Министерство образования
Российской федерации

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Т.В.Афанасьева

Основы визуальной алгоритмизации

Учебное пособие

Ульяновск 2002

УДК 681.3 (075)
ББК 32.81я73

Т.В.Афанасьева

Основы визуальной алгоритмизации: Учеб. пособие для студентов спец. 5102, 5525, 5501/ Сост :; Под ред. С.Г.Валеева. - Ульяновск , 2002. - с.

Учебное пособие разработано на кафедре прикладной математики и информатики в соответствии с учебными программами для студентов технических и математических специальностей. Содержание включает изложение методических приемов по практическому составлению визуальных алгоритмов, которые могут быть использованы для подготовки к выполнению практических заданий по курсу “Информатика” и "Программирование".

В данной работе определено место проектирования алгоритмов при решении задач на ЭВМ, рассмотрена технология проектирования и способ проверки несложных визуальных алгоритмов, приведено множество примеров и заданий для самостоятельного выполнения, алгоритмическое решение некоторых из них имеется в конце данного учебного пособия. Для проверки полученных знаний можно воспользоваться тестовыми заданиями, приведенными в приложении.

Учебное пособие предназначено для студентов вузов дневной, вечерней, заочной и дистанционной форм обучения.

УДК 681.3 (075)
ББК 32.81я73

Рецензент

Утверждено редакционно-издательским советом
Ульяновского государственного технического
университета в качестве учебного пособия.

ISBN 5-7695-0330-0

Оглавление

ВВЕДЕНИЕ

Решение любой задачи является творческим процессом, который состоит из нескольких последовательных этапов. К ним относятся :

А. Анализ постановки задачи и ее предметной области

1. понимание постановки и требований исходной задачи, определение предметной области, для которой поставлена задача,
2. анализ предметной области, выявление данных, которые фиксируют входную и выходную информацию (определение их структуры и свойств), определение отношений между данными, условий и ограничений, накладываемых на эти отношения,

Б. Формальное моделирование решения задачи

3. выбор и применение формальной системы для описания модели предметной области и решения задачи,
4. формирование основной идеи, выбор методов решения задачи,
5. определение технологий, средств и исполнителя решения задачи, построение алгоритмов, реализующих выбранные методы,

В. Практическое решение

6. применение выбранных методов и средств для решения ,
7. анализ полученных результатов.

Эти этапы ориентированы для получения решения не отдельно взятой, конкретной задачи, а некоторого класса задач данного типа. Этап построения алгоритмов , реализующих выбранные методы решения задачи, детализирует и визуализирует процесс ее решения. Алгоритмизация позволяет уже на этом этапе оценить эффективность решения, уточнить методы решения для различных потоков входных данных и выявить некоторые ошибки.

В этой последовательности наиболее трудоемким и рутинным является этап применения выбранных методов и средств для решения задачи. В настоящее время наиболее распространенным средством для решения задач является ЭВМ. Применение выбранных методов и алгоритмов для решения на ЭВМ включает дальнейшую детализацию ее решения за счет описания последовательности применяемых операций в виде программы для ЭВМ. Это придает процессу решения не только визуальные качества, но и качества интерактивности.

Не все задачи, решаемые с помощью ЭВМ, требуют составления сложных программ. Например, задачи вычислений в электронных таблицах или задачи поиска и выборки данных в базах данных. Решение некоторых задач благодаря внедрению новых информационных технологий вообще не требуют программирования, что расширяет сферу применения ЭВМ. Однако, и при решении этих задач необходимы вышеприведенные этапы.

Целью данной работы является рассмотрение всех вышеперечисленных этапов решения задачи с использованием ЭВМ, при этом наибольшее внимание уделяется этапам составления алгоритмов и соответствующих программ на языке TURBO PASCAL ,так как, на мой взгляд, эти этапы являются

достаточно трудоемкими и важными. Любые ошибки, возникающие на этих этапах, приводят к серьезным погрешностям при решении задач.

Эта работа предназначена для тех, кто не умеет, но стремится научиться использовать ЭВМ при решении задач, составлять корректные алгоритмы и правильные программы. Умение составлять алгоритмы позволит определить детальное решение и может быть использовано при любых технологиях проектирования программ от структурного программирования до объектно-ориентированной и компонентно-ориентированной технологии.

1. АНАЛИЗ ПОСТАНОВКИ ЗАДАЧИ И ЕЕ ПРЕДМЕТНОЙ ОБЛАСТИ

На первом этапе решения задачи уточняется постановка задачи, на основе чего выявляются отдельные явления, объекты, процессы, их связи и зависимости предметной области .

Здесь определяются такие понятия как исходные и результирующие данные, которые абстрактно представляют информацию о процессах предметной области реального мира, и каким образом из исходных данных могут быть получены результирующие.

Исходные данные должны быть полными, т.е. содержать данные, которые необходимы и достаточны для решения задачи. Если данные неполные необходимо приложить дополнительные усилия для сбора дополнительных сведений, но эта ситуация может возникнуть на следующих этапах при определении метода решения.

Различают исходные данные трех видов: постоянные, условно-постоянные и переменные.

Постоянные исходные данные - это такие данные, которые сохраняют свои значения в процессе решения задачи (математические константы, координаты неподвижных объектов) и не зависят от внешних факторов.

Условно-постоянные данные - это такие данные, которые могут иногда изменять свои значения, но эти изменения не зависят от процесса решения задачи, а определяются внешними факторами (величина подоходного налога, курса валют, количество дней в году).

Переменные данные - это данные, которые изменяют свои значения в процессе решения задачи.

На этом этапе важно не только классифицировать данные по отношению к процессу решения, но определить их наименование, тип, структуру и ограничения, накладываемые на значения. Желательно также определить допустимые и недопустимые операции по отношению к различным типам исходных данных.

Классификация данных по структурному признаку

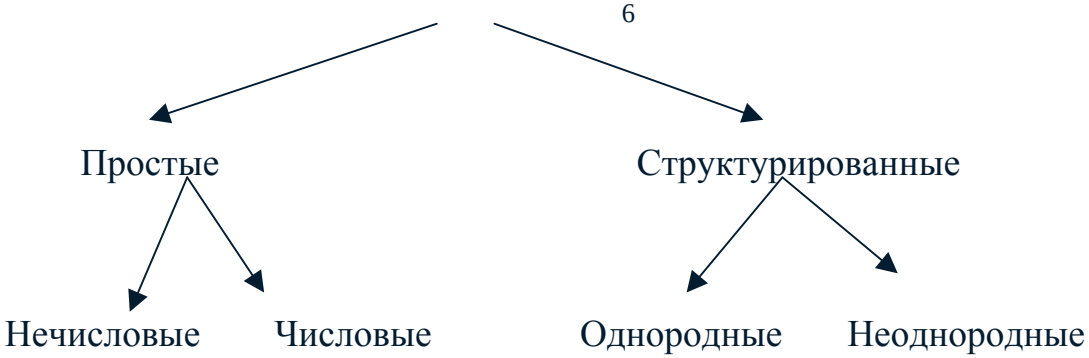


Рис. 1. Классификация данных

На рис.1 представлена классификация данных.

Данное относят к простому типу, если в любой момент времени оно определяет одно и только одно значение. Например, требуется вычислить площадь поверхности некоторого тела. Очевидно, что для представления информации о вычисляемой площади поверхности некоторого тела достаточно использовать данное простого числового типа. Простые данные определяют такое отношение: одно имя - одно значение.

Структурированные данные отличаются от простых тем, что к ним применимо другое отношение: одно имя - много значений. Если все элементы, входящие в такую структуру однотипны, то такая структура называется однородной, в противном случае - неоднородной. Классическим примером однородной структуры является некоторая последовательность простых данных, в виде **массива** значений, таких как, например, (2,51,3,7,88). Неоднородная структура в отличие от однородной содержит значения различных типов, относящихся к одному понятию или объекту, и значит, такое структурированное данное несет в себе больше информации. Для представления неоднородных структур используют **запись**. Запись - это структура, предназначенная для представления данных различного типа.

Рассмотрим простой пример. Задача заключается в определении в некоторой стране города с максимальным количеством жителей. Данные, которые необходимо проанализировать, это нечисловые данные, содержащие информацию о названии города и числовые данные, содержащие информацию о численности в этом городе. В качестве структуры, содержащей данные о названии города и количестве в нем жителей, следует выбрать неоднородную структуру запись, пример которой изображен в таблице 1.

Таблица 1.Пример записи

Название города	Количество жителей
Нечисловой тип	Числовой тип
Москва	8 578 676

В качестве структуры, содержащей информацию о множестве городов рассматриваемой страны, можно выбрать однородную структуру типа массив, состоящий из записей таблицы 1.

Определение отношений между данными, условий и ограничений, накладываемых на значения данных и эти отношения, зависит от конкретной постановки задачи и требований пользователя.

В результате анализа постановка и требования задачи могут быть представлены в обобщенном виде.

2.ФОРМАЛЬНОЕ РЕШЕНИЕ ЗАДАЧИ

После того как был проведен анализ постановки задачи, выявлены данные, их структура и отношения между данными можно приступить к построению формальной модели. Это самый важный этап в процессе решения задачи.

Модель - упрощенное представление о реальном объекте, процессе или явлении. Моделирование - построение моделей для исследования и изучения моделируемого объекта, процесса, явления с целью получения новой информации при решении конкретных задач.

Для описания модели предметной области решаемой задачи необходимо выбрать некоторую формальную систему. Обычно, исходя их постановки задачи, можно сразу определить один или несколько видов моделей, подходящих для описания и моделирования решения вашей задачи: математические, геометрические, структурные, логические и др.

Наиболее распространенными и хорошо изученными являются математические модели. Например, в качестве математической модели звезды можно использовать систему уравнений, описывающих процессы, происходящие в недрах звезды. Математической моделью другого рода являются математические соотношения, позволяющие рассчитать оптимальный план работы предприятия. К основным достоинствам математических моделей безусловно относятся хорошо изученные и широко применяемые математические методы решения большого класса задач, что значительно облегчает формирование основной идеи и выбор методов решения задачи. В дальнейшем будем рассматривать только математические модели.

Приступая к разработке модели, следует попытать решить задачу для конкретных входных данных, затем обобщить полученное решение на основе его анализа для любых значений входных данных. Перед тем, как определить решение задачи для конкретных входных данных целесообразно найти ответ на следующие вопросы:

Существуют ли решения аналогичных задач?

Какая математическая модель больше всего подходит для решения этой задачи?

*Пример 1.*Постановка задачи. Требуется определить подходит ли для проведения учебных занятий данная аудитория.

*Решение.*1 этап. Анализ постановки задачи и ее предметной области.

В результате анализа предметной области, выявляем, что эта предметная область связана с образовательными процессами. И постановка задачи может быть переформули-

рована таким образом. Определить подходит ли некоторая аудитория для проведения занятий группы учеников, при некоторой норме площади для каждого ученика. Введем обозначения для входных и выходных данных. Исходные данные: A - ширина аудитории, B - ее длина, K - количество учеников в группе, N - допустимое минимальное количество квадратных метров для одного ученика (норма), M - количество парт в аудитории.
В качестве выходных данных будут выступать сообщения: " Аудитория может быть использована для поведения учебных занятий " и " Аудитория не может быть использована для поведения учебных занятий ".
2. этап. Формальное решение
Определим отношения между входными и выходными данными. Введем дополнительные понятия: S - площадь аудитории, C - требуемая по нормам площадь для проведения занятий в группе из K учеников, D - требуемое количество парт для обучения группы из K учеников. Опишем соотношения между входными и выходными данными используя математические зависимости. Математическая модель:

S = A*B,
C=N*K, S>=C, K<=2*D.

3.ОСНОВЫ АЛГОРИТМИЗАЦИИ

Слово “алгоритм” появилось в 9-м веке и связано с именем математика Аль-Хорезми, который сформулировал правила выполнения четырех арифметических действий над многозначными числами.

В настоящее время понятие алгоритма - одно из фундаментальных понятий науки информатика. С одной стороны алгоритм является предметом изучения такой отрасли математики как теория алгоритмов (Марков [1]), с другой стороны в информатике существует неформальное определение алгоритма, и алгоритмизация выступает в качестве общего метода информатики.

Объектом приложения алгоритмов являются самые различные науки и области практической деятельности (Хохлюк[3],Ахо [2] []). Широкое применение алгоритмов для решения практических задач не только при использовании ЭВМ позволяет рассматривать эту область информатики как отдельную дисциплину - ***алгоритмику***.

Алгоритм – это точно определенная последовательность действий для некоторого исполнителя, выполняемых по строго определенным правилам и приводящих через некоторое количество шагов к решению задачи.

Исполнитель алгоритмов определяет элементарные действия, из которых формируется алгоритм. Отдельные действия, составляющие алгоритм, называются операциями. При этом под операцией понимается как какое-то единичное действие, например, сложение, так и группа взаимосвязанных действий.

Основными особенностями любого алгоритма являются решение задачи в обобщенном виде и возможность выполнять действия по решению задачи для конкретных значений (не только человеку, но и различным техническим устройствам (исполнителям)). Основным исполнителем несложных алгоритмов является человек. Достаточно вспомнить последовательность действий для решения систем линейных уравнений, вычисления корней уравнений.

При решении сложных задач исполнителем является ЭВМ и составление алгоритма решения задачи является необходимым

этапом, детализирующим метод решения для дальнейшего программирования. Программа осуществляет еще более глубокую детализацию решения и его визуализацию.

Свойства алгоритма:

Определенность – выполнив очередное действие, исполнитель должен точно знать, что ему делать дальше.

Дискретность – прежде, чем выполнить определенное действие, надо выполнить предыдущее.

Массовость – по одному и тому же алгоритму решаются однотипные задачи и неоднократно.

Понятность – алгоритм строится для конкретного исполнителя человеком и должен быть ему понятен.Это облегчает его проверку и модификацию при необходимости .

Результативность – алгоритм всегда должен приводить к результату.

Можно сказать, что в процессе формального решения задачи, ее решение сначала описывается на языке математики в виде системы формул, а затем на языке алгоритмов в виде некоторого процесса, в котором используются ранее определенные математические формулы и условия их выполнения. Таким образом, алгоритм может рассматриваться как связующее звено в цепочке "метод решения - реализующая программа"

4.ОСНОВНЫЕ СРЕДСТВА ПРЕДСТАВЛЕНИЯ АЛГОРИТМОВ

Алгоритм моделирует решение задачи в виде точно определенной последовательности действий для некоторого исполнителя по преобразованию исходных данных в результирующие.

Алгоритм, реализующий решение задачи, можно представить различными способами: с помощью графического или текстового

описания, в виде таблицы значений. Графический способ представления алгоритмов имеет ряд преимуществ, благодаря визуальности и явному отображению процесса решения задачи.

Алгоритмы, представленные графическими средствами, получили название визуальные алгоритмы. Текстовое описание алгоритма является достаточно

компактным и может быть реализовано на абстрактном или реальном языке программирования в виде программы для ЭВМ. Таблицы значений представляют алгоритм неявно, как некоторое преобразование конкретных исходных данных в выходные. Табличный способ описания алгоритмов может быть с успехом применен для проверки правильности функционирования разработанного алгоритма на конкретных тестовых наборах входных данных, которые вместе с результатами выполнения алгоритма фиксируются в "таблицах трассировки".

Таким образом, все три способа представления алгоритмов можно считать взаимодополняющими друг друга. На этапе проектирования алгоритмов наилучшим способом является графическое представление, на этапе проверки алгоритма - табличное описание, на этапе применения - текстовая запись в виде программы.

5.ВИЗУАЛЬНЫЕ АЛГОРИТМЫ

При проектировании визуальных алгоритмов используют следующие графические блоки, представленные на рис. 2.

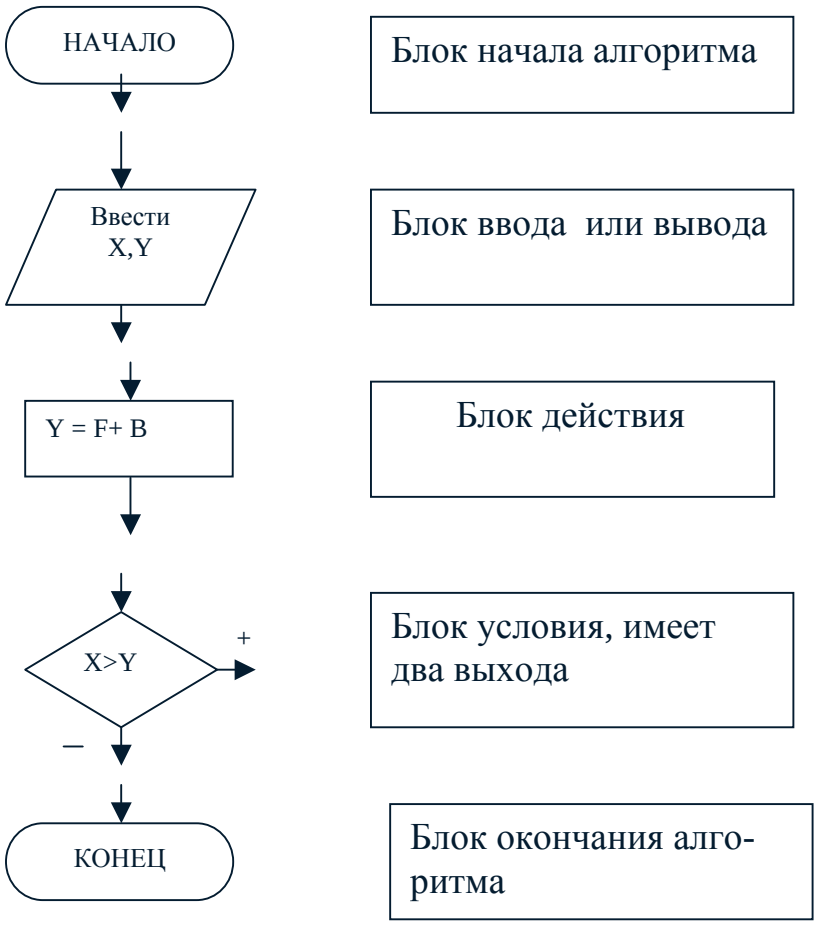


Рис.2. Основные блоки визуальных алгоритмов

Общими правилами при проектировании визуальных алгоритмов являются следующие:

- В начале алгоритма должны быть блоки ввода значений входных данных.
- После ввода значений входных данных могут следовать блоки обработки и блоки условия.
- В конце алгоритма должны располагаться блоки вывода значений выходных данных.
- В алгоритме должен быть только один блок начала и один блок окончания.
- Связи между блоками указываются направленными или ненаправленными линиями.

Этап проектирования алгоритма следует за этапом формального решения задачи, на котором определены входные и выходные данные, а также зависимости между ними.

При построении алгоритмов для сложной задачи используют системный подход с использованием декомпозиции (нисходящее проектирование сверху-вниз). Как и при разработке любой сложной системы, при построении алгоритма используют дедуктивный и индуктивный методы. При дедуктивном методе рассматривается частный случай общеизвестных алгоритмов. Индуктивный метод применяют в случае, когда не существует общих алгоритмических решений.

Одним из системных методов разработки алгоритмов является метод структурной алгоритмизации. Этот метод основан на визуальном представлении алгоритма в виде последовательности управляющих структурных фрагментов. Выделяют три базовые управляющие процессом обработки информации структуры: композицию,
альтернативу и итерацию.С помощью этих структур можно описать любые процессы обработки информации.

Композиция (следование) - это линейная управляющая конструкция, не содержащая альтернативу и итерацию. Она предназначена для описания единственного процесса обработки информации.

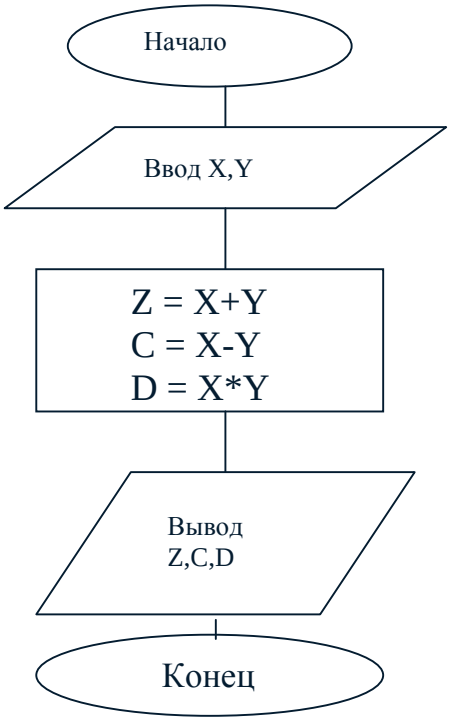
Альтернатива - это нелинейная управляющая конструкция, не содержащая итерацию. Она предназначена для описания различных процессов обработки информации, выбор которых зависит от значений входных данных.

Итерация - это циклическая управляющая структура, которая содержит композицию и ветвление. Она предназначена для организации повторяющихся процессов обработки последовательности значений данных.

В соответствии с наличием в алгоритмах управляющих структур композиции, альтернативы и итерации алгоритмы классифицируют на: линейные, разветвленные и циклические алгоритмы.

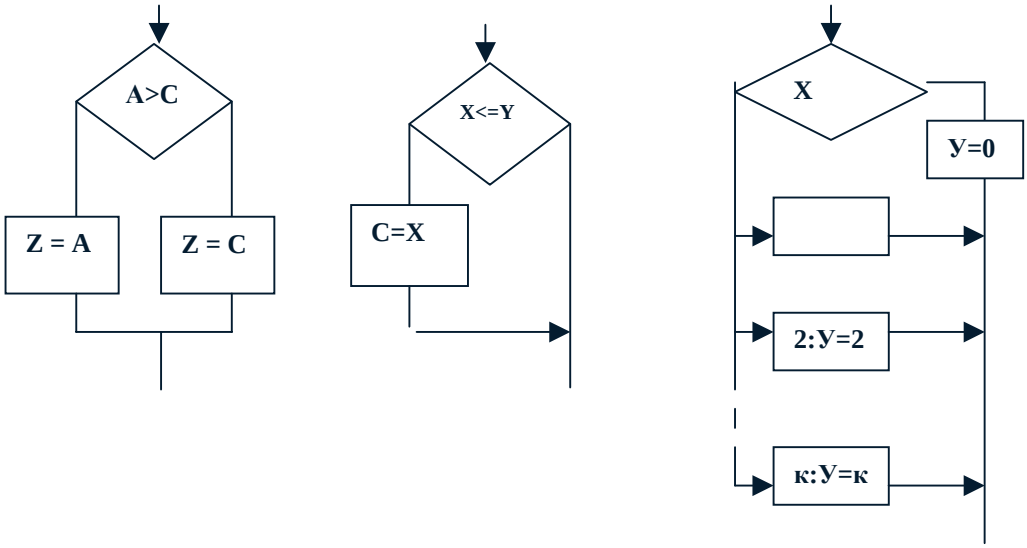
Линейные алгоритмы не содержат блока условия. Они предназначены для представления линейных процессов. Такие алгоритмы применяют для описания обобщенного решения задачи в виде последовательности модулей. Пример линейного алгоритма приведен на рисунке 3.

Рис. 3. Пример линейного визуального алгоритма



6.РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Разветвленные алгоритмы в своем составе содержат блок условия и различные конструкции ветвления. Ветвление - это структура, обеспечивающая выбор между альтернативами.



а) ветвление б)неполное ветвление в) многоальтернативный выбор

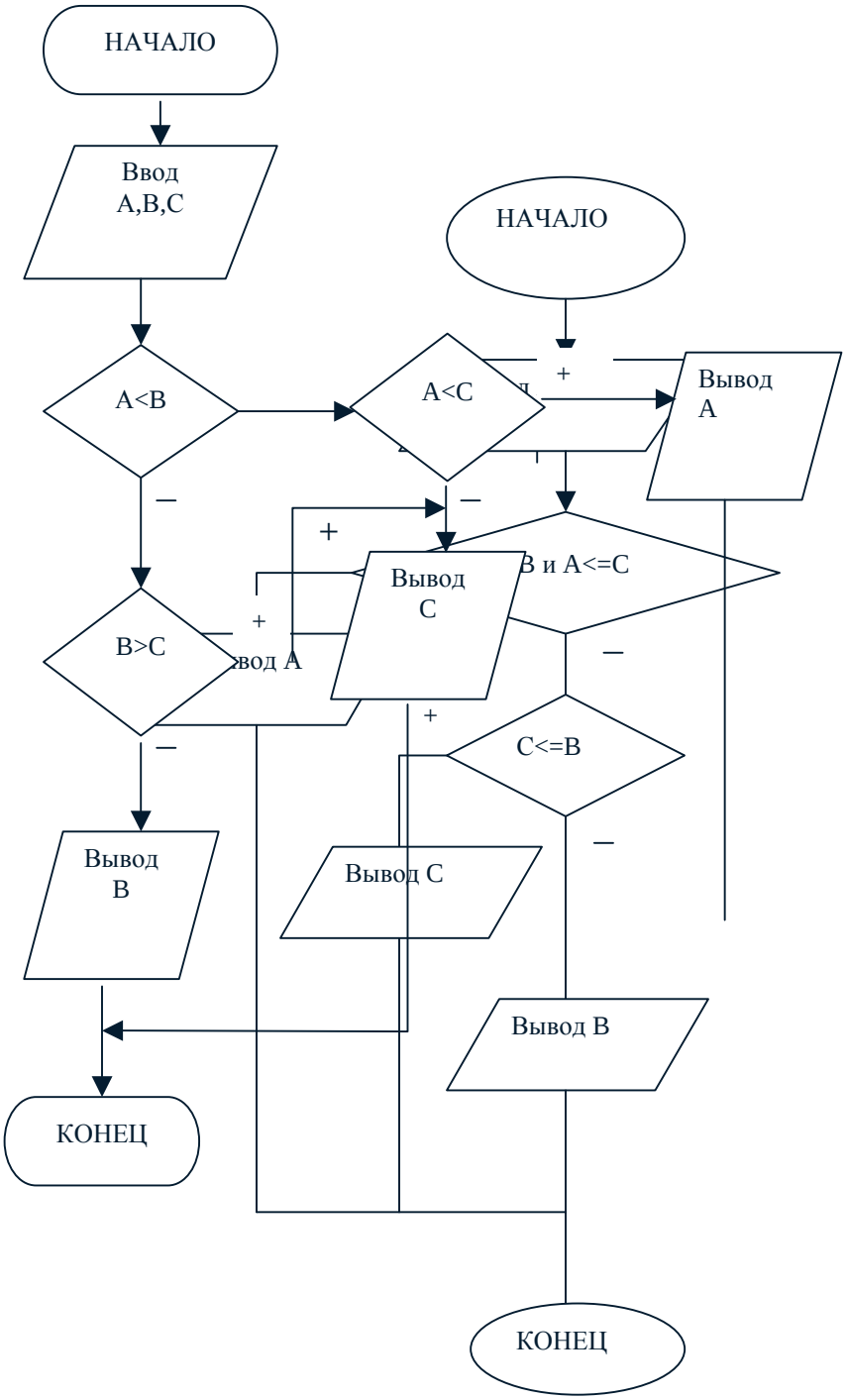
Рис. 4. Структуры ветвления

Каждая управляющая структура ветвления имеет один вход и один выход. Ветвления содержат блок условия, в котором записывают логические условия, такие как A >C , X<= Y. В зависимости от значений переменных A,C в управляющей структуре ветвления на рис. 4 а) условие A >C принимает значение "истина" или "ложь" и процесс вычислений включает блок действия Z=A или Z=C. Аналогично происходит и в управляющей структуре неполного ветвления (рис. 4 б)). Только в этом случае , если условие X<= Y истинно, то выполняется действие C=X, в противном случае никаких действий не выполняется.

В управляющей структуре многоальтернативный выбор (рис. 4в)) в блоке условия записывается переменная, в данном случае X, которая может принимать различные значения. Если значение пременной X совпадет с одним из значений в блоке действия, то выполняется действия , записанные в этом блоке. Например, если X=1, то выполнится действие У=1. Если значение X не совпало ни с одним из значений, указанных в блоках справа, то выполняется действие в блоке слева, которого также как и в неполном ветвлении может и не быть.

*Пример 2.*Составить алгоритм нахождения минимального значения из 3-х чисел. *Решение.* Для определения минимального значения будем использовать проверку пары значений. Визуальные разветвленные алгоритмы приведены на рис.5,6,7. Эти алгоритмы использует для обозначения чисел переменные A,B,C и вложенные структуры ветвления.

Рис. 5. Поиск минимального значения из трех чисел A,B,C при помощи двойного



сравнения.

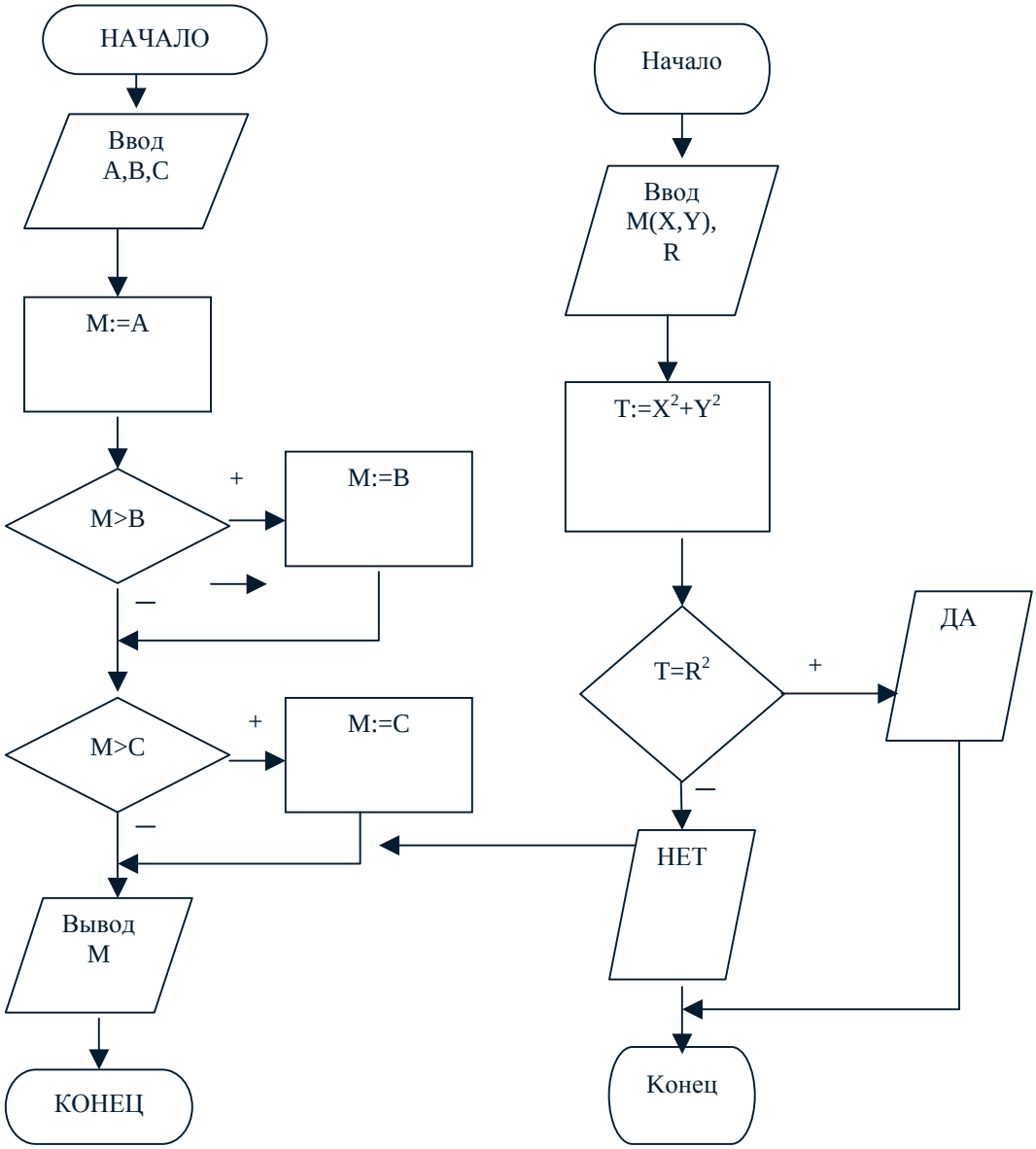


Рис. 6.Поиск минимального числа из трёх A,B,C.
Метод последовательного сравнения .

Пример 3. Составить алгоритм определения находится ли точка М с координатами Х,У на окружности радиуса R.

Решение. Визуальный алгоритм приведен на рис. 8.Для решения в нем используется математическая модель в виде формулы окружности $R^2 = X^2+Y^2$.

Рис. 7.Поиск минимального числа из трёх A,B,C. Метод сравнения с промежуточной переменной M.

Рис. 8. Определить находится ли точка M с координами X,У на окружности радиуса R.

Пример 4. Составить алгоритм определения корней уравнения (X^2+B*X+C=0).

Решение. При составлен[illegible]а надо рассмотреть случаи, когда уравнение не имеет корней и когда имеется только од[illegible]им корни уравнения через переменные X1,X2. D - промежуточная переменная для вычисления дискриминанта. Алгоритм вычисления корней уравнения заданного вида приведен
на рис. 9.

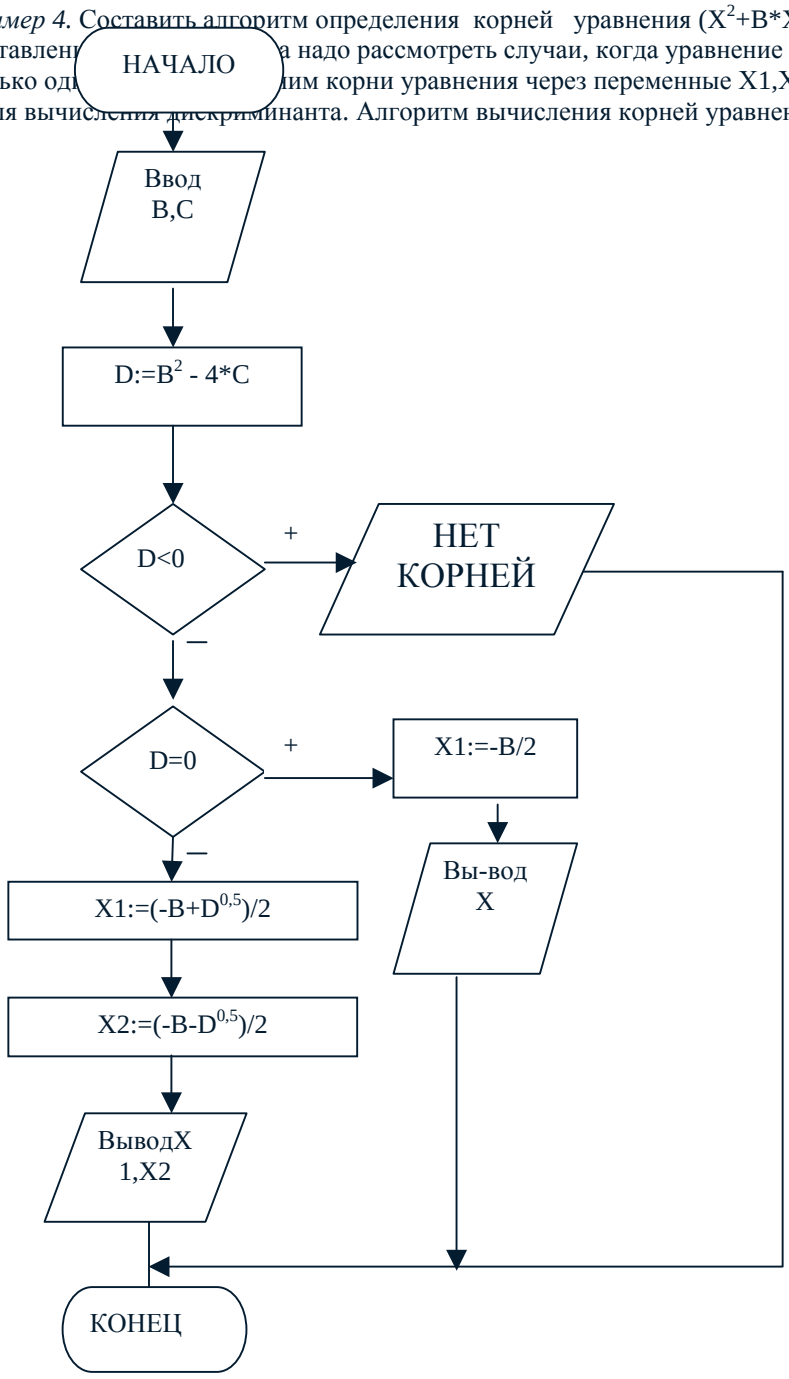


Рис.9. Алгоритм вычисления корней уравнения $X^2+B*X+C=0$

Задания для самостоятельного выполнения

Составить визуальные разветвленные алгоритмы для следующих задач.

1.Для двух чисел X,У определить, являются ли они корнями уравнения A*P^4+D*P^2+C=0
2.Если среди трех чисел A,B,C имеется хотя бы одно четное вычислить максимальное, иначе - минимальное
3.Ввести положительное A>=1. Найти наибольшее из выражений вида 1\A и SIN(A).
4.Ввести два числа . Меньшее заменить полусуммой, а большее - удвоенным произведением.
5.Ввести три числа A,B,C . Удвоить каждое из них , если A>=B>=C, иначе поменять значения A и B.
6.Определить является ли точка с координатами X,Y точкой пересечения диагоналей квадрата со стороной R ,одна вершина которого расположена в начале координат.
7.* Определить значения функции в зависимости от значения аргумента

$$y=\begin{cases} a*x^2, & \text{если } x > 10 \\ 1/x, & \text{если } -10 \le x \le 10 \\ \mathrm{Sin}(x), & \text{если } x < 10 \end{cases}$$

7.ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

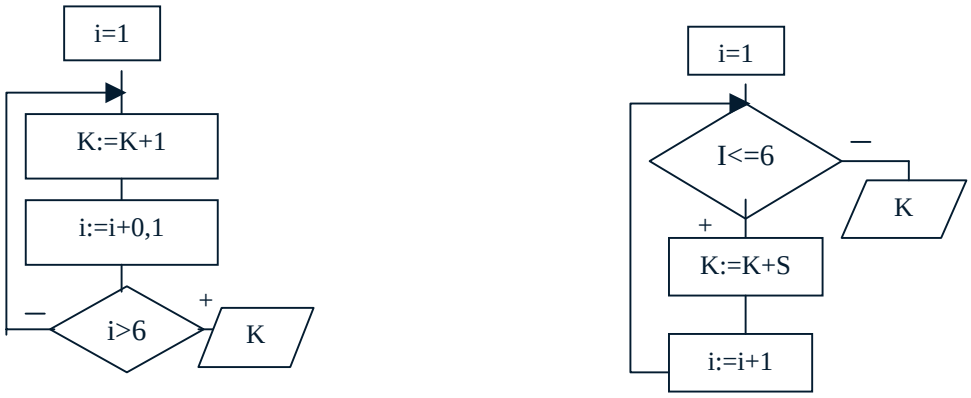
Циклические алгоритмы являются наиболее распространенным видом алгоритмов, в них предусматривается повторное выполнение определенного набора действий при выполнении некоторого условия. Такое повторное выполнение часто называют циклом.

Существуют два основных видов циклических алгоритмов: циклические алгоритмы с предусловием, циклические алгоритмы с постусловием. Они отличаются друг от друга местоположением условия выхода их цикла.

Цикл с предусловием начинается с проверки условия выхода из цикла. Это логическое выражение, например I<=6. Если оно истинно, то выполняются те действия, которые должны повторяться. В противном случае, если логическое выражение I<=6 ложно, то этот цикл прекращает свои действия.

Цикл с постусловием функционирует иначе. Сначала выполняется один раз те действия, которые подлежат повторению, затем проверяется логическое выражение , определяющее условие выхода из цикла, например, I>6 .Проверка его осуществляется тоже по-другому. Если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае - происходит повторение действий, указанных в цикле. Повторяющиеся действия в цикле

называются "телом цикла".Разновидности циклов приведены на рис. 10а),б).



а) Цикл с постусловием б) Цикл с предусловием

Рис. 10. Виды циклических алгоритмов

Классическим примером циклического алгоритма служит алгоритм для вычисления степени числа $Y=X^n$. Этот алгоритм может быть реализован на основе операции умножения. Табличное представление такого алгоритма, отражающего зависимость У от Х при изменении показателя степени n от 1 до 3, представлено в табл.3. В этой таблице показанны также реккурентные соотношения между У и Х, определяющие как на каждом шаге зависит значение У от значения Х и от значения У, вычисленного на предыдущем шаге.

Таблица 3.Реккурентные соотношения при вычислении $Y=X^n$

n	Y	Реккурентные соотношения
1	Y[1]=X	Y=X
2	Y[2]=X*X или Y[2]=Y[1]*X	Y=X*X или Y=Y*X
3	Y[3]=X*X*X или Y[3]=Y[2]*X	Y=X*X*X или Y=Y*X

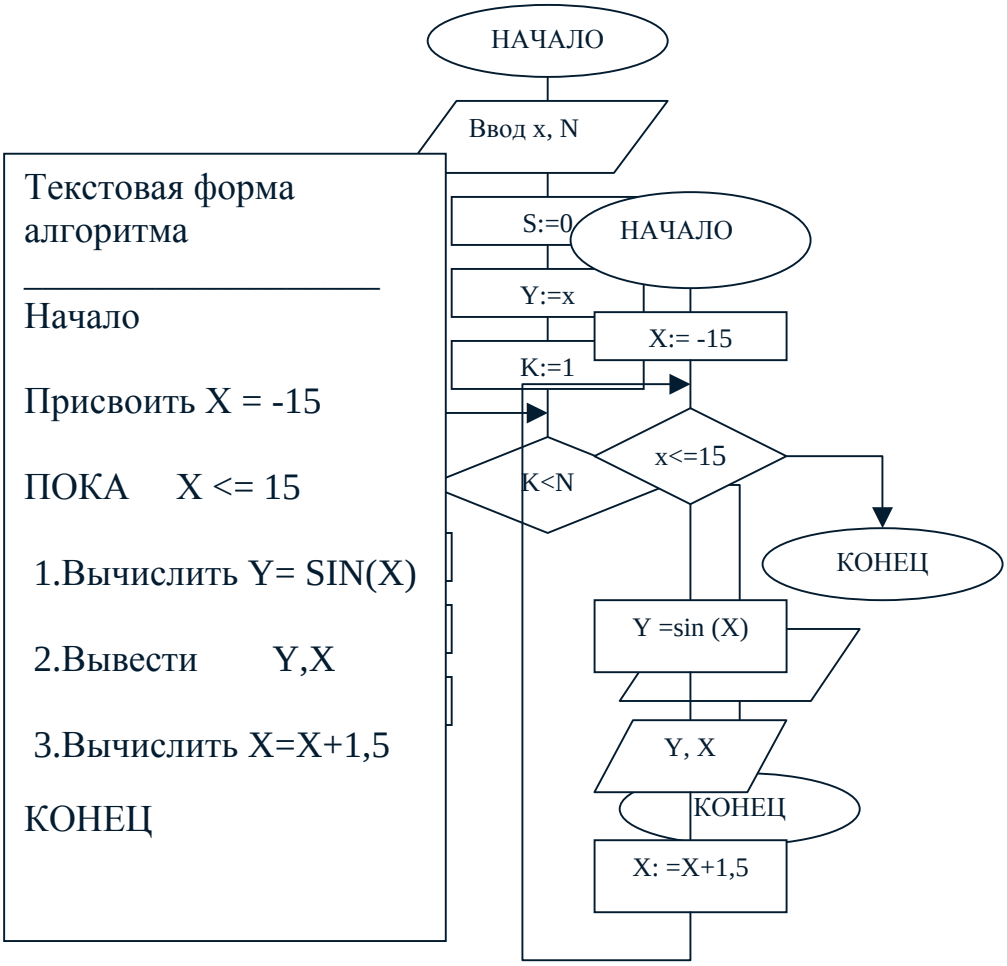


Рис.12. Алгоритм вычисления суммы ряда S=x+x^2+x^3+…+x^n

*Пример 5.*Пусть требуется составить алгоритм вычисления суммы ряда
S=x+x^2+x^3+…+x^n.

Решение. Исходные данные для алгоритма это переменные **X** и **n.** На каждом шаге будем вычислять очередной член суммы Y и прибавлять его к предыдущему значению суммы S.Для этого используем реккурентную формулу вычисления степени X (см. таблицу 3) Y=Y*X, тогда сумма ряда на каждом шаге итерации будет вычисляться по формуле S=S+Y. Количество итераций K изменяется от 1 до n и равно количеству членов ряда. Начальное значение суммы ряда S равно 0. На рис. 12 представлен циклический алгоритм с предусловием для вычисления заданной суммы ряда.

Пример 6. Требуется составить алгоритм получения на отрезке [-15,15] множества значений функции Y= SIN(X) в виде таблицы значений (X,Y) при изменении аргумента X по формуле X[k]=X[k-1]+h, где h=1,5.
Решение. Такие задачи относят к задачам табулирования функций. Из условия задачи определяем, что начальное значение отрезка табулирования X= -15, конечное значение - X=15. Процесс получения множества пар X,Y) является итерационным, значит проектируемый алгоритм будет циклическим. Условие выхода из цикла X>15. На рис. 13 представлен циклический алгоритм с предусловием вычисления табличного значения функ-

ции Y= SIN(X) на отрезке -15<X<15 при изменении X на каждом шаге итерации на величину 1,5. Результатом выполнения алгоритма является циклический вывод множеств пар (Y,X) .

Рис. 13. Циклический алгоритм табулирования функции Y =sin (X)

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы для следующих задач.

1. Вычислить число в факториале Y=X!
2. Вычислить сумму ряда , общий член которого задан формулой An=(x*n)/n!.
3. При табулировании функции y=cos(x+a) на отрезке [1,10] с шагом h=1 определить сумму значений y , больших p.
4. Подсчитать количество цифр в целом числе X.
5. Вычислить сумму значений функции y=x^2 на отрезке [1,5] с шагом 1.
6. * Найти минимальное значение функции Y=Sin(X)*X , на отрезке [C,D] с шагом 0.001. Реализовать цикл с постусловием.
7. Протабулировать функцию y=sin(x) на отрезке [1,5] с шагом h=0,5. Вывести предпоследнее положительное значение функции.
8. Определить постановку задачи и составить визуальный алгоритм для этой задачи, если табличное представление ее решения изображено ниже:

Условие N>0	**S**	**N**
	0	125
125>0 да	0+5=5	12
12>0 да	5+2=7	1
1>0 да	7+1=8	0
0>0 нет		

9. Составить визуальную и табличную формы алгоритма по его текстовому представлению, а также определить конечное значение S .

A) I=0; S=0;
ПОКА I<3
I=I+3
S=S+I*I
ВЫВОД S

B) I=1; S=0;
ПОКА I >1
S=S+1/I
I=I-1
ВЫВОД S

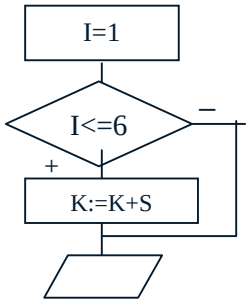
10. Составить визуальную и текстовую форму представления алгоритма, заданного в табличной форме.

I	**J**	**S**

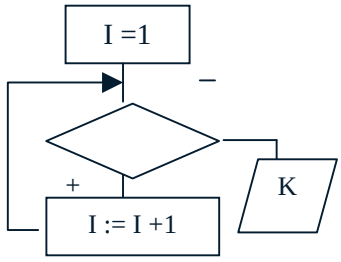
		0
1	2	0+1+2=3
	3	3+1+3=7
2	2	7+2+2=11
	3	11+2+3=16

11.Определить является ли данный фрагмент алгоритма циклом, если да, то какого вида и какое действие является телом цикла?

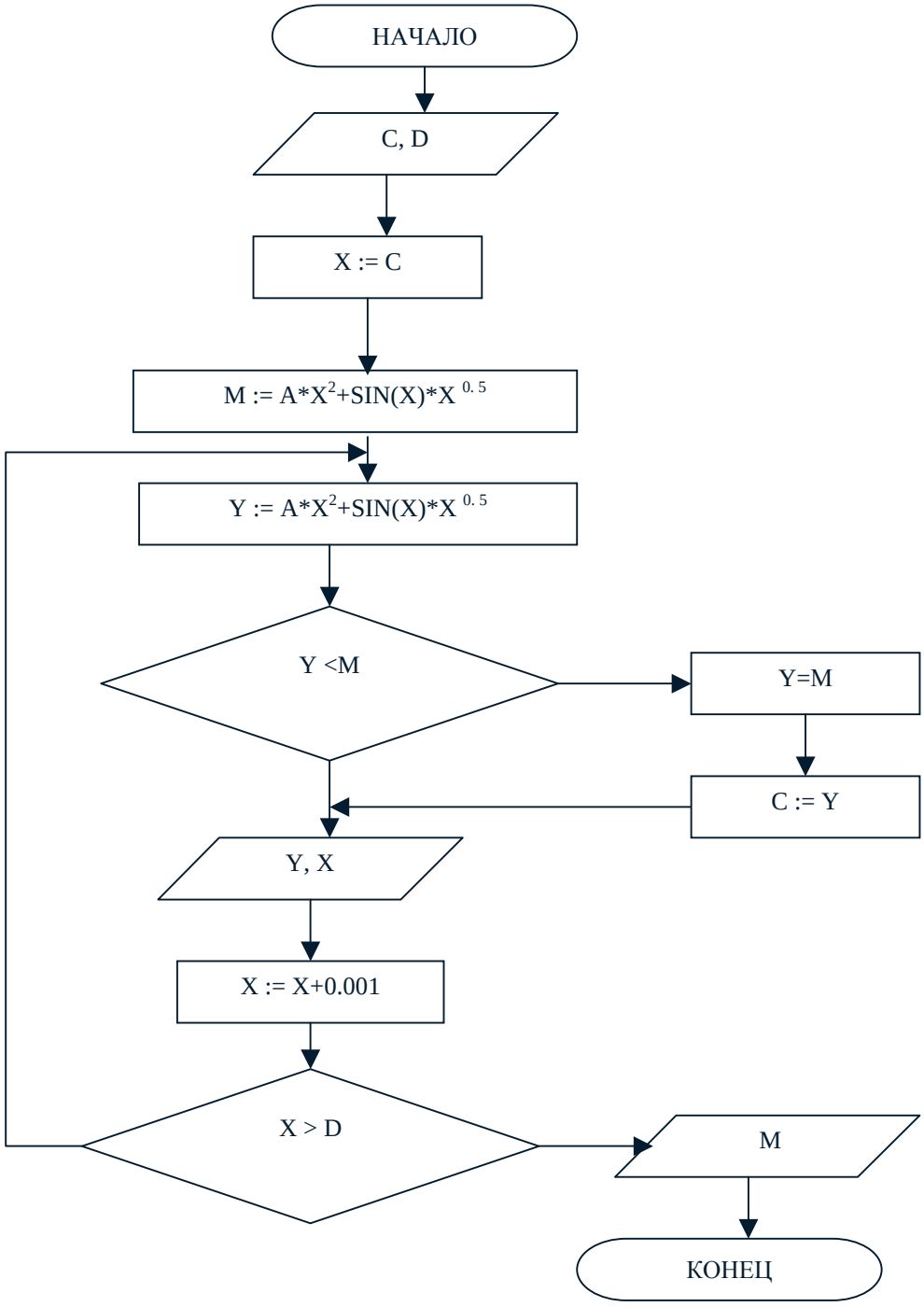
А)



В)



12. * Протабулировать функцию Y=tg(X), при изменении X на отрезке [A,B] с шагом K и определить количество точек разрыва(M) этой функции.
13. Определите местонахождение ошибок в алгоритмическом решении следующей задачи. Найти минимальное значение функции $Y=A*X^2+Sin(X)*X^{0,5}$, для X изменяющемся на отрезке [C,D] с шагом 0,01.



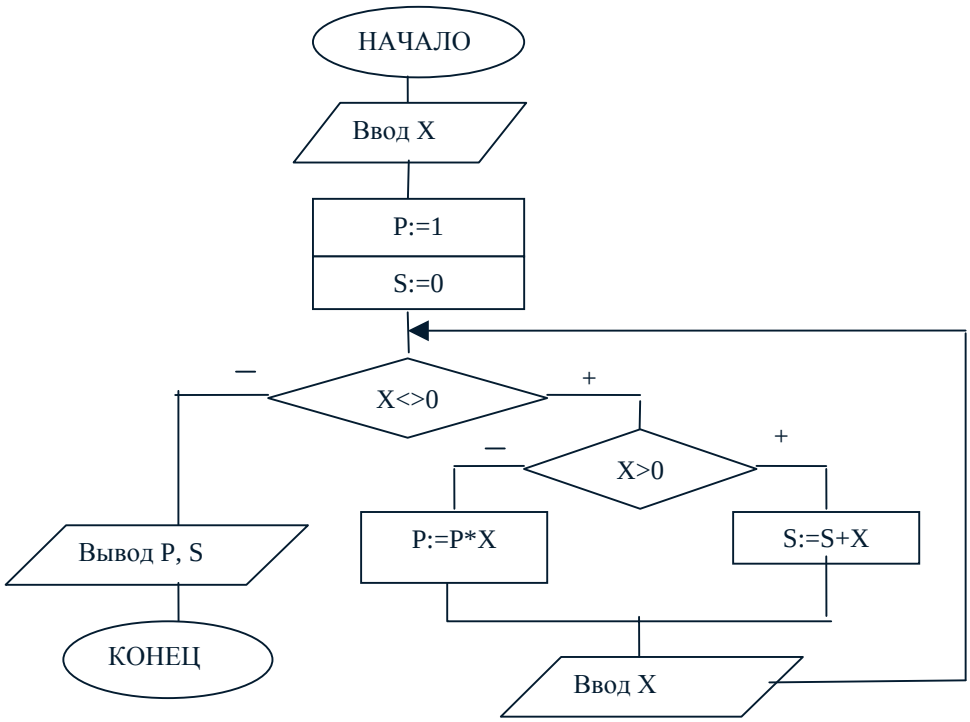
8.АЛГОРИТМЫ ОБРАБОТКИ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ЧИСЕЛ

Последовательность значений - это набор однотипных величин, которые вводятся и обрабатываются циклически. Примером последовательности целых чисел может быть следующий набор значений: (2,5,-4,10,1,0). Последовательности значений отличаются от массивов значений тем, что в памяти одновременно все значения последовательности не хранятся. Для обозначения значения последовательности используют одну переменную, в которую на каждом шаге итерации вводится очередное значение последовательности. Отличительной особенностью последовательности является также возможность содержания неопределенного или неизвестного заранее количества ее значений. В этом случае критерием окончания последовательности служит некоторое особое значение, например, ноль.

Пример 7. В числовой последовательности определить сумму положительных и произведение отрицательных чисел. Реализовать с помощью цикла с предусловием. Признак конца последовательности - значение 0.
Решение. Обозначим за X переменную, содержащую очередное значение последовательности, за S - сумму положительных значений , за P - произведение отрицательных значений. Полученный алгоритм приведен на рис. 14. Условие

Рис.14. Алгоритм вычисления суммы положительных и произведения отрицательных значений числовой последовательности

для выбора вычислений X>0. Для вычисления суммы значений воспользуемся реккурентной формулой S=S+X с начальным значением S=0, для вычисления произведения - реккурентной формулой P=P*X с начальным значением P=1. Условие выхода из цикла неравенство X<>0.



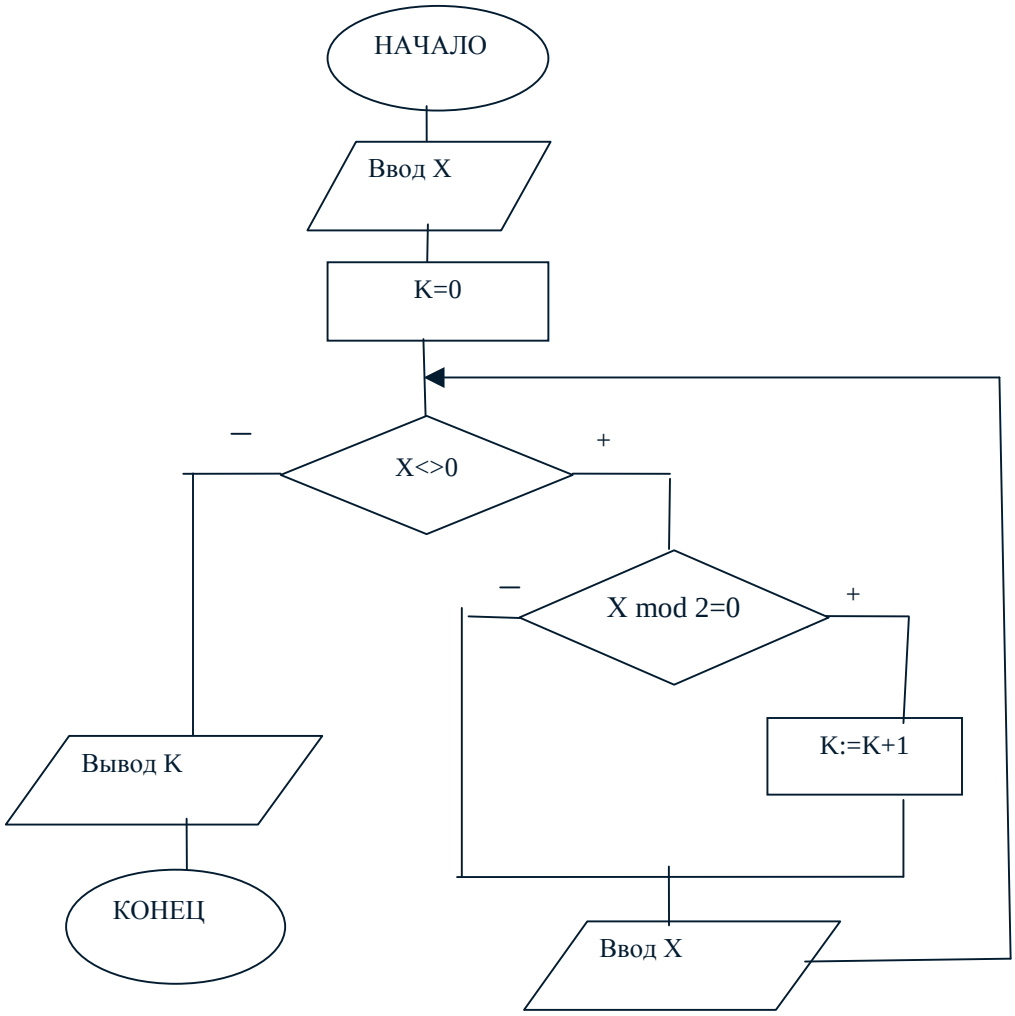
Пример 8 . Составить циклический алгоритм для определения в последовательности целых чисел количества четных чисел .
Решение. Обозначим за X переменную, содержащую очередное значение последовательности, за K - количество четных значений (рис. 15). Условие для выбора четных значений X mod 2=0 (остаток приделении X

на 2 равен 0). Для вычисления количества значений воспользуемся реккурентной формулой К=К+1 с начальным значением К=0.

Рис. 15. Алгоритм определения количества четных чисел в последовательности значений

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы для следующих задач обработки последовательности значений.



1. В последовательности вещественных чисел подсчитать произведение чисел, кратных 3.
2. В последовательности чисел сравнить ,что больше сумма положительных или произведение отрицательных.
3. В последовательности чисел определить предпоследнее отрицательное число.(При решении введите дополнительную переменную для хранения предпоследнего отрицательного числа).
4. В последовательности целых положительных чисел определить максимальное число (Рекомендуем реализовать такой алгоритм :

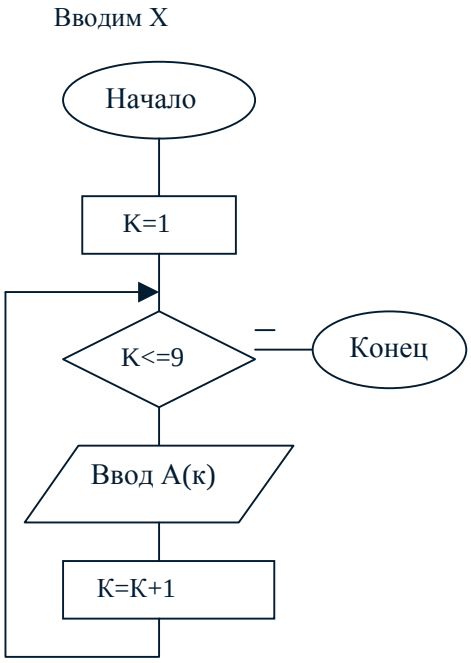


Рис. 16.Алгоритм ввода элементов

max=X
Цикл с постусловием
а. Если элемент X > max
то max:=X (значение этого элемента);
б. Вводим новый элемент последовательности X.
Условие выхода из цикла X=0)

5. В последовательности целых чисел определить третье положительное число и подсчитать количество цифр в нем.

9.АЛГОРИТМЫ ОБРАБОТКИ ОДНОМЕРНЫХ ЧИСЛОВЫХ МАССИВОВ

Под структурой данных типа массив понимают однородную структуру однотипных данных, одновременно хранящихся в последовательных ячейках оперативной памяти. Эта структура должна иметь имя и определять заданное количество данных (элементов). Однотипность данных определяет возможность использования циклических алгоритмов для обработки всех элементов массива. Количество итераций цикла определяется количеством элементов массива Одновременное хранение в памяти всех элементов массива позволяет решать большой набор задач, таких как поиск элементов, упорядочение и изменение порядка следования элементов.

Доступ к любому элементу массива осуществляется по его номеру (индексу). Поэтому для обращения к элементу массива используют имя_массива(номер элемента), например, A(5).

Массив называется одномерным , если для получения доступа к его элементам достаточно одной индексной переменной.

Рассмотрим простой алгоритм ввода элементов одномерного числового массива A из 9 элементов. В этом циклическом алгоритме условие выхода из цикла определяется значением специальной

переменной K, которая называется счетчиком элементов массива A (рис.16), эта же переменная K определяет количество итераций циклического алгоритма ввода элементов массива. На каждом шаге итерации переменная K(значение номера элемента массива A) изменяется на 1, то есть происходит переход к новому элементу массива.В дальнейшем, при рассмотрении алгоритмов обработки одномерных массивов в целях устранения дублирования алгоритм ввода элементов массива будем заменять одним блоком, подразумевая, что он реализуется по схеме, циклического алгоритма, представленного на ри- сунке 16.

Пример 9. Составить алгоритм определения в одномерном числовом массиве A из N элементов суммы положительных элементов.

Решение. Алгоритм представлен на рисунке 17. В этом алгоритме переменная K - является счетчиком элементов массива, S - сумма элементов массива, она вычисляется по реккурентной формуле S=S+A(K). Ввод количества и значений элементов массива осуществляется вначале в отдельном блоке ввода, который реализуется по
схеме алгоритма ввода элементов массива, изображенного на рис.16.

+

Часто для проверки правильности работы алгоритмов на конкретных наборах данных используют таблицу трассировки. Эта таблица содержит столько столбцов, сколько переменных и условий в алгоритме, в ней мы выполняем действия шаг за шагом от начала до конца алгоритма для конкретных наборов входных данных.

Пример 10. Составить алгоритм поиск[illegible]имальным значением в
одномерном массиве A(1..n) и его табл[illegible] для значений (3, 7, 0, 9).

Решение. Введем обозначения K[illegible] A[K] - текущее значение элемента массива, N=4 количество элементов о[illegible]р максимального элемента массива, A[M] - значение максимального элемента массива. Основной идеей ал[illegible]а является выполнение сравнения текущего элемента массива A[K] и элемента с ма[illegible]альным зна[illegible] A[M], определенным на предыдуще[illegible] шаге итерации. По алгоритму изображенному на рис.18 получено максимальное значение для массива (3, 7, 0, 9), процесс и правильный результат поиска которого показан[illegible] таблице 4.

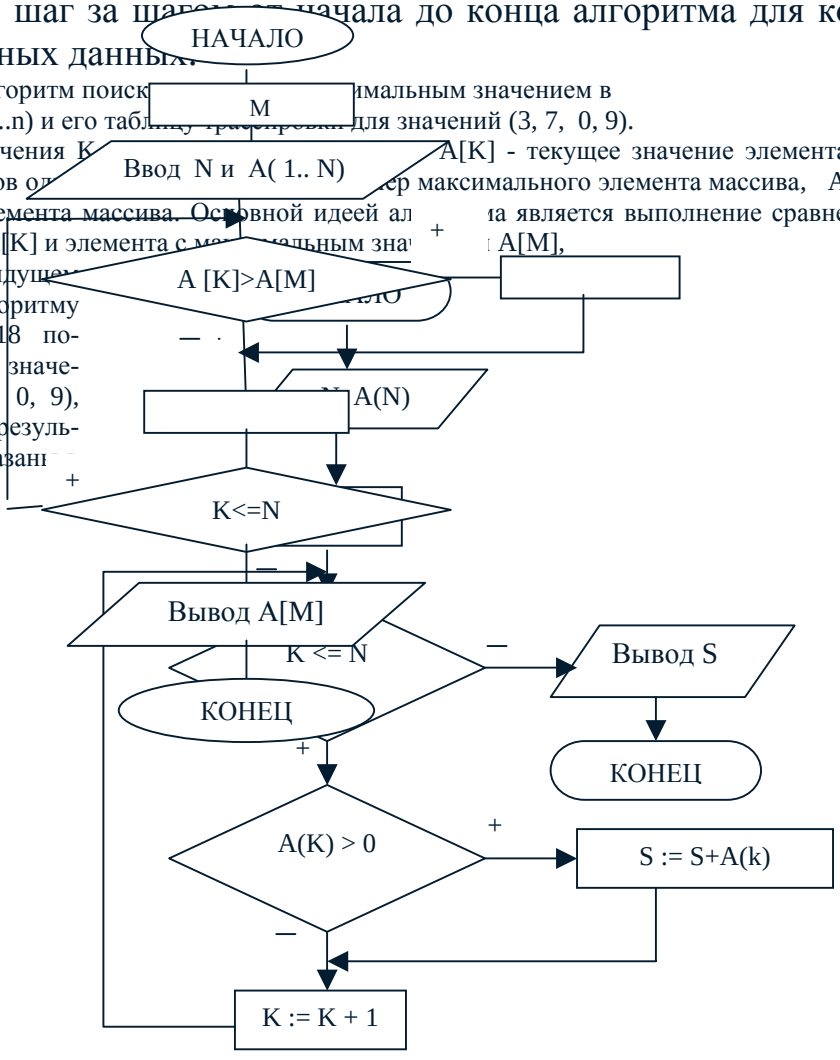


Рис. 17. Алгоритм вычисления суммы положительных элементов

Рис.18. Алгоритм поиска максимального значения в массиве

Таблица 4. Таблица трассировки алгоритма примера 10.

Номер элемента массива К	Значение элемента А (К)	Номер максимального М		Значение максимальнго А(М)		Проверка А(К)>А(М)
1	3	1		3		Нет
2	7	1	2	3	7	да
3	0	2		7		нет
4	9	2	**4**	7	**9**	да

Рассмотрим несколько более сложных алгоритмов, в которых осуществляется изменение порядка следования элементов в одномерном массиве . К таким алгоритмам относят алгоритмы с перестановкой элементов местами, алгоритмы удаления некоторых элементов или циклического переноса некоторых элементов в начало или конец массива.

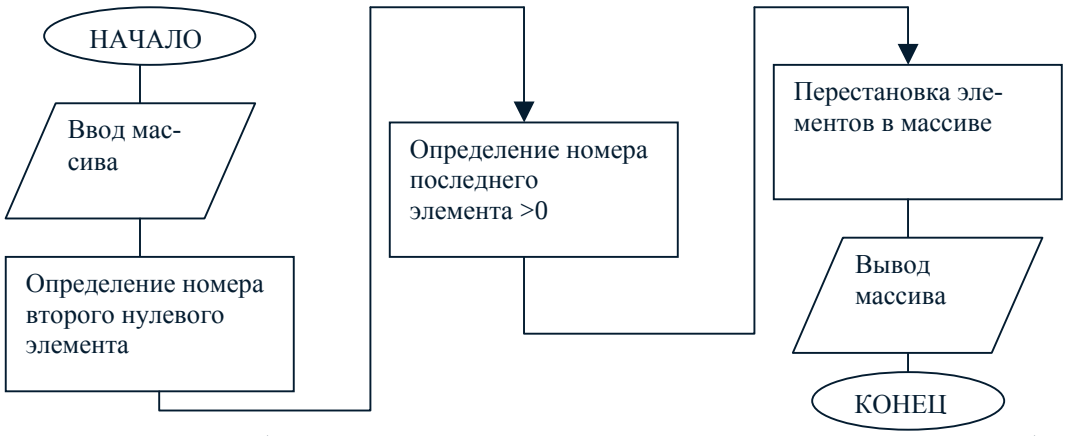
Основным требованием при составлении алгоритмов обработки массивов является неиспользование дополнительных массивов.

Чтобы точнее уяснить постановку задачи следует сначала рассмотреть частные решения для конкретных наборов входных данных (провести анализ) , затем обобщить полученное решение и определить набор решаемых задач. Составив визуальный алгоритм, его следует проверить на различных наборах исходных данных. Эти наборы исходных данных требуется подбирать таким образом, чтобы при заполнении таблиц трассировок проверить все пути вычислений данного алгоритма от начальной вершины до конечной.

Пример 11.Составить алгоритм решения и таблицу трассировки следующей задачи. В одномерном массиве поменять местами 2-ой нулевой элемент и последний положительный элемент. Применить нисходящее проектирование алгоритма.

Решение. Пусть одномерный массив содержит 9 элементов: (5, 0, 4, -3, -7, 0, -2, -4, 0). Среди этих элементов имеются три нулевых значения, отрицательные и положительные значения. Второй нулевой элемент имеет порядковый номер 6, а последний положительный элемент - номер 3, его значение равно 4. Если поменять местами 2-ой нулевой элемент и последний положительный в исходном массиве (5, 0, 4, -3, -7, 0, -2, -4, 0) то получим новый массив, в котором изменен порядок следования элементов 5, 0, 0, -3, -7, 4, -2, -4, 0 **.**

Основными операциями алгоритма будут: поиск второго нулевого элемента массива, поиск последнего положительного элемента массива и перестановка найденных элементов. Отметим, что для пере-



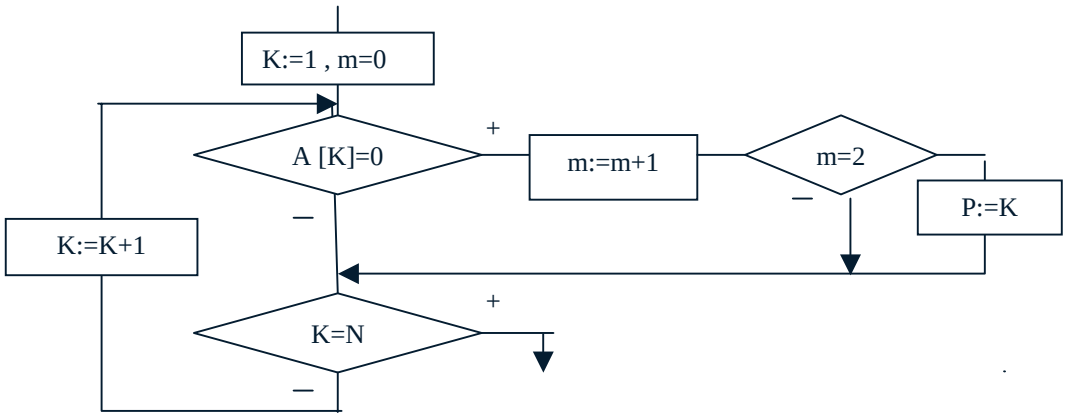
становки элементов необходимо знать номера переставляемых элементов. На рис. 19 изображен обобщенный алгоритм решения задачи, который в дальнейшем будет детализирован.

Рис. 19. Обобщенный алгоритм к примеру 11

Ввод количества и значений элементов массива осуществляется вначале в отдельном блоке ввода, который реализуется по схеме алгоритма ввода элементов массива, изображенного на рис.16.

Поиск второго нулевого элемента массива будем осуществлять в цикле, проверяя значение очередного элемента на 0. Если очередной элемент равен 0, то для подсчета количества таких элементов используем реккурентную формулу M=M+1. В тот момент времени, когда значение M=2, нам необходимо запомнить номер текущего элемента массива, так как это и есть искомый второй положительный элемент исходного массива. На рис.20 приведен фрагмент алгоритма, реализующего поиск второго нулевого элемента в некотором массиве A из N элементов.

Рис. 20. Фрагмент алгоритма поиска второго нулевого элемента в массиве A, состоящем из N элементов. Здесь K- номер очередного элемента, A(K) - значение



очередного элемента, m -количество нулевых элементов, P- номер второго нулевого элемента в массиве

Поиск последнего положительного элемента массива будем осуществлять аналогично в цикле, запоминая номер очередного элемента, значение которого больше нуля, в дополнительной переменной S. Учитывая тот факт, что после того как будут просмотрены и проверены на положительность все элементы массива в переменной S будет храниться номер последнего положительного элемента массива. На рис. 21 представлен фрагмент алгоритма поиска последнего положительного элемента в массиве A.

Рис. 21. Фрагмент алгоритма поиска последнего положительного элемента

Рассмотрим фрагменты алгоритма, приведенные на рис. 20 и 21. Можно заметить, что оба этих фрагмента выполняют поиск под управлением цикла, с одинаковым числом повторений, равным количеству элементов исходного массива N. Поэтому в итоговом алгоритме решения задачи примера 11 вместо композиции этих двух фрагментов в виде двух последовательных циклов можно использовать параллельное выполнение операций поиска под управлением одного цикла, который приведен на рис. 22.

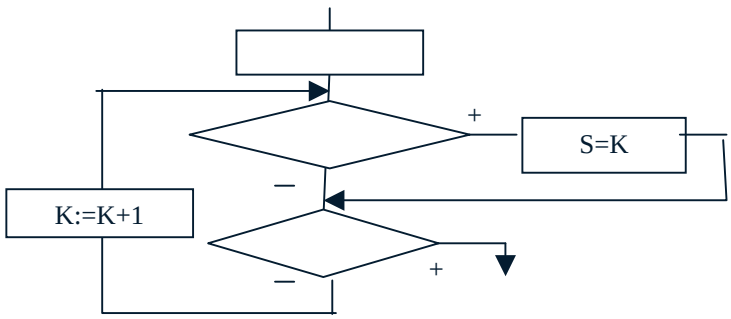
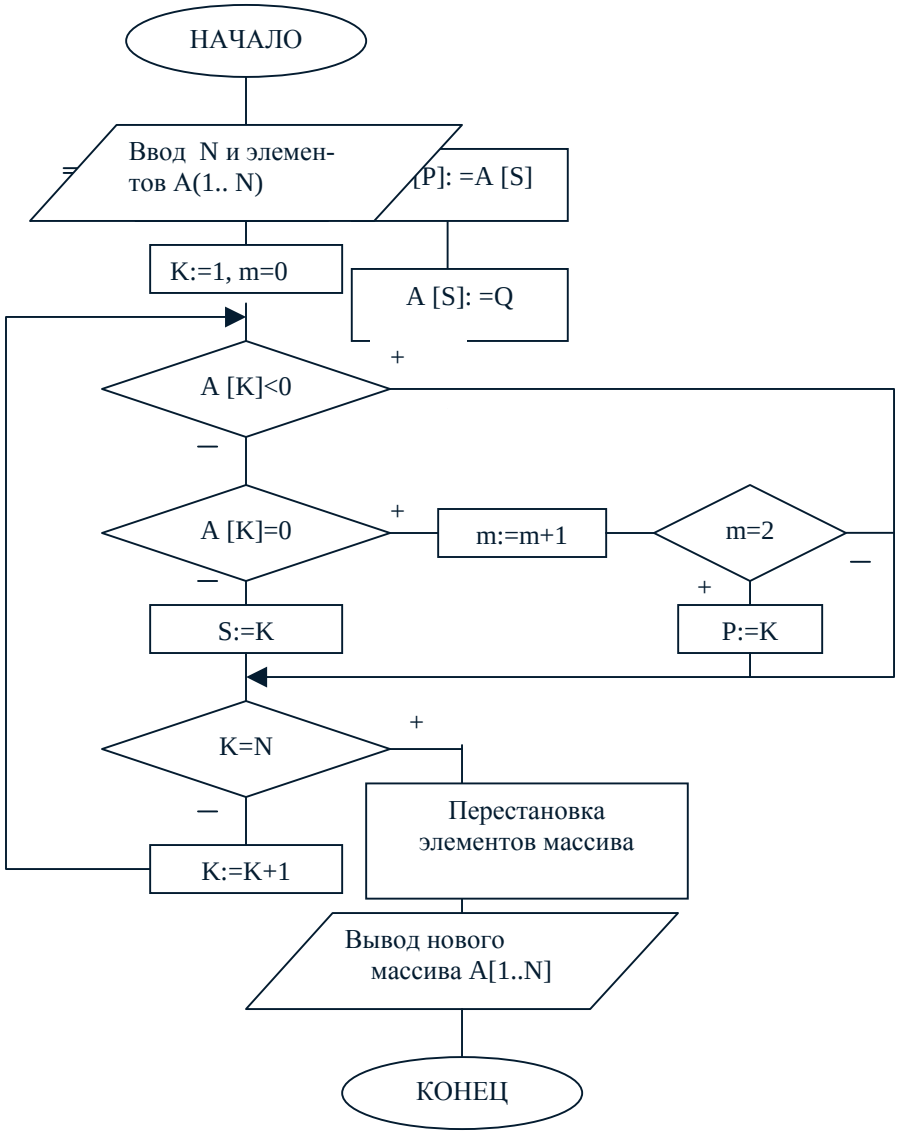


Рис 22. Алгоритм поиска второго нулевого и последнего положительного элементов массива A.



Напомним, что исходный одномерный массив содержит 9 элементов: (5, 0, 4, -3, -7, 0, -2, -4, 0).По условию задачи необходимо поменять местами 2-ой нулевой элемент и последний положительный элемент.
Так как задачи поиска необходимых для перестановки элементов алгоритмически решены (рис. 22), то рассмотрим задачу перестановки элементовДля реализации перестановки найденных элементов достаточно воспользоваться управляющей структурой следования. Для того, чтобы во время перестановки не потерять переставляемые значения используем дополнительную переменную Q, временно хранящую одно из переставляемых значений элемента массива, в частности второе нулевое значение.На рис. 23 представлен универсальный алгоритм перестановки двух элементов одномерного массива, имеющих порядковые номера P и S .

Рис. 23.Фрагмент перестановки двух элементов массива, с номерами S и P

Подводя итог для алгоритмического решения примера 11, приведем на рис. 24 алгоритм для вывода элементов преобразованного массива после перестановки найденных элементов.

Рис. 24. Алгоритм вывода элементов массива

Таким образом, процесс проектирование первоначального алгоритма перестановки второго нулевого элемента и последнего положительного элемента в одномерном массиве завершен. Полученный алгоритм представлен на рис.25. Следует заметить, что целью решения примера 11 было показать процесс нисходящего проектирования алгоритма " сверху-вниз " с использованием декомпозиции и метода структурной алгоритмизации. Отметим, что с целью упрощения в полученном алгоритме не рассматриваются ситуации, когда в составе элементов массива отсутствуют положительные значения или нулевые значения в количестве большем одного.

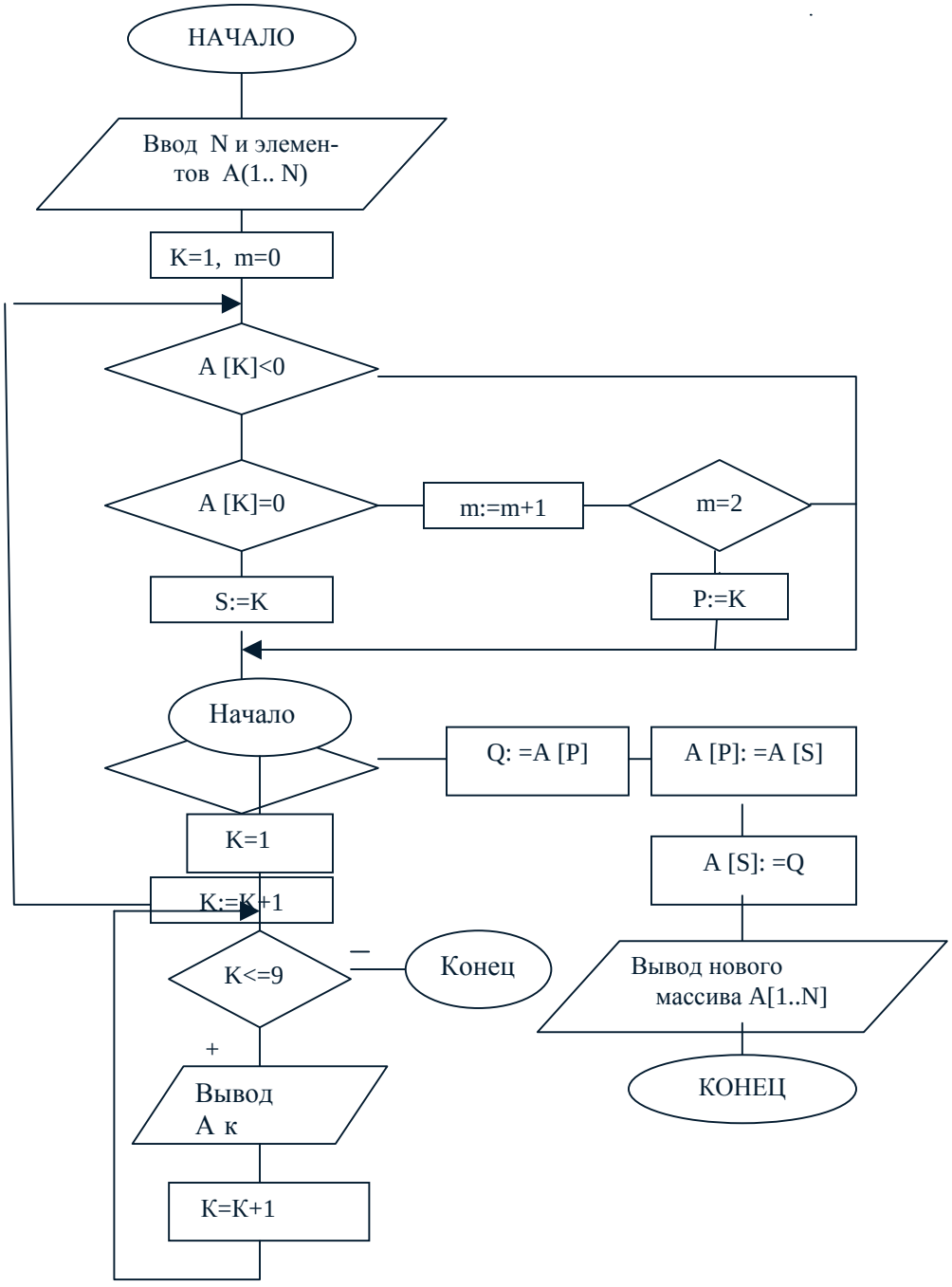


Рис.25. Алгоритм перестановки второго нулевого и последнего положительного элемента в одномерном массиве

Для проверки правильности работы полученного алгоритма составим таблицу трассировки для одномерного массива из 9 элементов: (5, 0, 4, -3, -7, 0, -2, -4, 0), приведенных в примере 11. Напомним, что заполнение таблицы происходит по строкам. Считаем, что все начальные значения числовых переменных равны 0. В таблице 5 выполнена трассировка фрагмента поиска второго нулевого и последнего положительного элементов (их номеров), а в таблице 6 выполнена трассировка следующего фрагмента алгоритма по перестановке в массиве найденных элементов, где K- текущий номер элемента, A[K] - текущее значение элемента массива, M- количество нулей, обнаруженных в массиве при анализе очередного элемента, P- номер второго нулевого элемента массива, S- номер последнего положительного элемента массива, A[S] - значение последнего положительного элемента массива, A[P] - значение второго нулевого элемента массива.

Таблица 5.Таблица трассировки операций поиска второго нулевого элемента и последнего положительного

K	Входной массив A(K)	M	M=2	P	S
1	5	0	Нет		1
2	0	1	Нет		
3	4				3
4	-3				
5	-7				
6	0	2	Да	6	
7	-2				
8	-4				
9	0	3	Нет		

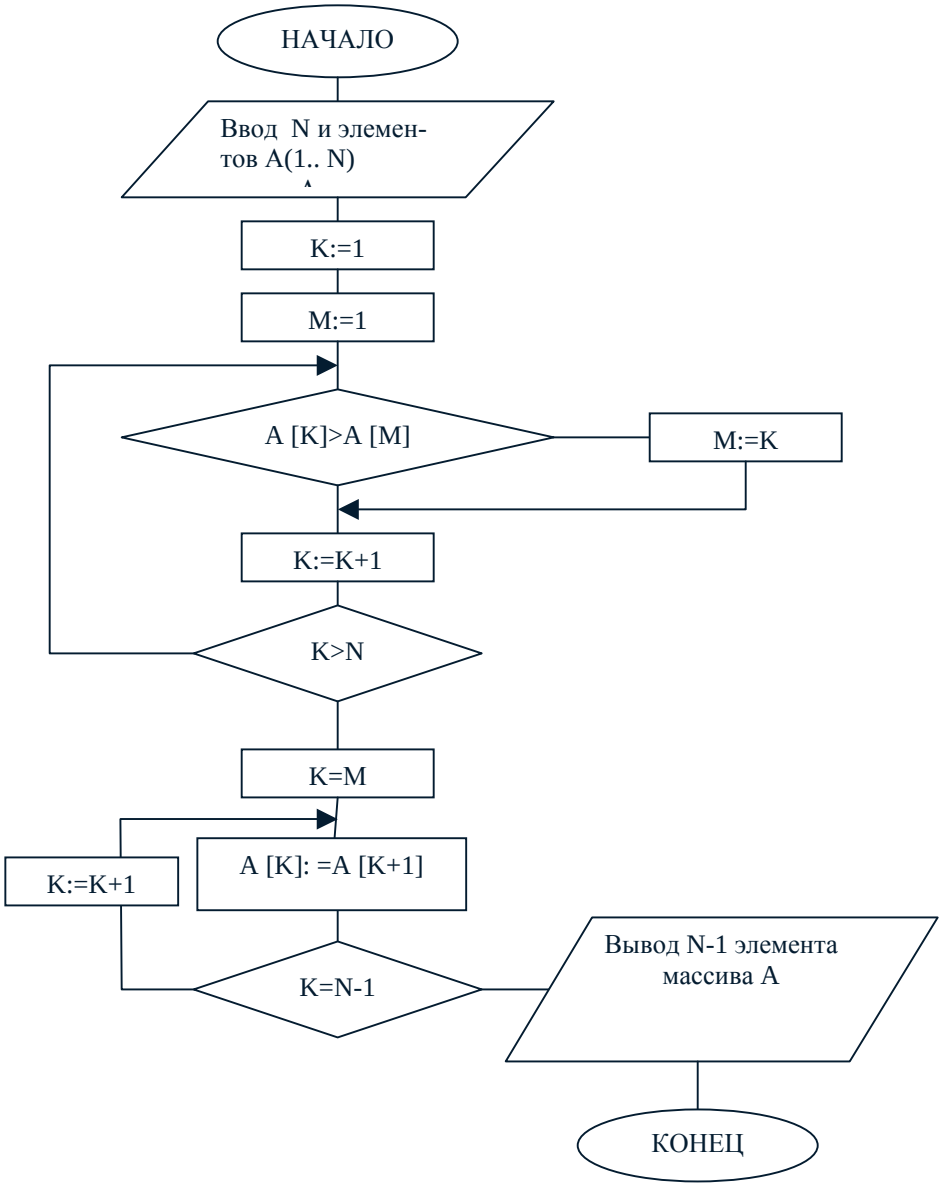
Таблица 6. Таблица трассировки перестановки найденных элементов массива

A(P) A(6)	Q	A(S) A(3)	Выходной массив A(K)	
0	0	4	1	5
4	0	4	2	0
4	0	0	3	0
			4	-3
			5	-7
			6	4
			7	-2
			8	-4
			9	0

Пример 12. Составить алгоритм удаления в одномерном массиве элемента с максимальным значением.
Решение. Анализ постановки задачи позволяет выделить две последовательно решаемые задачи: поиск элемента с максимальным значением и удаление этого значения из массива. Алгоритм первой задачи был рассмотрен ранее в примере 10 (рис.18). В этом алгоритме был определен номер максимального значения M, а максимальное значение определялось как A(M). Удаление элемента из массива приводит к уменьшению количества элементов массива за счет их перемещения на позицию удаляемого. Например, требуется удалить максимальное значение в массиве (2,4,13,5,7). Максимальное значение в этом примере равно 13. После удаления количество элементов данного массива уменьшится на 1 и станет равным 4, а массив примет вид (2,4,5,7). Таким образом, можно сделать вывод , что для удаления элемента из массива необходимо знать его номер, например M, удаление производится путем сдвига на одну позицию влево всех следующих за удаляемым элементов A(M)=A(M+1), этот сдвиг должен осуществляться под управлением цикла. Цикл завершит свою работу, когда последний элемент массива сдвинется на место предпоследнего элемента .
После приведенных рассуждений и используя алгоритмическое решение примера 10, изображенное на рис.18, составим алгоритм удаления элемента с максимальным значением из одномерного массива из N элементов (см. рис.26).

Рис. 26. Алгоритм удаления элемента с максимальным значением

К - номер очередного элемента, М- номер элемента с максимальным значением, N-1 - уменьшенное



в результате удаления одного элемента количество элементов массива A, A [K]: =A [K+1] - удаление путем сдвига влево следующих за удаляемым элементов на одну позицию.

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы и таблицы трассировки для следующих задач обработки одномерных массивов.

1. *В одномерном массиве определить первый отрицательный элемент и его номер.
2. Исключить из массива A1..AN первый отрицательный элемент.
3. Исключить из массива A1..AN первый четный элемент, следующий за максимальным.
4. Дан массив целых чисел a1,..an.Выяснить, какая из трех ситуаций имеет место:все числа a1,..an равны нулю,в последовательности a1,...,an первое ненулевое число-положительное,первое ненулевое число-отрицательное.
5. Дан массив целых чисел a1,..an.Выяснить, какая из трех ситуаций имеет место:все числа a1,..an равны нулю,в последовательности a1,...,an первое ненулевое число-положительное,первое ненулевое число-отрицательное.
6. Даны целые числа a1,..,an.Определить количество целых чисел,входящих в последовательность a1,...,an по одному разу.
7. Даны действительные числа a1,..,an.Требуется найти B равное среднему арифметическому чисел a1,..,an,и наибольшее отклонение от среднего,т.е. max(/a1-b/,/a2-b/,../an-b/).
8. Дан массив действительных чисел a1,...,an.Найти

максимальный элемент среди отрицательных элементов и поменять его местами с минимальным положительным.

9. *В одномерном массиве перенести в начало максимальный элемент.
10. Перенести в начало одномерного массива второй нулевой элемент.
11. Ввести массив a1,...,a16. Получить новый массив по правилу (a1 + a16, a2+a15,...,a8+a9). Найти минимальный элемент полученного массива.
12. *В одномерном массиве перенести в конец минимальный элемент .
13. Перенести в хвост одномерного массива все отрицательные элементы.
14. Перенести в начало одномерного массива все нечетные элементы.
15. В одномерном массиве найти первую группу повторяющихся элементов.
16. Выполните примеры 10 и 11, реализуя ввод элементов массива в цикле, в котором производится их обработка.

10. АЛГОРИТМЫ СОРТИРОВКИ ОДНОМЕРНЫХ МАССИВОВ

Под сортировкой понимают процесс перестановки объектов данного массива в определенном порядке. Целью сортировки являются упорядочение массивов для облегчения последующего поиска элементов в данном массиве. Рассмотрим основные алгоритмы сортировки по возрастанию числовых значений элементов массивов. Существует много методов сортировки массивов. В этой работе будут рассмотрены алгоритмы двух методов: модифицированного метода простого выбора и метода парных перестановок.

10.1. Сортировка модифицированным методом простого выбора

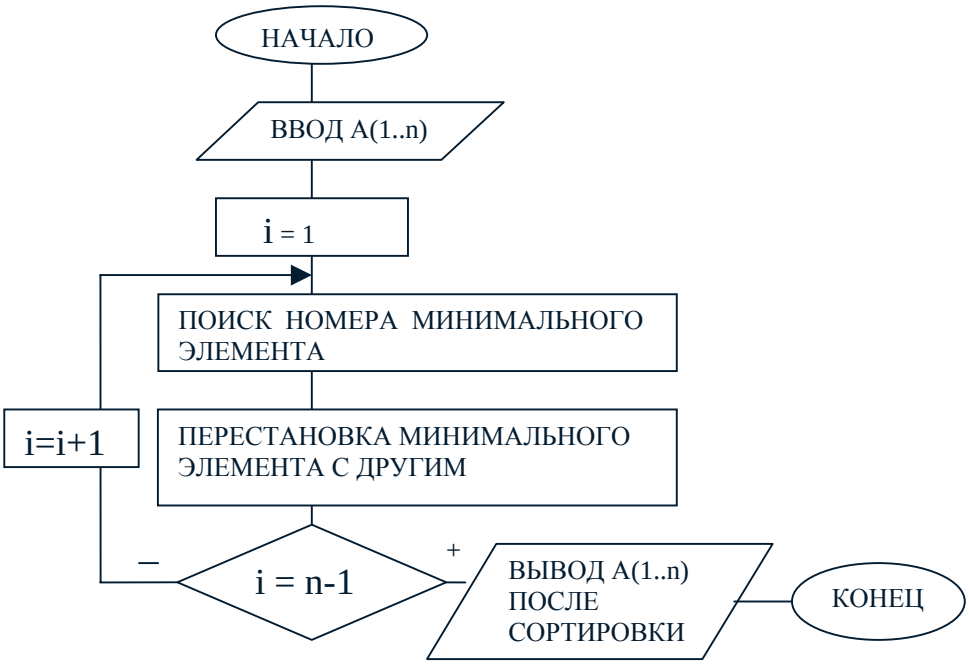
Этот метод основывается на алгоритме поиска минимального элемента. В массиве A(1..n) отыскивается минимальный элемент, который ставится на первое место . Для того, чтобы не потерять элемент , стоящий на первом месте , этот элемент устанавливается на место минимального . Затем в усеченной последовательности, исключая первый элемент, отыскивается минимальный элемент и ставится на второе место и так далее n-1 раз пока не встанет на свое место предпоследний n-1 элемент массива A, сдвинув максимальный элемент в самый конец.

Рассмотрим алгоритмическое решение задачи на примере сортировки некоторого массива значений по возрастанию. В соответствии с вышеописанным методом нам необходимо несколько раз выполнять операции поиска

минимального элемента и его перестановку с другим элементом, то есть потребуется несколько раз просматривать элементы массива с этой целью. Количество просмотров элементов массива согласно описанию модифицированного метода простого выбора равно n-1, где n- количество элементов массива. Таким образом, можно сделать вывод, что проектируемый алгоритм сортировки будет содержать цикл, в котором будет выполняться поиск минимального элемента и его перестановка с другим элементом.
Обозначим через i - счетчик (номер) просмотров элементов массива и изобразим обобщенный алгоритм сортировки на рис.27.

Рис.27. Обобщенный алгоритм сортировки массива модифицированным методом простого выбора

Отметим, что для перестановки элементов местами необходимо знать их порядковые номера, алгоритм перестановки элементов массива был рассмотрен ранее (см. рис. 23). Алгоритмы ввода исходного массива и вывода этого же массива после сортировки изображены на рисунках 16 и 24 соответственно. Алгоритм поиска в массиве минимального элемента и его номера будет аналогичен рассмотренному в примере 10 алгоритму поиска максимального элемента, который представлен на рис.18. Однако, в этом алгоритме будут внесены изменения.Для того, чтобы определить какие изменения следует внести рассмотрим выполнение сортировки данным методом с акцентом на поиск минимального элемента на конкретном примере. Пусть исходный массив содержит 5 элементов
(2,8,1,3,7). Количество просмотров согласно модифицированному методу простого выбора будет равно 4. Покажем в таблице 7, как будет изменяться



исходный массив на каждом просмотре.

Таблица 7. Пример сортировки

Номер просмотра массива i	Исходный Массив	Минимальный Элемент		Переставляемый элемент		Массив после перестановки
		Номер	Значе-ние	Номер	Значе-ние	
1	(2,8,1,3,7)	3	1	1	2	(1,8,2,3,7)
2	1,(8,2,3,7)	3	2	2	8	1,(2,8,3,7
3	1,2,(8,3,7)	4	3	3	8	1,2,(3,8,7)
4	1,2,3,(8,7)	5	7	4	8	1,2,3,7,8

Из данных, приведенных в таблице 7, следует, что поиск минимального значения в массиве на каждом просмотре осуществляется в сокращенном массиве, который сначала начинается с первого элемента,а на последнем просмотре массив, в котором ищется минимальный элемент начинается уже с четвертого (или n-1) элемента. При этом можно заметить, что номер первого элемента массива для каждого поиска и перестановки совпадает с номером просмотра i.

Введем следующие обозначения :

К- номер минимимального элемента,

J - номер элемента массива,

M и A(K)- одно и тоже значение минимального элемента массива,

i - номер переставляемого с минимальным элемента,

A(i)- значение переставляемого элемента.

Тогда циклический алгоритм сортировки модифицированным методом простого выбора будет выглядеть следующим образом (рис.28).

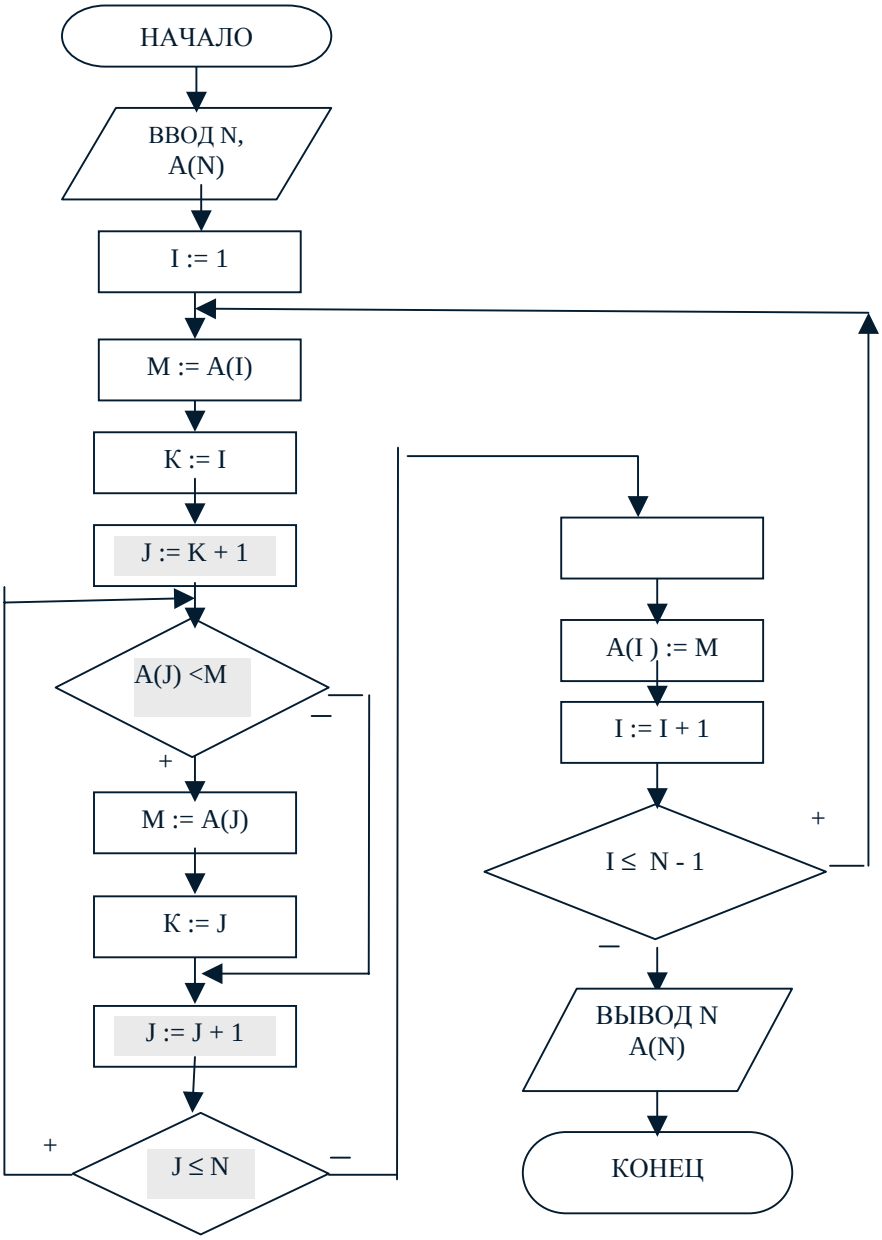


Рис.28. Алгоритм сотрировки массива модифицированным методом простого выбора

В этом алгоритме два цикла, внутренний цикл выделен цветом.

10.2.Сортировка методом парных перестановок

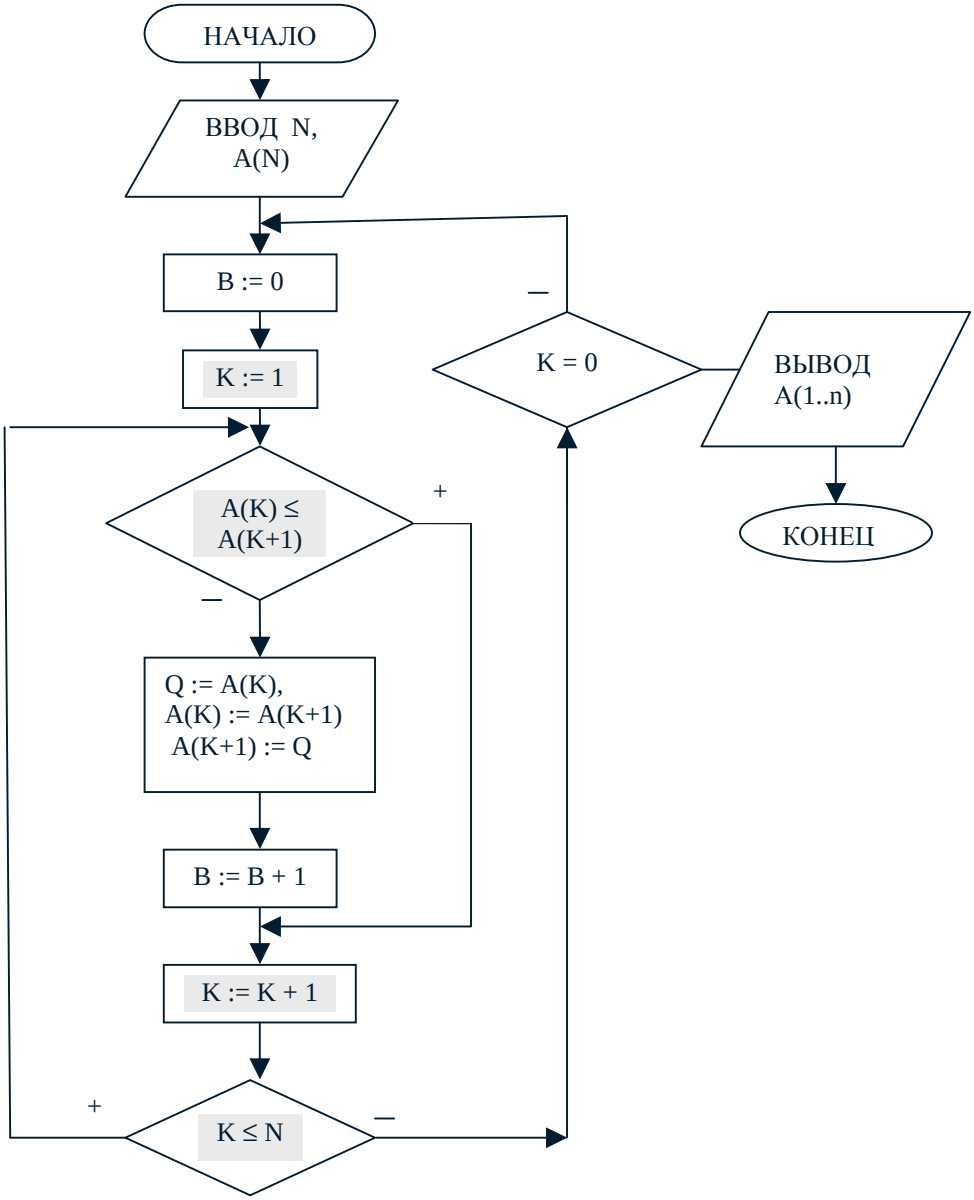
Самый простой вариант этого метода сортировки массива основан на принципе сравнения и обмена пары соседних элементов.

Процесс перестановок пар повторяется просмотром массива с начала до тех пор , пока не будут отсортированы все элементы , т.е. во время очередного просмотра не произойдет ни одной перестановки. Для подсчета количества перестановок целесообразно использовать счетчик - специальную переменную B. Если при просмотре элементов массива значение счетчика перестановок осталось равным нулю, то это означает, что все элементы отсортированы (см.рис.29).

Рис.29. Алгоритм сортировки методом парных перестановок содержит два цикла, внутренний цикл выделен цветом

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы и таблицы трассировки для



следующих задач сортировки одномерных массивов.

1. Ввести массив a1,a2,...,a15. Расположить ненулевые элементы по убыванию.

2. Ввести массив x1,x2,...,x20. Элементы, на нечетных местах, расположить в порядке возрастания, а на нечетных в порядке убывания.
3. Ввести массив a1,a2,...,a15. Требуется упорядочить его по возрастанию абсолютных значений элементов
4. Ввести массив x1,x2,...,x20. Требуется расположить отрицательные элементы в порядке убывания.

11. АЛГОРИТМЫ ОБРАБОТКИ УПОРЯДОЧЕННЫХ МАССИВОВ

Рассмотренные выше алгоритмы сортировки считаются одними из важнейших процедур упорядочивания структурированной данных, хранимых в виде массивов. Одной из главных целей задач сортировки массивов является облегчение их дальнейшей обработки, так как для упорядоченных данных разработаны эффективные методы поиска и обновления. Так, например, поиск минимального или максимального значения в упорядоченном массиве сводится к выборке первого или последнего элемента массива. Рассмотрим некоторые алгоритмы обработки упорядоченных массивов.

11.1.Поиск элементов в упорядоченном массиве

Задача поиска значения X в упорядоченном по возрастанию значений массиве A(1)<A(2)<,..,<A(n) формулируется следующим образом. Требуется выяснить входит ли значение X в этот упорядоченный массив, и если входит, то определить значение k, для которого A(k)=X. Для такого типа задач применяется эффективный метод бинарного поиска, который также известен, как метод деления пополам. Сущность этого метода поиска заключается в последовательном определении номера S элемента, расположенного в точке деления упорядоченного массива пополам и сравнении искомого значения X с этим элементом массива A(s). Если A(s)=X, то поиск заканчивается. В противном случае возможны две ситуации: если A(s)<X, то все элементы, имеющие номера с 1 по s также меньше X, если A(s)>X, то все элементы, имеющие номера с S по n также больше X в силу упорядоченности массива по возрастанию значений. Поэтому для дальнейшего поиска половину значений массива можно исключить из рассмотрения. В первом случае - левую, во втором случае - правую половину. Рассмотрим пример. Допустим, что требуется определить совпадает ли значение X=12 с каким-либо элементом в упорядоченном массиве A и если совпадает, то определить порядковый номер S этого элемента. Иллюстрация применения метода бинарного поиска для поиска элемента X=12 в массиве (2,3,4,6,10,12,20,30,35,45) приведена на рис. 30.

Элементы массива A (2,3,4,6,***10***,12,20,30,35,45).
Номера элементов 1 2 3 4 ***5*** 6 7 8 9 10.
Первое деление S=5, A(5)=10 A(5)<X), исключаем левую половину.

Элементы массива A (2,3,4,6,10,12,20,***30***,35,45).
Номера элементов 1 2 3 4 5 6 7 ***8*** 9 10.

Второе деление S=8, A(8)=30 A(8)>X), исключаем правую половину.

Элементы массива A (2,3,4,6,10,***12***,20,30,35,45).
Номера элементов 1 2 3 4 5 ***6*** 7 8 9 10.
Третье деление S=6, A(6)=12 A(6)=X), элемент X=12 найден.

Рис.30. Иллюстрация применения метода бинарного поиска

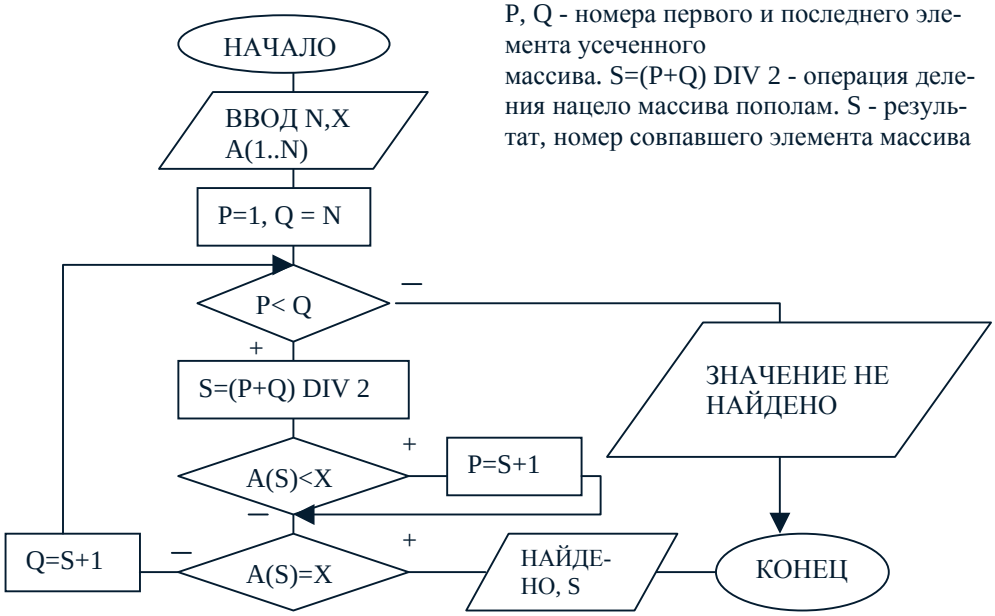
На рис.31 представлен алгоритм, реализующий метод бинарного поиска в упорядоченном массиве.

Рис.31.Алгоритм поиска методом бинарного поиска

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы для следующих задач обработки упорядоченных одномерных массивов.

1. Ввести упорядоченный по неубыванию массив X(1) < = X(2) < =…X(n).
   Найти количество различных чисел среди элементов этого массива.
2. Ввести два упорядоченных по возрастанию числовых массива
   X(1)<X(2)<X(3) <…X(n) и Y(1)<Y(2)<…Y(m). Найти количество общих элементов в этих массивах, то есть количество K таких чисел X(i)= Y(j).
3. Известно, что некоторое число содержится в каждом из трех целочисленных неубывающих массивов X(1) < = X(2) < =…X(n), Y(1)< =Y(2)< =…Y(m) и
   Z(1) < = Z(2) < = … Z(k).Найти одно из этих чисел.
4. Вставить значение P в упорядоченный неубыванию массив X(1) < = X(2) < =…X(n) так, чтобы упорядоченность не нарушилась.
5. Удалить значение P в упорядоченный неубыванию массиве X(1) < = X(2) < =…X(n).
6. Соединить два упорядоченных массива X(1) < = X(2) < =…X(n) и Y(1)< =Y(2)< =…Y(m) в массив Z(1) < = Z(2) < = … Z(k), при этом каждый элемент должен входить в массив Z столько раз, сколько раз он входит в массивы



В этом алгоритме X - искомое значение, P, Q - номера первого и последнего элемента усеченного
массива. S=(P+Q) DIV 2 - операция деления нацело массива пополам. S - результат, номер совпавшего элемента массива

X и Y.

12.АЛГОРИТМЫ ОБРАБОТКИ ОДНОМЕРННЫХ СИМВОЛЬНЫХ МАССИВОВ

Одномерные символьные масивы - это массивы, составленные из определенной последовательности символов, которые образуют тексты. Основными операциями, выполняемыми над текстами, являются операции по определению слов, выделению слова с максимальной длиной, удаление и перестановка слов,сортировка по алфавиту идр.

Для простоты будем считать, что символьный массив представляет одну строку произвольного текста, слово - любую последовательность подряд идущих символов не содержащую пробела. Пробел - это специальный символ, используемый для отделения слов, он не может располагаться перед первым словом. Учитывая все эти допущения можно предложить для решения задачи определения количества слов использовать подсчет количества элементов массива, равных пробелу минус 1.

Рассмотрим алгоритмическое решение распространенной задачи определения в массиве символов слова с максимальной длиной.

Пусть исходный массив A содержит N символов. Для определения слова с максимальной длиной будем использовать сравнение длины текущего слова M с длиной предыдущего слова MAX. Длина слова определяется как содержащееся в нем количество символов. Для того, чтобы вывести слово с максимальной длиной, необходимо запомнить номер элемента S, с которого начинается это слово.

Алгоритм поиска в символьном массиве слова с максимальной длиной приведен на рис. 32, а его таблица трассировки для массива (Дул теплый ветер)- в таблице 8.

Рис. 32.Алгоритм поиска в символьном массиве слова с максимальной длиной

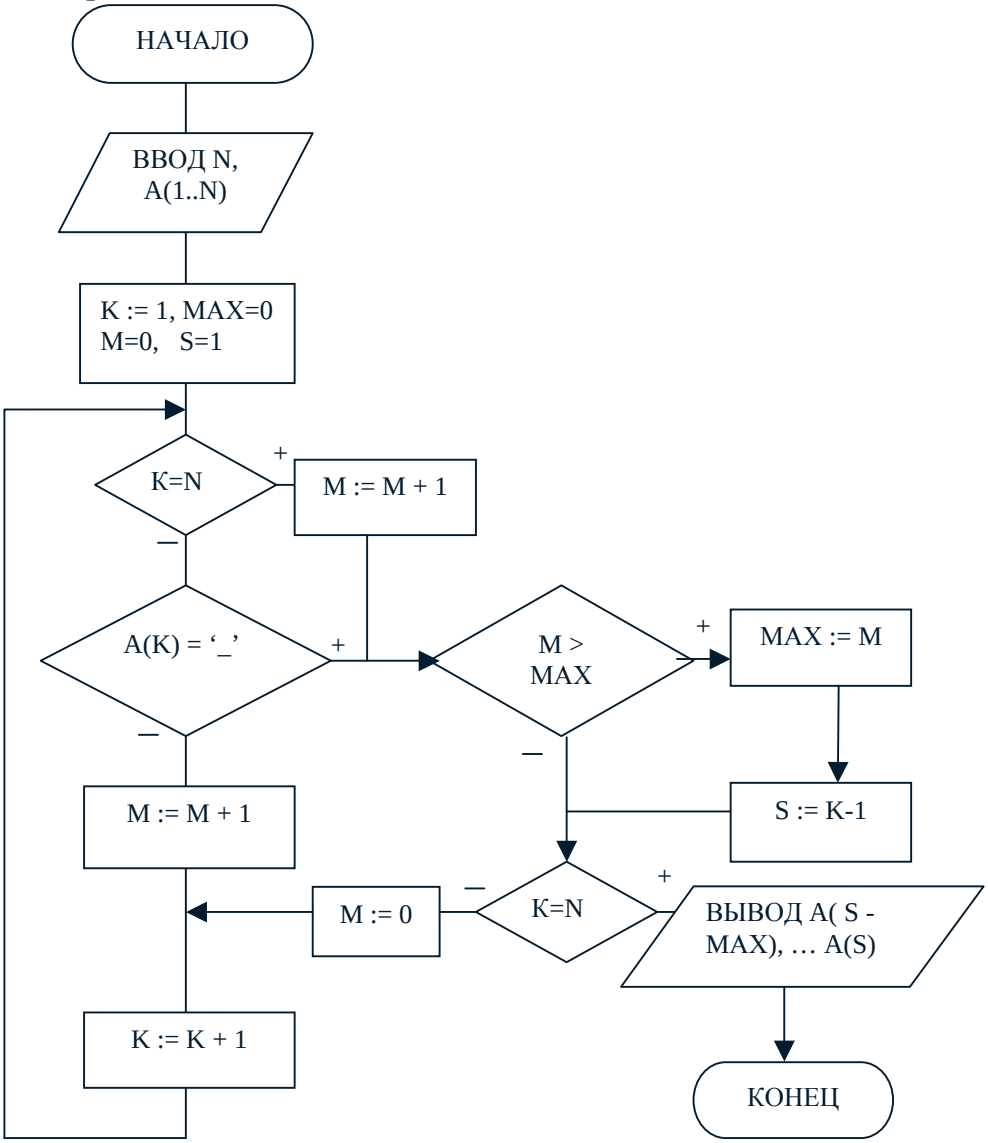


Таблица 8. Таблица трассировки алгоритма поиска слова с максимальной длиной при N= 16 в тексте : "Дул теплый ветер"

К	A(К)	К=N	A(К)=" "	M>MAX	M	MAX	S	НОВОЕ S

1	Д	Нет	Нет		1	0	1	1
2	у	Нет	Нет		2			
3	л	Нет	Нет		3			
4	" "	Нет	ДА	Да	0	3	3	4
5	т	Нет	Нет		1			
6	е	Нет	Нет		2			
7	п	Нет	Нет		3			
8	л	Нет	Нет		4			
9	ы	Нет	Нет		5			
10	й	Нет	Нет		6			
11	" "	Нет	ДА	Да	0	6	10	11
12	в	Нет	Нет		1			
13	е	Нет	Нет		2			
14	т	Нет	Нет		3			
15	е	Нет	Нет		4			
16	р	Да	Нет	Нет	5			

Рассмотрите результат, приведенный в таблице 8, для конкретного входного символьного массива "Дул теплый ветер" без последнего столбца. Однако, после выполнения приведенного на рис. 32 алгоритма для предложения "Дул теплый ветер" будет выведено слово из 7 символов, начинающихся с пробела :" теплый". Значит, формулу определения номера символа S = K-1 , с которого начинается слово с максимальной длиной, следует изменить на S = K. При этом надо будет изменить содержание блока вывода результата: вместо A(S -MAX), … A(S) следует использовать A(S -MAX), … A(S-1). Таким образом, таблица трассировки показала наличие ошибок в алгоритме, изображенном на рис. 32. После внесения изменений этот алгоритм будет работать правильно (см. модернизированный алгоритм поиска в символьном массиве слова с максимальной длиной на рис. 33).

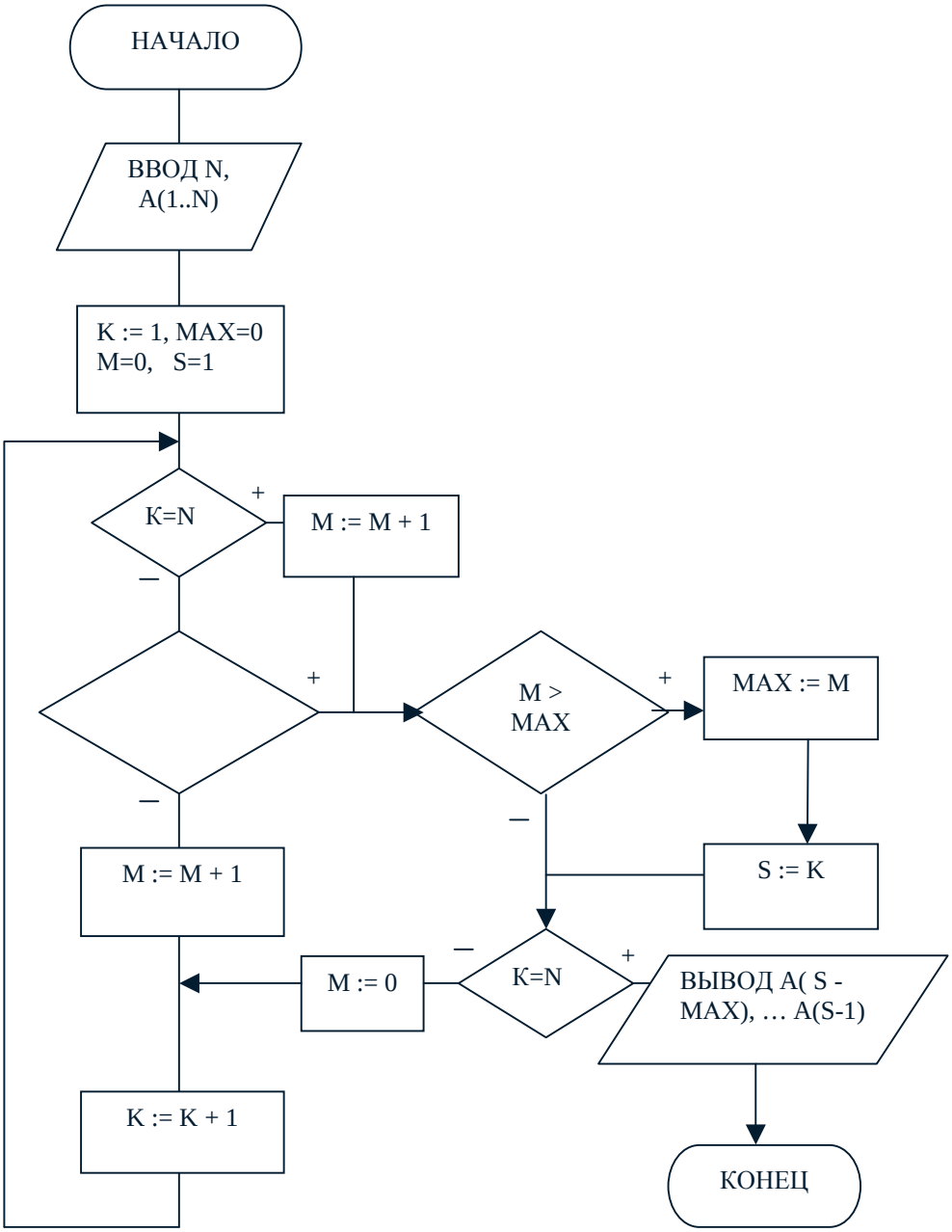


Рис. 33. Модернизированный алгоритм поиска в символьном массиве слова с максимальной длиной

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы для следующих задач обработки символьных одномерных массивов.

1.Найти и вывести слово , содержащее наибольшее количество гласных букв.
2.В слове , в котором обнаружено наибольшее количество шипящих букв , заменить их на символ "*".
3.Вывести все гласные буквы , содержащиеся в слове наибольшей длины и вывести число повторений каждой этой буквы.
4.Подсчитать количество слов и количество символов во всех словах , отличных от заглавных латинских букв.
5.Вывести все слово , содержащее наибольшее количество цифр и вывести число цифр в каждом слове.
6. Слово с минимальной длиной удалить из данного предложения.
7. В предложении перенести в его конец все, встречающиеся в тексте цифры .
8. В предложении расставить все слова в алфавитном порядке.

13.АЛГОРИТМЫ ОБРАБОТКИ ДВУМЕРНЫХ МАССИВОВ

Двумерный массив - это структура однотипных элементов, расположенных в виде таблицы значений. Такое представление значений соответствует математическому понятию двумерный массив. Каждый элемент в двумерном массиве идентифицируется номером строки и номером столбца, на пересечении которых он расположен. Например, в двумерном массиве А, изображенном на рис. 34, элемент со значением 5 расположен на пересечении третьей строки и второго столбца. Этот элемент будет обозначаться как А(3,2). А элемент А(1,4) имеет значение , равное нулю. Такое представление набора значений позволяет выполнять обработку как отдельных значений в двумерном массиве, так и последовательности значений, расположенных в строках или столбцах.

```
3   7 -1  0
4   2  6  1
9   5  12 4 6
22 31 1
```

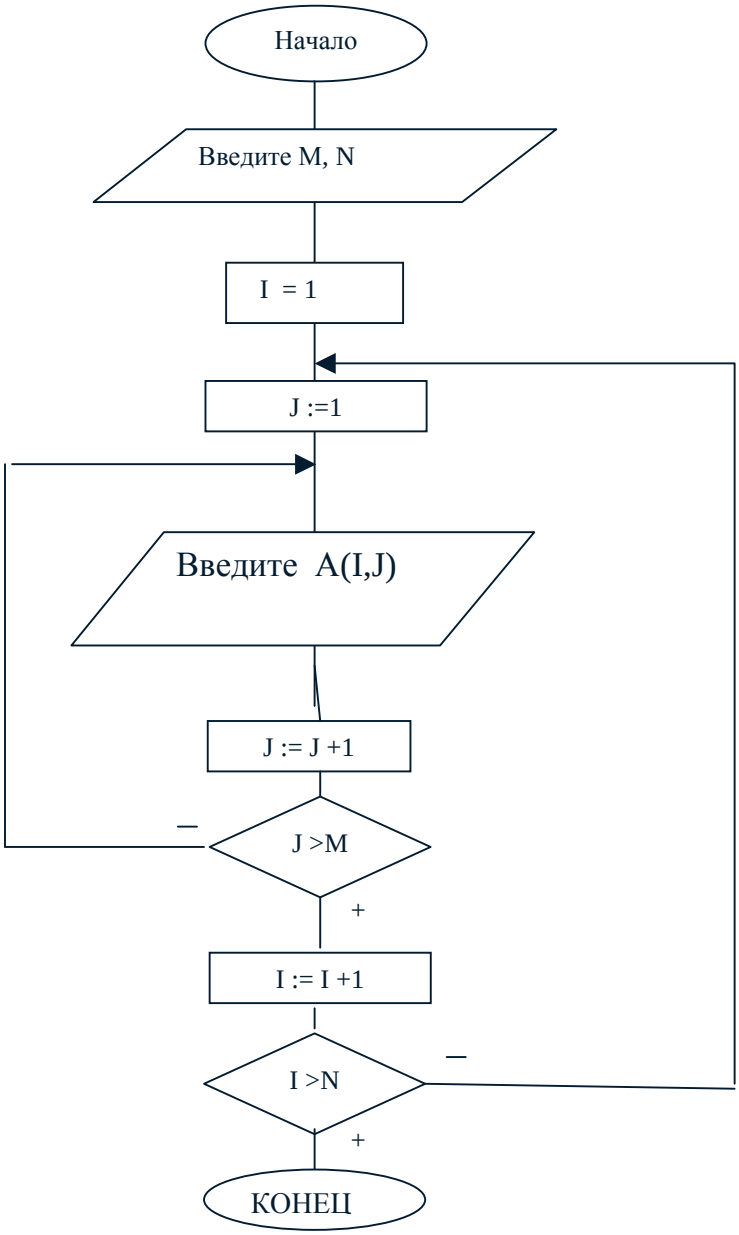
Рис.34. Пример двумерного массива

В дальнейшем будем считать, что для двумерного массива A(N,M) в обозначении элемента A(i,j) первое значение i соответствует номеру строки и изменяется от1 до N, а j - номеру столбца и изменяется от 1 до M. В отличие от одномерного массива, в котором использовался только один номер для определения местоположения элемента и требовался только один цикл для ввода элементов, в двумерном массиве для обработки элеменов необходимы два вложенных друг в друга цикла. Внешний цикл предназначен для изменения номера строки i, а второй, внутренний, - для изменеия номера столбца j в текущей строке i.

На рис. 35 представлен простой алгоритм ввода элементов, построенный в виде структуры из вложенных циклов.

Рис. 35. Алгоритм ввода элементов двумерного массива

При рассмотрении в дальнейшем алгоритмов обработки элементов двумерного массива в целях сокращения их размера фрагмент ввода элементов бу-



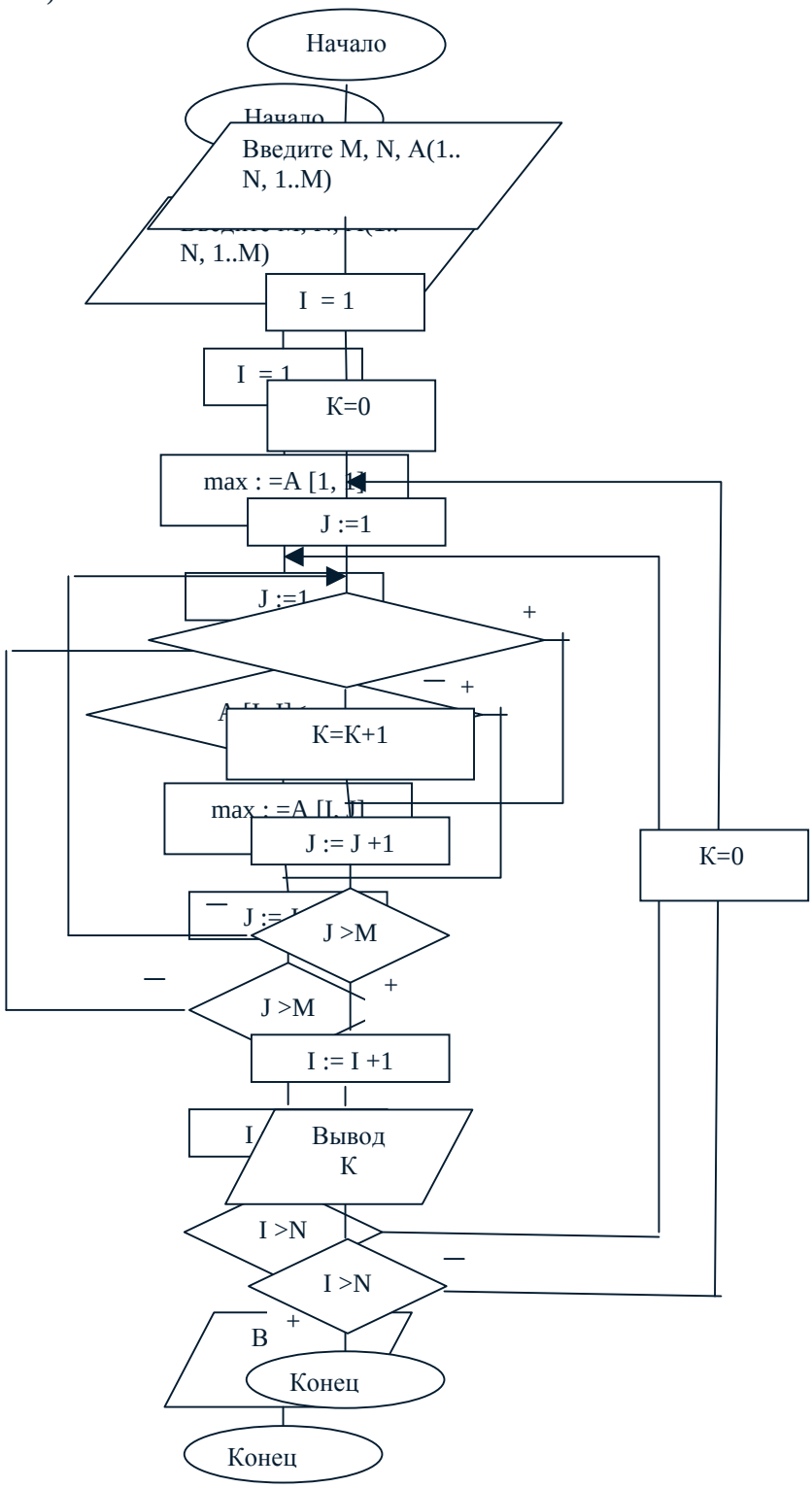
дем заменять отдельным блоком ввода.

Пример 13. Составить алгоритм поиска максимального значения в двумерном массиве A(N,M).

Решение. Поиск максимального элемента в двумерном массиве осуществляется аналогично поиску в одномерном массиве. Отличие состоит в том, что для обработки двумерного массива используем влож Обозначим максимальный элемент переменной MAX. Значение этой переменной будет меняться на каждой итерации цикла, если очередной значение элемента массива окажется больше MAX (см. рис.36).

Рис.36. Алгоритм поиска максимального значения в двумерном массиве

*Пример 14.*Составить алгоритм вычисления количества нечетных элементов в каждой строке двумерного массива A(1.. N, 1..M).



Решение. Для определения нечетных элементов будем использовать проверку на нечетность A[I,J] mod 2 ≠ 0, для подсчета количества нечетных значений - формулу K=K+1и вывод значения K столько раз, сколько строк в массиве. Алгоритм решения представлен на рис. 37.

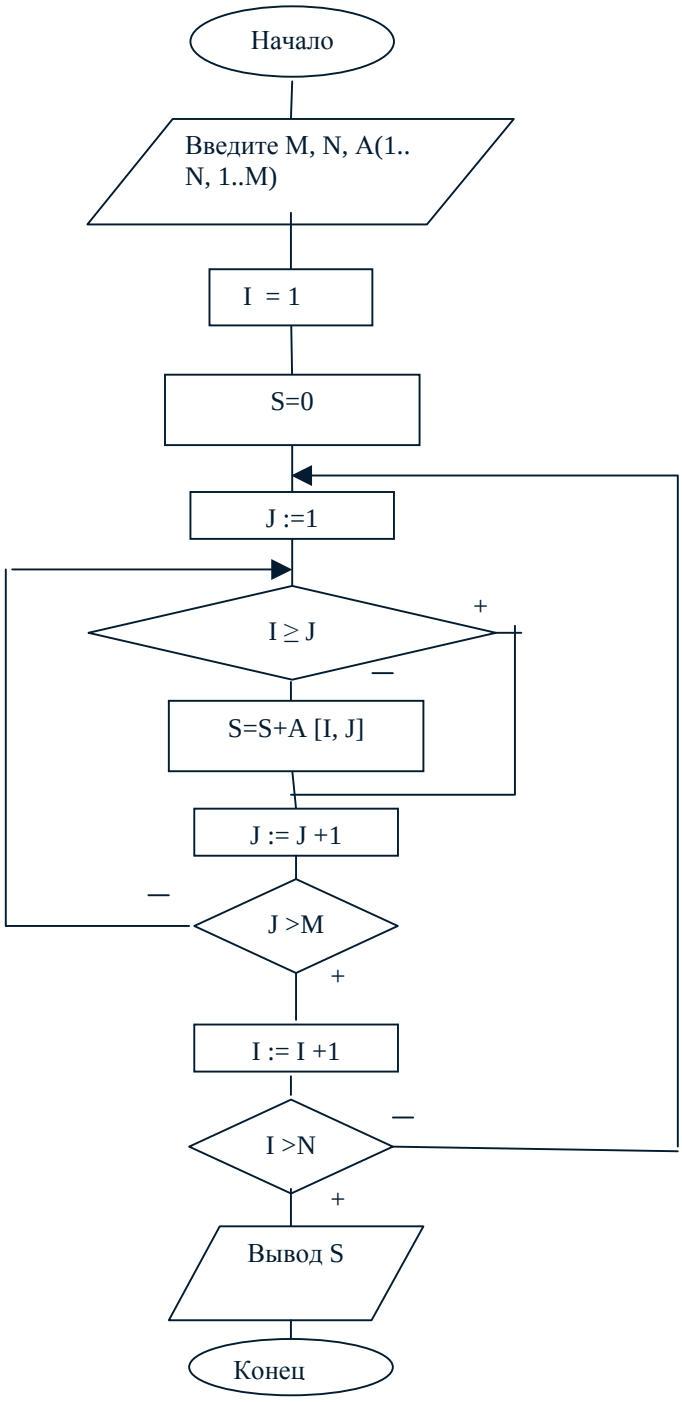
Рис.37.Алгоритм вычисления в каждой строке двумерного массива количества нечетных элементов

*Пример 15.*Составить алгоритм вычисления суммы элементов двумерного массива A(1.. N, 1..M), расположенных выше главной диагонали.

Решение. Для определения условия расположения элементов выше главной диагонали рассмотрим двумерный массив в обобщенном виде на рис. 38. Обратим внимание на диагональные элементы: номер строки и номер столбца совпадают. Значит для определения элементов на главной диагонали достаточно использовать условие I=J, где I- номер строки, J -номер столбца.

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

Рис.38. Пример двумерного массива



Для для определения элементов выше главной диагонали достаточно использовать условие I<J, ниже главной диагонали I>J. По условию задачи нам требуется найти сумму элементов двумерного массива A(1.. N, 1..M), расположенных выше главной диагонали, значит применим условие I<J, связывающее такие параметры элемента массива как номер строки I и номер столбца J.

Алгоритмическое решение задачи вычисления суммы элементов двумерного массива, расположенных выше главной диагонали приведено на рисунке 39.

Данный алгоритм содержит два вложенных цикла, каждый из которых относится к циклу с постусловием.

Рис.39. Алгоритм вычисления суммы элементов двумерного массива, расположенных выше главной диагонали

Задания для самостоятельного выполнения

Составить визуальные циклические алгоритмы для следующих задач обработки двумерных массивов.

1. Ввести двумерный массив A(N,M).Составить визуальный алгоритм замены всех нулевых элементов на минимальный элемент.
2. Ввести двумерный массив A(N.N) . Составить визуальный алгоритм подсчета среднего арифметического значений двумерного массива . Найти отклонение от среднего у элементов первой строки.

3. Ввести двумерный массив A(N,N) . Составить визуальный алгоритм подсчета среднего арифметического значения двумерного массива. Вычислить отклонение от среднего для всех элементов двумерного массива .
4. Ввести двумерный массив A(N,N).Составить визуальный алгоритм замены всех отрицательных элементов на среднее арифметическое значение элементов двумерного массива.
5. Составить визуальный алгоритм нахождения числа строк двумерного массива A(N,N) , количество отрицательных элементов в которых больше P.
6. Ввести двумерный массив размером 7*4 . Найти наибольший элемент двумерного массива . Удалить строку с максимальным элементом.
7. Ввести двумерный массив размером 7*4. Поменять столбец с максимальным элементом с первым столбцом двумерного массива .
8. Ввести двумерный массив размером 7*7. Найти максимальный элемент двумерного массива , расположенный ниже побочной диагонали.
9. Ввести двумерный массив размером 7*4 . Найти наименьший элемент двумерного массива . Перенести строку , содержащую этот элемент в конец.
10. Ввести двумерный массив размером 7*4.Найти максимальный элемент двумерного массива . Поменять столбец, содержащий этот элемент с последним столбцом двумерного массива .
11. Ввести двумерный массив размером 6*4.Найти минимальный элемент двумерного массива . Переставляя строки и столбцы, добиться того , чтобы он оказался в правом нижнем углу.

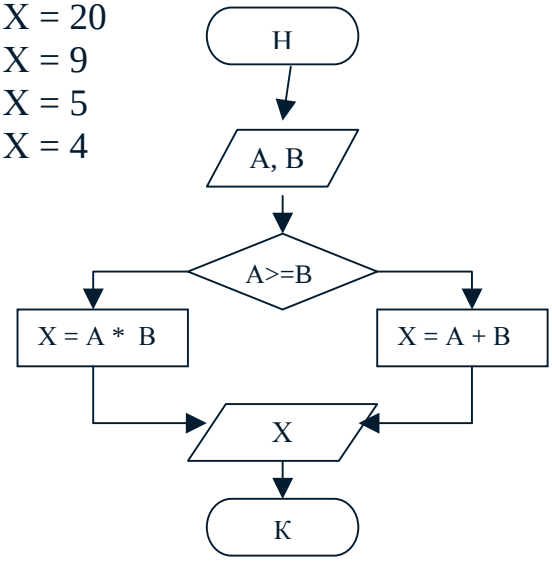
ЗАКЛЮЧЕНИЕ

В данной работе определено место проектирования алгоритмов при компьютерном решении задач, рассмотрена технология проектирования и способ проверки визуальных алгоритмов, приведено множество примеров и заданий для самостоятельного выполнения, алгоритмическое решение некоторых из них имеются в конце данного учебного пособия. Для проверки полученных знаний можно воспользоваться тесовыми заданиями, представленными в приложении 1. Так как построение алгоритмов предшествует процессу написания программы, то в целях ускорения преобразования алгоритмов в текст программ в приложении 2 приведена таблица соответствия основных алгоритмических структур фрагментам программ, написанных на языке Паскаль.

ПРИЛОЖЕНИЕ 1. Тестовый самоконтроль

1. При исходных данных A: = 5, B: = 4 определите результат выполнения визуального алгоритма, изображенного ниже. Определите из каких структурных фрагментов состоит он и к какому виду он относится.

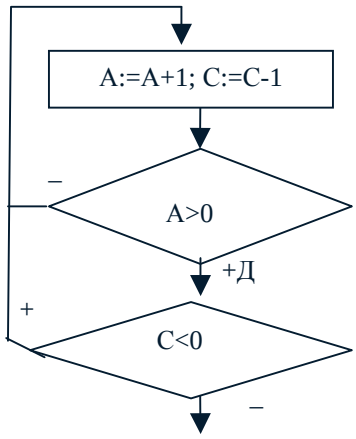
1) X = 20
2) X = 9
3) X = 5
4) X = 4



2. Алгоритм закончит работу при начальных значениях

1) A = 0; C = 1
2) A = 1; C = 0
3) A = -1; C = 1
4) A = 1; C = -1
5) При любых A и C

Проверьте с помощью таблицы трассировки



3.Если элементы массива R [1..4] равны соответственно (5,5,1,-2), то значение выражения R[1+R[4] + R[- 3 + R[1]] равно

1) 0 2) 1 3) -2 4) 6 5) 4

4. Дан двумерный массив a[1..n,1..n]

3	-7	-1
2	- 4	9
- 1	- 6	8

В результате работы нижеприведенного фрагмента алгоритма

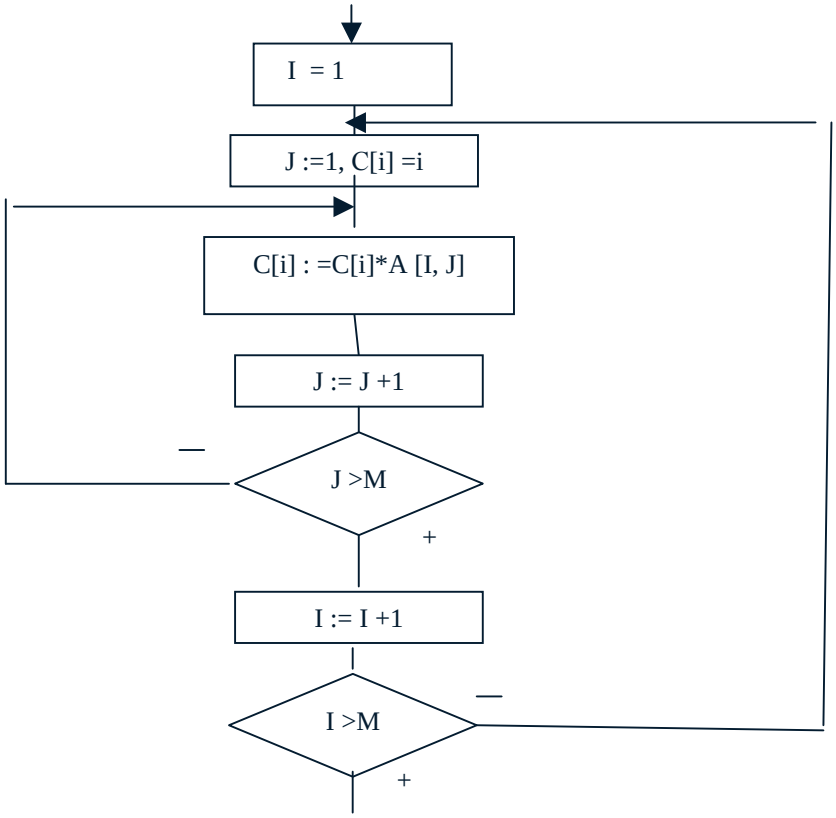
чему будет равно значение переменной c[3]

1) – 48 2) – 18 3) 4 4) 48 5) 144

5.Если элементы массива p [1..4] равны соответственно (4,2,1,3) , то значение выражения p[p[4] – 2 + p[5 – p[2]] равно

1) 5 2) 4 3) 6 4) 3 5) 2

6. Отгадывая целое число, задуманное в промежутке от 1 до 100 можно зада-



вать вопросы, на которые вы получаете ответ "да" или "нет". Чтобы отгадать число, минимально необходимое число вопросов будет (воспользуйтесь методом бинарного поиска)

1) 1 2) 100 3) 7 4) 8 5) 9

7.Записи в таблице

Автор	**Серия**	**Наименование**	**Год издания**	**Кол. стр.**
Визе М.	Компьютер для носорога	Access 2.0	1994	255
Кирсанов Д.	Для чайников	Word 7.0	1996	236

Султанов И.	Для пользователей	Энциклопедия Delphi	1997	300
Уолш Р.	Для начинающих	Windows 95	1996	128

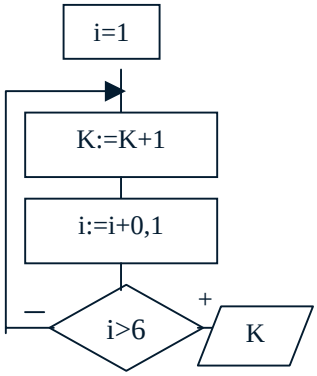
отсортированы по полю:

1) **Наименование**
2) **Кол.Стр.**
3) **Автор**
4) **Год издания**
5) **Серия**

8. Дан фрагмент алгоритма .Как называется данная управляющая структура?

1) Ветвление полное
2) Ветвление неполное
3) Цикл с постусловием
4) Цикл с предусловием
5) Композиция

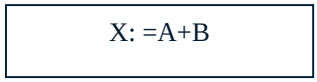
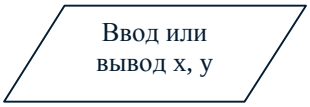
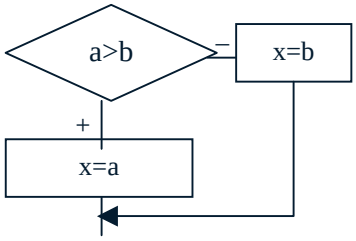
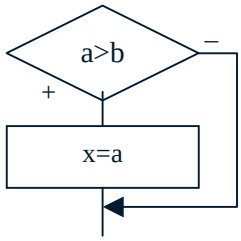
9. Дан некоторый двумерный массив A(1..M,1..M) какое условие позволит определить элементы, кратные 10 и расположенные ниже главной диагонали

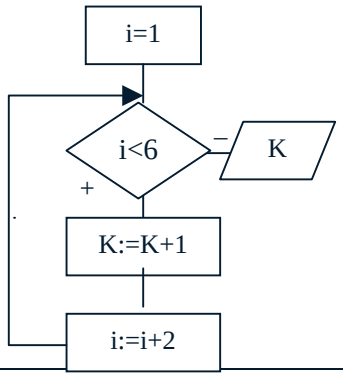


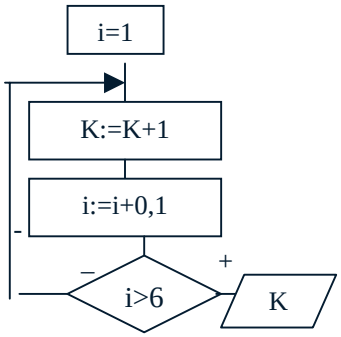
1) i<J 2) i>j 3)A(i,j) mod 10 = 0
4) i>j and A(i,j) mod 10 = 0
5) i<J and A(i,j) mod 10 = 0

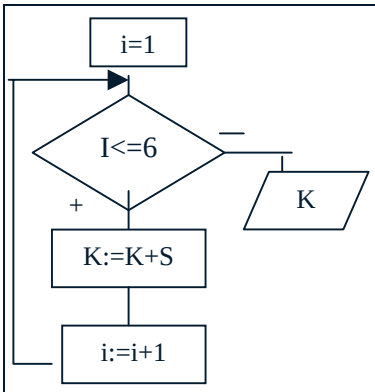
ПРИЛОЖЕНИЕ 2.Таблица соответствия алгоритмических и программных фрагментов

Фрагменты блок-схем алгоритмов	**Назначение**	**Соответствующие фрагменты программ на языке Паскаль**
Начало Конец	**Начало и конец алгоритма**	**Begin** **End**

	**Блок обработки, в нем вычисляются новые значения или производится вызов подпрограммы.**	**X: =A+B**
	**Ввод** исходных данных или **вывод** результатов.	**Read (x, y)** **Write (x, y)**
	**Ветвление полное.** Если значение переменной a больше b, то выполняется x=a , иначе x=b.	**If a>b then X: =a** **else X:=b**
	**Ветвление неполное**. Если значение переменной a больше b, то выполняется x=a.	**If a>b then x:=a**

	**Цикл с предусловием.**
	Пока значение условия **i<6 истино** выполняется тело цикла, то есть действия K=K+1 и i=i+2. Переменная i определяет количество повторений и называется **счетчиком цикла.**
		i: =1;
		While i<6 do
		Begin
		K: =K+1;
		i: =i+2;
		End;
		Write(K);

	**Цикл с постусловием.** Пока значение условия **i>6 ложно** выполняется тело цикла, то есть действия K=K+1 и i=i+0,1.Переменная i определяет количество повторений в цикле и называется **счетчиком цикла.**	**i: =1;** **Repeat** **K: =K+1;** **i: =I+0.1;** **Until I>6;** **Write(K);**

	**Цикл с постоянным приращением счетчика.** В этом цикле изменение счетчика цикла i происходит только на единицу. Пока значение счетчика цикла меньше или равно 6 Выполняется тело цикла, то есть действия K=K+S и i=i+1.	**For i: =1 to 6 do** **K: =K+S;** **Write(K);**

Словарь основных понятий и терминов

Алгоритм – это точно определенная последовательность действий для некоторого исполнителя, выполняемых по строго определенным правилам и приводящих через некоторое количество шагов к решению задачи.

Альтернатива - это нелинейная управляющая конструкция, не содержащая итерацию. Она предназначена для описания различных процессов обработки информации, выбор которых зависит от значений входных данных.

Ветвление - это структура, обеспечивающая выбор между альтернативами.

Визуальные алгоритмы - это алгоритмы, представленные графическими средствами, получили название

Двумерный массив - это структура однотипных элементов, расположенных в виде таблицы значений. Такое представление значений соответствует математическому понятию двумерный массив. Каждый элемент в двумерном массиве идентифицируется номером строки и номером столбца, на пересечении которых он расположен.

Исполнитель алгоритмов определяет элементарные действия, из которых формируется алгоритм.

Итерация - это циклическая управляющая структура, которая содержит композицию и ветвление. Она предназначена для организации повторяющихся процессов обработки последовательности значений данных.

Композиция (следование) - это линейная управляющая конструкция, не содержащая альтернативу и итерацию. Она предназначена для описания единственного процесса обработки информации.

Линейные алгоритмы - алгоритмы,несодержащие блока условия. Они предназначены для представления линейных процессов.

Массив - это однородная структура однотипных данных, одновременно хранящихся в последовательных ячейках оперативной памяти. Эта структура должна иметь имя и определять заданное количество данных (элементов).

Метод бинарного поиска, который также известен, как метод деления пополам. Сущность этого метода поиска заключается в последовательном определении номера S элемента, расположенного в точке деления упорядоченного массива пополам и сравнении искомого значения X с этим элементом массива A(s). Если A(s)=X, то поиск заканчивается. В противном случае возможны две ситуации: если A(s)<X, то все элементы, имеющие номера с 1 по s также меньше X, если A(s)>X, то все элементы, имеющие номера с S по n также больше X в силу упорядоченности массива по возрастанию значений. Поэтому для дальнейшего поиска половину значений массива можно исключить из рассмотрения. В первом случае - левую, во втором случае - правую половину.

Метод структурной алгоритмизации. Этот метод основан на визуальном представлении алгоритма в виде последовательности управляющих структурных фрагментов. Выделяют три базовые управляющие процессом обработки информации структуры: композицию,альтернативу и итерацию.С по-

мощью этих структур можно описать любые процессы обработки информации.

Метод парных перестановок сортировки массива основан на принципе сравнения и обмена пары соседних элементов. Процесс перестановок пар повторяется просмотром массива с начала до тех пор , пока не будут отсортированы все элементы , т.е. во время очередного просмотра не произойдет ни одной перестановки.

Метод модифицированный простого выбора сортировки основывается на алгоритме поиска минимального элемента. В массиве A(1..n) отыскивается минимальный элемент, который ставится на первое место . Для того, чтобы не потерять элемент , стоящий на первом месте , этот элемент устанавливается на место минимального . Затем в усеченной последовательности, исключая первый элемент, отыскивается минимальный элемент и ставится на второе место и так далее n-1 раз пока не встанет на свое место предпоследний n-1 элемент массива A, сдвинув максимальный элемент в самый конец.

Модель - упрощенное представление о реальном объекте, процессе или явлении.

Моделирование - построение моделей для исследования и изучения моделируемого объекта, процесса, явления с целью получения новой информации при решении конкретных задач.

Одномерный массив - это однородная структура однотипных данных, для получения доступа к его элементам достаточно одной индексной переменной

Одномерные символьные масивы - это массивы, составленные из определенной последовательности символов, которые образуют тексты.

Переменные данные - это данные, которые изменяют свои значения в процессе решения задачи.

Последовательность значений - это набор однотипных величин, которые вводятся и обрабатываются циклически.

Постоянные данные - это такие данные, которые сохраняют свои значения в процессе решения задачи (математические константы, координаты неподвижных объектов) и не зависят от внешних факторов.

Разветвленные алгоритмы в своем составе содержат блок условия и различные конструкции ветвления. Ветвление - это структура, обеспечивающая выбор между альтернативами.

Сортировка - процесс перестановки объектов данного массива в определенном порядке. Целью сортировки являются упорядочение массивов для облегчения последующего поиска элементов в данном массиве.

- метод парных перестановок сортировки массива основан на принципе сравнения и обмена пары соседних элементов. Процесс перестановок пар повторяется просмотром массива с начала до тех пор , пока не будут отсортированы все элементы , т.е. во время очередного просмотра не произойдет ни одной перестановки.
- модифицированный метод простого выбора сортировки основывается на алгоритме поиска минимального элемента. В массиве A(1..n) отыскивается минимальный элемент, который ставится на первое место . Для того, чтобы не потерять элемент , стоящий на первом месте , этот элемент устанавливается на место минимального . Затем в усеченной последовательности, исключая первый элемент, отыскивается минимальный элемент и ставится на второе место и так далее n-1 раз пока не встанет на свое место предпоследний n-1 элемент массива A, сдвинув максимальный элемент в самый конец.

Таблица трассировки - это таблица содержащая столько столбцов, сколько переменных и условий в алгоритме, в ней мы выполняем действия шаг за шагом от начала до конца алгоритма для конкретных наборов входных данных.

Циклические алгоритмы являются наиболее распространенным видом алгоритмов, в них предусматривается повторное выполнение определенного набора действий при выполнении некоторого условия. Такое повторное выполнение часто называют циклом. Существуют два основных видов циклических алгоритмов: циклические алгоритмы с предусловием, циклические алгоритмы с постусловием. Они отличаются друг от друга местоположением условия выхода их цикла.

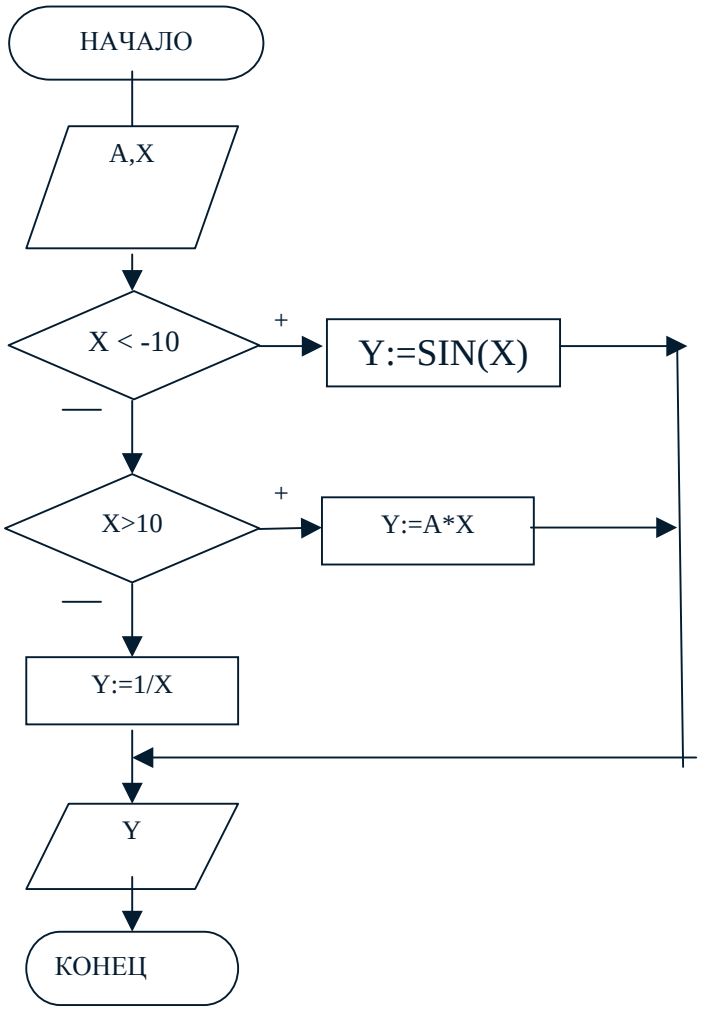
Условно-постоянные данные - это такие данные, которые могут иногда изменять свои значения, но эти изменения не зависят от процесса решения задачи, а определяются внешними факторами

Литература

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ.-М.: Мир,1979.
2. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.- М.: Мир, 1981.
3. Вирт Н. Алгоритмы и структуры данных.- М.Мир,1989.
4. Гейн А.Г. и др. Основы информатики и вычислительной техники.- М.Просвещение , 1992.
5. Гудман С., Хидетниели С. Введение в разработку и аналих алгоритмов. - М.: Мир, 1981.
6. Дайтибегов Д.М., Черноусов Е.А. Основы алгоритмизации и алгоритмические языки.- М.: Финансы и статистика, 1992.
7. Коллинз Г. Блэй Дж. Структурные методы разработки систем: от стратегического планирования до тестирования.Пер. с англ./ Под ред. В.М. Савинкова.- М.:Финансы и статистика, 1986.
8. Кузнецов А.А. и др. Основы информатики.- М.:Дрофа, 1998.
9. Кушниренко А.Г. и др. Информатика.- М.:Дрофа, 1998.
10. Ландо С.К. Алгоритмика: Методическое пособие. - М.: Дро фа,1997.
11. Марков А.А., Нагорный Н.М. Теория алгорифмов.-М.:Наука. Главная редакция физико-математической литературы, 1984.
12. Матросов В.Л. Теория алгоритмов. - М.: Прометей, 1989.
13. Могилев и др. Информатика: Учебное пособие для вузов / А.В.Могилев,Н.И.Пак, Е.К.Хеннер; Под ред. Е.К. Хеннера. - М.: Изд. центр "Академия", 2000.
14. Светозарова Г.Н. и др. Практикум по программированию на языке Бэйсик.- М.:Наука, 1988.
15. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения.- М.: Наука, 1987.

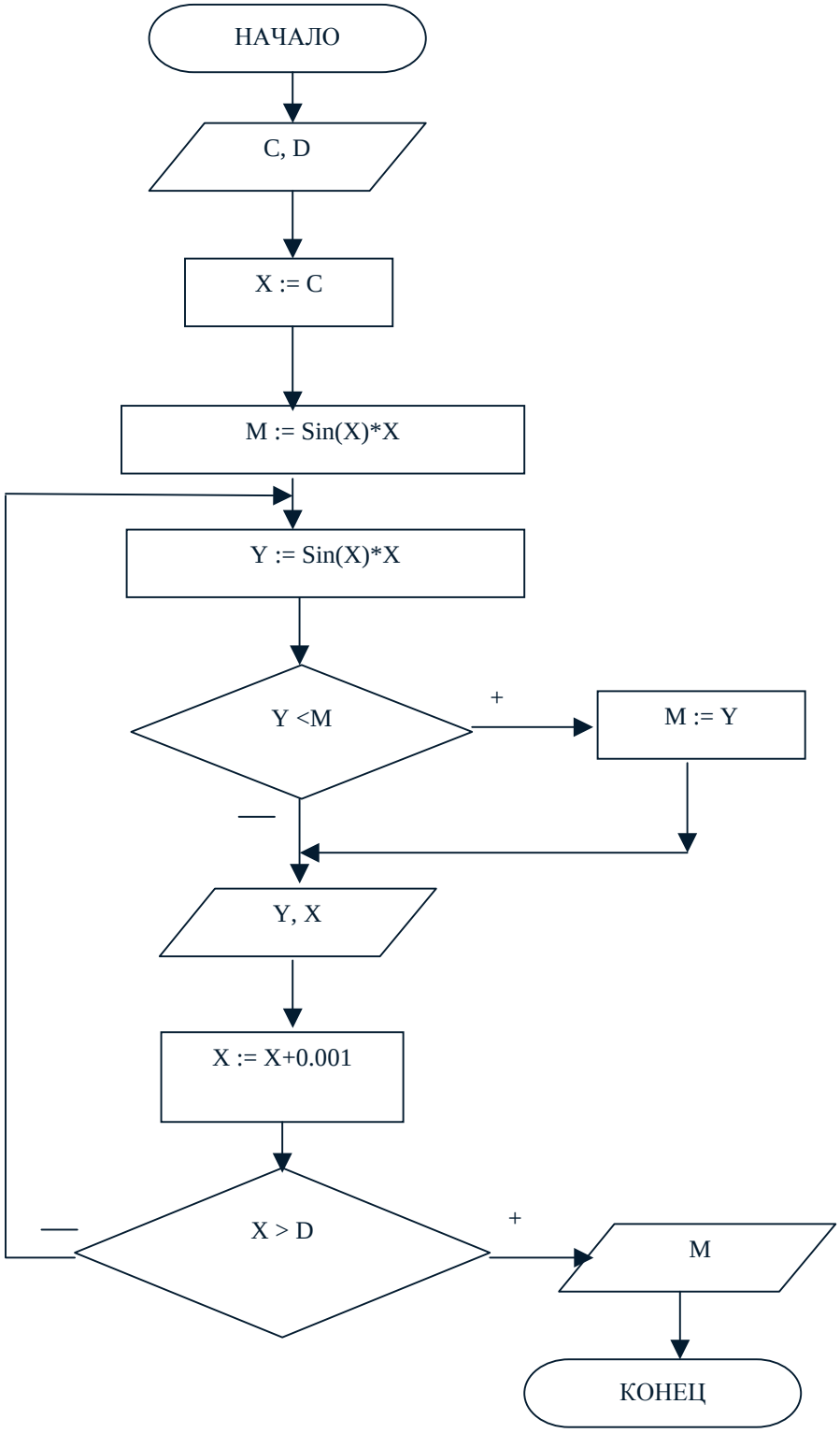
16.Хохлюк В.И. Параллельные алгоритмы целочисленной оптимиза ции.- М.: Радио и связь,1987.

Ответы и решения

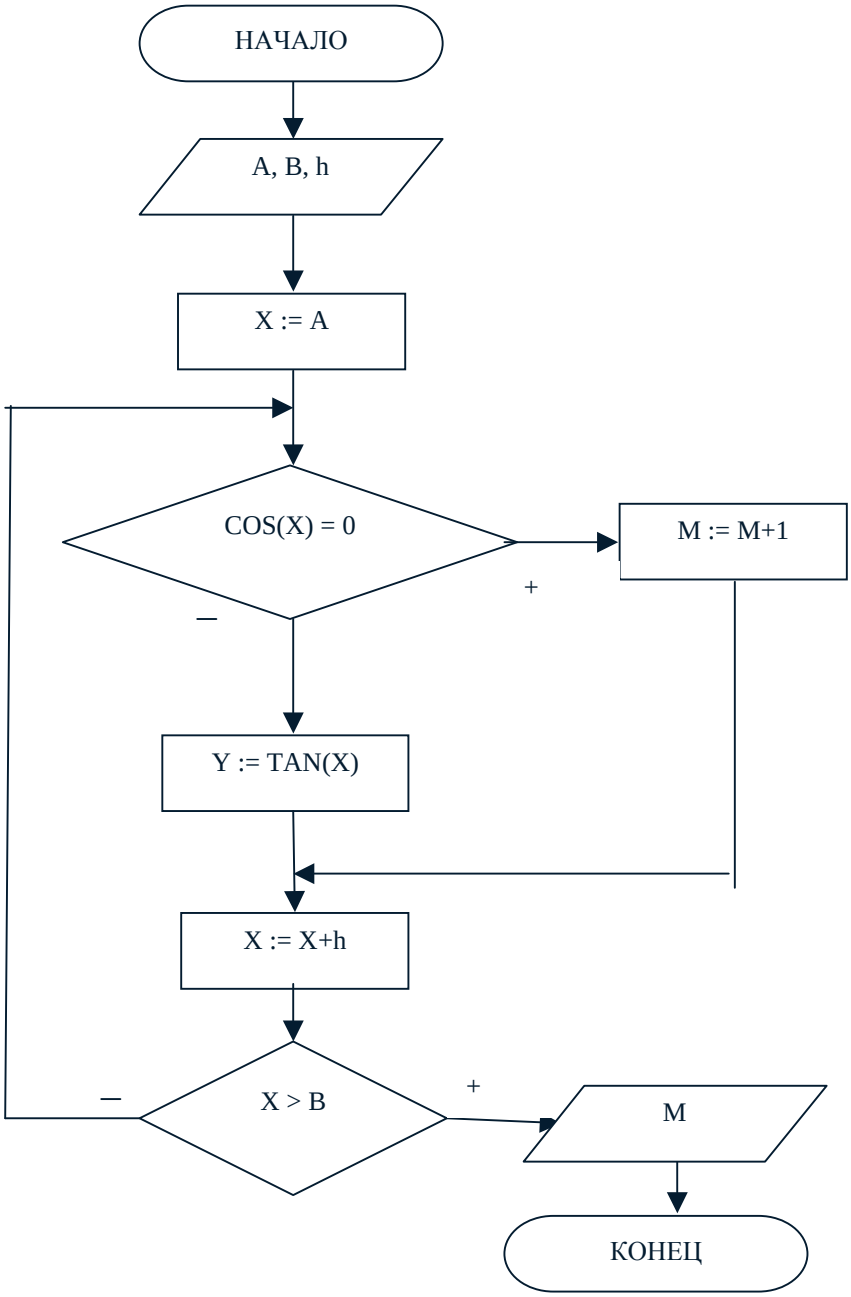


Ответы к теме разветвленные алгоритмы
Задача 7

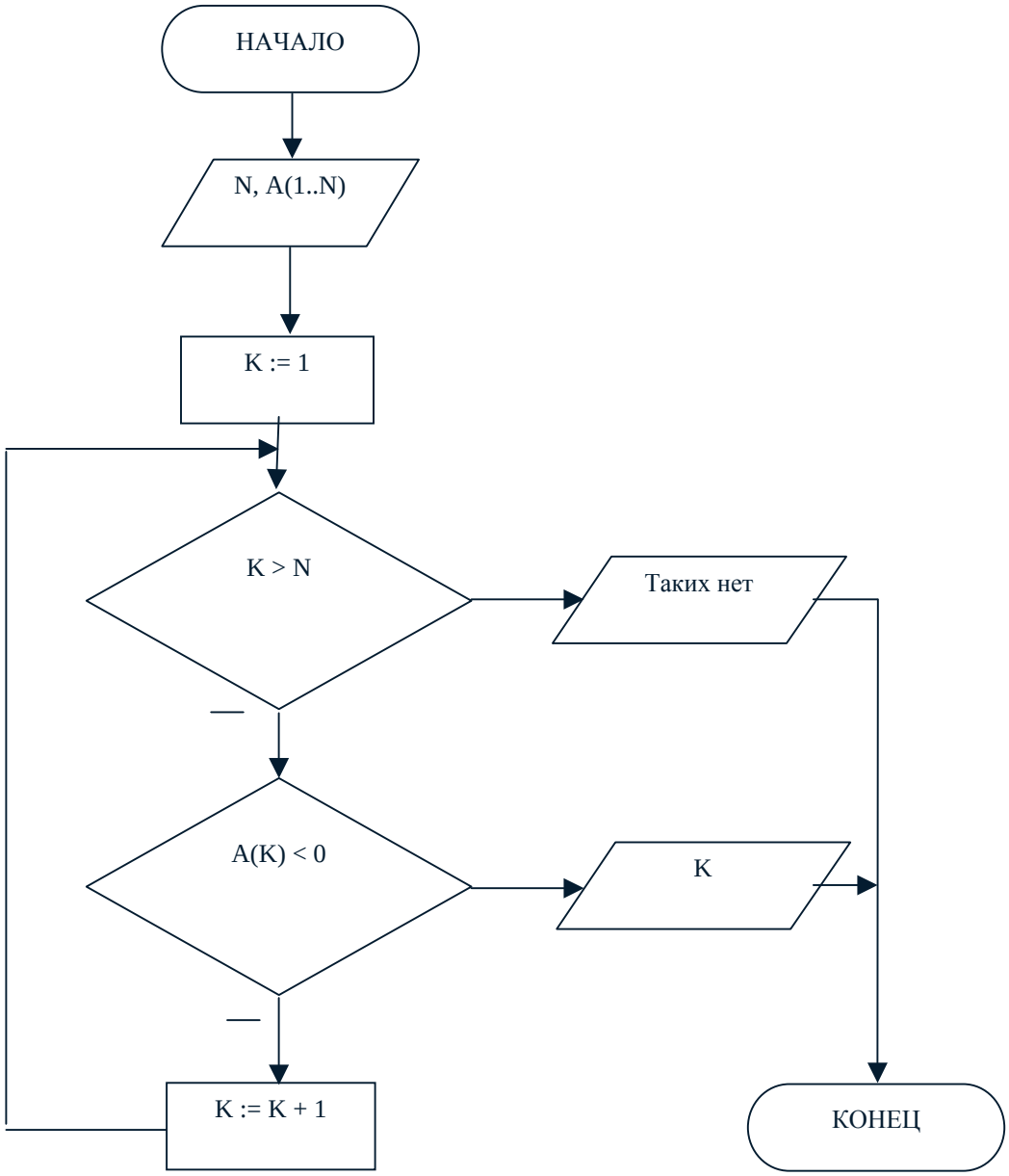
Ответы к теме циклические алгоритмы
Задача 6.



Ответы к теме циклические алгоритмы
Задача 12.

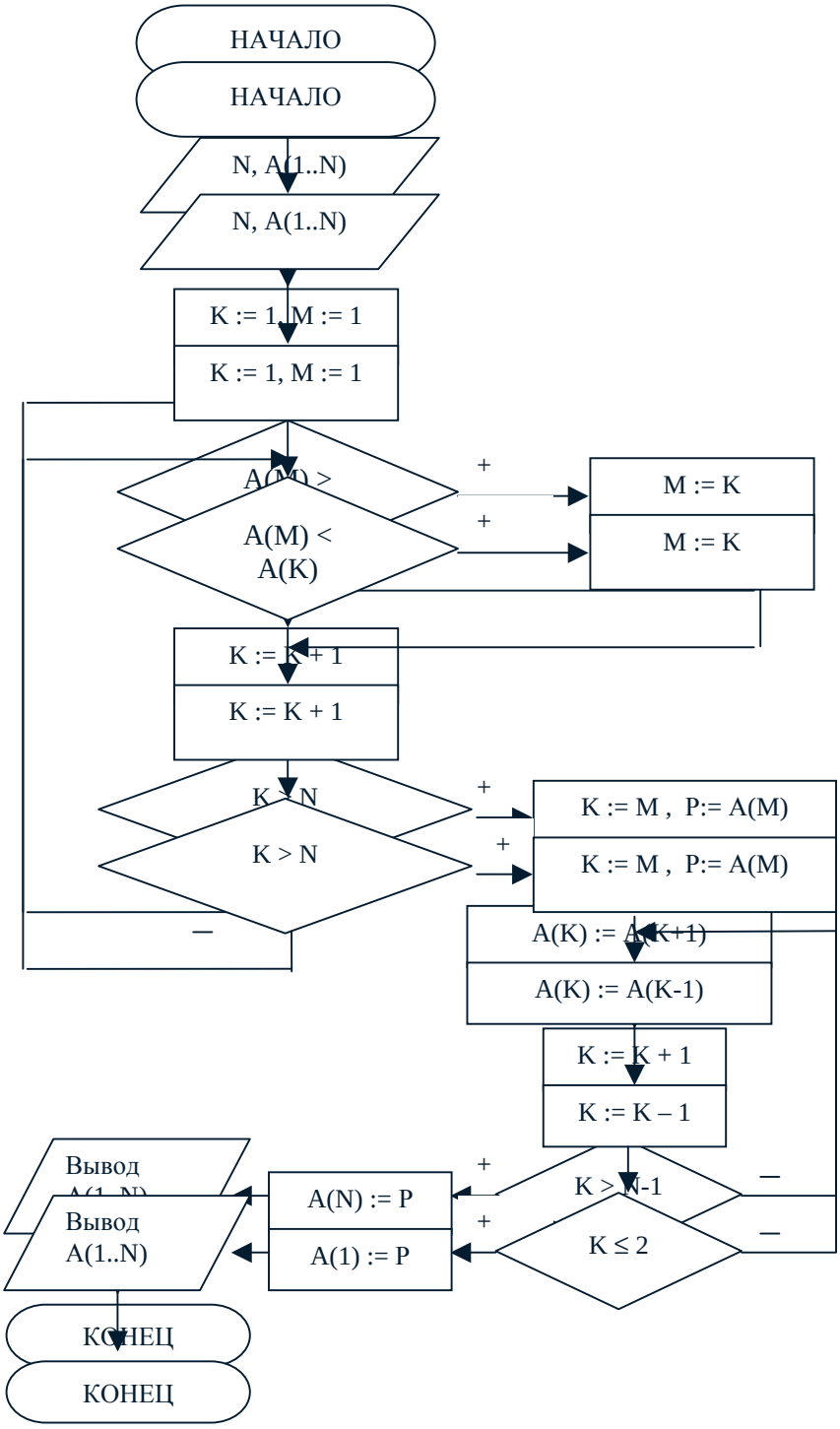


Ответы к теме одномерные массивы
Задача 1.



Ответы к теме одномерные массивы

Задача 10.



Ответы к теме одномерные массивы
Задача 13.

Ответы к тестовому контролю, представленному в приложении 1.

Номер вопроса	Номер правильного ответа
1	1
2	1
3	3
4	5
5	5
6	3
7	3
8	4
9	4

Учебное издание

АФАНАСЬЕВАТатьяна Васильевна.

Основы визуальной алгоритмизации
Учебное пособие