1 i - номер очередного слагаемого кц вывод S кон </pre>	<pre> graph TD Start([Начало]) --> Input[/x, ε/] Input --> Init[S := 0; i := 1; m := 1; p := -1] Init --> Decision{abs(m) > ε} Decision -- да --> Process[p := -p*x; m := p/i; S := S+m; i := i+1] Process --> Decision Decision -- нет --> Output[/S/] Output --> End([Конец]) </pre>

Алгоритм, в состав которого входит итерационный цикл, называется **итерационным алгоритмом**. Итерационные алгоритмы используются при реализации итерационных численных методов.

В итерационных алгоритмах необходимо обеспечить обязательное достижение условия выхода из цикла (**сходимость итерационного процесса**). В противном случае произойдет "заикливание" алгоритма, т.е. не будет выполняться основное свойство алгоритма — **результативность**.

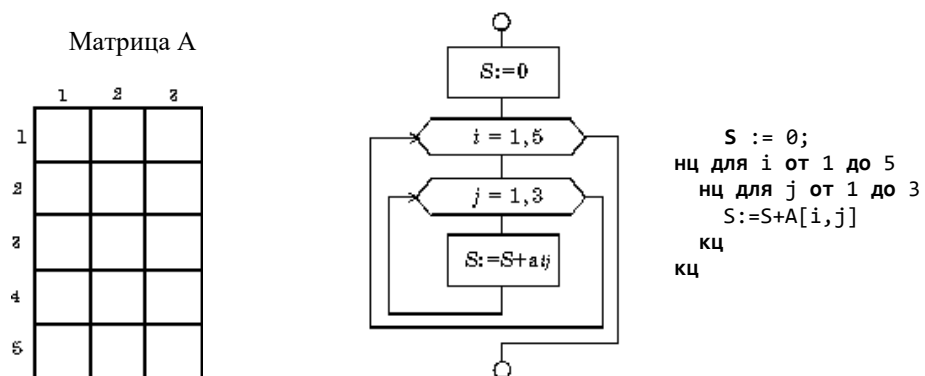
1.2.4. Вложенные циклы

Возможны случаи, когда внутри тела цикла необходимо повторять некоторую последовательность операторов, т. е. организовать внутренний цикл. Такая структура получила название **цикла в цикле** или **вложенных циклов**. Глубина вложения циклов (то есть количество вложенных друг в друга циклов) может быть различной.

При использовании такой структуры для экономии машинного времени необходимо выносить из внутреннего цикла во внешний все операторы, которые не зависят от параметра внутреннего цикла.

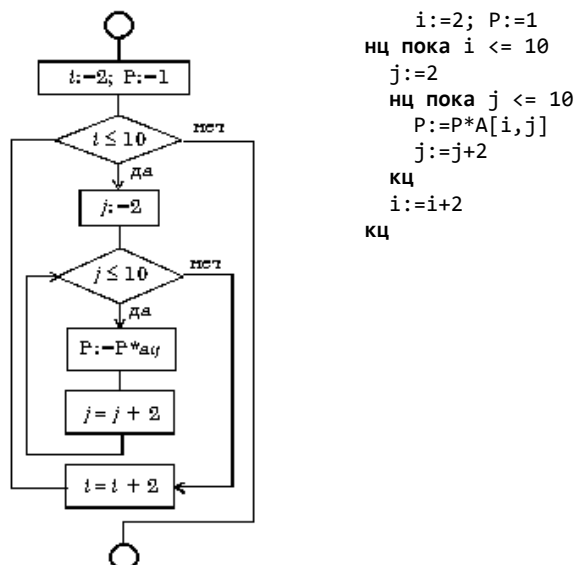
Пример вложенных циклов **для**

Вычислить сумму элементов заданной матрицы $A(5,3)$.



Пример вложенных циклов **пока**

Вычислить произведение тех элементов заданной матрицы $A(10,10)$, которые расположены на пересечении четных строк и четных столбцов.



1.3. Разработка алгоритмов решения задач

Различают алгоритмы линейной, разветвляющейся, циклической структуры и смешанной структуры.

2.2.1. Разработка алгоритмов линейной структуры

Алгоритмы линейной структуры. - это алгоритм, в котором блоки выполняются друг за другом, в порядке, заданном схемой.

Такой порядок выполнения называется естественным. В основу алгоритмов линейной структуры положена алгоритмическая структура - последовательность.

Пример: Вычислить высоты треугольника со сторонами a, b, c, используя формулы:

$$h_a = \frac{2}{a} \sqrt{p(p-a)(p-b)(p-c)} \quad \text{где } p = (a+b+c)/2$$

$$h_b = \frac{2}{b} \sqrt{p(p-a)(p-b)(p-c)}$$

$$h_c = \frac{2}{c} \sqrt{p(p-a)(p-b)(p-c)}$$

Решение: При решении задачи для исключения повторений следует вычислить высоты не приведенным выше формулам, а используя промежуточную переменную.

$$t = \sqrt{p(p-a)(p-b)(p-c)}, \text{ тогда } h_a = t/a, h_b = t/b, h_c = t/c.$$

При этом схема алгоритма решения имеет вид, представленный на рис. 4

2.2.2. Разработка алгоритмов разветвляющейся структуры

2.2.3. Разработка алгоритмов циклической структуры

