

ВВЕДЕНИЕ

В настоящее время применение средств вычислительной техники как инструмента решения инженерных задач требует углубленных знаний в различных областях человеческой деятельности. Часто на практике необходимо преобразовать исходную задачу с учетом дискретного характера машинных вычислений и представить процесс ее решения на ЭВМ в виде последовательности шагов. Такой подход должен выработать у будущего специалиста «алгоритмическое мышление», на основе которого дальнейший процесс разработки программ не вызывает затруднений.

Решение любой задачи на ЭВМ содержит следующие этапы обработки.

1. *Постановка задачи* – формулирование задачи, определение конкретной цели ее решения и результатов, которые должны быть получены, выработка критериев оценки этих результатов.

2. *Формализация задачи* – выбор математических методов решения задачи с учетом их применимости для машинных вычислений.

3. *Алгоритмизация* – разработка алгоритма решения задачи, т. е. представление процесса ее решения в виде шагов, этапов.

4. *Программирование* (кодирование алгоритма) – перевод алгоритма решения задачи на язык ЭВМ.

5. *Отладка программы* – выявление возможных синтаксических или семантических (смысловых) ошибок и их устранение. На этом этапе разрабатываются также тестовые примеры с целью проверки работоспособности программы.

6. *Получение результатов и их анализ* – полученные результаты должны быть проанализированы на предмет их достоверности и возможности использования в практической или научной деятельности. Если результаты не удовлетворяют поставленным требованиям, то осуществляется проверка правильности выполнения предыдущих этапов. Таким образом, процесс решения инженерных задач на ЭВМ носит обычно *итерационный* характер.

Целью методических указаний является обучение студента основным правилам и приемам составления схем алгоритмов решения вычислительных задач. Схема алгоритма позволяет наглядно представить пошаговое решение поставленной задачи на ЭВМ и является универсальным средством, не зависящим от языка программирования. При этом будем считать, что первые два этапа решения любой задачи на ЭВМ выполнены, т. е. осуществлена корректная постановка задачи и выбран метод ее решения.

В методических указаниях приведены схемы алгоритмов решения типовых вычислительных задач.

Для более детального освоения методов и приемов, используемых на этапе алгоритмизации задач, можно воспользоваться литературой [1].

1. ПРОГРАММА ДИСЦИПЛИНЫ «АЛГОРИТМИЗАЦИЯ ИНЖЕНЕРНЫХ ЗАДАЧ»

Раздел 1. Этапы решения инженерных задач на ЭВМ

От постановки задачи до получения и оценки результатов ее решения на ЭВМ. Итерационный процесс решения инженерных задач.

Раздел 2. Алгоритмизация задач. Схемы алгоритмов

Роль этапа «Алгоритмизация» в процессе решения инженерных задач. Схемы алгоритмов. Условные обозначения в схемах алгоритмов. Виды алгоритмов. Примеры.

Раздел 3. Алгоритмизация ветвящихся вычислительных процессов

Организация вычислений по условию. Алгоритмы решения типовых задач. Примеры.

Раздел 4. Алгоритмизация циклических вычислительных процессов

Организация вычислений с использованием циклов. Алгоритмы решения типовых задач. Примеры.

Раздел 5. Алгоритмизация вычислительных процессов с использованием подпроцессов

Организация вычислений с использованием подпроцессов. Алгоритмы решения типовых задач. Примеры.

Дисциплина «Алгоритмизация инженерных задач» изучается студентами в первом семестре. Общий объем – 77 часов, из них – 27 часов аудиторных в период сессии, 50 часов – самостоятельная работа.

Вид итогового контроля – зачет.

Для зачетной аттестации по дисциплине студент должен:
выполнить контрольную работу и *выслать ее на проверку до сессии*;
выполнить лабораторную работу на тему «Обработка массивов данных» (*во время сессии*). Задания к лабораторной работе выдаются преподавателем.

В качестве темы для самостоятельной работы предлагается изучение раздела 5 рабочей программы дисциплины:

подпроцессы в алгоритмах – будущие процедуры и функции в программах, составленных на каком-либо языке программирования;
 типовые подпроцессы: примеры их использования.

2. ВВЕДЕНИЕ В АЛГОРИТМИЗАЦИЮ ЗАДАЧ

Основные понятия и определения

Алгоритмизация задачи представляет собой процесс составления алгоритма ее решения. *Алгоритм* – строго определенная процедура, гарантирующая получение результата за конечное число шагов.

Все многообразие вычислительных алгоритмов включает в себя в виде фрагментов три типовых вычислительных процесса:

1) *линейный* процесс – последовательность операций, выполняемых одна за другой;


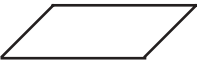

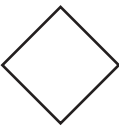
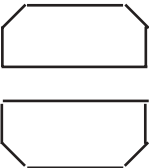

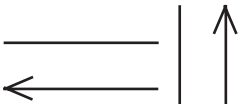

2) *ветвящийся* процесс – выполнение операций по одному из возможных направлений (ветвей алгоритма) в зависимости от некоторого условия;

3) *циклический* процесс – многократное выполнение некоторого набора операций, составляющих тело цикла, в соответствии с заданным правилом.

С целью наглядного представления вычислительного процесса решения задачи используются схемы алгоритмов, которые составляются в соответствии с требованиями ГОСТ «Единая система программной документации» (ЕСПД): ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем.

Рассмотрим основные символы (геометрические фигуры), используемые для графического представления алгоритмов.

В схеме алгоритма каждый символ может иметь порядковый номер, который записывается слева над символом. Нумерация производится слева направо и сверху вниз.

Символ	Наименование символа	Выполняемые функции
	Терминатор	Начало/конец схемы алгоритма
	Данные	Ввод данных из внешней среды или вывод результатов (носитель данных не определен)
	Процесс	Обработка данных любого вида
	Решение	Переключение вычислительного процесса на одну из ветвей алгоритма после вычисления условий, определенных внутри этого символа. Результаты вычисления условий указываются рядом с линиями, выходящими из вершин символа
	Граница цикла	Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце, в зависимости от расположения операции, проверяющей условие
	Предопределенный процесс	Использование процесса, состоящего из операций, определенных в другом программном блоке
	Линия	Отображает поток данных или управления. Направления потоков справа налево и снизу вверх обозначаются стрелками
	Соединитель	Используется для обрыва линии потока и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение

Ниже приведены примеры построения схем алгоритмов ветвящегося и циклического вычислительных процессов.

Пример 1.

Составить схему алгоритма вычисления значения y :

$$y = \begin{cases} a + x, & \text{если } a < b, \\ b - 2x & \text{в остальных случаях.} \end{cases}$$

Исходные данные: a, b, x .

Решение задачи показано на рис. 2.1.

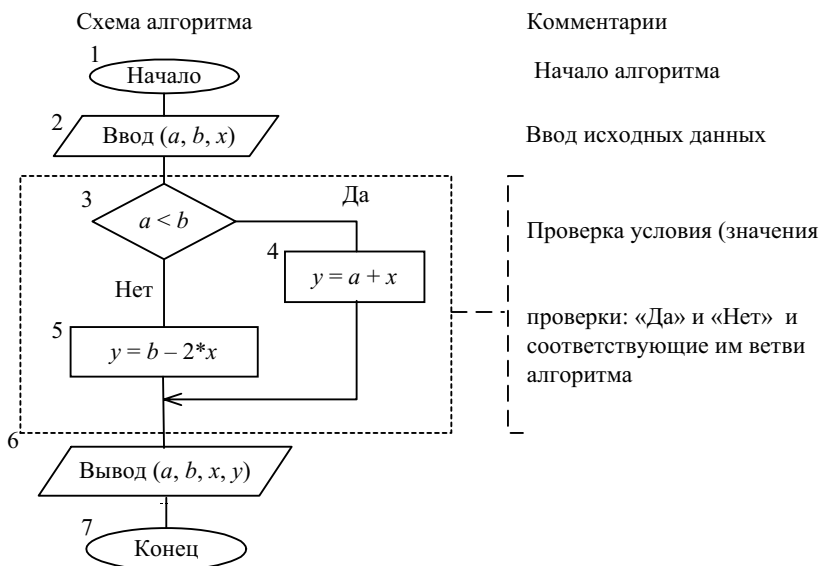


Рис. 2.1. Схема алгоритма вычисления значения y

Пример 2.

Составить схему алгоритма вычисления значения суммы:

$$S = \sum_{k=1}^n \sin(p + k) / k$$

Исходные данные: p, n

Решение задачи показано на рис. 2.2.

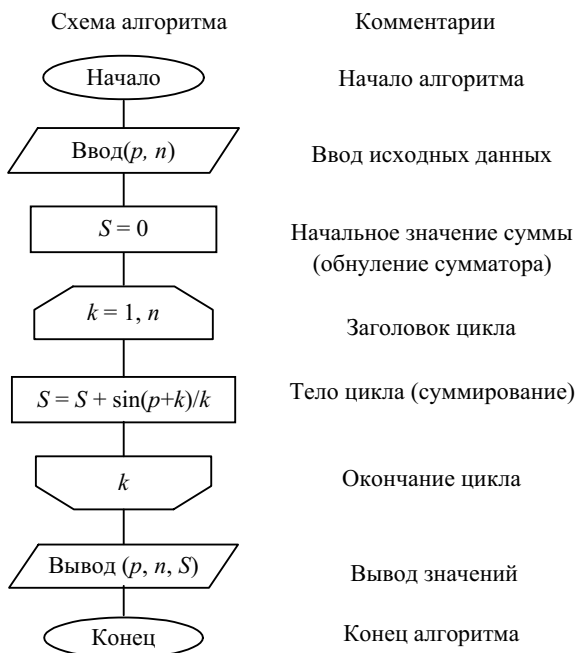


Рис. 2.2. Схема алгоритма вычисления значения суммы

Алгоритм поиска экстремума среди нескольких величин

Задача поиска экстремального значения среди нескольких величин может быть реализована следующими способами:

- а) на основе предположений с последующими проверками;
- б) последовательное сравнение пар.

Пример 3.

Составить схему алгоритма нахождения максимального значения среди трех величин: a , b , c .

Решение задачи на основе предположений с последующими проверками приведено на рис. 2.3.

Аналогично решается задача поиска минимального значения среди нескольких величин.

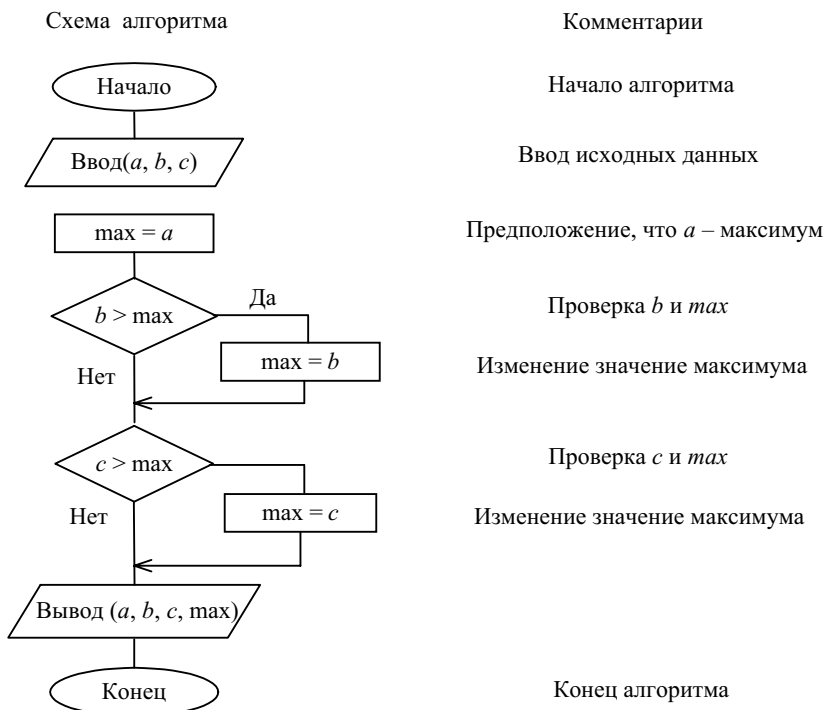


Рис. 2.3. Схема алгоритма нахождения максимального значения

3. АЛГОРИТМ ОБРАБОТКИ ЧИСЛОВОЙ ПОСЛЕДОВАТЕЛЬНОСТИ

Пусть задана последовательность $a_1, a_2, \dots, a_k, \dots, a_N, \dots$, где a_k – общий член последовательности. Любые операции с такой последовательностью возможны лишь при выполнении условия сходимости (критерий Коши): если для каждого сколь угодно малого положительного числа ϵ существует такой номер N , что из $m > N$ и $n > N$ следует $|a_n - a_m| < \epsilon$, то последовательность считается сходящейся. Проверку этого условия можно не делать, если операции проводятся с конечным числом членов последовательности.

В выражение для a_k могут входить различные функции: степенные, показательные, тригонометрические, логарифмические, а также факториалы. Для заданных значений аргументов функций, входящих в a_k , можно вычислить числовые значения членов последовательности.

В этом случае говорят о числовой последовательности. При вычислении степенных функций и факториалов с ростом k резко возрастает расход машинного времени и уменьшается точность вычислений. В этих случаях используются *рекуррентные* соотношения, позволяющие вычислить очередной член числовой последовательности a_k или его компоненты через a_{k-1} :

$$a_k = f(a_{k-1}), \quad k = 2, \dots, N \quad (3.1)$$

Рекуррентная зависимость (3.1) используется также при вычислении значения суммы (произведения) членов последовательности. Действительно, частичные суммы членов последовательности:

$$S_1 = a_1,$$

$$S_2 = a_1 + a_2,$$

...

$$S_k = a_1 + a_2 + \dots + a_{k-1} + a_k$$

можно представить рекуррентной формулой

$$S_k = S_{k-1} + a_k \text{ (аналогично для произведения } P_k = P_{k-1} \cdot a_k \text{)}.$$

Особенностью вычислений по рекуррентной формуле (3.1) является то, что для получения значения a_k достаточно знать только вычисленное на предыдущем шаге значение a_{k-1} . Таким образом, достигается экономия памяти ЭВМ, так как результат каждого шага вычислений по формуле (3.1) заносится в одну и ту же ячейку памяти, при этом предыдущее значение (a_{k-1}) стирается. Аналогичные рассуждения можно привести для вычисления S_k и P_k .

Следует иметь в виду, что перед вычислениями по рекуррентным формулам необходимо определить a_1 , S_1 или P_1 .

Пример 1.

Составить схему алгоритма вычисления значения суммы первых l из n членов последовательности, общий член которой $a_k = (-1)^{k+1} \sin^k(x) p^{2k} / k!$, где $x = x_0 + (i-1)h$, $i = 1 \dots m$. Исходными являются значения параметров: n , m , x_0 , h , p .

В примере используется факториал $k! = k \times (k-1) \times (k-2) \times \dots \times 1$.

Решение. Для получения рекуррентной зависимости (3.1) можно воспользоваться отношением

$$\frac{a_k}{a_{k-1}} = \frac{(-1)^{k+1} \sin^k(x) p^{2k} (k-1)!}{k! (-1)^k \sin^{k-1}(x) p^{2(k-1)}} = \frac{-\sin(x) p^2}{k}.$$

Таким образом, согласно формуле (3.1), $a_k = -a_{k-1} \sin(x) p^2 / k$.

Для определения a_1 в формулу для общего члена подставим значение $k = 1$. Получим $a_1 = p^2 \sin(x)$, откуда $S_1 = a_1 = p^2 \sin(x)$.

Схема алгоритма решения задачи приведена на рис. 3.1.

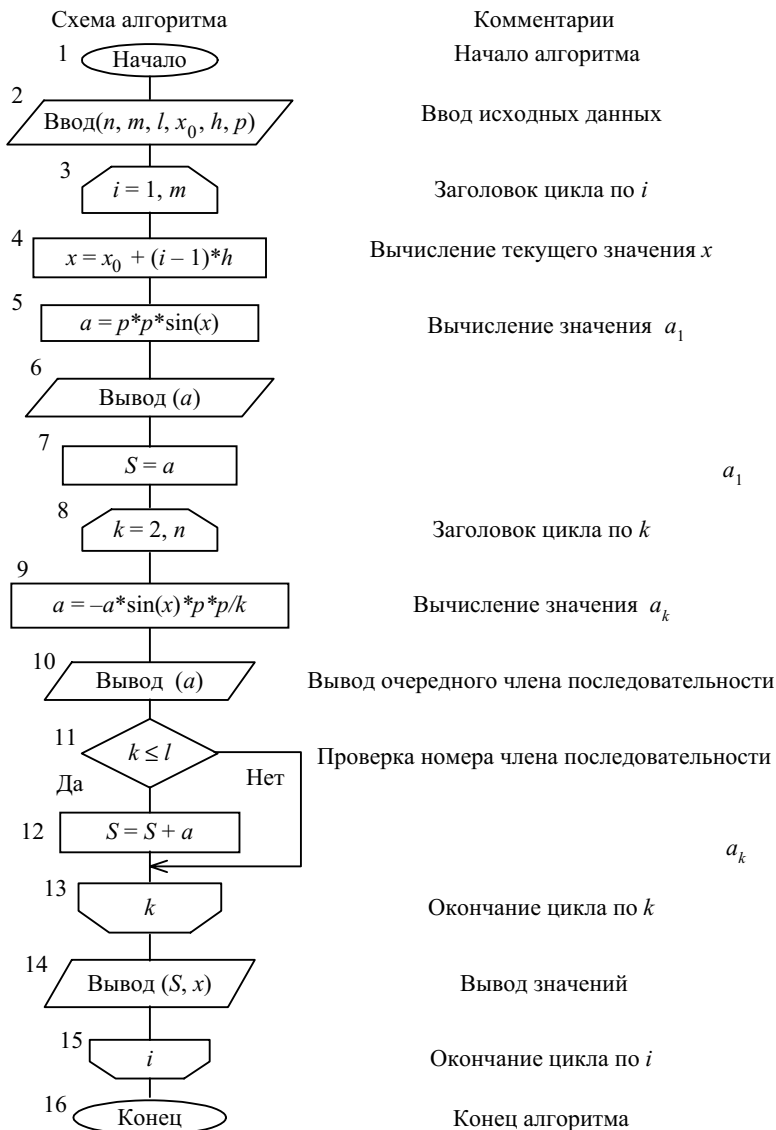


Рис. 3.1. Схема алгоритма обработки числовой последовательности

В некоторых задачах рекуррентное соотношение целесообразно найти только для некоторой компоненты общего члена последовательности.

Пример 2.

Составить схему алгоритма вычисления значения произведения P всех n членов последовательности, общий член которой $a_k = 1 + e^{-pk} x^{k+1} / (k+1)!$, где $x = x_0 + (i-1)h$, $i = 1..m$. Исходными являются значения параметров: n, m, x_0, h, p .

Решение. Предварительно необходимо произвести преобразование: $a_k = 1 + b_k$ и рекуррентную формулу вывести для b_k :

$$\frac{b_k}{b_{k-1}} = \frac{e^{-pk} x^{k+1} (k-1+1)!}{(k+1)! e^{-p(k-1)} x^{(k-1)+1}} = \frac{e^{-p} x}{k+1}.$$

Таким образом, согласно выражению (3.1), $b_k = b_{k-1} e^{-p} x / (k+1)$.

В этом случае $b_1 = e^{-p} x^2 / 2$, $a_1 = 1 + b_1$.

Для решения этой задачи можно воспользоваться схемой алгоритма, представленной на рис. 3.1, произведя в ней следующие изменения:

в блоке 5 записать $b = e^{-p} \times x \times x / 2$;

в блоке 7 записать $P = 1 + b$;

в блоке 9 записать $b = b \times e^{-p} \times x / (k+1)$;

блок 11 удалить;

в блоке 12 записать $P = P + b$;

в блоке 14 записать *Вывод* (P, x).

4. АЛГОРИТМ ОБРАБОТКИ МАССИВА ДАННЫХ

Под *массивом* будем понимать некоторым образом организованный набор данных (например, чисел), называемых *элементами*, или *компонентами*, массива. Элементам массива приписывается общее имя, при этом элементы имеют индексы, которые определяют местоположение каждого элемента в массиве. Количество индексов, используемых для указания *координат* (положения) элемента в массиве, зависит от размерности массива. Различают *одномерные* (векторы в математике), *двумерные* (матрицы в математике или таблицы) и *многомерные* массивы. В дальнейшем будем рассматривать только одномерные и двумерные массивы.

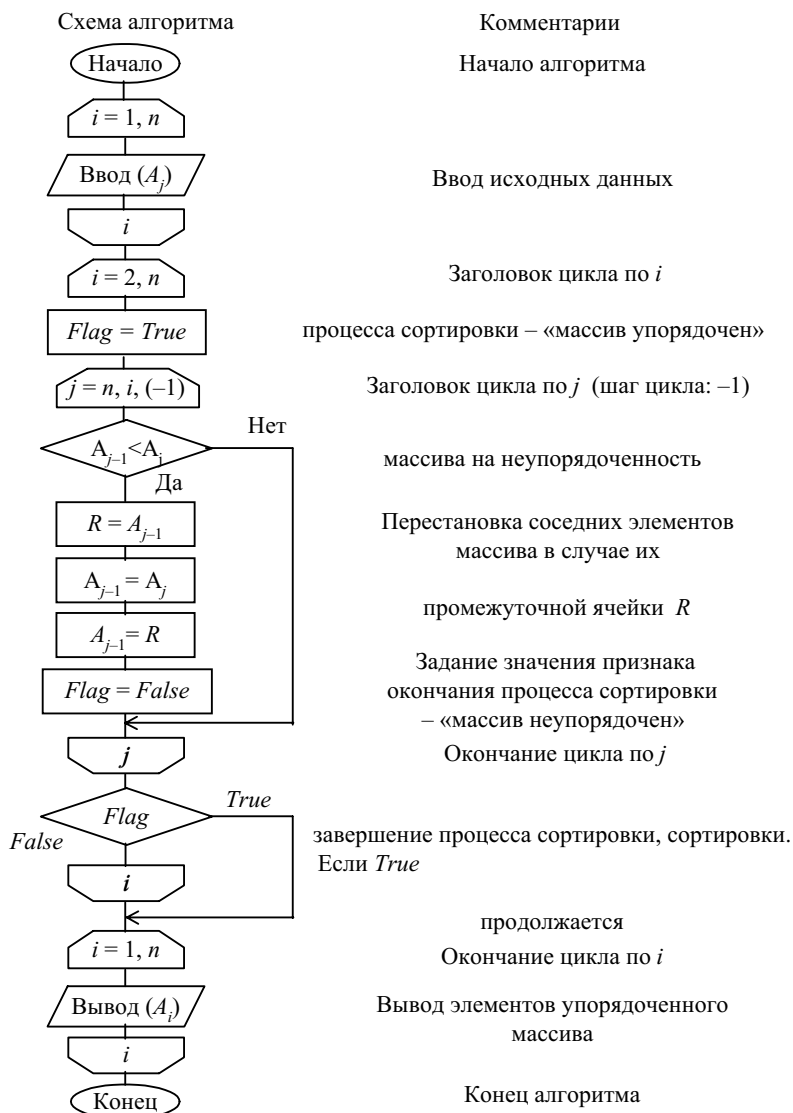


Рис. 4.1. Схема алгоритма сортировки методом «пузырька»

Например,

$A_n = \{ a_1, a_2, \dots, a_n \}$ – одномерный массив (вектор) размерностью n ,

$$B_{m \times n} = \begin{cases} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{12} & b_{13} & \dots & b_{1n} \\ & \dots & & & \\ b_{m1} & b_{m2} & b_{m3} & \dots & b_{mn} \end{cases} \quad \text{— двумерный массив (матрица) размерно-}$$

стью $m \times n$. (m — число строк, n — число столбцов матрицы).

В общем случае массивы не упорядочены по значениям элементов, но в то же время упорядочены по индексам, что позволяет производить обработку массивов путем организации циклов по индексам.

К задачам обработки массивов данных относятся задачи сортировки (упорядочения), вычисления значений суммы и произведения элементов массивов, поиска экстремальных элементов и их координат. Кроме того, обработка массивов используется при решении задач линейной алгебры, таких как сложение и умножение векторов и матриц, транспонирование матриц и т. д. [2, 3].

Сортировка массива данных (обычно одномерного) заключается в перестановке элементов массива в заданном порядке, например, по возрастанию или убыванию значений элементов массива или значений функций от этих элементов. Существуют несколько методов сортировки массивов. Рассмотрим один из наиболее применимых на практике метод «пузырька». Суть метода состоит в следующем. В ходе просмотра элементов неупорядоченной последовательности сравниваются два соседних числа. Если эти числа расположены в заданном порядке, то они остаются на своих местах, иначе их меняют местами. Затем переходят к следующей паре, в которой одно число из предыдущей пары. Обычно просмотр элементов начинается с последней пары. Сортировка считается законченной, если в ходе просмотра элементов последовательности не была произведена ни одна перестановка, иначе процесс повторяется.

Пример 1.

Составить схему алгоритма упорядочения одномерного массива A_n в порядке убывания значений элементов, используя метод «пузырька».

Схема алгоритма решения задачи приведена на рис. 4.1.

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Для более эффективного освоения дисциплины студенты должны выполнить контрольную работу на тему «Обработка числовой последо-

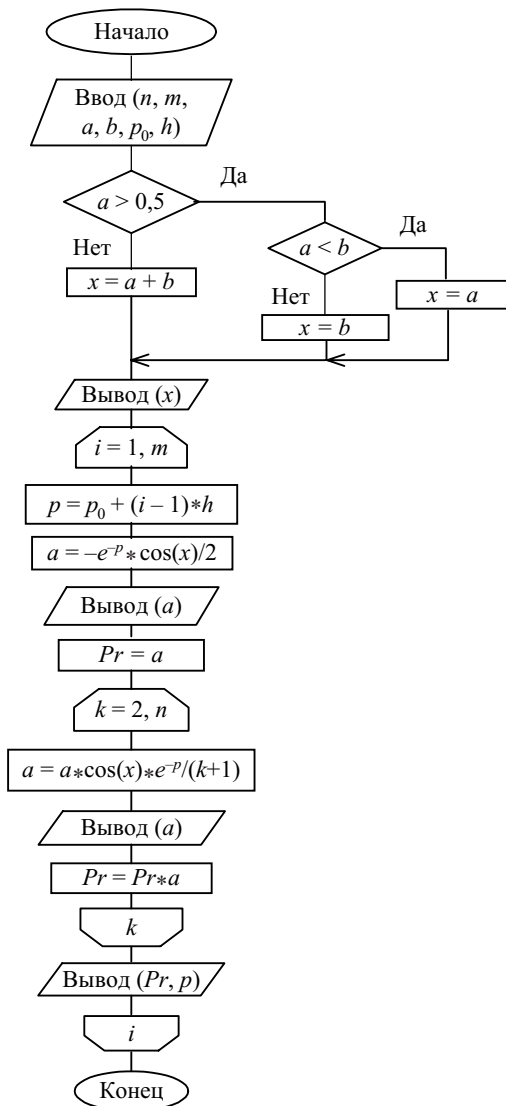


Рис. 5.1. Схема алгоритма решения примера контрольной работы

вательности». Методические указания к алгоритмизации задач обработки числовых последовательностей приведены в разд. 3.

Рассмотрим порядок выполнения контрольной работы на примере.

Пример 1.

Составить схему алгоритма обработки числовой последовательности в соответствии с указанным заданием:

21.	$\frac{(-1)^k e^{-pk} \cos^k(x)}{(k+1)!}$	Найти произведение всех членов	$x = \begin{cases} \min(a, b), & \text{если } a > 0,5 \\ a + b, & \text{иначе} \end{cases}$ $p = p_0 + (i-1)h, i = 1..m$	$a, b,$ $p_0, h,$ m, n
-----	---	--------------------------------	---	--------------------------------

Перед тем как составлять схему алгоритма, необходимо свести вычисления членов последовательности к рекуррентным формулам в соответствии с методикой, изложенной в разд. 3. Найдем соотношение

$$\frac{a_k}{a_{k-1}} = \frac{(-1)^k e^{-pk} \cos^k(x) k!}{(k+1)! (-1)^{k-1} e^{-p(k-1)} \cos^{k-1}(x)} = \frac{-\cos(x) e^{-p}}{(k+1)}.$$

Таким образом, согласно выражению (3.1),

$$a_k = -a_{k-1} \cos(x) e^{-p} / (k+1), a_1 = -e^{-p} \cos(x) / 2.$$

При составлении схемы алгоритма необходимо до обработки членов последовательности вычислить значения параметров x и p .

Схема алгоритма решения примера контрольной работы приведена на рис. 5.1.

Варианты учебных заданий на выполнение контрольной работы приведены в табл. 1.

Таблица 1

Учебные задания по теме «Обработка числовой последовательности»

№ варианта	Последовательность из n членов		Параметры	
	Общий член a_k	Вид обработки	вычисляемые	исходные
1	$\frac{(-1)^k e^{-pk} \sin^k(x)}{k(k-1)!}$	Найти сумму всех членов	$x = \begin{cases} \max(a, b), & \text{если } a + b > 0 \\ a \cdot \sin(b), & \text{иначе} \end{cases}$ $p = p_0 + (I-1)h, i = 1, \dots, m$	$a, b, p_0,$ h, m, n
2	$\frac{(-1)^k e^{-2pk} \cos^k(x)}{(2k+1)!}$	Найти произведение всех членов	$x = \begin{cases} \min(a, b), & \text{если } a + b < 0 \\ a, & \text{иначе} \end{cases}$ $p = p_0 + (I-1)h, i = 1, \dots, m$	$a, b, p_0,$ h, m, n
3	$\frac{(-1)^{k+1} e^{-pk} x^k}{(k-1)!}$	Найти сумму первых пяти членов	$p = \begin{cases} \max(a, b) + \min(c, d), & \text{если } a > d \\ 1, & \text{иначе} \end{cases}$ $x = x_0 + (I-1)h, I = 1, \dots, m$	$a, b, c,$ $d, x_0, h,$ m, n

№ вари- анта	Последовательность из n членов		Параметры	
	Общий член a_k	Вид обработки	вычисляемые	исходные
4	$\frac{(-1)^{k-1} e^{-pk} (2x)^k}{2k(2k-1)!}$	Найти сумму последних четырех членов	$p = \begin{cases} \max(a, b), & \text{если } a + b > 0 \\ \min(c-1, d), & \text{иначе} \end{cases}$ $x = x_0 + (I-1)h, I = 1, \dots, m$	a, b, c, d, x_0, h, m, n
5	$\frac{(-1)^{k+1} x^{-k} (2p)^{k-1}}{(2k+2)!}$	Найти \max из всех членов	$p = \min(a, b, c, d)$ $x = x_0 + (I-1)h, i=1, \dots, m$	a, b, c, d, x_0, h, m, n
6	$\frac{(-1)^k x^{-k+1} (p)^{k-1}}{(2k-2)!}$	Найти \min из всех членов	$p = \max(a, b, c, d)$ $x = x_0 + (I-1)h, I = 1, \dots, m$	a, b, c, d, x_0, h, m, n
7	$\frac{(-1)^{k+2} p^{-2k} (x)^{k-2}}{(2k)!}$	Найти сумму положительных членов	$x = \max(a, b, c) + d$ $p = p_0 + (I-1)h, i = 1, \dots, m$	a, b, c, d, p_0, h, m, n
8	$\frac{(-1)^{k-3} p^{-k} (x+1)^{k-1}}{k(2k-2)!}$	Найти сумму отрицательных членов	$x = \min(a, b, c) + a*b*c$ $p = p_0 + (I-1)h, I = 1, \dots, m$	a, b, c, p_0, h, m, n
9	$\frac{(-1)^{-k+3} p^{-2(k+1)} (x+1)^k}{(k+1)!}$	Найти количество положительных членов	$p = \begin{cases} \max(a, b, c), & \text{если } a + b > c \\ c - d + a \cdot c, & \text{иначе} \end{cases}$ $x = x_0 + (I-1)h, I = 1, \dots, m$	a, b, c, x_0, h, m, n
10	$\frac{(-1)^k e^{-2pk} (x-1)^k}{(3k-2)!}$	Найти количество отрицательных членов	$p = \begin{cases} \min(a+b, c-d), & \text{если } a \neq c \\ 0, & \text{иначе} \end{cases}$ $x = x_0 + (I-1)h, i=1, \dots, m$	a, b, c, d, x_0, h, m, n
11	$\frac{(-1)^k e^{-k} (x+p)^{k+1}}{2k(k+1)!}$	Найти \max и \min среди всех членов	$x = \min(a, b) + a \cdot b \cdot \sin(a+b)$ $p = p_0 + (I-1)h, i = 1, \dots, m$	a, b, p_0, h, m, n
12	$\frac{(-1)^{k-1} p^{-2k} x^{k+1}}{(k+1)(k+1)!}$	Найти произведение положительных членов	$x = \max(a, b) + (a+b) \cdot \cos(a-b)$ $p = p_0 + (I-1)h, I = 1, \dots, m$	a, b, p_0, h, m, n
13	$\frac{(-1)^k p^{-k} (x-1)^k}{k!}$	Найти произведение отрицательных членов	$x = \max(a+b, 2I)p = p_0 + (I-1)h, I = 1, \dots, m$	a, b, p_0, h, m, n
14	$\frac{(-1)^{-k} p^{-k} (x+2)^{k-1}}{(k-1)!}$	Найти произведение первых шести членов	$p = \begin{cases} \max(a, b), & \text{если } a + b + d > c \\ \min(c, d), & \text{иначе} \end{cases}$ $x = x_0 + (I-1)h, i=1, \dots, m$	a, b, c, d, x_0, h, m, n
15	$\frac{(-1)^{k+2} p^{-2k} x^{k-1}}{(k+3)!}$	Найти произведение последних четырех членов	$p = \begin{cases} \min(a, b), & \text{если } a + 1 > c - 2 \\ \min(c, d), & \text{иначе} \end{cases}$ $x = x_0 + (i-I)h, I = 1, \dots, m$	a, b, c, d, x_0, h, m, n

ЗАКЛЮЧЕНИЕ

Рассмотренный в методических указаниях этап алгоритмизации вычислительных задач предваряет этап программирования, и то, насколько правильно и корректно составлен алгоритм решения задачи, определяет эффективность и качество программы, реализующей этап машинных вычислений.

Приведенные примеры алгоритмов решения типовых вычислительных задач должны помочь студенту освоить основные правила составления алгоритмов и научить решать сложные задачи, представляя их решение в виде элементарных шагов.

Библиографический список

1. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов / Пер. с англ. под ред. В. В. Мартынюка. М.: Мир, 1981. 368 с.
2. Беллман Р. Введение в теорию матриц / Пер. с англ. под ред. В. Б. Лидского. М.: Наука, 1976. 352 с.
3. Гильберт А. Как работать с матрицами/ Пер. с нем. Я. Ш. Палпэ. М.: Статистика, 1981. 157 с.

СОДЕРЖАНИЕ

Введение	1
1. Программа дисциплины «Алгоритмизация инженерных задач»	2
2. Введение в алгоритмизацию задач	3
3. Алгоритм обработки числовой последовательности	7
4. Алгоритм обработки массива данных	10
5. Методические указания к выполнению контрольной работы	12
Заключение	13
Библиографический список	13