Федеральное агентство морского и речного транспорта
Федеральное государственное образовательное учреждение
высшего профессионального образования
«Волжская государственная академия водного транспорта»

Кафедра информатики, систем управления
и телекоммуникаций

Серия «Информационные технологии
в системах управления и телекоммуникаций»

Выпуск 2

В.И. Логинов, Л.Н. Шемагина

Основы алгоритмизации

Учебно-методическое пособие
для студентов очного и заочного обучения
технических специальностей

Рекомендовано Государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский морской технический университет» в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности 180404 «Эксплуатация судового электрооборудования и средств автоматики»

Нижний Новгород
Издательство ФГОУ ВПО «ВГАВТ»
2010

УДК 681.3.06
Л69

Логинов, В.И.
Основы алгоритмизации : учеб.-метод. пособие для студ. оч. и заоч. обуч. технич. специальностей / В.И. Логинов, Л.Н. Шемагина. – Н. Новгород : Изд-во ФГОУ ВПО «ВГАВТ», 2010. – 81 с.

Излагаются основные принципы и типовые приемы алгоритмизации при решении инженерных задач. Приводятся примеры алгоритмов решения типовых задач: табулирование функций, поиск элементов с заданными признаками и сортировка данных.

Пособие направлено на формирование начальных навыков алгоритмизации и программирования, составляющих фундаментальную базу профессиональной деятельности современного инженера.

Методические концепции пособия базируются на многолетнем опыте обучения основам алгоритмизации и программирования студентов технических специальностей ВГАВТ.

Для студентов, обучающихся по техническим специальностям и направлениям подготовки в высших учебных заведениях водного транспорта.

Работа рекомендована к изданию кафедрой информатики, систем управления и телекоммуникаций (протокол № 3 от 19.11.2008 г.).

Введение

Основу инженерной деятельности составляет умение ставить задачи, разрабатывать алгоритмы и получать решения, производить анализ полученных данных и делать выводы. Поэтому в своей профессиональной деятельности инженер должен уметь грамотно применять персональный компьютер (ПК) для решения научных и инженерных задач.

В пособии представлены алгоритмы решения задач начальной и средней сложности. Как правило, эти задачи содержат немного параметров, их легко можно сформулировать и смоделировать. Данные задачи отличаются от задач, описывающих большие системы, но каждая большая программная система состоит из небольших программ, реализующих несложные известные алгоритмы.

Изложение базируется на современных принципах синтеза алгоритмов с использованием концепции структурного программирования. Сформулированы этапы подготовки задач для программирования, способы и основные принципы алгоритмизации при решении инженерных задач. Рассмотрены типовые структуры алгоритмов и типовые приемы алгоритмизации. Приведены примеры алгоритмов типовых задач: табулирование функций, поиска элементов с заданными признаками и сортировки данных.

Все темы рассматриваются по мере усложнения. При этом основополагающими являются первая, вторая и третья главы. Материал расположен именно в том порядке, в котором большинство начинающих пользователей изучает самостоятельно основы алгоритмизации.

Цель пособия – научить студентов анализировать и ставить задачи, разрабатывать алгоритмы их решения.

Все главы завершаются контрольными вопросами и упражнениями, предназначенными для закрепления изученного материала. В пособии приведены задания для выполнения лабораторных работ, охватывающих основные типы вычислительных процессов.

1. Основы алгоритмизации

Алгоритмизация является основным, базовым компонентом компьютерной грамотности в динамично развивающемся совре-

менном компьютерном мире. Для достижения положительных результатов важную роль играет умение разрабатывать оптимальный алгоритм решения поставленной задачи, что требует от исполнителя наличия определённых навыков алгоритмизации и системного анализа. А также знание математики, общей физики, механики и других инженерных дисциплин.

Алгоритмизация – это общая последовательность действий, которые необходимо выполнить для построения алгоритма решения задачи, в том числе – выделение конкретных шагов алгоритмического процесса, определение вида формальной записи для каждого шага и установление определённого порядка выполнения каждого шага.

Процесс подготовки решения задачи и её непосредственная реализация на компьютере происходит за некоторое количество самостоятельных этапов:

1. Постановка задачи.
2. Математическая формулировка задачи.
3. Выбор метода решений.
4. Разработка алгоритма.
5. Составление программы на алгоритмическом языке.
6. Отладка и решение задачи на ПК.
7. Анализ полученных результатов.

1.1. Постановка задачи

На этом этапе на основе информационной модели (словесной постановки) задачи формируется цель решения задачи и подробно описывается её содержание. Проводится анализ характера и сущности известных и неизвестных данных, рассматривается область их существования, определяются условия, при которых задача может быть решена.

Таким образом, процесс формулировки задачи сводится к постановке следующих основных вопросов:

1. Что дано?
2. Что нужно вычислить?
3. Что представляют собой неизвестные и сколько их?
4. Какие данные необходимо ввести в ПК, чтобы получить ответ?
5. Как определить решение?
6. Какие следует сделать допущения?

7. Каковы требования к точности решения?

И т.д.

Иногда формулировка задачи допускает неточности и много толкований. В этом случае нужно сделать условия более точными, чтобы четко изложить, что должен сделать алгоритм.

При постановке задачи определяются не только характеристики заданных величин, их свойства и отношения между ними, но и требования к решению задачи. От правильности постановки задачи зависит успех ее решения.

1.2. Математическая формулировка задачи

Задача, предназначенная для решения на персональном компьютере (ПК), должна быть задана в математической формулировке, т.е. выражена в терминах понятий, формально определённых в математике, имеющих точно определённые свойства и находящихся между собой в строго определённых отношениях.

Условие задачи записывается либо в виде уравнений, либо в виде последовательности формул или логических соотношений, необходимых для решения задачи. На этом этапе могут добавляться некоторые дополнительные по сравнению с постановкой задачи условия, выделяющие единственное решение. Основным компонентом этого этапа является построение математической модели.

> ***Построением модели*** *называется формализация описания задачи с использованием математических, логических и других методов, при которых существующие соотношения между величинами, определяющие результат, выражаются с использованием математических формул и логических соотношений.*

Выбор модели существенно влияет на остальные этапы процесса решения. В зависимости от содержания задачи построение её модели может быть областью исследования различных дисциплин (например: методы оптимизации, численный анализ и др.).

При построении модели в первую очередь следует рассмотреть следующие вопросы.

Существует ли опыт решения аналогичных задач?

Какие известные математические и другие методы формализации подходят для решения поставленной задачи?

Затем следует рассмотреть математическое описание того, что мы знаем и что нужно найти. На выбор модели влияют такие факторы, как простота и однозначность вычислений, удобство представления и уровень знаний. При этом следует определить следующее.

Вся ли информация, необходимая для решения задачи, полностью описана математически?

Существует ли математическая величина, соотносимая с результатом?

Все ли соотношения между характеристиками модели описаны?

Метод решения задачи следует из принятой модели.

1.3. Выбор метода решений

Выбрать метод решения задачи значит преобразовать математическую формулировку задачи, включающую символы математического анализа (например: $\sum$, max, min, $\frac{d}{dx}$, $\int$…, dx и т.д.) в последовательность действий и логических связей между ними.

Если одна и та же задача может быть решена с помощью различных методов, выбирают тот, который наилучшим образом удовлетворяет её требованиям. При этом учитывается точность решения, быстрота получения результата, объём памяти для сохранения исходных и промежуточных данных, результатов и сложность программой реализации.

Существует два типа методов: точные и численные. Сущность точных методов состоит в последовательности выполнения арифметических и логических действий, позволяющих получить точное решение. Например: вычисление корня квадратного уравнения.

Однако для большинства задач, встречающихся в инженерной практике, точные методы неизвестны. Для решения таких задач используются численные методы, которые обеспечивают отыскание приближенного значения с требуемой точностью.

В некоторых случаях пригодность (или непригодность) избранного метода может выявиться только на последующих этапах. Тогда нужно возвратиться к этапу выбора метода.

1.4. Разработка алгоритма

Понятие алгоритма относится к числу основных понятий современной вычислительной математики и информатики.

Если задача содержит вычисления, то алгоритм – это последовательность указаний, содержащих расчётные формулы и сведения о том, как ими пользоваться. По мере развития параллельности в работе компьютеров слово *последовательность* стали заменять более общим словом *порядок*.

Единого общепринятого определения *алгоритм* нет. Разные авторы, в том числе Д.Э. Кнут, А. Колмогоров и др., дают различные толкования. Нет определения алгоритма и в ГОСТ 19781–90. Проанализировав все известные определения можно сказать, что в общем случае алгоритм – это система формальных правил, определяющая порядок действий и приводящая к решению поставленной задачи. И если различные исполнители будут действовать в соответствии с этой системой правил, то все они будут получать одинаковые результаты, т.е.

> ***Алгоритмом решения задачи*** *называется путь решения задачи (определенный порядок действий), который необходимо выполнить для достижения результата.*

Основной целью вычислительного процесса является исполнение алгоритма с заданными исходными данными и получение результата.

Если выбранный для задачи численный метод уже реализован в стандартной библиотеке программ, то алгоритмом будет описание и ввод данных, обращение к стандартной программе, вывод результатов на печать или экран дисплея. В инженерной практике более часты случаи, когда стандартные программы решают лишь какую-то часть задачи.

Разработка схемы алгоритма в общем случае состоит в чёткой структуризации задачи, разбиении её на последовательность подзадач (шагов) и построении алгоритма для каждого шага.

Алгоритм должен обладать следующими свойствами.

Дискретность. Весь вычислительный процесс разбит на мелкие этапы (шаги, дискреты). И решение задачи сводится к решению

простых задач, при этом каждое действие выполняется только после того, как закончится исполнение предыдущего.

Детерминированность (определённость), которая заключается в том, что чётко определён порядок выполнения алгоритма. Это обеспечивает однозначность результата при заданных исходных данных.

Результативность. Получение вполне определённого результата за определённое число шагов.

Конечность. При работе с численными методами строится бесконечный, сходящийся к искомому решению процесс. Процесс обрывается, когда очередное приближённое решение достигает заданной точности. Таким образом, за конечное число шагов получается решение задачи.

Массовость. С помощью выбранного алгоритма можно получать вполне определённый результат при различных исходных данных для некоторого класса задач.

Формальность. Исполнитель, незнакомый с содержанием алгоритма, но правильно выполнивший его предписание, получает искомый результат.

Современные методы программирования, основанные на структурном подходе, предусматривают использование различных специальных приёмов. Например, пошаговая детализация до команд решателя. На первом этапе разработки алгоритм рассматривается как некоторая совокупность действий для обработки исходной информации, а все последующие этапы – это уточнение (выявление) всё более частных особенностей.

При записи алгоритма на любом этапе нужно учитывать, что исполнителем будет компьютер, который сможет выполнять только вполне определённые действия – присваивание, ветвление, циклическое повторение, безусловный переход. Поэтому любая детализация должна приводить к реализации таких конструкций. При этом следует обратить внимание на исполнение алгоритма. Пошаговое выполнение последовательности операторов позволит получить результат.

В итоге процесс разработки алгоритма должен быть направлен на получение четкой структуры алгоритмических конструкций.

1.5. Способы описания алгоритмов

Для строгого задания различных структур данных и алгоритмов их обработки нужно иметь такую систему формальных обозначений и правил их использования, чтобы всякое действие трактовалось точно и однозначно. Соответствующие системы правил называют языками описаний.

В настоящее время используются следующие языки описания:

– словесная запись;

– псевдокод;

– схемы.

В больших коллективах программистов при разработке сложных проектов работают со специальными программными средствами для автоматизированного проектирования алгоритмов и программ.

1.5.1. Словесная запись алгоритма

Способ основан на использовании общепринятых средств общения между людьми и содержит набор фраз, который не допускает лишних слов, повторений и неоднозначностей. Действия, предусмотренные алгоритмом, нумеруются, что даёт возможность на них ссылаться. Допускается использование математической символики.

Задача 1. *Записать алгоритм нахождения наибольшего делителя двух натуральных чисел (X и Y).*

Решение.

1. Сравнить данные числа (X равно Y, X меньше, больше Y) и перейти к следующему пункту.

2. Если числа равны, то необходимо взять любое из них в качестве ответа и перейти к п. 6

3. Если числа не равны, то перейти к следующему пункту.

4. Определить большее из чисел.

5. Заменить большее число разностью большего и меньшего чисел. Перейти к началу п. 1.

6. Конец алгоритма.

Описанный алгоритм применим к любым числам и должен приводить к решению поставленной задачи.

Задача 2. *Заданы координаты двух точек. Координаты 1-й точки – X1, Y1, координаты 2-й точки – X2, Y2. Определить, какая из точек наиболее удалена от начала координат.*

Решение.

1. Вычисляем расстояние 1-й точки до начала координат:

$$R1 = \sqrt{X1^2 + Y1^2}.$$

2. Вычисляем расстояние 2-й точки до начала координат:

$$R2 = \sqrt{X2^2 + Y2^2}.$$

3. Определяем большее из расстояний. Если R1 больше R2, то переход к следующему пункту, если R1 меньше, то переход к п. 5.
4. Печать сообщения «Первая точка лежит ближе к началу координат». Переход к п. 6.
5. Печать сообщения «Вторая точка лежит ближе к началу координат».
6. Конец алгоритма.

Действия (пункты) алгоритма выполняются в естественной последовательности. Вид действий не формализован. Такой вид записи алгоритма имеет серьёзный недостаток – отсутствие наглядности. Поэтому этот способ записи алгоритма не имеет широкого распространения.

1.5.2. Псевдокод

Псевдокод – это язык записи структурированных алгоритмов. Основан на формализованном представлении предписаний, задаваемых с помощью ограниченного набора типовых синтаксических конструкций. Набор конструкций состоит из смеси алгоритмического языка высокого уровня и фраз родного языка исполнителя. Как правило, стандартов на псевдокод не существует.

Псевдокод используется для облегчения разработки программ. Так же как и программа, псевдокод имеет все достоинства структурированной записи, поэтому алгоритм, написанный на псевдокоде, достаточно легко преобразовать в программный код.

Пример псевдокода для нахождения наибольшего элемента из двух натуральных чисел (A и B). Обозначим наибольший элемент – Z.

Решение.

Начало
Ввод A, B
Если A≥B, то Z=A
Иначе Z=B
Конец Если
Вывод Z
Конец

Основными достоинствами псевдокода являются:

– близость к языкам программирования;

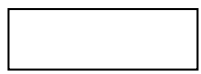
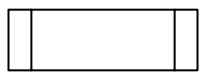
– возможность разобраться в самом длинном и сложном алгоритме, поэтому псевдокод используется чаще всего для документирования программ.

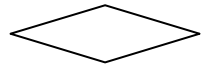
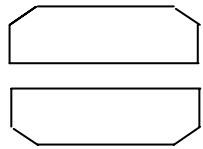
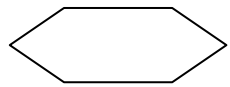
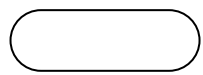
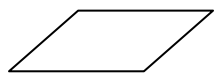
1.5.3. Схемы алгоритмов

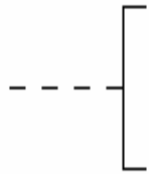
Наиболее распространенным является описание алгоритма в виде схемы из графических символов. *Схема алгоритма* – это графическое представление метода решения задачи, в котором используются символы, отображающие операции (действия) и данные.

В схеме алгоритма каждому типу действий (например: ввод исходных данных, вычисление значений выражений, проверка условий и т.д.) соответствует геометрическая фигура, представленная символом действия. Символы действия соединяют линиями переходов, которые определяют очерёдность выполнения действий. Форма символов и правила составления схем установлены Единой Системой Программной Документации (ЕСПД) ГОСТ 19701-90 [2]. Наиболее часто употребляемые символы действий указанного стандарта приведены в таблице.

Применение символов

Название символа	Обозначение	Пояснение
Процесс		Выполнение определенной операции или группы операций
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме

Решение		Проверка условий
Граница цикла		Символ состоит из двух частей и отображает начало и конец цикла. Обе части имеют один и тот же идентификатор
Подготовка		Модификация команды или группы команд на некоторую последующую функцию. Например, модификация параметров цикла Используется для циклов с параметром
Терминатор		Начало и конец программы, вход и выход из подпрограммы
Данные		Символ отображает ввод данных, носитель которых не определен
Документ		Символ отображает данные, представленные на носителе в удобочитаемой форме. Используется как символ печати результатов
Линия		Символ отображает поток данных или управления. При необходимости могут быть добавлены стрелки-указатели
Соединитель		Символ отображает выход в часть схемы и вход из другой части схемы. Используется для обрыва линии и продолжения её в другом месте

Комментарий		Используется для добавления описательных комментариев или пояснительных записей в целях объяснений примечаний. Пунктирная линия связана с соответствующим символом или может обводить группы символов. Текст помещается около ограничивающей фигуры

Символы **Данные** и **Документ** используются для обозначения операций ввода-вывода.

Символ **Процесс** применяется для обозначения выполнения одной или группы операций, приводящих к изменению значения, формы или размещения информации. Представление операций достаточно свободно. Например, для обозначения вычислений можно использовать математические выражения или текст.

Символ **Решение** используется для обозначения переходов управления по условию. Имеет один вход и ряд выходов, при этом только один из них может быть активизирован после вычисления условий, определённых внутри этого символа. Результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

Символ **Граница цикла** используется для организации циклических вычислительных процессов. Параметр цикла записывается в обеих частях символа. Начальное значение, граничное условие и правило изменения параметра цикла указывается в начале или в конце в зависимости от расположения операции, проверяющей условие.

Символ **Подготовка** используется, как правило, для организации цикла с параметром. Внутри символа записывается параметр цикла, указывается его начальное значение, граничное условие и правила изменения значения параметра. Символ размещается в начале циклической конструкции.

Линии переходов используются для обозначения порядка выполнения действий.

Соединители используются в том случае, когда схема алгоритма разделяется на автономные части, особенно если она не умеща-

ется на одном месте, или когда необходимо избежать излишних пересечений линий переходов.

Комментарий позволяет включать в схемы алгоритмов пояснения к функциональным блокам. Частое использование комментариев нежелательно, так как это усложняет схему, делает её менее наглядной.

Алгоритм начинается и заканчивается символами **Начало** и **Конец**.

Основные правила применения символов и выполнения схем алгоритмов:

1. Символы в схеме должны быть расположены равномерно. Нужно придерживаться разумной длины соединений и минимального числа длинных линий.
2. Символы должны быть по возможности одного размера и, предпочтительно, горизонтальной ориентации.
3. Внутри символа помещается минимальное количество текста, необходимое для понимания функции символа. Для записи используется естественный язык с элементами математической символики. Если объем текста превышает размер символа, то нужно использовать символ **Комментарий**.
4. Потоки данных и потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.
5. Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.
6. Каждый символ имеет один вход и один выход. Исключением является символ **Решение**, который имеет один вход и несколько выходов. При этом каждый выход должен сопровождаться значениями условий, чтобы указать логический путь, который он представляет.

Схема алгоритма является самым наглядным способом представления алгоритма, при этом нет никаких ограничений на степень детализации.

1.5.4. Реализация алгоритмов

После разработки алгоритма встает задача его реализации в виде программы, которую можно выполнить на вычислительной машине (компьютере). В ГОСТ 19781–90 дано следующее определение программы.

Программа *– это данные, предназначенные для управления конкретными компонентами обработки информации в целях реализации определенного алгоритма.*

Для написания программы необходимо в первую очередь выбрать язык программирования (алгоритмический язык). В общем случае алгоритмический язык – это набор символов с заданными правилами образования из этих символов конструкций, с помощью которых описывается процесс выполнения алгоритма. В ГОСТ 19781–90 дано следующее определение алгоритмического языка.

Алгоритмический язык *– искусственный язык, предназначенный для выражения алгоритмов.*

Основная цель любого алгоритмического языка – дать пользователю удобные средства для реализации алгоритмов.

Выбор языка определяется:

– типом решаемой задачи;

– предпочтениями и привычками пользователя;

– требуемыми затратами времени на разработку;

– операционной системой и доступными средами программирования.

Например, для программирования сложных задач выбирают язык Ассемблер или C++, а для решения инженерных задач предпочтение отдают Fortran (Visual Fortran), Basic (Visual Basic) или Pascal (Delphi или Lazarus).

Во вторую очередь необходимо перевести исходную программу, написанную на алгоритмическом языке, в исполняемую программу для вычислительной машины (процессора) или промежуточной исполняемой среды.

Этапы подготовки исполняемой программы.

1. Трансляция (компиляция) – это преобразование (перевод) исходного текста программы в объектный модуль, представляющий из себя набор машинных инструкций (команд) без стандартных

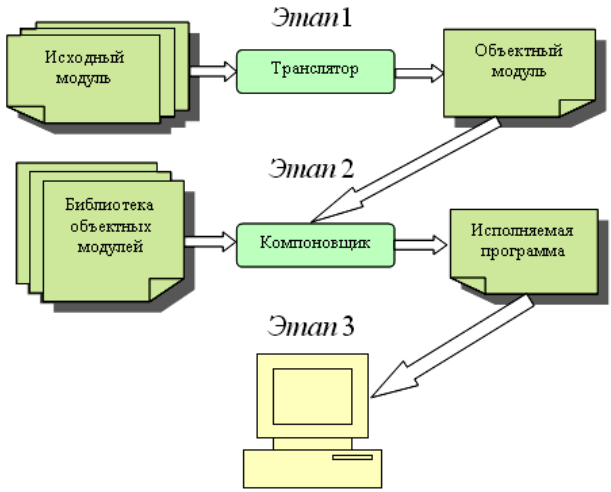
программ функций (программ вычисления синуса, логарифма и т.д.), модулей ввода исходных данных и вывода результатов или активных элементов управления, представляющих элементы современных операционных систем. При этом выполняется проверка правильности синтаксиса исходной программы. Если в программе будут обнаружены ошибки, то обработка программы прекращается. Если ошибок нет, то программа передается на следующий этап. Трансляция выполняется специальной программой (Транслятор, Компилятор), которая может входить в среду программирования.

2. Компоновка (или редактирование связей) – это сборка объектного модуля программы, модулей ввода-вывода и компонентов стандартной библиотеки объектных модулей в один модуль, который называется выполняемым файлом или загрузочным модулем.

3. Выполнение файла загрузочного или исполняемого модуля. На этом этапе получаем решение поставленной задачи. При этом возможно прерывание решения (аварийный останов, например, деление на ноль), зацикливание или неправильные результаты. Это возможно по следующим причинам.

- Ошибки в исходных данных. Во избежание этой ошибки рекомендуется проверять исходные данные путем их вывода, печати или любым другим способом.
- Ошибки в алгоритме – в этом случае необходимо вернутся к начальным этапам разработки алгоритма. Для выявления этих ошибок рекомендуется выполнить тестирование алгоритма.

Схема этапов подготовки программы представлена ниже.



1.5.5. Тестирование алгоритмов

Это важный этап при решении алгоритмических задач. На этом этапе выявляются ошибки реализации алгоритма.

В общем случае отсутствуют универсальные правила проведения тестирования.

При решении физических и инженерных задач общее правило тестирования заключается в необходимости знания или результатов решения задачи для одного или нескольких наборов данных или оценки достоверности полученных результатов любыми доступными математическими или экспертными методами. Это дает предварительную оценку правильности работы программы.

Для более детального анализа правильности алгоритма и программы необходимо при отладке программы предусмотреть вывод этапов прохождения алгоритма, промежуточных переменных и переменных цикла.

1.6. Величины в алгоритмах

Современные компьютеры могут получать, накапливать и хранить большие объемы информации, содержащие формализованные сведения о реальном мире. И при постановке задачи пользователь сознательно или невольно должен выбрать некоторое абстрактное

представление предмета рассмотрения, т.е. определить множество данных, отражающих реальную ситуацию и зафиксировать их.

Применение компьютера для хранения и обработки данных приводит к разделению данных и их смыслового содержания. Компьютер имеет дело с данными как таковыми, а интерпретирующая информация в явной форме не фиксируется.

В алгоритмах для данных используются специальные символические обозначения, которые аналогичны обозначениям в математике, представляют имена величин или их **идентификаторы** и при выполнении алгоритма заменяются конкретными для данной задачи числовыми значениями. В зависимости от вида задачи используются данные разных типов.

Типы данных: целые, вещественные, логические, текстовые (символьные) и другие. Мы определили только те типы, которые будут использоваться в этом учебном пособии.

В алгоритмах и программах используются два вида величин: константы и переменные.

Константа *– это величина, значение которой не изменяется в процессе выполнения алгоритма и программы.*

Переменная *– это величина, значение которой изменяется в процессе выполнения алгоритма и программы.*

Используются простые и индексные переменные (переменные с индексом). Переменная с индексом – это элемент некоторой заданной последовательности значений, которые называются *массивом.*

Массив состоит из компонентов одного типа, поэтому структура массива однородна. Индекс или порядковый номер элемента имеет значение целого типа. С помощью индекса можно выделить любой элемент в массиве. Имена массивов выбираются по тому же правилу, что и имена простых переменных. При этом следует учесть, что имя массива не должно совпадать с именем ни одной простой переменной, используемой в этом же алгоритме.

Массивом *называется совокупность однотипных данных, связанных общим именем.*

Массив может быть одномерным (вектор), количество индексов – один; двумерным, количество индексов – два и т.д (матрицы). Количество индексов определяет размерность массива.

Первый компонент одномерного массива – это элемент с номером 1, второй – элемент с номером 2 и т.д., запись в математике – X_1, X_2,… При программировании запись более формализована: X(1), X(2),… (в некоторых языках программирования допускается использование нулевых и отрицательных индексов).

Двумерный массив – это матрица из горизонтальных строк и вертикальных столбцов. Первый из индексов определяет номер строки, второй – номер столбца. Номер строки изменяется от 1 до N, где N – полное число строк, номер столбца от 1 до M, где M – полное число столбцов. При программировании элементы двумерного массива по имени Y будут записаны Y(1,1), Y(1,2), Y(1,3) и т.д. Индексы отделяются запятыми.

При работе с элементами массивов необходимо задать соответствующие значения индекса (для одномерного массива) или индексов (для двумерного массива). Текущие значения индексов не должны выходить за пределы заданного диапазона, иначе переменная с индексами не может быть определена.

Основными характеристиками массива являются:

– имя массива;
– тип элементов массива;
– размерность, равная количеству индексов (измерений) массива;
– значения верхней и нижней границы для каждого индекса;
– размер (длина) массива – количество компонентов.

Вопросы для самопроверки

1. Назовите основную цель вычислительного процесса.
2. При подготовке задачи к решению, какой этап реализуется раньше: Выбор метода решения или Разработка алгоритма?
3. На каком этапе подготовки задачи к решению определяются требования к точности решения?
4. Назовите методы решения задач. Что между ними общего и в чём отличия?
5. Если задача может быть решена с помощью различных методов, то какой метод следует выбрать?
6. Назовите свойства алгоритмов.
7. Назовите способы записи алгоритма.
8. На чём основана словесная запись алгоритма? В чём недостаток этого способа?
9. На чём основан псевдокод?

10. Назовите наиболее распространенный способ описания алгоритма.
11. Назовите символы, без которых схема алгоритма невозможна.
12. Какой символ используется для проверки условия?
13. Сколько символов Решения будет в схеме вычисления выражения

$$Z = \min(A,B) + \max(C,D)?$$

14. Сколько символов Процесс будет в схеме вычисления выражения, приведенного выше?
15. Как называется величина, значение которой не изменяется в процессе выполнения алгоритма и программы?
16. Что такое массив?
17. В чём отличие между простыми переменными и переменными с индексами?
18. Этапы подготовки исполняемой программы.

2. Типовые структуры алгоритмов

2.1. Линейный алгоритм

Вычислительные процессы описываются следующими типовыми структурами алгоритмов: линейной, разветвлённой и циклической.

Алгоритм линейной структуры – это алгоритм, действия которого выполняются последовательно, одно за другим. Такой порядок выполнения действий называется естественным. Поэтому в схемах алгоритмов линейной структуры нет символа **Решения.**

Задача. *Составить алгоритм вычисления площади треугольника со сторонами A, B, C по формуле Герона:*

$$S = \sqrt{p(p-A)(p-B)(p-C)}, \quad \text{где } p = (A + B + C)/2.$$

Решение. Словесное описание алгоритма будет иметь вид:

1. Ввести A, B, C.
2. Вычислить $p = (A + B + C)/2$.
3. Вычислить $S = \sqrt{p(p-A)(p-B)(p-C)}$.
4. Вывести S.
5. Конец.

Алгоритм имеет линейную структуру при любых исходных данных. И каждое последующее действие следует из предыдущего. Для алгоритмов этой структуры одинакова, наглядна как словесная схема, псевдокод, так и схема алгоритма, представленного на рис. 1. Потоки данных и управления в данной схеме совпадают со стандартными, поэтому стрелки не используются. Такой подход при составлении схем вычислительного процесса будет использован и в дальнейшем (см. ГОСТ 19701-90, п.4.2.1 и 4.2.2 [2]).

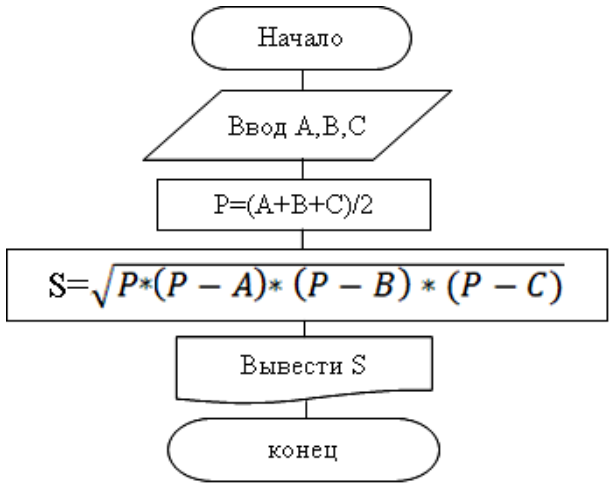


Рис. 1. Схема линейного алгоритма

В большинстве инженерных задач вычислительный процесс зависит от выполнения некоторых условий и естественный порядок выполнения алгоритма нарушается, т.е. имеет место или разветвлённый, или циклический вычислительный алгоритм.

2.2. Разветвлённый алгоритм

Разветвленный (разветвляющийся) вычислительный процесс – это процесс, в котором предусмотрено разветвление выполняемой последовательности действий в зависимости от результата проверки какого-либо условия. В данных алгоритмах естественный поря-

док выполнения действий нарушается. Словесно разветвление описывается следующим образом:

ЕСЛИ условие справедливо (истина), то выполняется Действие 1,
ИНАЧЕ выполняется Действие 2.

Разветвлённый алгоритм содержит блок проверки условия **Решение**, и в зависимости от результата проверки выполняется то или иное действие. Если присутствуют оба действия, то говорят *о полной альтернативе* (рис. 2).

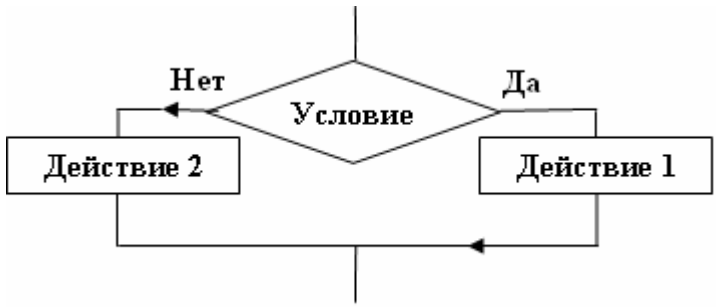


Рис. 2. Полная альтернатива

В алгоритмических языках данный алгоритм реализуется структурным (блочным) оператором IF. Например, в языке Фортран оператор имеет вид:

```
IF (<логическое выражение>) THEN
  <Действие 1>
ELSE
  <Действие 2>
ENDIF
```

Если вместо действия 2 стоит указание **«перейти к пункту №»**, то такая форма записи называется *неполной альтернативой* (рис. 3).

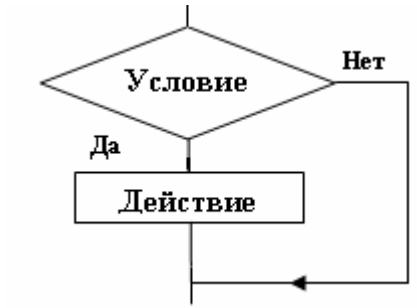


Рис. 3. Неполная альтернатива

Условие – это логическое выражение, которое может принимать два значения – «ДА» (истина) или «Нет» (ложь). Если условие верно, действие выполняется, в противном случае – действие не выполняется.

Если имеется выбор нескольких альтернативных решений, то в схеме нужно предусмотреть несколько выходов. Это можно выполнить несколькими линиями от данного символа к другим, либо одной линией, которая имеет число разветвлений, соответствующих числу выходов.

При выполнении алгоритма после вычисления условий, записанных внутри символа **Решение**, один из выходов будет активизирован.

Соответствующие результаты вычисления могут быть записаны рядом с линиями, отображающими эти пути.

Задача. *Судно заданного проекта с количеством груза G встало под выгрузку сыпучего материала (песчано-гравийная смесь). Выгрузка производится по слоям. Весь груз условно разделен на два слоя: верхний – легкодоступный и нижний – труднодоступный. Процентное содержание груза в каждом слое K_1 и K_2 зависит от типа проекта. Производительность работы крана зависит от его типа, рода груза и номера слоя соответственно P_1 и P_2.*

Рассчитать количество груза, выгруженного в конце 1-й смены. Длительность одной смены 7 часов.

Решение. Проанализируем условие задачи и определим, какие исходные данные, условия и формулы нужно иметь для составления алгоритма решения задачи.

Исходные данные:

- количество груза **G**;
- процентное содержание груза в 1-м слое $\mathbf{K_1}$;
- производительность крана по слоям $\mathbf{P_1}$ и $\mathbf{P_2}$.

Условия:

- время выгрузки 1-го слоя больше времени 1-й смены;
- время выгрузки 1-го слоя меньше времени 1-й смены.

Формулы:

- вычисление количества груза в слое;
- время выгрузки слоя;
- определение количества груза, выгруженного за **n** часов.

Так как задача имеет два пути решения, то алгоритм будет иметь разветвленную структуру.

Словесное описание алгоритма будет иметь вид:

1. Ввести $\mathbf{G, K_1, P_1, P_2}$.
2. Вычислить количество груза в 1-м слое $\mathbf{G_1 = G*K_1}$.
3. Вычислить время выгрузки 1-го слоя $\mathbf{t_1 = G_1:P_1}$.
4. Проверить – за время первой смены кран выгружал только 1-й слой? Т.е. сравнить $\mathbf{t_1}$ и 7.
5. Если $\mathbf{t_1}$ больше или равен 7, то количество выгруженного груза $\mathbf{Z = 7*P_1}$. И перейти к п. 7.
6. Если $\mathbf{t_1}$ меньше 7, то количество выгруженного груза равно $\mathbf{Z = G_1 + (7 – t_1)*P_2}$.
7. Вывод Z.
8. Конец.

Алгоритм имеет разветвлённую структуру. Схема алгоритма представлена на рис. 4.

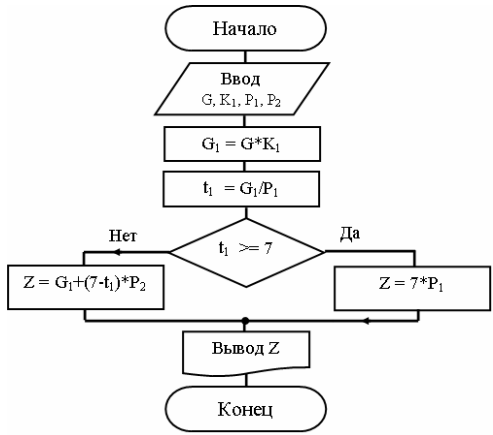


Рис. 4. Схема разветвленного алгоритма

Алгоритм имеет две ветви. В первой ветви, если выполнено условие $\mathbf{t_1} \geq 7$, вычисляется количество выгруженного груза только из 1-го слоя, во второй ветви, если $\mathbf{t_1}$ меньше 7, вычисляется количество груза, выгруженного не только из 1-го слоя, но и из 2-го слоя. После вычисления выполняются общие для обеих ветвей действия ***«Вывод Z»*** и ***«Конец»*** вычислений.

В этой схеме реализован алгоритм с полной альтернативой.

2.3. Циклический алгоритм

Большинство задач, решаемых в инженерной практике, имеют циклическую структуру. Циклическая структура позволяет существенно сократить объём алгоритма, представить его компактно за счет организации повторений большого числа одинаковых вычислений над разными данными для получения необходимого результата.

> *Алгоритмом **циклической** структуры называется алгоритм, в котором предусмотрено неоднократное выполнение одной и той же последовательности действий при различных значениях входящих в них величин.*
>
> *Многократно повторяющиеся участки называются **циклами** или **телом цикла**.*

*Переменная алгоритма, которая при каждом выполнении цикла принимает новое значение, называется **параметром цикла*** (или ***переменной цикла***).

Для организации любого цикла необходимо выполнение следующих условий:

– задание начального значения параметра (переменной) цикла перед началом цикла;

– изменение параметра (переменной) цикла перед каждым новым повторением тела цикла;

– проверка условия окончания (выхода из цикла) или повторения цикла;

– переход к началу цикла, если цикл не закончен, или выход из цикла, если условие выхода выполнено.

Рассмотрим классификацию циклов для лучшего понимания их разновидностей и основных отличий друг от друга. **По месту расположения** условий проверки повторения или окончания цикла можно выделить циклы с предусловием и постусловием.

1. В цикле с предусловием (с предварительным условием) проверка выхода стоит перед телом цикла. Условие записывается в виде логического выражения. Операторы цикла (тело цикла) выполняются, пока условие *истинно.* Если при входе в цикл условие *ложь* (не выполняется), то будет выход из цикла. В этом случае цикл не выполнится ни одного раза.

2. В цикле с постусловием (с последующим условием) проверка выхода стоит после тела цикла. Операторы цикла будут выполняться до тех пор, пока не станет возможным условие выхода из цикла. Цикл выполнится хотя бы один раз.

В инженерных расчетах наиболее распространены циклы с постусловием. В общем виде вычислительная схема представлена на рис. 5.

Реализация циклических алгоритмов в алгоритмических языках выполняется с помощью структурного оператора цикла. Например, в языке Фортран один из вариантов оператора цикла:

```
DO  n= НачалоЦикла, КонецЦикла, ШагЦикла
  <Тело цикла>
ENDDO
```

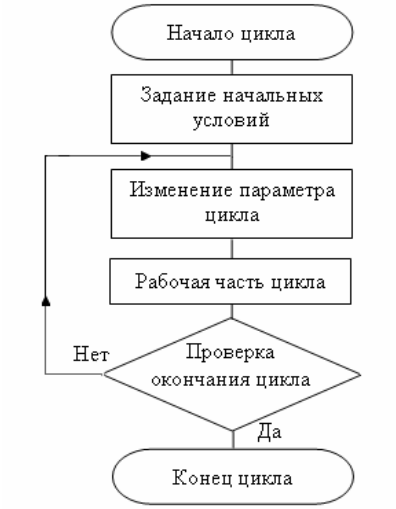


Рис. 5. Типовая схема
циклического алгоритма
с постусловием

По **способу контроля окончания цикла** различают следующие типы циклов.

1. Количество повторений цикла неизвестно (цикл с неизвестным числом итераций). Выход из цикла выполняется по дополнительному условию. Например, вычислить сумму сходящегося ряда

$$S = \sum_{n=1}^{\infty} \frac{n}{n^2 + A}.$$

Условие окончания вычисления

$$\left| \frac{n}{n^2 + A} \right| \le EPS,$$

где *EPS* – очень малая величина, которая определяет точность решения задачи.

Данный тип характерен для математических задач. Схема алгоритма представлена на рис. 6.

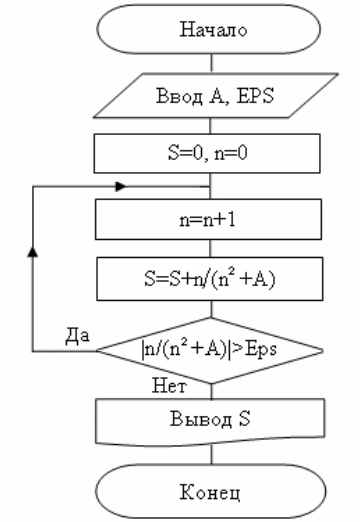


Рис. 6. Схема циклического алгоритма
с неизвестным числом повторений

В инженерной практике при использовании численных методов часто применяется разновидность циклов с неизвестным числом повторений, которые называются итерационными. В итерационных циклах может отсутствовать переменная цикла. В итерационных циклах выполняется закон

$$X_n = F(X_{n-1}),$$

где F – функция;

X_{n-1}, X_n – параметры.

Цикл заканчивается при выполнении условия, связанного с проверкой значения изменяющейся в цикле величины. Как правило, это условие имеет вид

$$\left| X_n - X_{n-1} \right| \le EPS.$$

2. Тип арифметической прогрессии (цикл с известным числом итераций). В этих циклах параметр (переменная цикла) изменяется от заданного начального до заданного конечного значения, получая при каждом выполнении цикла постоянное приращение, которое называется *шагом параметра* цикла. Другое

название этого типа – циклы с параметром. Например, вычислить сумму 100 элементов ряда, представленного выше. Схема алгоритма представлена на рис. 7.

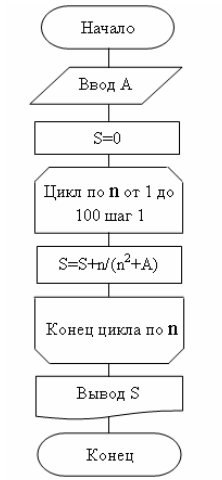


Рис. 7. Схема циклического алгоритма
типа арифметической прогрессии
(число повторений цикла известно)

Алгоритмы решения сложных задач могут включать все перечисленные структуры. Например, внутри одного цикла могут находиться один или несколько других циклов. Охватывающий цикл называется *внешним*, а вложенные в него циклы – *внутренними (вложенными)*, при этом область действия внутреннего цикла должна полностью находиться в области внешнего цикла, т.е. циклы не должны пересекаться. Для этого параметр (переменная цикла) каждого вложенного цикла должен иметь своё **имя.** Правила организации как внешнего, так и внутреннего циклов такие же, как и правила организации простого цикла. Параметры внешнего и внутреннего циклов изменяются не одновременно.

Условия правильного использования циклов. Начальное, конечное значения и шаг должны иметь тот же тип, что и параметр цикла. Если начальное и конечное значения равны, то цикл выпол-

няется один раз. Внутри цикла не рекомендуется изменять параметр цикла и его конечное значение, т.к. эти значения устанавливаются в самом начале работы цикла. Цикл заканчивается, когда параметр цикла принимает конечное значение.

При организации цикла следует особое внимание уделить правильному оформлению изменения параметра цикла, потому что ошибка на этом этапе может привести к «зацикливанию» вычислительного процесса.

Вопросы для самопроверки

1. Назовите основные виды структур алгоритмов.
2. Из каких символов состоит схема алгоритма линейной структуры? Какие символы она не может содержать?
3. Какой порядок выполнения действий называется естественным?
4. Что общего и в чём разница между полной и неполной альтернативой для разветвлённого алгоритма?
5. Что даёт для алгоритма применение циклических структур?
6. Что называется параметром цикла?
7. Что называется телом цикла?
8. Может ли тело цикла не выполняться ни одного раза? В каких случаях?
9. Чем отличается цикл с предусловием от цикла с постусловием?
10. Назовите типы циклов по способу контроля окончания цикла.
11. Какие типы циклов характерны для численных методов?
12. Можно ли в качестве параметра цикла во внешнем и внутреннем цикле использовать одну и ту же переменную?
13. Нарисовать общую структурную схему цикла с предусловием.

3. Примеры алгоритмов разветвлённой структуры

В алгоритмах разветвлённой структуры порядок выполнения действий зависит от проверки условий, заданных в задаче. Происходит выбор одного из нескольких возможных путей вычислений. В этой главе рассмотрим примеры наиболее распространенных алгоритмов для решения задач этого типа. Примеры расположены в порядке возрастания сложности.

3.1. Выбор наибольшего из двух чисел

Задача. *Даны два числа X и Y. Составить схему алгоритма для нахождения наибольшего.*

Решение. Обозначим наибольшее как max, следовательно, нужно составить схему алгоритма вычисления Z = max(X,Y). В данном примере возможны два варианта ответа: X или Y, т.е. реализуется полная альтернатива. Исходные данные могут иметь любые значения.

Выбор варианта будет проведён по результату проверки условия: X > Y? Для однозначности решения считаем, что при X = Y, Z = X. В общем случае местоположение знака равенства определяется постановкой самой задачи.

Алгоритм вычисления будет иметь вид

$$Z = \begin{cases} X, \text{ если} \quad X \geq Y, \\ Y, \text{ если} \quad X < Y. \end{cases}$$

Фрагмент схемы вычисления представлен на рис. 8 (не указаны символы действий **Начало**, **Останов**, **Ввод исходных данных X, Y**).

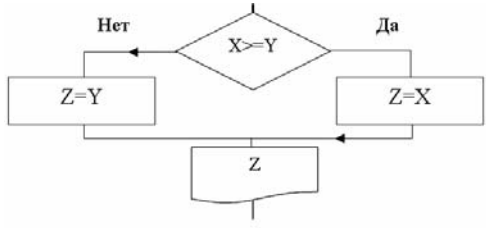


Рис. 8. Фрагмент схемы алгоритма
вычисления Z = max(X, Y)

3.2. Выбор наименьшего из двух чисел

Задача. *Даны два числа X и Y. Составить схему алгоритма вычисления минимального из двух чисел Z = min(X, Y).*

Решение. Как и в предыдущем примере, будет два варианта ответа, и алгоритм можно записать следующим образом:

$$Z = \begin{cases} X, \text{если} \quad X < Y, \\ Y, \text{если} \quad X \geq Y. \end{cases}$$

Фрагмент схемы вычисления представлен на рис. 9.

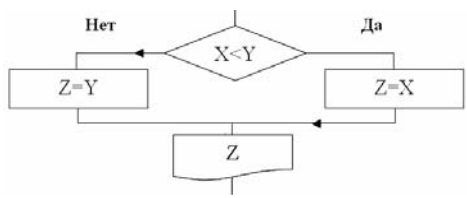


Рис. 9. Фрагмент схемы алгоритма
вычисления Z = min(X, Y)

Как и в предыдущем примере в этой схеме реализована полная альтернатива. Оба алгоритма имеют две ветви вычисления Z. Если условие выполняется, т.е. логическое выражение принимает значения «ДА» (истина), то работает правая ветка. Если условие не выполняется, т.е. выражение принимает значение «НЕТ» (ложь), то работает левая ветка. Одновременно две ветви никогда не будут

работать. В качестве исходных данных могут выступать любые числовые значения.

3.3. Выбор наибольшего из трех чисел

Задача. *Даны три числа A, B, C. Составить схему алгоритма вычисления Z = max(A, B, C).*

Решение. В данном примере возможны три варианта ответа: или A, или B, или C. Выбор может быть выполнен двумя способами.

1. В результате последовательных сравнений: A и B, A и C, далее B и A, B и C. И в конце – C и A, C и B – получаем три альтернативных варианта решения.
2. Используя промежуточную переменную R, выбор можно будет выполнить по результатам проверки двух условий, т.е. получаем два альтернативных решения.

Выберем второй способ, как более рациональный и содержащий наименьшее число действий.

Введем промежуточную переменную R и следующие обозначения:

$$R = \max(A, B),$$

тогда

$$Z = \max(R, C).$$

Алгоритм выбора max из двух переменных рассмотрен в предыдущем примере. Если в предыдущем примере рассматривались исходные данные и ответ, то в этом примере введена дополнительная рабочая переменная **R**. Число вводимых рабочих переменных в любой программе не ограничено (определяется свободной оперативной памятью компьютера).

Алгоритм вычисления будет иметь вид:

$$R = \begin{cases} A, \text{ если} & A \geq B, \\ B, \text{ если} & A < B, \end{cases} \qquad Z = \begin{cases} R, \text{ если} & R \geq C, \\ C, \text{ если} & R < C. \end{cases}$$

Схема вычисления представлена на рис. 10.

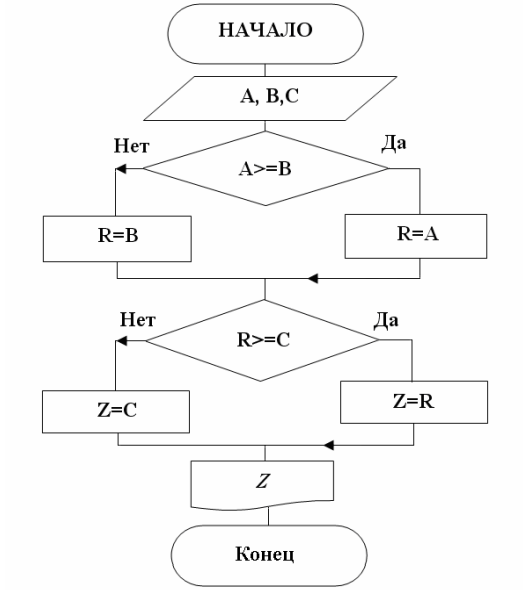


Рис. 10. Схема алгоритма
вычисления Z = max(A, B, C)
Способ с использованием промежуточной переменной

Хотя и эту схему можно уменьшить на один оператор присваивания, если в качестве промежуточной переменной R использовать выходную переменную Z, не вводить переменную R и выполнять следующие вычисления:

$$Z = \max(A, B),\ Z = \max(Z, C).$$

Изменить схему предлагается самостоятельно.

Исходные данные A, B, C могут иметь любые числовые значения.

3.4. Вычисление функции

Задача. *Даны действительные числа A, B, C. Составить схему алгоритма вычисления* $Z = max(A^2 + B, min(B + 1, C))$.

Решение. В данном примере возможно три варианта ответа: или $A^2 + B$, или $B + 1$, или C. И выбор будет выполнен только по результатам проверки условий.

Вычисление величины Z выполняется в два этапа. На первом этапе выбираем min из двух величин B + 1 и C. Результат выбора обозначим через промежуточную переменную R, т.е. R = min(B + 1, C).

Тогда формула для вычисления Z будет иметь вид

$$Z = \max(A^2 + B, R).$$

В предыдущих примерах были разработаны фрагменты схем алгоритмов выбора min и max из двух величин.

Алгоритм вычисления будет иметь вид

$$R = \begin{cases} C, & \text{если} \quad B+1 \geq C, \\ B+1, & \text{если} \quad B+1 < C, \end{cases} \qquad Z = \begin{cases} A^2 + B, & \text{если} \quad A^2 + B \geq R, \\ R, & \text{если} \quad A^2 + B < R. \end{cases}$$

Схема представлена на рис. 11.

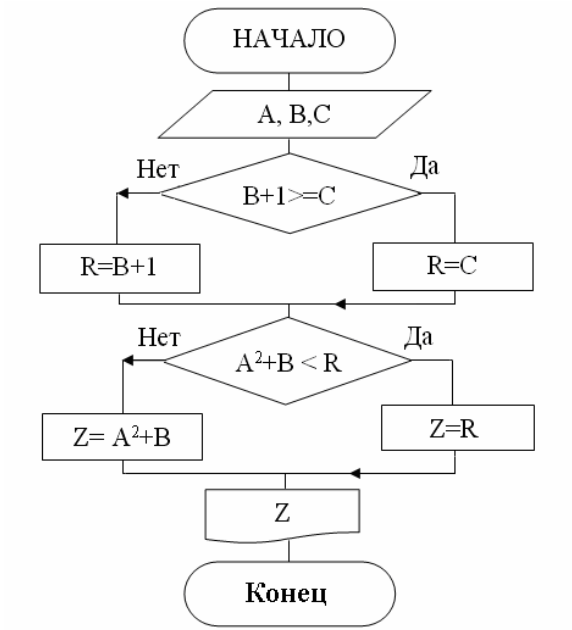


Рис. 11. Схема алгоритма вычисления
$Z = \max(A^2 + B, \min(B + 1, C))$

3.5. Выбор из нескольких условий

Задача. *Даны действительные числа A, B, C. Составить схему алгоритма вычисления Z.*

$$Z = \begin{cases} \max(A, B) + C^3 & \text{, если } A + B + C \geq 0, \\ 0.5(A - B^2) + e^{0.1C} & \text{, если } A + B + C < 0,\ A \geq 0, \\ A^3 + \min(B, C) & \text{, если } A + B + C < 0,\ A < 0. \end{cases}$$

Решение. Схема алгоритма данной задачи имеет разветвленную структуру. Переменная Z вычисляется по одной из формул в зависимости от условий. Первым должно проверяться условие A + B + C ≥ 0. Если условие выполняется, т.е. логическое выражение имеет значение “Да” (истина), то Z вычисляется по первой ветке $Z = \max(A, B) + C^3$.

Алгоритм вычисления max из двух чисел приведен в первом примере. Таким образом, алгоритм вычисления Z по первой ветке будет состоять из следующих действий.

1. Проверка условия A ≥ B? Если условие выполнено, т.е. «Да», то переход к пункту 2, иначе пункт 3.
2. Вычисление $Z = A + C^3$. Переход к пункту 4.
3. Вычисление $Z = B + C^3$.
4. Вывод результата Z.

Если условие A + B + C ≥ 0 не выполняется, т.е. логическое выражение имеет значение «Нет» (ложь), то Z вычисляется либо по второй, либо по третьей ветке. Для выбора ветки нужно проверить дополнительное условие: A ≥ 0. Если условие выполняется, то Z вычисляется по формуле $Z = 0{,}5(A - B^2)^2 + e^{0{,}1C}$ и никаких дополнительных проверок для вычисления Z не требуется. Затем вывод результата Z.

Если A < 0, то Z вычисляется по третьей ветке $Z = A^3 + \min(B, C)$. Алгоритм вычисления min из двух чисел приведен во втором примере.

Вычисление Z будет состоять из следующих действий.

1. Проверка условия B ≥ C? Если условие выполнено, т.е. «Да», то переход к пункту 2, иначе пункт 3.
2. Вычисление $Z = A^3 + C$. Переход к пункту 4.

3. Вычисление $Z = A^3 + B$.
4. Вывод результата Z.

Для рациональности алгоритма организован один общий вывод результата Z. Исходные данные и результат имеют вещественный тип.

В итоге алгоритм вычисления Z имеет пять ветвей, т.е. возможно пять вариантов ответа: или $A + C^3$, или $B + C^3$, или $0{,}5(A - B)^2 + e^{0{,}1C}$, или $A^3 + C$, или $A^3 + B$.

Схема алгоритма представлена на рис. 12.

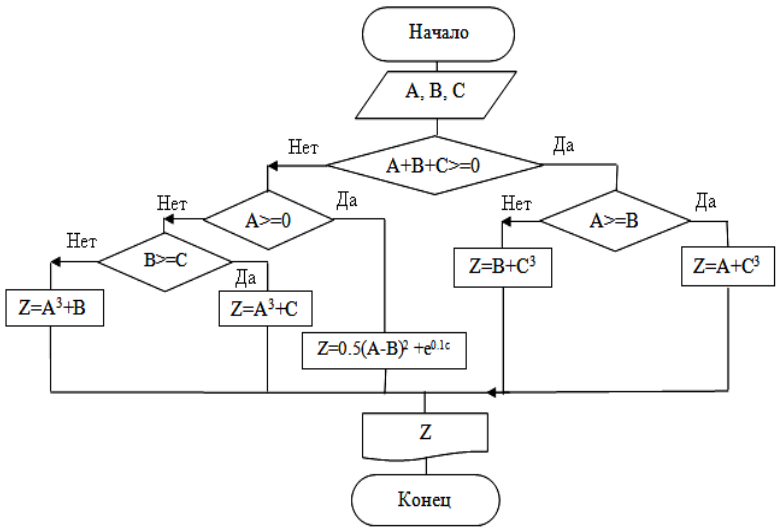


Рис. 12. Схема алгоритма вычисления функции, состоящей из пяти ветвей

Вопросы для самопроверки

Составить алгоритмы для решения следующих задач.

1. Определить, попадает ли точка с координатами X и Y в круг радиуса R. Вывести на печать сообщение о местоположении точки по отношению к кругу.
2. Дана точка с координатами X и Y. Определить, в какой полуплоскости она находится. В верхней или нижней? Правой или левой?

3. Даны две точки с координатами X_1, Y_1 и X_2, Y_2. Вопрос: какая точка наиболее удалена от начала координат?
4. Даны три точки с координатами X_1, Y_1, X_2, Y_2 и X_3, Y_3. Определить, какая из двух точек, первая или вторая, лежит наиболее близко к третьей точке.
5. Дана точка с координатами X и Y. Определить, в каком квадранте она находится.
6. Даны числа A, B, C, D, определяющие длины отрезков. Определить, можно ли из данных четырех отрезков составить прямоугольник.
7. Даны числа A, B, C. Определить, можно ли из данных трёх отрезков составить прямоугольный треугольник.
8. Вычислить значения «y», если:

а) $y = \begin{cases} -1, \text{если} & x < 0, \\ 0, \text{если} & x = 0, \\ 1, \text{если} & x > 0; \end{cases}$

б) $y = \begin{cases} \ln(x), \text{если} & x \geq 1, \\ 1, \text{если} & -1 < x < 1, \\ e^x, \text{если} & x > 1; \end{cases}$

в) $y = \begin{cases} -x - 1, \text{если} & x < -1, \\ 0, \text{если} & -1 \leq x \leq 1, \\ x - 1, \text{если} & x > 1. \end{cases}$

9. Даны числа A, B, C. Вычислить Z = max(A, B, C), не используя промежуточную переменную. Сравнить полученную схему со схемой на рис. 10. На какое число операторов они отличаются?

4. Типовые приемы алгоритмизации

При решении большинства инженерных задач, встречающихся в практике, используется определенный набор типовых приемов алгоритмизации. Далее рассмотрим наиболее распространенные приемы.

4.1. Вычисление суммы и произведения

При вычислении суммы используется прием накопления. Вычисление суммы сводится к ее накоплению в виде значения пере-

менной в цикле, в котором вычисляются соответствующие слагаемые. При этом вновь вычисленное слагаемое прибавляется к сумме предыдущих слагаемых, т.е. в цикле последовательно вычисляются все промежуточные суммы. Поэтому формула, предназначенная для накопления суммы, имеет вид

$$S = S + y,$$

где y – очередное слагаемое;
S – промежуточная сумма.

После первого выполнения цикла первая промежуточная сумма должна быть равна значению первого слагаемого. Следовательно, начальное значение S должно быть равно нулю.

При вычислении произведения используется тот же прием накопления. Вычисление произведения сводится к его накоплению в цикле в виде значения переменной, при этом в цикле вычисляются последовательно все промежуточные произведения. Формула для вычисления произведения имеет вид

$$P = P*y,$$

где y – очередной сомножитель;
P – промежуточное произведение.

Начальное значение P, которое задается перед циклом, должно быть равно единице. Среди начинающих пользователей распространена следующая ошибка: величине P не присваивается начальное значение. Но во многих алгоритмических языках, если переменная не определена, то ей присваивается значение 0, следовательно, произведение вычисляться не будет. Среди членов произведения не должно быть нулевых значений.

Вычисление суммы и произведения рассмотрим на следующем примере.

Задача. *Задан массив по имени A, состоящий из 20 элементов A_i; i = 1, …, 20. Составить схему алгоритма вычисления суммы и произведения элементов этого массива.*

Решение. В соответствии со смыслом описываемых величин выбираем имя переменных: для суммы – S, произведения – P.

Алгоритм вычисления будет состоять из следующих шагов.

1. Ввод массива A_i; i = 1, …, 20.

2. Задание начальных значений переменных S и P. S = 0, P = 1.
3. Организация цикла. Задаются параметры цикла: начальное значение параметра цикла 1, конечное значение 20, шаг 1.
4. Вычисление промежуточных значений S и P, т.е. накопление S и P. $S = S + A_i$; $P = P*A_i$.
5. Проверка окончания цикла. Если параметр цикла меньше конечного значения, то увеличиваем переменную цикла на шаг, т.е. на 1 и переходим к п. 4. Если переменная цикла больше конечного значения, то следующим выполняется п. 6.
6. Печать вычисленных значений S и P.
7. Конец.

Схема вычисления представлена на рис. 13.

Рис. 13. Схема алгоритма вычисления суммы и произведения элементов массива

4.2. Вычисление количества элементов

При вычислении количества элементов используется прием накопления, как и при вычислении суммы и произведения. Но в отличие от этих величин переменная, описывающая количество, должна иметь целый тип. Если не задано дополнительных условий, то правило изменения количества таково:

$$K = K + 1.$$

Начальное значение K = 0.

Задача. *Задан массив X, состоящий из 20 элементов, X_i; i = 1, ..., 20. Составить схему алгоритма вычисления суммы и количества положительных элементов массива.*

Решение. В предыдущем примере была разобрана схема вычисления суммы 20 элементов массива. В этом примере вычисляется сумма и количество только положительных элементов. Поэтому внутри цикла нужно сделать проверку очередного элемента на знак. Алгоритм вычисления будет состоять из следующих пунктов.

1. Ввод массива X_i; i = 1, …, 20.
2. Задание начальных значений переменных S = 0, K = 0.
3. Организация цикла. Задаются начальное и конечное значение переменной цикла и шаг цикла.
4. Проверка очередного элемента X_i на знак. Если условие $X_i \geq 0$? «Да» (истина), то переход к п. 5, если «Нет» (ложь) – то к п. 6.
5. Накопленные суммы $S = S + X_i$ и увеличение счетчика K на 1, K = K + 1.
6. Проверка окончания цикла.
7. Печать значений S и K.
8. Конец.

Схема алгоритма вычисления представлена на рис. 14.

4.3. Нахождение максимального и минимального элементов в заданной последовательности

Задача. *Задан массив y_j; j = 1, ..., 30. Найти максимальный элемент этого массива.*

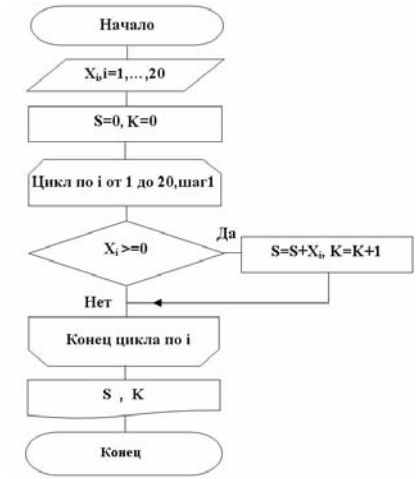


Рис. 14. Схема алгоритма вычисления суммы
и количества положительных элементов массива

Решение. Поиск максимального (наибольшего) элемента массива выполняется в цикле путём последовательного сравнения значения текущего элемента массива с максимальным элементом из всех предыдущих. И если значение текущего элемента больше максимального из всех предыдущих, то максимуму присваивается значение текущего элемента. В *j*-м цикле для выбора максимального элемента используется следующая формула:

$$y_{\max} = \begin{cases} y_i, & \text{если } y_i > y_{\max}, \\ y_{\max}, & \text{если } y_i \le y_{\max}. \end{cases}$$

После окончания цикла значение $y_{\max}$ будет максимальным из всёх рассмотренных значений y_j.

Для применения указанного способа необходимо перед началом цикла задать начальное значение $y_{\max}$, некоторый эталон переменной. Например, значение первого элемента массива. И поиск в цикле начинается со второго элемента. При первом выполнении цикла ($j = 2$) $y_{\max}$ будет сравниваться с y_2. И если y_2 будет больше $y_{\max}$, то меняем эталон $y_{\max}$, присваивая переменной $y_{\max}$ значения y_2. И продолжаем сравнение, теперь уже со следующим элементом.

Можно взять в качестве эталона очень маленькое число (значение маленького числа определяется типом используемого процессора в компьютере и типом этого числа). Тогда после выполнения первого цикла ($j = 1$) y_{max} будет равен y_1. Данный приём используется в циклах с простыми переменными.

Алгоритм вычисления будет состоять из следующих пунктов.

1. Ввод исходного массива y_j; $j = 1, \ldots, 30$.
2. Задание начального значения $y_{max} = y_1$.
3. Организация цикла. Задаются параметры цикла: начальное значение переменной цикла 2, конечное значение 30, шаг цикла 1.
4. Сравнение очередного j-го элемента y_j и y_{max}. Проверяется условие $y_j \geq y_{max}$? Если условие выполняется, т.е. «Да», то переход на пункт 5, если «Нет», то – на конец цикла.
5. Присвоение $y_{max} = y_j$.
6. Конец цикла.
7. Печать максимального элемента массива y_{max}.
8. Конец.

Схема алгоритма вычисления максимального элемента массива представлена на рис. 15.

Если надо найти минимальный (наименьший) элемент массива, то для выбора минимального элемента используется формула

$$y_{min} = \begin{cases} y_i \text{ , если } y_i < y_{min}, \\ y_{min} \text{ , если } y_i \geq y_{min}, \end{cases}$$

и алгоритм выбора аналогичен рассмотренному выше.

В качестве начальных значений y_{min} может быть взято значение первого элемента массива, и поиск в цикле начнётся со второго элемента. Или заведомо большое число. Больше любого элемента в исходном массиве, и поиск в цикле начнётся с первого элемента.

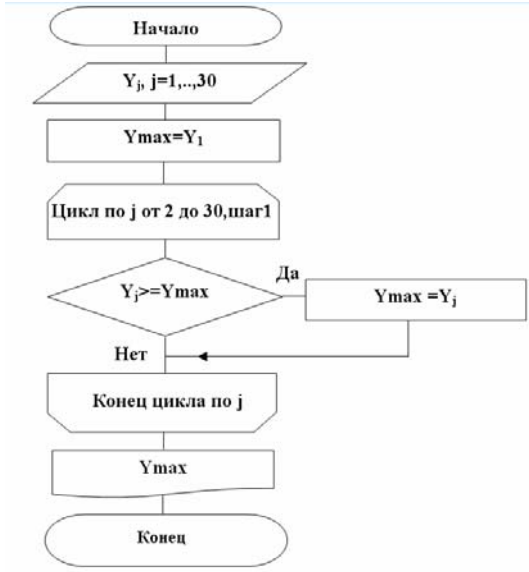


Рис. 15. Схема алгоритма вычисления максимального элемента массива

Задача. *Для массива Y, заданного в предыдущем примере, найти минимальный элемент* $Y_{\min}$ *и его порядковый номер* ($J_{\min}$).

Решение. Особенностью примера является то, что нужно найти не только минимальный элемент, но и его номер. Для решения данной задачи в алгоритм предыдущего примера нужно внести следующие изменения.

2. Задание начального значения $Y_{\min} = Y_1$, $J_{\min} = 1$.
4. Сравнение очередного элемента Y_j и $Y_{\min}$. Если Y_j меньше $Y_{\min}$, то переход на п. 5, если больше – то на конец цикла.
5. Присвоение $Y_{\min} = Y_j$, $J_{\min} = j$.
7. Печать номера минимального элемента $J_{\min}$ и значение минимального элемента $Y_{\min}$.

Так как цикл начнет выполняться со второго элемента массива, то может оказаться, что первый элемент Y_1 будет минимальным. Поэтому в п. 2 задаётся начальное значение не только $Y_{\min} = Y_1$, но и $J_{\min} = 1$.

Схема алгоритма аналогична схеме на рис. 15 с соответствующими дополнениями.

4.4. Структуры с вложенными циклами

Программы решения многих задач требуют нескольких циклов. Например:

– упорядочение массивов;
– обработка массивов;
– расчет таблицы значений функций, заданной степенным рядом.

В этих случаях важно правильно определить структуру алгоритма, прежде всего количество и относительное расположение циклов. В этих структурах могут использоваться рассмотренные приёмы алгоритмизации, но при этом необходимо определить, в каком цикле (внешнем или внутреннем) будет использоваться тот или иной приём.

Например, вычислить сумму всех элементов матрицы. В этом примере начальное значение суммы элементов нужно задать перед внешним циклом, а накапливать её во внутреннем цикле.

Если необходимо вычислить сумму элементов каждой строки матрицы, то начальное значение суммы нужно задать перед внутренним циклом, в котором перебираются и суммируются элементы одной строки матрицы. При этом внешним обязательно должен быть цикл, изменяющий номер строки, а внутренним – изменяющий номер столбца.

Задача. *Задана матрица X(N*M). Определить количество положительных элементов в каждой строке матрицы.*

Решение. Матрица X имеет размер N*M, т.е. в ней N строк и M столбцов. Вводить и печатать матрицу будем в общепринятом виде, т.е. по строкам. Вывод матрицы на печать необходим для получения результата в удобном для чтения виде.

Алгоритм решения состоит из следующих пунктов.

1. Ввод матрицы X(N, M), где i = 1, …, N; j = 1, …, M.
2. Печать матрицы X(N, M), где i = 1, …, N; j = 1, …, M.
3. Организация внешнего цикла по строкам. Переменная цикла i изменяется от 1 до N, шаг 1.

4. Задание начального значения счетчика количества положительных элементов для каждой строки матрицы K = 0.
5. Организация внутреннего цикла по столбцам. Параметр цикла j изменяется о 1 до M, шаг 1.
6. Проверка очередного элемента X(i, j) на знак. Если X(i, j) больше или равно нулю, то переход на п. 7, иначе – на п.8.
7. Увеличение счетчика количества положительных элементов K на 1, K = K + 1.
8. Конец внутреннего цикла по столбцам.
9. Печать номера строки i и количества положительных элементов K.
10. Конец внешнего цикла по строкам.
11. Конец.

Схема алгоритма вычисления приведена на рис. 16.

Анализ данной схемы позволяет дать следующую рекомендацию: внутренний цикл необходимо переводить в исходное состояние непосредственно перед его началом (положение пункта обнуления счетчика количества положительных элементов).

Следующая рекомендация касается написания программ. Для наглядности вложенность циклов можно указывать отступами.

Вопросы для самопроверки

1. Почему при вычислении суммы начальное значение задают равным нулю?
2. Чему равно начальное значение произведения?
3. Каков закон изменения количества элементов?
4. В каком случае при нахождении максимального элемента последовательности чисел в качестве начального значения берётся очень маленькое число?
5. В массиве несколько совпадающих по значению максимальных элементов. Номер какого максимума алгоритм выведет на печать – первого или последнего?

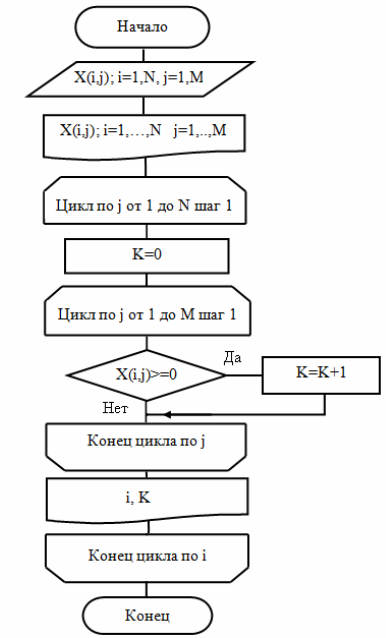


Рис. 16. Схема алгоритма определения количества положительных элементов в каждой строке матрицы

6. Заданный массив содержит положительные и отрицательные числа. Что можно сказать о соотношении между максимальным элементом всего массива и максимальным элементом среди положительных элементов? А как соотносятся между собой минимальный элемент всего массива и минимальный элемент среди отрицательных элементов?
7. Разработать схему алгоритма нахождения максимального и минимального элементов среди положительных элементов массива.
8. Массив содержит положительные и отрицательные элементы. Вычислены: сумма всех элементов массива S и сумма абсолютных величин всех элементов S1. Что можно сказать о соотношении между S и S1? Они равны, S больше S1 или S меньше S1?
9. Разработать схему алгоритма вычисления среднего арифметического значения положительных элементов массива.

10. Если при отыскании экстремального элемента массива в качестве начального значения берется очень большое число, то какой экстремум отыскивается в массиве? Минимум или максимум?
11. Разработать схему алгоритма нахождения максимальных элементов в каждом столбце матрицы.
12. Разработать схему алгоритма вычисления суммы элементов всей матрицы и сумм элементов в каждой строке.
13. В массиве несколько совпадающих по значению минимальных элементов. Что нужно сделать, чтобы вывести номер первого найденного минимального элемента?
14. Можно ли в качестве начального значения для минимума / максимума из положительных взять первый элемент массива?

5. Табулирование функций

В инженерных расчетах и при создании вычислительных программ широко используется задача табулирования функции, т.е. расчет значений функции на заданном интервале с заданным шагом. При этом вид функции и шаг может меняться.

Например, в численных методах эта задача используется при отыскании корня трансцендентного уравнения. Для решения этой задачи выполняется поиск интервала, в котором расположен корень. На концах этого интервала функция имеет разные знаки.

Пример из инженерной практики. Расчет грузовой шкалы судна (таблица грузового размера судна). По этой шкале при ведении грузовых работ определяется масса погруженного груза. Таблица рассчитывается с постоянным шагом по осадке, от осадки порожнем до полной осадки.

Расчет режима наполнения камеры шлюза. Расчет ведется по времени от начального момента до момента заполнения камеры.

Расчет для заданной балки, на которую действует определенная нагрузка эпюр поперечных сил и изгибающих моментов, действующих по длине балки, расчет прогибов балки.

При организации печати рекомендуется выводить на печать исходные данные с соответствующими пояснениями. Например, при расчете балки надо указать ее характеристики и нагрузку.

Таблицам результатов должны предшествовать заголовки, а если функция зависит от параметров, то нужно указать и значения параметров. Таким образом, печать должна быть такой, чтобы в последующем пользоваться этим результатом, не обращаясь к тексту задачи.

5.1. Табулирование функции одной переменной

Математическая формулировка задачи следующая.

Вычислить и напечатать таблицу значений аргумента X и функции Y = F(x) при изменении аргумента X на отрезке от X_{min} до X_{max} с шагом Dx.

Вывод результатов оформим в следующем виде:

X	Y
X_{min}	Y1
X_{min} + Dx	Y2
	Y3
……	
X_{max}	Yn

Алгоритм состоит из следующих этапов.

1. Ввод исходных данных X_{min}, X_{max}, DX.
2. Печать исходных данных.
3. Организация цикла по x от начального значения X_{min} до конечного значения X_{max} с шагом DX.
4. Вычисление Y = F(x).
5. Печать X, Y.
6. Конец цикла.
7. Конец.

Схема алгоритма представлена на рис. 17.

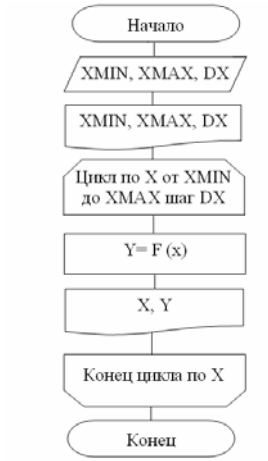


Рис. 17. Схема алгоритма табулирования функции одной переменной

При анализе результатов проверяется верность изменения аргумента X (аргумент увеличивается при каждом цикле на DX) и совпадение значений Y с контрольными расчетами. Обычно для контроля расчет проводится в начальный и конечной точке таблицы, т.е. при $X = X_{min}$ и $X = X_{max}$.

5.2. Табулирование функции на двух участках с разными шагами

Математическая формулировка задачи.

Вычислить и напечатать таблицу значений аргумента X и функции Y при условии, что X изменяется на отрезке от X_{min} до X_{max}, причем шаг изменения X и выражение для вычисления функции Y различны в разных участках отрезка изменения аргумента X. Формула вычисления Y имеет вид

$$Y = \begin{cases} F_1, & \text{при } X_{\min} \le X \le X_{SR} \text{ с шагом } DX_1, \\ F_2, & \text{при } X_{SR} < X \le X_{\max} \text{ с шагом } DX_2. \end{cases}$$

Данный алгоритм содержит два цикла.

Первый цикл выполняет табулирование функции $Y = F_1(x)$ на отрезке от X_{min} до X_{SR} с шагом DX_1. Схема этого алгоритма описана в предыдущем примере.

Второй цикл вычисляет значение функции $Y = F_2(x)$ от $X_{SR}+DX_2$ до X_{max} с шагом DX_2. Начальная точка $X_{SR} + DX_2$ выбрана из условия задачи $X_{SR} < X$. Схема вычисления аналогична схеме в первом цикле.

Общая схема алгоритма вычисления представлена на рис. 18.

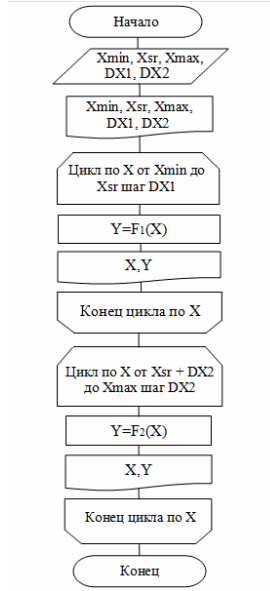


Рис. 18. Схема алгоритма табулирования функции на двух участках с разными шагами

5.3. Табулирование функции двух переменных

Математическая формулировка задачи следующая.

Вычислить и напечатать таблицу значений аргументов X, Y и функции Z = F(X, Y) при изменении аргумента X на отрезке от X_{min} до X_{max} с шагом DX и аргумента Y – на отрезке от Y_{min} до Y_{max} с шагом DY.

Алгоритм решения данной задачи – это алгоритм циклического типа с вложенными циклами. При этом по одному аргументу X или Y будет внешний цикл, а по другому – внутренний, что определяется условиями задачи. При этом циклы не

должны «пересекаться» и область действия внутреннего цикла должна полностью находиться в области действия внешнего цикла.

Рассмотрим табулирование функции Z = F(X,Y). Внешний цикл организуем по X, а внутренний по Y.

Вывод результатов оформим в следующем виде:

X	Y	Z
X_{min}		
	Y_{min}	Z1
	Y_{min}+DY	Z2
…….	.....	
	Y_{max}	Zn
X_{min}+dx		
	Y_{min}	Zn+1
……		

Вывод результатов на печать можно оформить и по-другому.

Алгоритм состоит из следующих этапов.

1. Ввод исходных данных X_{min}, X_{max}, DX, Y_{min}, Y_{max}, DY.
2. Печать исходных данных.
3. Организация внешнего цикла по аргументу X от начального значения X_{min} до конечного значения X_{max} с шагом DX.
4. Печать значения X.
5. Организация внутреннего цикла табулирования Z = F(X, Y) по аргументу Y от Y_{min} до Y_{max} с шагом DY.
6. Конец цикла по Y.
7. Конец цикла по X.
8. Конец.

Если табулирование Z = F(X,Y) будет проводиться для фиксированного значения Y, то изменения будут в параметрах циклов. Внешний цикл – параметр Y, внутренний – параметр X (рис. 19).

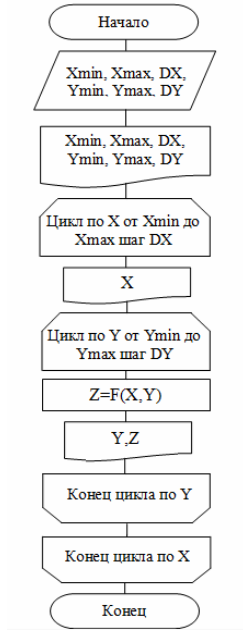


Рис. 19. Схема алгоритма
табулирования функции
двух переменных

Вопросы для самопроверки

1. Как будет работать программа табулирования на отрезке $[X_{min}, X_{max}]$, если при вводе вместо значения X_{min} по ошибке будет введено значение X_{max}?
2. Что будет выдано на печать при табулировании Y = F(X) на отрезке $[X_{min}, X_{max}]$, если оператор печати будет стоять в конце цикла?
3. Какой вид будет иметь схема табулирования Y = F(X) на отрезке $[X_{min}, X_{max}]$, если Y вычисляется по формуле

$$Y = \begin{cases} F(x), & \text{если } X_{min} \le X < X_{SR}, \\ F(x) + F1(X), & \text{если } X_{SR} \le X \le X_{max}. \end{cases}$$

4. Составить схему алгоритма вычисления таблицы значений X и Y = F(X) при изменении X на отрезке $[X_{min}, X_{max}]$ с шагом DX и найти:

a) сумму положительных и отрицательных значений функции;
b) произведение положительных и отрицательных значений функции;
c) отношение произведения всех значений функции к их сумме;
d) среднее арифметическое значение функции;
e) максимальное значение функции;
f) минимальное значение функции;
g) минимальное значение из положительных значений функции;
h) максимальное значение из отрицательных значений функции.

5. Составить схему алгоритма вычисления сложной функции Y = F(X), которая имеет следующий закон:

   $Y = F_1(X)$ на интервале $X_{min} \le X < X_{sr}$ шаг DX;
   $Y = F_2(X)$ на интервале $X_{sr} \le X \le X_{max}$ шаг DX2.

Определить:

a. На каком отрезке лежит абсолютный максимум – на 1-м или 2-м?
b. Количество положительных значений Y на 1-м и 2-м отрезках. Указать, на каком отрезке их больше.
c. Сумму положительных значений Y на 1-м и 2-м отрезках. Указать, на каком отрезке сумма больше.
d. Среднее арифметическое отрицательных значений функции на 1-м и 2-м отрезках.
e. Минимальное значение функции на всем интервале. На каком отрезке оно лежит?
f. Произведение отрицательных значений на 1-м и 2-м отрезках.
g. Произведение положительных и отрицательных значений на всем интервале. Какая из найденных величин больше на всем интервале? Какая из найденных величин больше по абсолютной величине?

6. Составить схему алгоритма вычисления таблицы значений аргументов X, Y и функции Z = F(X,Y) при изменении X на отрезке $[X_{min}, X_{max}]$ с шагом DX и Y на отрезке $[Y_{min}, Y_{max}]$ с шагом DY и найти:

a. Максимальное значение функции Z и значения аргументов, при которых оно достигнуто.
b. Максимальное значение функции Z для каждого фиксированного значения аргумента X.
c. Минимальное значение функции Z для каждого фиксированного значения аргумента Y.
d. Сумму положительных и отрицательных значений функции Z. Какая сумма больше по абсолютной величине?

e. Произведение всех значений функции Z для каждого фиксированного значения X.
f. Количество положительных и отрицательных значений функции Z. Каких значений больше?
g. Среднее арифметическое значение функции Z в заданной области.
h. Среднее арифметическое положительных значений функции Z для каждого фиксированного значения аргумента X.

6. Алгоритмы поиска данных

В программировании часто встречаются задачи поиска в заданной последовательности элемента или нескольких элементов с заданными свойствами. Существуют два основных варианта организации поиска данных.

1. Поиск номера элемента последовательности с заданным значением. Или поиск элемента больше (меньше) заданного значения и т.д.

2. Поиск максимального или минимального элемента и его номера в заданной последовательности.

Последовательность для поиска может быть как упорядоченная, так и неупорядоченная.

Алгоритмы вычисления задач второго варианта изложены в главе «Типовые приёмы алгоритмизации». В этой главе рассмотрим примеры первого варианта поиска данных.

6.1. Поиск номера элемента последовательности с заданным значением

Задача. *Дан массив X_i, $i = 1, ..., N$ и число Y. Найти номер элемента в массиве, значение которого равно Y, $X_i = Y$.*

Решение. Поиск в массиве реализуется методом простого перебора элементов массива, пока не будет найден элемент, равный эталонному значению Y. Если элемент будет найден, то его индекс будет минимально возможным. Но в заданном массиве может не оказаться элемента со значением, равным эталонному, хотя массив будет просмотрен от начала до конца.

Это циклический алгоритм типа арифметической прогрессии, поэтому цикличность поиска, количество повторений не превышает количество элементов в заданной последовательности.

Если предполагается, что в заданном массиве может быть только один элемент, равный эталонному значению, то после того как будет найден этот элемент, поиск можно закончить.

Если в массиве несколько элементов, равных эталону, то нужно задать дополнительное условие: номер какого из найденных элементов необходим. Иначе список просматривается от начала до конца (i = 1, …, N) и последний элемент массива, равный Y, будет считаться результатом поиска. Алгоритм схемы такого поиска представлен на рис. 20.

В алгоритм введена переменная K, перед началом поиска ей присваивается начальное значение K = 0. Если элемент будет найден, то K равно номеру этого элемента, если элемент не найден, то K = 0.

Алгоритм поиска состоит из следующих этапов.

1. Ввод Y и массива X(i), i = 1, …, N.
2. Присвоение начального значения K = 0.
3. Организация цикла по i от 1 до N, шаг 1.
4. Сравнение: X(i) = Y? Если «Да», то переход на п. 5, если «Нет» – то переход на п. 6.
5. Присвоение K = i.
6. Конец цикла по i.
7. Сравнение: K = 0? Если «Да», то переход к п. 8, если «Нет» – то к п. 9.
8. Печать сообщения «Элемент не найден». Переход на 10.
9. Печать номера элемента K.
10. Конец.

Схему алгоритма см. на рис. 20.

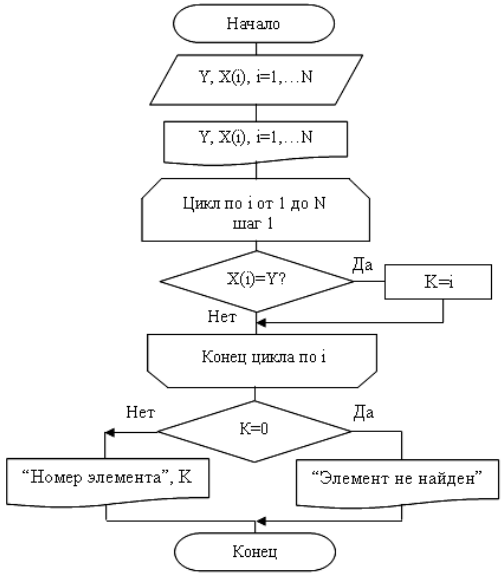


Рис. 20. Схема алгоритма поиска заданного элемента

6.2. Формирование новой последовательности

Рассмотрим следующую задачу: задана последовательность элементов. Сформировать новую последовательность из элементов, обладающих определёнными признаками.

Для поиска элементов, обладающих определёнными свойствами, в первую очередь организуется цикл, в котором просматриваются все элементы массива.

Во вторую очередь необходимо организовать формирование текущего индекса новой последовательности. Для этого выбирается целая переменная – индекс, перед циклом ей присваивается нуль. И как только элемент исходного массива последовательности удовлетворяет заданному признаку, к индексу прибавляется единица, а элементу нового массива присваивается значение элемента исходной последовательности.

Для контроля правильности результатов рекомендуется вывести на печать исходную и вновь сформированную последовательности.

Задача. *Итоги первого вступительного экзамена заданы массивом оценок M_i, i = 1, ..., 50. Сформировать списки абитуриентов, допущенных к следующему (второму) экзамену. Ко второму экзамену будут допущены абитуриенты с оценками $M_i \geq 3$. В списке результатов указать текущие номера студентов, допущенных ко второму экзамену.*

Решение. В этой задаче нужно отыскать элементы больше или равные трем и сформировать из них новый массив. В схеме введены следующие обозначения:

K – счётчик числа абитуриентов на 2-й экзамен (индекс нового массива);
L – массив номеров абитуриентов, допущенных ко 2-му экзамену;
ML – массив положительных оценок.

Алгоритм поиска состоит из следующих этапов.

1. Ввод массива оценок M(i), i = 1, …, 50.
2. Печать исходного массива M(i), i = 1, …, 50.
3. Присвоение начального значения K = 0.
4. Организация цикла по i от 1 до 50, шаг 1.
5. Сравнение M(i) ≥ 3? Если «Да», то переход на п. 6, если «Нет» – то на п. 7.
6. Формирование элементов нового массива.
   K = K + 1
   L(K) = I
   ML(K) = M(i).
7. Конец цикла по i.
8. Печать новых массивов L(j), ML(j), j = 1, …, K.
9. Конец.

Схема алгоритма формирования нового массива представлена на рис. 21.

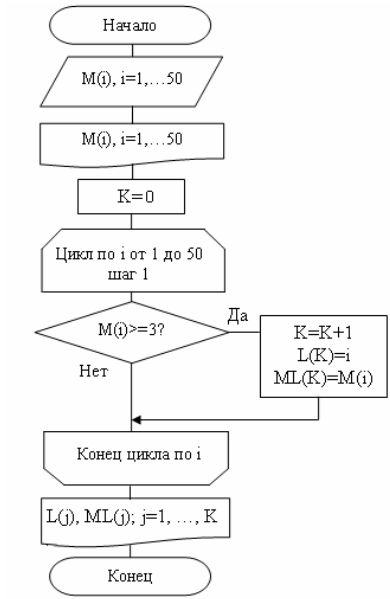


Рис. 21. Схема алгоритма
формирования нового массива

6.3. Согласование с государственным стандартом расчётной величины

В инженерной практике широко распространена задача согласования с государственным стандартом величины, полученной в результате расчётов. Рассмотрим это на примере следующей задачи.

Задача. *Согласовать с государственным стандартом расчётный модуль зацепления открытой зубчатой передачи.*

Дано: M – расчётный модуль и таблица стандартных значений модулей зацепления в виде массива STM(i), i = 1, …, 20.

Решение. Математическая формулировка: найти интервал в таблице стандартов, внутри которого лежит расчётная величина.

$$STM(i) < M < STM(i + 1).$$

Затем согласовать, т.е. заменить расчётное значение на значение из таблицы или на STM(i), или на STM(i + 1). Есть разные типы согласования:

1. Выбор ближайшего к расчётной величине стандартного значения.
2. Выбор наибольшего стандартного значения, т.е. STM(i + 1).

В заданном примере приведем согласование по 1-му условию.

Алгоритм согласования состоит из следующих этапов.

1. Ввод расчётной величины M и таблицы стандартных значений STM(i), i = 1, …, 20.
2. Печать исходных данных M, STM(i), i = 1, …, 20.
3. Организация цикла по i от 1 до 20, шаг 1.

Этапы 4–6. Нахождение интервала, внутри которого лежит M, и запоминание параметра цикла i.

7. Конец цикла по i.

Этапы 8–9. Согласование с государственным стандартом. M – STM(k) > STM(k + 1) – M?

Если «Да», то MGOST = STM(k+1), если «Нет», то MGOST = STM(k).

10. Печать MGOST.
11. Конец.

Схема алгоритма согласования представлена на рис. 22.

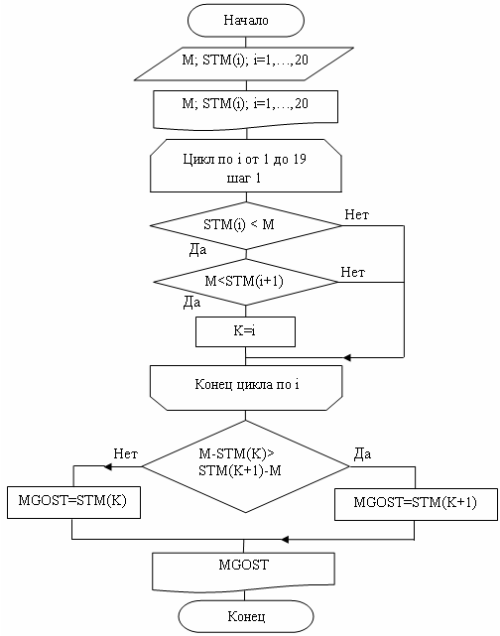


Рис. 22. Схема алгоритма согласования
с ГОСТом расчетной величины

6.4. Нахождение элементов квадратной матрицы, лежащих выше, ниже и на главной диагонали

Рассмотрим квадратную матрицу A(N*N).

1. Выше главной диагонали расположены следующие элементы:

$A_{12}\ A_{13}\ A_{14}\ldots..\ A_{1n}$
$A_{23}\ A_{24}\ldots..\ A_{2n}$
$A_{34}\ldots..\ A_{3n}$
……
A_{n-1n}

Получаем следующий закон изменения индексов i, j

i = 1, j = 2, 3, 4, …, N
i = 2, j = 3, 4, …, N
i = 3, j = 4, …, N
……..

i = N – 1, j = ……. N
i = N, нет элементов

Номер строки i изменяется от 1 до N – 1, номер столбца j изменяется от i + 1 до N, т.е. **j > i**.

2. Ниже главной диагонали расположены элементы:

A_{21}
A_{31} A_{32}
A_{41} A_{42} A_{43}
……
A_{n1} A_{n2} A_{n3} … A_{nn-1}

Закон изменения индексов i, j имеет следующий вид:

i = 1, нет элементов
i = 2, j = 1
i = 3, j = 1, 2
i = 4, j = 1, 2, 3
…….
i = N, j = 1, 2, 3, …, N – 1

Номер строки i изменяется от 2 до N, номер столбца j – от 1 до i – 1, т.е. **j<N**.

3. На главной диагонали лежат элементы, у которых **i = j**.

Задача. *Дана квадратная матрица Y(N*N). Найти минимальный элемент среди элементов, лежащих выше главной диагонали, номер строки и номер столбца, в котором он находится.*

Решение. Алгоритм нахождения минимального элемента последовательности разобран в подразделе 4.3.

Обозначим:

Y_{min} – минимальный элемент матрицы;
I_{min} – номер строки, в которой лежит Y_{min};
J_{min} – номер столбца, в котором лежит Y_{min}.

Алгоритм вычисления состоит из следующих этапов.

1. Ввод матрицы Y(i, j), где i и j изменяются независимо друг от друга от 1 до N с шагом 1.
2. Печать матрицы в общепринятом виде, т.е. по строкам.
3. Задание начального значения Y_{min}.

4. Организация цикла по строкам i.
5. Организация цикла по столбцам j.
6. Сравнение значения текущего элемента матрицы Y(i, j) и Y_{min}. Если $Y(i, j) < Y_{min}$, то переход на п. 7, иначе – на п. 8.
7. Запоминание Y_{min}, номера строки I_{min}, номера столбца J_{min}.
8. Конец цикла по столбцам j.
9. Конец цикла по строкам i.
10. Печать I_{min}, J_{min}, Y_{min}.
11. Конец.

Схема алгоритма определения минимального элемента среди элементов, лежащих выше главной диагонали, представлена на рис. 23.

Вопросы для самопроверки

1. Дан массив Z, состоящий из 15 элементов. Расположить элементы заданного массива в обратном порядке.
2. Дан массив A, состоящий из 30 элементов. Сформировать два новых массива B и C. Массив B состоит из положительных элементов массива A, массив C из отрицательных элементов.
3. Дана информация о прибытии судов в порт в течение месяца. Найти:
   a) судно, первым прибывающее в порт;
   b) судно, последним прибывающее в порт;
   c) суда, прибывающие в порт во 2-й декаде;
   d) количество судов, прибывающих после 20-го числа.
4. Группа из 30 студентов сдала за сессию 5 экзаменов. Сформировать три списка:
   a) студентов, сдавших все экзамены на «отлично»;

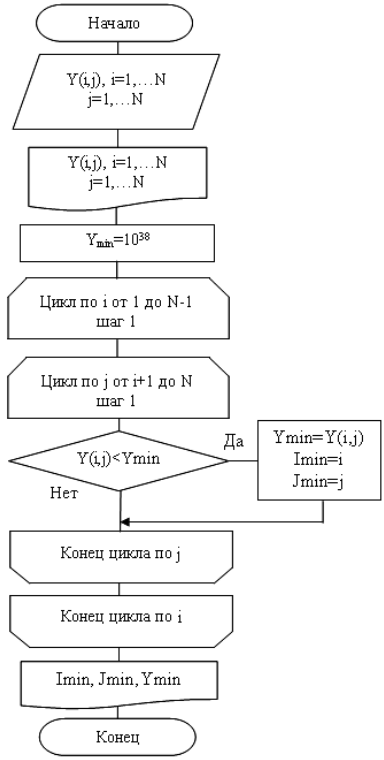


Рис. 23. Схема алгоритма
определения минимального элемента
среди элементов выше главной диагонали

b) студентов, которые будут отчислены (студенты отчисляются, если сдали на «неудовлетворительно» три экзамена). Если таких нет, то напечатать сообщение об этом;
c) студентов, которые будут продолжать обучение.

5. На участке работало 5 кранов. По нормативам кран должен перегрузить за смену количество груза, равное P, а за сутки (3 смены) – количества груза Q. Количество перегруженного за сутки груза представлено матрицей G_{ij}, i = 1, …, 5; j = 1, …, 3.

Определить:

a) номер крана, перегрузившего наибольшее количество груза;
b) номер крана, перегрузившего наименьшее количество груза;

c) номера кранов, перегрузивших больше сменного норматива в каждой из смен;
d) номера кранов, которые выполняли норматив P каждую смену. Если таких нет, то напечатать сообщение об этом;
e) среднее количество груза, перегруженное каждым краном.

7. Сортировка данных

Сортировка – важная задача информатики и вычислительной математики, которая относится к ресурсоемким методам решения алгоритмических задач.

Сортировка данных – это обработка информации, в результате которой ее элементы (записи) располагаются в определенной последовательности в зависимости от значения некоторых признаков этой информации.

Сортировка данных позволяет сократить во много раз продолжительность решения задач, которые связаны с обработкой больших массивов информации. Когда элементы отсортированы, как в телефонном справочнике, их проще найти, обновить, исключить, легче отыскать, какие элементы пропущены.

Смысл любой сортировки заключается в перестановке элементов последовательности в определенном заданном порядке. Упорядочение осуществляется в процессе многократного просмотра исходного массива.

В зависимости от того, где выполняется сортировка, во внутренней оперативной памяти компьютера или на внешних носителях данных, различают методы внутренней и внешней сортировки. В данном пособии рассматриваются только методы внутренней сортировки. И только те из них, которые наглядно показывают их внутренние механизмы.

Не существует алгоритма сортировки универсально наилучшего в любой ситуации. Имеется много наилучших способов, но только в случаях, когда известно, что́ сортируется, на каком компьютере и с какой целью. Эффективность алгоритма будет зависеть от множества факторов:

- сколько элементов участвует в сортировке;
- в какой степени элементы уже отсортированы;

- какой диапазон значений и распределение сортируемых элементов;
- предполагается ли, что элементы будут периодически исключаться или дополняться;
- можно ли сравнивать элементы параллельно.

Метод сортировки является устойчивым, если относительный порядок элементов с равными значениями не меняется после упорядочения.

Для оценки алгоритмов сортировки обычно используют функциональную зависимость времени от количества N сортируемых элементов.

Рассмотрим основные методы сортировки. При разработке алгоритмов рекомендуется выводить на печать исходные данные с соответствующими пояснениями, что позволяет повысить наглядность решения задачи.

7.1. Простой выбор

Идея метода простого выбора состоит в последовательном поиске наименьшего (или наибольшего) элемента массива, начиная с первого элемента до конца массива, и замене первого элемента на найденное значение. Первый элемент встает на место наименьшего элемента.

Далее рассматриваем второй элемент и опять находим наименьший элемент в последовательности, начиная с третьего. Затем меняем их местами.

Выбор продолжается до предпоследнего элемента массива.

Задача. *Дан массив A(i); i = 1, …, N. Отсортировать его методом простого выбора по возрастанию.*

Решение. Алгоритм содержит два цикла вложенной структуры. Во внешнем цикле переменная цикла i изменяется от 1 до N – 1.

Во внутреннем цикле переменная j изменяется от i + 1 до N с шагом 1. В этом

цикле выбирается наименьшее значение среди элементов массива A(j), начиная с i + 1 до N. Поэтому перед началом внутреннего цикла задаются начальные значения минимального элемента массива и его номера.

После окончания внутреннего цикла i-й элемент и найденный наименьший элемент меняются местами.

Схема алгоритма сортировки данных простым выбором по возрастанию представлена на рис. 24.

Этот алгоритм выполняет в среднем N(N – l) / 2 сравнений и 3(N – 1) присваиваний.

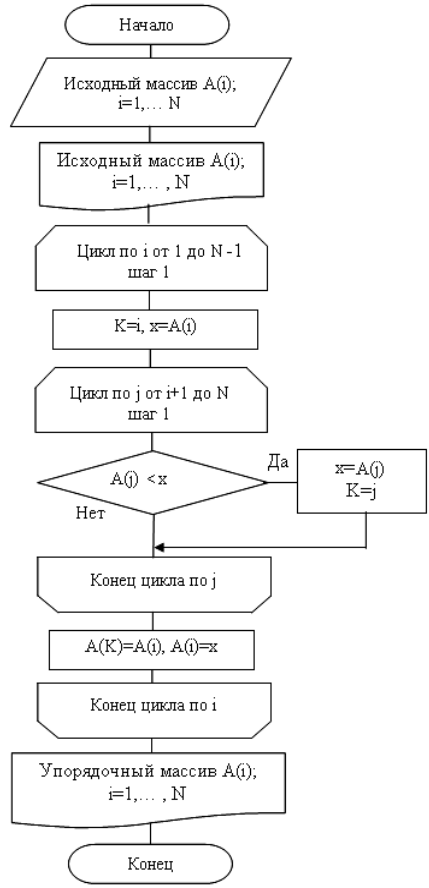


Рис. 24. Схема алгоритма сортировки простым выбором

7.2. Простой обмен

Идея метода заключается в том, что если два стоящих рядом элемента расположены не по порядку, то они меняются местами. Этот процесс повторяется до тех пор, пока элементы не будут упорядочены. Иначе этот метод называется методом пузырьковой сортировки (или просто метод пузырька).

Задача. *Дан массив A(i); i = 1, ..., N. Отсортировать его ме-*

тодом простого обмена по возрастанию.

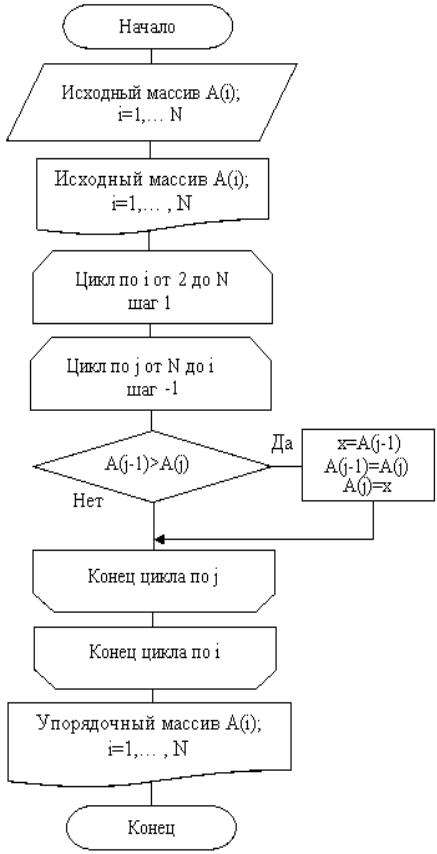


Рис. 25. Схема алгоритма сортировки по возрастанию методом простого обмена

Суть сортировки состоит в многократном сравнении элементов массива, стоящих рядом, и перестановке этих элементов в порядке возрастания. Таким образом, поочередно сравниваются соседние элементы A1 и A2, A2 и A3, A3 и A4, … . И если Ai – 1 > Ai , то элементы меняются местами.

Алгоритм содержит два цикла вложенной структуры. Во внешнем цикле переменная цикла i изменяется от 2 до N.

Во внутреннем цикле переменная цикла j изменяется в обратном порядке от N до i, что напоминает всплывание пузырька в стакане воды. Во внутреннем цикле происходит сравнение смежных элементов и их перестановка, если это необходимо.

Схема алгоритма сортировки данных методом простого обмена представлена на рис. 25.

Алгоритм обменной сортировки осуществляет максимально возможное количество сравнений, много раз приходится просматривать список и выполнять много перестановок. Это дает повод считать его неэффективным алгоритмом. Увеличить воз-

можности алгоритма можно, например, чередованием проходов «вперед» и «назад».

7.3. Последовательное упорядочение пар смежных элементов

Идея метода заключается в последовательном сравнении смежных элементов и перестановке их местами, если это требуется.

Задача. *Дан массив X(i); i = 1, ..., N. Отсортировать его методом упорядочения пар смежных элементов по убыванию.*

Решение. Обозначим через K счетчик количества выполненных перестановок. Алгоритм содержит два цикла вложенной структуры. Во внешнем цикле осуществляется проверка ненулевого значения переменной K.

Во внутреннем цикле по переменной цикла i организуется последовательное сравнение смежных элементов X(i) и X(i + 1). И если X(i) > X(i + 1), то меняем их местами и к счетчику прибавляем единицу.

После окончания внутреннего цикла проверяется число перестановок в последовательности. И если K = 0, то перестановок не было и массив упорядочен. Если K <> 0, то внутренний цикл повторяется снова.

Схема алгоритма сортировки данных методом упорядочения пар смежных элементов представлена на рис. 26.

Использование счетчика количества выполненных перестановок делает данный алгоритм более эффективным, чем предыдущие.

Вопросы для самопроверки

1. Изложите идею сортировки методом пузырька.
2. В каком методе меняются местами два стоящих рядом элемента?
3. Чем отличается метод выбора от метода пузырька?

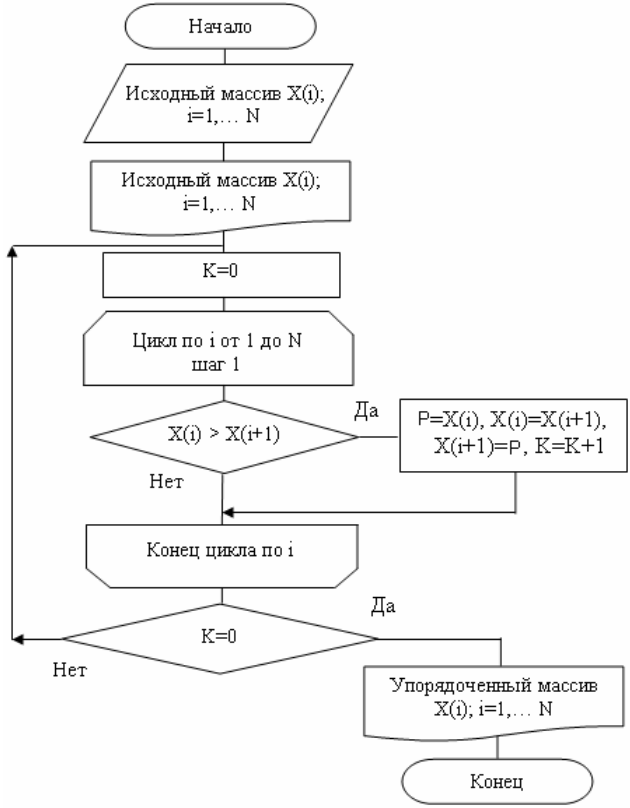


Рис. 26. Схема алгоритма сортировки
методом упорядочения пар смежных элементов

4. От каких факторов зависит эффективность сортировки последовательностей данных?
5. Какая методика используется для оценки эффективности алгоритмов сортировки?
6. В каком методе вводится счетчик количества перестановок и в чем смысл введения этого счетчика?
7. Дан график прибытия судов в порт. Определить очередность их прибытия.
8. Дан результат тестирования группы студентов. Составить рейтинг оценки знаний студентов.

8. Задания для лабораторных работ

Составить схему вычислительного процесса и написать программу на алгоритмическом языке высокого уровня. Входные данные задать самостоятельно.

Задание 1. Разветвляющиеся вычислительные процессы.

№ п/п	Вычислить выражение
1	Z = max(A, min(B, C, D))
2	Z = min(A + 1, max(B, C + D))
3	Y = min(A, max(B + C, D + 1))
4	Z = min(A, max(B, C + D))
5	Z = min(A, B) + max(B, C)
6	Z = max(A + 1, min(B, C + D))
7	Y = max(A, min(B, C – 1))
8	Z = min(A, max(B – D, C))
9	Z = max(A, min(B, C + 1, D))
10	Y = min(A+1, min(B – 1, C + D))
11	Z = max(A, max(B + C, D + 1))
12	Z = min(A^2, max(B, C, D + 1))
13	Y = min(A, B^2) + 2.5max(B, C + 1)
14	Z = max(A^3 + 1, min(B, C + D))
15	Z = max(A, B + 1) – 0.5min(C, D)
16	Y = min(A + C, min(B – 4, C + 2, D))
17	Z = max(A + 1, B) + 1.5min(C, D – 0.5))
18	Y = min(A, min(B – 4, C, D + 1))
19	Z = max(A + B^3, min(B, C, D))
20	Z = min(A^5 + 1, min(B, C + D))
21	Z = max(A, min(B – C, D – 1))
22	Z = min(A + D, min(B^3 + 2, C))
23	Y = min(A, B) – 0.1max(B, C + 1)
24	Z = max(A + 1, min(B, C + D))
25	Y = max(A, min (B, C – 1, D))
26	Z = max(A + D, max(B – A, C))
27	Y = min(A, B, C) + max(B, C)
28	Z = max(A, min (B, C – 1), D)
29	Z = min(A, B + 0.6) + 0.5max(C, D)
30	Z = min(A + D, max(B^3 + 2, C))

Задание 2. Разветвляющиеся вычислительные процессы (сложное выражение).

№ п/п	Вычислить сложное выражение
1	$Z=\begin{cases}0.25\min(A,B)+\lvert C\rvert, & \text{если } A+B+C\ge 0\\ 0.1(1-A^2)+\dfrac{B}{C+1}, & \text{если } A+B+C<0,\ A\ge 0\\ \min(A,B,C), & \text{если } A+B+C<0,\ A<0\end{cases}$
2	$Z=\begin{cases}A+\max(A+B,C), & \text{если } A+B+C\ge 0\\ B^2+\min(A,B,C), & \text{если } A+B+C<0\end{cases}$
3	$Z=\begin{cases}A+\min(A,B,C), & \text{если } A^2-B^2+C\ge 0\\ 0.2(A+B)^2+e^{0.1C}, & \text{если } A^2-B^2+C<0,\ B\ge 0\\ \max(B,C)+A^2, & \text{если } A^2-B^2+C<0,\ B<0\end{cases}$
4	$Z=\begin{cases}\min(x,y)+R^2, & \text{если } x+y+R^2\ge 0\\ x^3+\max(y,R), & \text{если } x+y+R^2<0,\ R\ge 1\\ 0.1(x-y)^3+e^R, & \text{если } x+y+R^2<0,\ R<1\end{cases}$
5	$Z=\begin{cases}0.5\min(X,Y)+X^{\sin(Y^2+1)}, & \text{если } X+Y\ge 0\\ \sqrt{\max(X,0.5Y)+Y^2}, & \text{если } X+Y<0\end{cases}$
6	$Z=\begin{cases}\max(A,B,C+1)+0.5(C+1)^2, & \text{если } A^2+B+C\ge 0\\ \min(B,C)+0.5\max(A,C), & \text{если } A^2+B+C<0\end{cases}$
7	$Z=\begin{cases}\max(A,B)+C^2, & \text{если } A+2.5B\ge C^2\\ A^3+\min(B,C), & \text{если } A+2.5B<C^2,\ C\ge 0\\ 0.1(A-B)^2+\dfrac{1}{\lvert C+1\rvert}, & \text{если } A+2.5B<C^2,\ C<0\end{cases}$

8	$Z=\begin{cases}\max(A,B+C)+C^2, & \text{если} \quad A+B\ge C\\ A^4+0.5\min(A,B,C), & \text{если} \quad A+B<C\end{cases}$
9	$Z=\begin{cases}\max(A,B)+\min(B,C^2), & \text{если} \quad A^2-B\ge C\\ A^3+\min(B,C), & \text{если} \quad A^2-B<C\end{cases}$
10	$Z=\begin{cases}0.25\min(A,B)+\lvert C\rvert, & \text{если} \quad A+B+C\ge 0\\ 0.1(1-A^2)+\dfrac{B}{C+1}, & \text{если} \quad A+B+C<0,\ A\ge 0\\ \min(A,B,C), & \text{если} \quad A+B+C<0,\ A<0\end{cases}$
11	$Z=\begin{cases}A^2+2.5\max(A+1,B), & \text{если} \quad A+B+C\ge 0\\ B+\min(A,C), & \text{если} \quad A+B+C<0\end{cases}$
12	$Z=\begin{cases}A+\min(A,B,C), & \text{если} \quad A+B+C\ge 0\\ 0.3(1-A^2)+Ln(A+C), & \text{если} \quad A+B+C<0,\ A\ge 0\\ \max(A,B,C), & \text{если} \quad A+B+C<0,\ A<0\end{cases}$
13	$Z=\begin{cases}2.5\min(x,y,R)+R^2, & \text{если} \quad x+y+R\ge 0\\ x^3+\max(y,R), & \text{если} \quad x+y+R<0\end{cases}$
14	$Z=\begin{cases}0.5\max(A,B,C)+A, & \text{если} \quad A+B+C\ge 0\\ 2.1(1-A^2)+\cos\left(\dfrac{B}{C+1}\right), & \text{если} \quad A+B+C<0,\ A\ge 0\\ A^2+\min(B,C), & \text{если} \quad A+B+C<0,\ A<0\end{cases}$
15	$Z=\begin{cases}\max(A,B,C+1)+\sqrt{C+1}, & \text{если} \quad A^2+B+C\ge 0\\ \min(A,B-1,C)+A^3, & \text{если} \quad A^2+B+C<0\end{cases}$
16	$Z=\begin{cases}\max(A,B)+0.4\min(B,C^2), & \text{если} \quad A^2+B^2\ge C\\ \min(A,B,C), & \text{если} \quad A^2+B^2<C\end{cases}$

17	$Z=\begin{cases}\min(A,B)+e^{C}, & \text{если} \quad A+1.5B\ge C^2\\ A^3+\max(B,C), & \text{если} \quad A+1.5B<C^2,\ C\ge 0\\ 3.1(A-B)^3+\sin^2(C+1), & \text{если} \quad A+1.5B<C^2,\ C<0\end{cases}$
18	$Z=\begin{cases}\max(A,B)+0.2\min(B,C), & \text{если} \quad A^2+B-C\ge 1\\ A^2+\min(A+1,B-1,C), & \text{если} \quad A^2+B-C<1\end{cases}$
19	$Z=\begin{cases}\max(A^2-1,B+C)+C^2, & \text{если} \quad A+B\ge C\\ A^4+\min(B-2,C^2), & \text{если} \quad A+B<C\end{cases}$
20	$Z=\begin{cases}\max(A-1,B,C+1)+C^2, & \text{если} \quad A+B^2\ge C\\ 0.1\sin(A)+\min(B,A-C), & \text{если} \quad A+B^2<C\end{cases}$
21	$Z=\begin{cases}0.5\min(A,B-C)+\lvert C\rvert, & \text{если} \quad A+B+C\ge 0\\ 2.1(1-A)+\dfrac{B}{\lvert C+1\rvert}, & \text{если} \quad A+B+C<0,\ A\ge 0\\ \min(A,B,C-1), & \text{если} \quad A+B+C<0,\ A<0\end{cases}$
22	$Z=\begin{cases}A^2+\max(A+B,A-C), & \text{если} \quad A+B+C\ge 0\\ 0.25(C-A)^3+\min(A,B,C), & \text{если} \quad A+B+C<0\end{cases}$
23	$Z=\begin{cases}1.5\max(A,B+C)-\lvert C\rvert, & \text{если} \quad A+B+C\ge 0\\ 2.2(1-A)+\dfrac{B}{\lvert C+1\rvert}, & \text{если} \quad A+B+C<0,\ A\ge 0\\ \max(B-C,C)+A^2, & \text{если} \quad A+B+C<0,\ A<0\end{cases}$
24	$Y=\begin{cases}\min(x,y,R)+R^2, & \text{если} \quad x+y+R\ge 0,\\ \max(x,y)+\min(y-R,R), & \text{если} \quad x+y+R<0\end{cases}$

25	$Z=\begin{cases}1.5\min(A,B+C)-\lvert C-1\rvert, & \text{если } A-B+C\ge 0\\ 2(B-A^2)+\dfrac{B}{\lvert C+1\rvert}, & \text{если } A-B+C<0,\ A\ge 0\\ \max(B+C,C)+A^2, & \text{если } A-B+C<0,\ A<0\end{cases}$
26	$Z=\begin{cases}\max(A,B-C)+\min(A,C), & \text{если } A+B-C\ge 0\\ \min(A,B,C-1)+\sqrt{A^3}, & \text{если } A+B-C<0\end{cases}$
27	$Z=\begin{cases}\max(A,B)+0.2\min(B,C^2), & \text{если } A+B^2\ge C\\ A^3+\min(A,B+1,C-1), & \text{если } A+B^2<C\end{cases}$
28	$Z=\begin{cases}\max(A^2-B,B+C)+\ln(C), & \text{если } A+B\ge C^2\\ A^3+\min(A,B,C), & \text{если } A+B<C^2\end{cases}$
29	$Z=\begin{cases}A^2+\max(A,B,C+1), & \text{если } A^2-B+C\ge 0\\ 2(A+B)^2+e^{0.1C}, & \text{если } A^2-B+C<0,\ B\ge 0\\ \min(B,C)+A^2, & \text{если } A^2-B+C<0,\ B<0\end{cases}$
30	$Z=\begin{cases}C^2+\max(A,B+1,C+1), & \text{если } A^2-B+C\ge 0\\ \ln(A+B)+\cos(C+1), & \text{если } A^2-B+C<0,\ B\ge 0\\ \max(B,C)+\min(A,B), & \text{если } A^2-B+C<0,\ B<0\end{cases}$

Задание 3. Циклические вычислительные процессы.

№ п/п	Условие задачи
1	Вычислить $Z = 0.2X - Y^2$, где X – минимальный элемент массива Ai, i = 1, …, 25, Y – максимальный элемент массива Bj, j = 1, …, 30.
2	Даны массивы Ai, i = 1, … , 15 и Dj, j = 1, …, 20. Найти минимальные элементы этих массивов A_{min} и D_{min}. Определить, какой элемент меньше.
3	Дан массив Xi, i = 1, …, 15. Найти среднее арифметическое значение положительных и отрицательных элементов массива R1 и R2. Определить, какой из найденных значений больше по абсолютной величине.

4	Дан массив Ai, i = 1, 2, …, 15. Найти сумму положительных элементов и сумму отрицательных элементов массива S1 и S2. Определить, что больше по абсолютной величине – S1 или S2.
5	Даны массивы Xi, i = 1, …, 10 и Yj, j = 1, 2, …, 15. Найти максимальные элементы массивов X_{max} и Y_{max}. Определить, какой элемент больше – X_{max} или Y_{max} – и на сколько.
6	Дан массив Xi, i = 1, 2, …, 15. Найти произведения положительных и отрицательных элементов массива P1 и P2. Определить, что больше по абсолютной величине – P1 или P2.
7	Дан массив Xi, i = 1, …, 15. Найти кол-во положительных и отрицательных элементов массива K1 и K2. Определить, каких элементов больше – положительных или отрицательных – и на сколько?
8	Дан массив Xi, i = 1, 2, …, 15. Найти номера минимального X_{min} и максимального X_{max} элементов массива. Определить, какой из элементов лежит ближе к началу массива.
9	Даны массивы Xi, i = 1, …, 10 и Yj, j = 1, 2, …, 15. Найти номера максимальных элементов массивов X_{max} и Y_{max}. Определить, какой из элементов лежит ближе к началу массиву?
10	Вычислить $Z = 0.2X - Y^2$, где X – максимальный элемент массива Ai, i = 1, 25, Y – минимальный элемент массива Bj, j = 1, 30.
11	Даны массивы Ai, i = 1, … , 13 и Dj, j = 1, …, 15. Найти максимальные элементы этих массивов A_{max} и D_{max}. Определить, какой элемент больше.
12	Дан массив Bi, i = 1, …, 15. Найти среднее арифметическое значение положительных и отрицательных элементов массива R1 и R2. Определить, какой из найденных элементов меньше по абсолютной величине.
13	Дан массив Ai, i = 1, 2, …, 15. Найти сумму элементов, стоящих на четных местах массива, S1 и сумму элементов, стоящих на нечетных местах, S2. Определить, что больше по абсолютной величине – S1 или S2.
14	Даны массивы Xi, i = 1, …, 10 и Yj, j = 1, 2, …, 15. Найти номера максимальных элементов массивов X_{max} и Y_{max}. Определить, какой из элементов лежит дальше от начала массива.
15	Дан массив Bi, i = 1, …, 27. Найти произведение элементов, стоящих на четных местах массива, P1 и произведение элементов, стоящих на нечетных местах, P2. Определить, что меньше по абсолютной величине – P1 или P2.
16	Дан массив Di, i = 1, …, 14. Найти кол-во положительных и отрицательных элементов массива K1 и K2. Определить, каких элементов больше, положительных или отрицательных и на сколько?
17	Дан массив Xi, I = 1, 2, …, 15. Найти произведения положительных и отрицательных элементов массива P1 и P2. Определить, что меньше по абсолютной величине – P1 или P2.
18	Даны массивы Xi, i = 1, …, 10 и Yj, j = 1, 2, …, 15. Найти максимальные элементы массивов X_{max} и Y_{max}. Определить, какой из X_{max} или Y_{max} больше и на сколько.
19	Вычислить $Z = 0.3X - Y^3$, где X – максимальный элемент массива Ai, i = 1, …, 25, Y – минимальный элемент массива Bj, j = 1, …, 30.

20	Даны массивы Bi, i = 1, … , 15 и Cj, j = 1, …, 20. Найти минимальные элементы этих массивов B_{min} и C_{min}. Определить, какой элемент меньше.
21	Даны массивы Ai, i = 1, …, 15 и Cj, j = 1, …, 22. Найти среднее арифметическое значение положительных элементов массива Ra и Rc. Определить, какое из найденных значений меньше по абсолютной величине.
22	Дан массив Ci, i = 1, 2, …, 15. Найти сумму положительных элементов и сумму отрицательных элементов массива S1 и S2. Определить, что меньше по абсолютной величине – S1 или S2.
23	Даны массивы Xi, i = 1, …, 10 и Yj, j = 1, 2, …, 15. Найти номера максимальных элементов массивов X_{max} и Y_{max}. Определить, какой из X_{max} или Y_{max} лежит дальше от начала массива.
24	Даны массивы Ai, i = 1, …, 20 и Cj, j = 1, …, 25. Найти количество положительных элементов в каждом массиве. Определить в каком массиве больше положительных элементов.
25	Дан массив Ai, i = 1, …, 30. Найти кол-во положительных и отрицательных элементов массива K1 и K2. Определить, каких элементов больше – положительных или отрицательных – и на сколько?
26	Дан массив Yi, i = 1, …, 25. Найти произведения положительных и отрицательных элементов массива P1 или P2. Определить, что меньше по абсолютной величине – P1 или P2.
27	Даны массивы Xi, i = 1, …, 15 и Yj, j = 1, …, 12. Найти максимальные элементы массивов X_{max} и Y_{max}. Определить, какой элемент меньше – X_{max} или Y_{max} – и на сколько.
28	Дан массив Ai, i = 1, …, 18. Найти сумму положительных и произведение отрицательных элементов массива S и P. Определить, какое значение больше – S или P – по абсолютной величине и на сколько.
29	Даны массивы Ai, i = 1, …, 30 и Cj, j = 1, …, 25. Найти количество отрицательных элементов в каждом массиве. Определить в каком массиве больше отрицательных элементов.
30	Дан массив Bi, i = 1, …, 18. Найти сумму отрицательных и произведение положительных элементов массива S и P. Определить, какое значение больше – S или P – по абсолютной величине и на сколько.

Библиографический список

1. ГОСТ 19781–90. Обеспечение систем обработки информации программное. Термины и определения.

2. ГОСТ 19701–90. ЕСПД. Схемы алгоритмов, программ, данных систем. Условные обозначения и правила выполнения.

3. **Гурьяшова, Р.Н.** Основы алгоритмизации и программирования: метод. указания / Р.Н. Гурьяшова, Ю.С. Федосенко, Л.Н. Шемагина. – Горький : ГИИВТ, 1989. – 90 с.

4. **Князева, М.Д.** Алгоритмика: от алгоритма к программе : учеб. пособие / М.Д. Князева. – М. : КУДИЦ-ОБРАЗ, 2006. – 192 с.

5. Вычислительная техника и программирование : учеб. для техн. вузов / А.В. Петров [и др.]. – М. : Высш. шк., 1990. – 479 с.

Оглавление

Введение…… 3
1. Основы алгоритмизации……... 3
1.1. Постановка задачи…….. 4
1.2. Математическая формулировка задачи…… 5
1.3. Выбор метода решений…….. 6
1.4. Разработка алгоритма……. 6
1.5. Способы описания алгоритмов……. 8
1.5.1. Словесная запись алгоритма……... 9
1.5.2. Псевдокод…….. 10
1.5.3. Схемы алгоритмов…… 10
1.5.4. Реализация алгоритмов…… 14
1.5.5. Тестирование алгоритмов…… 16
1.6. Величины в алгоритмах……. 16
2. Типовые структуры алгоритмов……... 19
2.1. Линейный алгоритм……... 19
2.2. Разветвлённый алгоритм…… 21
2.3. Циклический алгоритм…….. 24
3. Примеры алгоритмов разветвлённой структуры…… 30
3.1. Выбор наибольшего из двух чисел……... 30
3.2. Выбор наименьшего из двух чисел……... 31
3.3. Выбор наибольшего из трех чисел……... 32
3.4. Вычисление функции……. 33
3.5. Выбор из нескольких условий……... 35
4. Типовые приемы алгоритмизации……... 37
4.1. Вычисление суммы и произведения……. 37
4.2. Вычисление количества элементов…….. 40
4.3. Нахождение максимального и минимального элементов в заданной последовательности……. 40
4.4. Структуры с вложенными циклами…….. 44
5. Табулирование функций……... 47
5.1. Табулирование функции одной переменной……... 48
5.2. Табулирование функции на двух участках с разными шагами…… 49
5.3. Табулирование функции двух переменных……. 50
6. Алгоритмы поиска данных……... 54
6.1. Поиск номера элемента последовательности с заданным значением……..... 54

6.2. Формирование новой последовательности…….. 56
6.3. Согласование с государственным стандартом расчётной величины……. 58
6.4. Нахождение элементов квадратной матрицы, лежащих выше, ниже и на главной диагонали…… 60
7. Сортировка данных……... 64
7.1. Простой выбор…… 65
7.2. Простой обмен…… 66
7.3. Последовательное упорядочение пар смежных элементов 68
8. Задания для лабораторных работ……. 70
Задание 1. Разветвляющиеся вычислительные процессы…. 70
Задание 2. Разветвляющиеся вычислительные процессы (сложное выражение)……. 71
Задание 3. Циклические вычислительные процессы……. 74
Библиографический список……... 77

Логинов Вячеслав Иванович
Шемагина Людмила Николаевна

Основы алгоритмизации

Учебно-методическое пособие

Редактор *Н.С. Алёшина*
Корректор *Д.В. Богданов*
Компьютерная вёрстка *М.Е. Савинова*

Подписано в печать 15.12.09.
Формат бумаги 60×84 $^{1}/_{16}$. Гарнитура «Таймс».
Ризография. Усл. печ. л. 4,8. Уч.-изд. л. 5,0.
Тираж 670 экз. Заказ 000.

Издательско-полиграфический комплекс ФГОУ ВПО «ВГАВТ»
603950, Нижний Новгород, ул. Нестерова, 5а