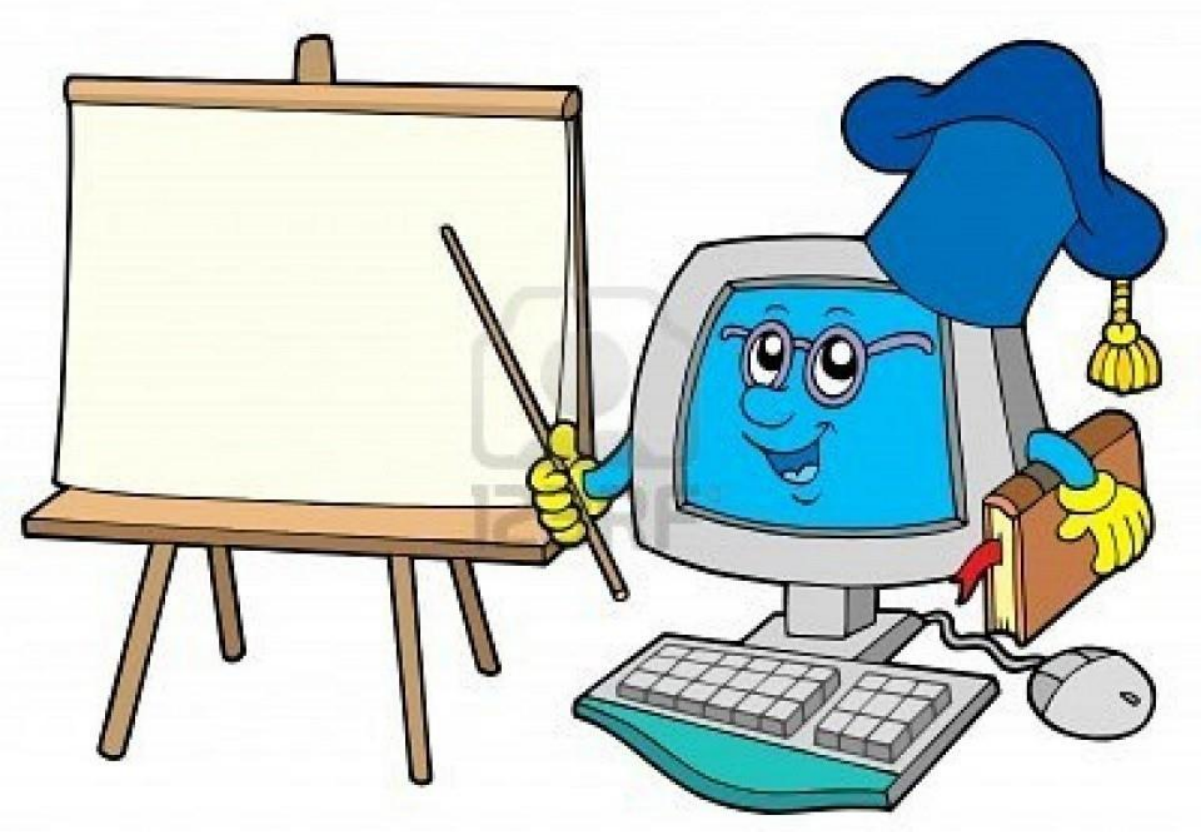
42-2020

ИНФОРМАТИКА

НЕФОРМАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ



Методические указания
к изучению дисциплины «Информатика»
для студентов направления 08.03.01
«Строительство» всех форм обучения

Воронеж 2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования

«Воронежский государственный технический университет»

ИНФОРМАТИКА

НЕФОРМАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Методические указания
к изучению дисциплины «Информатика»
для студентов направления 08.03.01
«Строительство» всех форм обучения

Воронеж 2020

УДК 004.9 (07)
ББК 32,81 я 73

Составители: А. Д. Кононов, А. А. Кононов

Информатика: Неформальное программирование и основы алгоритмизации вычислительных процессов: методические указания к изучению дисциплины «Информатика» для студ. направления 08.03.01 «Строительство» / ФГБОУ ВО «Воронежский государственный технический университет»; сост.: А. Д. Кононов, А. А. Кононов. – Воронеж: Изд-во ВГТУ, 2020. – 30 с.

В доступной форме рассматриваются практические задания и приемы, используемые при построении алгоритмов различной структуры. Приводятся примеры, после подробного анализа которых, предлагаются контрольные вопросы и упражнения для проверки усвоения студентами изучаемого материала.

Предназначены для использования при изучении дисциплины «Информатика» студентами направления 08.03.01 «Строительство» всех форм обучения.

Ил. 7. Библиогр.: 10 назв.

УДК 004.9 (07)
ББК 32,81 я 73

Рецензент – *Д. В.Сысоев, к.т.н., доцент кафедры прикладной математики и механики Воронежского государственного технического университета,*

Печатается по решению редакционно-издательского совета Воронежского государственного технического университета

Введение

Современный специалист, работающий в области строительства, должен уметь пользоваться вычислительной техникой и информационными технологиями. Но при освоении программирования наибольшую трудность для студентов составляет овладение навыками конструирования алгоритмов, которые для наглядности представляются в виде блок-схем. При усвоении методики построения алгоритмов студент должен научиться выявлять структуру решения задачи и сводить ее к четко сформулированной последовательности действий.

Цель данных методических указаний – повысить эффективность самостоятельной работы студентов по овладению методикой конструирования алгоритмов, показать на конкретных примерах типовые, часто встречающиеся конструкции.

В методических указаниях рассматриваются практические примеры, используемые при построении алгоритмов различной структуры: линейных, разветвляющихся, циклических. Представлены задания, после подробного анализа которых предлагаются контрольные вопросы и упражнения для проверки качества усвоения студентами изложенного материала.

В рамках дальнейшего обучения предполагается использовать предложенные задания для решения на компьютере в ходе изучения конкретного алгоритмического языка.

Понятие алгоритма

Методические указания посвящены изучению основ алгоритмизации. Можно уверенно утверждать, что каждый, читающий эти строки, знаком с термином «алгоритм». Его применяют весьма широко, и не только в области вычислительной техники и программирования. Понятие алгоритма, относящееся к фундаментальным концепциям информатики, возникло задолго до появления компьютеров и стало одним из основных понятий математики.

Для решения задачи на ЭВМ необходимо выбранный метод ее решения выразить в виде определенной последовательности операций, выполняемых компьютером. При этом следует детально описать решение задачи, предусмотрев все возможные случаи, переходы и проверки. Разработка алгоритма - необходимый этап в процессе под-

готовки и решения задачи на ЭВМ, и в связи с этим алгоритмы представляют самостоятельную ценность как интеллектуальные ресурсы общества.

Понятие алгоритма широко используется как в математике, так и в программировании. Алгоритм представляет собой описание вычислительного процесса, ведущего от варьируемых начальных данных к искомому результату. Правильно разработанный алгоритм должен обладать следующим набором свойств.

Свойства алгоритма

Дискретность. Алгоритм должен представлять процесс решения задачи как последовательность выполнения простых действий (шагов, этапов).

Детерминированность (Однозначность). Каждое действие (шаг, этап) должно быть четким, однозначным, исключающим произвольное толкование и не оставляющим места для двусмысленности. Выполнение алгоритма носит, по сути, механический характер и не требует никаких дополнительных указаний, а применение алгоритма к одним и тем же исходным данным должно приводить к одному и тому же результату.

Результативность. Алгоритм должен приводить к получению решения задачи или сигнализировать о невозможности получения решения при имеющихся исходных данных. Каждое отдельное действие, как и весь алгоритм, должны иметь возможность реального исполнения, при этом результат должен быть получен за конечное число шагов.

Массовость. Алгоритм должен быть пригодным для решения любых задач, для которых он предназначен. Таким образом, алгоритм разрабатывается в общем виде так, чтобы его можно было применять для класса задач, различающихся только исходными данными. При этом исходные данные выбираются из некоторой области, которая называется областью применимости алгоритма. Например, для решения квадратного уравнения $ax^2 + bx + c = 0$, коэффициенты a, b, c – различные действительные числа, $a \neq 0$.

Разработка алгоритма решения задачи является очень ответственным этапом при подготовке задачи к решению на ЭВМ. Ошибки, допущенные на этапе разработки алгоритма, приведут к тому, что

созданная по этому алгоритму программа будет неработоспособной, то есть не позволит получить правильного результата.

В частности, для удовлетворения свойств массовости и результативности алгоритма необходимо, чтобы результат мог быть получен для любого множества исходных данных, если на них не накладываются ограничения условиями задачи. Это значит, что при построении алгоритма необходимо предусмотреть те варианты исходных данных, при которых возникают «невычислительные» ситуации – деление на ноль, вычисление корня четной степени из отрицательного числа, вычисление логарифма отрицательного или равного нулю аргумента и т.д. – и выдавать сообщение о возникновении такой ситуации в качестве результата.

Контрольные вопросы и упражнения

1. Дайте определение алгоритма.
2. Перечислите основные свойства алгоритма.

Этапы подготовки и решения задачи на ЭВМ

Подготовка задачи к решению на ЭВМ является достаточно сложной процедурой, и ее непосредственная реализация включает ряд этапов.

1. Содержательная постановка задачи – определение цели и условий решения задачи. Подробно описывается характер и сущность всех величин, используемых в задаче и сопровождающих ее условиях. Этап может носить качественный характер и выполняется руководителем работ.

2. Математическая формулировка задачи - построение математической модели (описание связей между объектами задачи математическими соотношениями выполняется специалистом той области, к которой относится задача).

3. Выбор существующего или разработка нового метода решения задачи. При обосновании выбора численного метода определяющими факторами могут быть точность вычислений, время решения задачи на компьютере, требуемый объем основной памяти и др. (выполняется специалистом по прикладной математике).

4. Разработка алгоритма для выбранного метода. На этом этапе составляется алгоритм решения задачи в соответствии с действиями,

задаваемыми выбранным методом. Процесс переработки данных разбивается на самостоятельные участки вычислений и устанавливается порядок выполнения этих участков (выполняет программист).

5. Собственно программирование – запись алгоритма изобразительными средствами конкретного языка программирования (выполняет программист).

6. Подготовка исходных данных, кодировка и трансляция программы. Трансляция заключается в переводе текста программы с языка высокого уровня (ЯВУ) на внутренний язык компьютера. Ее отладка включает визуальный и синтаксический контроль, решение тестового (контрольного) примера. Синтаксические ошибки в программе выявляются транслятором и автоматически выводятся на экран в соответствии с диагностикой, предусмотренной в системе программирования. После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными значениями исходных данных. В современных системах программирования для упрощения процесса обнаружения ошибок в исходной программе могут использоваться специальных программы - отладчики (этап выполняет оператор ЭВМ).

7. Решение задачи на компьютере, получение и анализ результатов (выполняет специалист – постановщик задачи).

Контрольные вопросы и упражнения

1. Перечислить этапы подготовки и решения задачи на ЭВМ.
2. Охарактеризовать каждый из перечисленных этапов.
3. Кто выполняет этап постановки задачи?
4. В чем заключается построение математической модели задачи?
5. Что должно учитываться при выборе численного метода решений?
6. В чем заключается трансляция программы? Отладка программы?

Неформальное программирование

Структурный подход к алгоритмизации определяет так называемое неформальное программирование. Отметим, что понятие алгоритма имеет более широкую область применения, чем математика. В целом мы будем иметь дело с объектами, над которыми выполняется последовательность действий. Объекты могут быть материальны (процесс переработки сырья в производстве, продукты в кулинарии,

переход улицы) или абстрактные (числа, понятия). Таким образом, в целом мы будем рассматривать алгоритмы обработки информации, то есть объектами будут *данные* - в том или ином виде закодированная информация. Результатом работы алгоритма будут *новые данные*. Такие алгоритмы называются вычислительными, понимая вычисления шире, чем вычисления по формуле, а именно как обработку данных. Вычислительный алгоритм можно определить как последовательность действий по переработке исходных данных в результаты. Другими словами, *алгоритм* - точное предписание, которое задает алгоритмический процесс, начинающийся с произвольных исходных данных (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определенного этими исходными данными результата.

Алгоритмический процесс – процесс последовательного преобразования конструктивных объектов (слов, букв, чисел, предложений, графов, логических выражений и т. п.), происходящий дискретными шагами. Каждый шаг состоит в смене одного конструктивного объекта другим.

Если алгоритм представляет собой последовательность инструкций, которые могут быть выполнены на ЭВМ (непосредственно или после автоматической обработки – трансляции, состоящей в приведении алгоритма к исполнимому в ЭВМ виду), то такой алгоритм называется *программой*. Устройство ЭВМ, которое выполняет инструкции программы, называется *центральным процессором* (ЦП); с ним непосредственно связано одно из устройств ЭВМ, именуемое *оперативной памятью* (ОП). ЦП и ОП, вместе с устройствами обмена информацией с окружающим миром (ввода исходных данных и текста программы и вывода результатов), образуют вычислительный комплекс ЭВМ – *вычислитель*. Таким образом, вычислитель - это устройство, которое исполняет алгоритм. Вычислители бывают узкоспециализированными (выполняющими однотипные операции) и универсальными (могут перестраиваться на выполнение того или иного алгоритма). ОП состоит из элементов – *ячеек памяти*, каждая из которых используется для запоминания числа или цифрового кода.

Ячейка памяти имеет следующие свойства:

а) в ней значение может сохраняться до выключения ЭВМ;

б) если в ячейку не заносилось значение, она имеет неопределенное состояние, воспринимаемое как некоторое случайное значение;

в) занесение в ячейку нового значения приводит к автоматическому стиранию прежнего;
г) хранимое в ячейке значение может многократно использоваться в вычислениях.

Контрольные вопросы и упражнения

1. Дать определение алгоритмического процесса.
2. Дать определение программы.
3. Перечислить устройства, входящие в состав компьютера, и выполняемые ими функции.
4. Дать определение центрального процессора, оперативной памяти, вычислителя.
5. Назвать свойства ячейки памяти.

Способы описания схем алгоритмов

Алгоритм решения задачи можно описать различными способами: с помощью словесной или формульно-словесной формулировки, в виде блок-схем или с помощью того или иного языка программирования. Соответствующая программа (греческое слово, означает «предписание») – это текст, содержащий описание алгоритма на том или ином алгоритмическом языке.

Алгоритмические языки могут быть неформальными, если они адресованы людям (компьютеру требуются формализованные языки). Языки программирования весьма разнообразны, однако существует ряд общих понятий в программировании, не связанных со спецификой языка, которые могут быть изучены на неформальном уровне.

Словесная запись алгоритма представляет собой перечисление простейших действий (арифметических, логических и др.), которые нужно выполнить для получения результата, в той последовательности, в какой они должны исполняться.

Пример 1. Например, для алгоритма Евклида нахождения наибольшего общего делителя (НОД) двух положительных чисел N и M (которому более двух тысяч лет), описание с помощью словесного способа будет иметь следующий вид:

Этап 1. Положить X, равным N, и Y, равным M. Перейти к этапу 2.

Этап 2. Проверить: X равно Y? Если Да, то перейти к этапу 6, иначе перейти к этапу 3.

Этап 3. Проверить: X больше, чем Y? Если Да, то перейти к этапу 5, иначе перейти к этапу 4.

Этап 4. Произвести обмен значениями ячеек X и Y. Перейти к этапу 5.

Этап 5. Определить разность значений X и Y. Затем положить X, равным Y, и Y, равным значению полученного остатка. Перейти к этапу 2.

Этап 6. Принять X или Y за искомое значение НОД и прекратить процесс вычисления.

С помощью данного способа можно описывать алгоритмы с произвольной степенью детализации. Недостатком словесного способа записи алгоритма является отсутствие более или менее строгой формализации и наглядности вычислительного процесса.

Формульно-словесный способ записи алгоритма основан на задании инструкций о выполнении конкретных действий в определенной последовательности с использованием математических символов и выражений в сочетании со словесными пояснениями.

Пример 2. Требуется решить квадратное уравнение $ax^2 + bx + c = 0$ в области действительных чисел. Математической моделью этой задачи является известная формула корней квадратного уравнения

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

На основании этой формулы запишем алгоритм:

1. Задать значения a, b, c
2. Вычислить дискриминант $d = b^2 - 4ac$
3. Сравнить дискриминант с нулем, если он больше нуля, то вычислить корни по формуле $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ и перейти к п.4, иначе сообщить: «В области действительных чисел уравнение решений не имеет»
4. Записать результат: «Корни уравнения x_1 *и* x_2».

Пример 3. Формульно-словесная запись вычисления выражения

$$y = \begin{cases} \dfrac{x}{2} & , \quad если \quad x \le 0 , \\ \sqrt{x} & , \quad если \quad x > 0 \end{cases} .$$

будет иметь следующий вид:

Этап 1. Если x>0, то перейти к этапу 2, иначе перейти к этапу 3.

Этап 2. Положить $y = \sqrt{x}$. Перейти к этапу 4.

Этап 3. Положить $y = \dfrac{x}{2}$. Перейти к этапу 4.

Этап 4. Принять значение y за искомый результат и прекратить процесс вычисления.

Пример 4. В этом примере формульно-словесный способ записи алгоритма Евклида нахождения D=НОД(N,M) двух конечных положительных чисел N и M (N>M) может быть записан в виде:

1) разделим N на M, получим остаток

$N=Mq_1+R_1$,

если $R_1=0$, то конец: D=M, иначе

2) разделим M на R_1

$M=R_1q_2+R_2$,

если $R_2=0$, то конец: $D= R_1$, иначе

3) разделим R_1 на R_2

$R_1=R_2q_3+R_3$

и т.д.

. . . . . .

к) разделим R_{k-2} на R_{k-1}

$R_{k-2}=R_{k-1}q_k+0$ и тогда $D=R_{k-1}$.

Здесь q_i – частные, а R_i – остатки на каждом i-ом этапе деления. Остатки – целые положительные числа, они уменьшаются до значения, равного нулю на каком-то *k*-ом этапе. Это описание алгоритма Евклида понятно человеку, но недоступно ЭВМ, так как содержит «и т. д.».

Запишем этот алгоритм более формально. Суть алгоритма в том, что каждый следующий шаг отличается от предыдущего тем, что делитель становится делимым, а остаток – делителем и мы приходим к следующему словесному описанию алгоритма Евклида:

1) Возьмем в качестве Делимого N, в качестве Делителя M

2) Разделим Делимое на Делитель, получим Остаток

3) Если Остаток равен нулю, то конец: перейдем к пункту 4, иначе возьмем в качестве Делимого Делитель, в качестве Делителя Остаток, перейдем к пункту 2

4) Результат – последний Делитель.

В этой записи алгоритма компьютеру не нужно ничего «домысливать», так как отсутствуют многоточие и «и т. д.».

В процессе выполнения алгоритма под словами Делимое, Делитель, Остаток понимаются числа, которые на разных этапах меняют свои значения.

Переменная – это ячейка памяти вычислителя вместе с ее содержимым, которое называется значением переменной. Действия, которые записаны в тексте программы как действия над переменными, фактически выполняются над их значениями.

Операция придания переменной значения называется *присваиванием.* Операция, записанная в пункте 1 – сообщение вычислителю исходных данных, называется *вводом* или чтением. Операция ввода подобна операции присваивания, но при присваивании новое значения переменной берется из памяти вычислителя, а при вводе это значение получается из внешнего мира.

Теперь алгоритм Евклида может быть записан в виде:

1) Ввод (Делимое, Делитель)
2) Остаток := MOD (Делимое, Делитель)
3) Если Остаток = 0 то перейти к п.4,
   иначе { Делимое := Делитель,
   Делитель := Остаток,
   Перейти к п.2 }
4) Вывод (Делитель).

В последней форме алгоритма Евклида использованы обозначения:

:= операция присваивания; MOD (A,B) – операция вычисления целочисленного остатка от деления A на B.

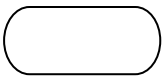
Вновь переходя к словесно-формульному способу описания алгоритма, заменим слово Делимое на буквенное обозначение N, Делитель на M, Остаток на R. Тогда получим:

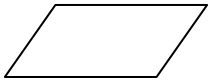
1) Ввод (N,M)
2) R := MOD (N,M)
3) Если R = 0, то перейти к п.4, иначе { N:=M, M := R, перейти к п. 2 }

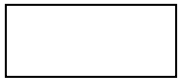
4) Вывод (М).

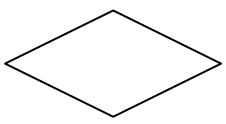
Формульно-словесный способ записи алгоритма по сравнению с предыдущим способом более компактен и нагляден. Однако, при реализации практических задач содержание каждого автономного этапа подлежит дальнейшему описанию и анализу для выявления его элементарных операций, которые необходимо выполнять в пределах этапа, что порождает многообразные допустимые направления вычислительного процесса, сложные связи между этапами и т.д. Изобразительным средством, предназначенным для разрешения подобных затруднений, являются блок - схемы.

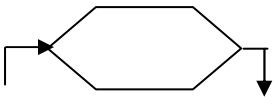
Графический способ описания алгоритма иначе называют блок-схемой. В блок-схемах используются геометрические фигуры, каждая из которых изображает какую-либо операцию, действие, или этап процесса решения задачи. Каждая фигура называется блоком, таким образом, блок-схема – графическое изображение логической структуры алгоритма, в которой каждый этап процесса переработки данных представляется в виде последовательности блоков, имеющих определенную конфигурацию и выполняющих определенные функции. Порядок выполнения этапов показывается стрелками, соединяющими блоки и показывающими связи между ними. Внутри блоков указывается информация, характеризующая выполняемые ими функции, которые записываются словесно или с помощью формул. Блоки размещают сверху вниз или слева направо в порядке их выполнения.

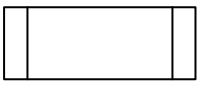

– блок, определяющий начало, конец или прерывание процесса вычисления;


– блок ввода-вывода информации;


– блок вычислений;

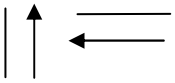

– блок проверки выполнения условия (логический блок);


– блок цикла (модификация);

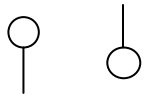


– вычисление по подпрограмме, стандартной программе;

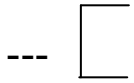
– печать результатов на бумаге;



– линии потока, изображают последовательность связей между блоками;



– соединители, указывают связи между прерванными линиями потока, связывающими блоки;



– пояснения, содержание подпрограмм, формулы.

Рассмотренный выше алгоритм Евклида может быть представлен в виде блок-схемы на рис. 1.

Правила построения алгоритмов на языке блок-схем:

1. Блок-схема строится сверху вниз.
2. В любой блок-схеме имеется один элемент, соответствующий началу, и один элемент, соответствующий концу.
3. Должен быть хотя бы один путь из начала блок-схемы к любому элементу.
4. Должен быть хотя бы один путь от каждого элемента алгоритма в конец блок-схемы.

Описание на алгоритмическом языке. Алгоритм можно рассматривать как задание для исполнителя, который получит правильный результат, если точно выполнит то, что в нем написано. Человек, автоматическое устройство, компьютер – это разные исполнители. Для того, чтобы компьютер мог выполнить алгоритм, он должен быть написан на понятном ему языке. Компьютер понимает машинный язык. Человеку трудно писать и читать алгоритмы на машинном языке, ему понятен естественный язык. Но научить компьютер понимать естественный язык затруднительно потому, что в естественном языке слишком много слов и нет строгих правил записи предложений.

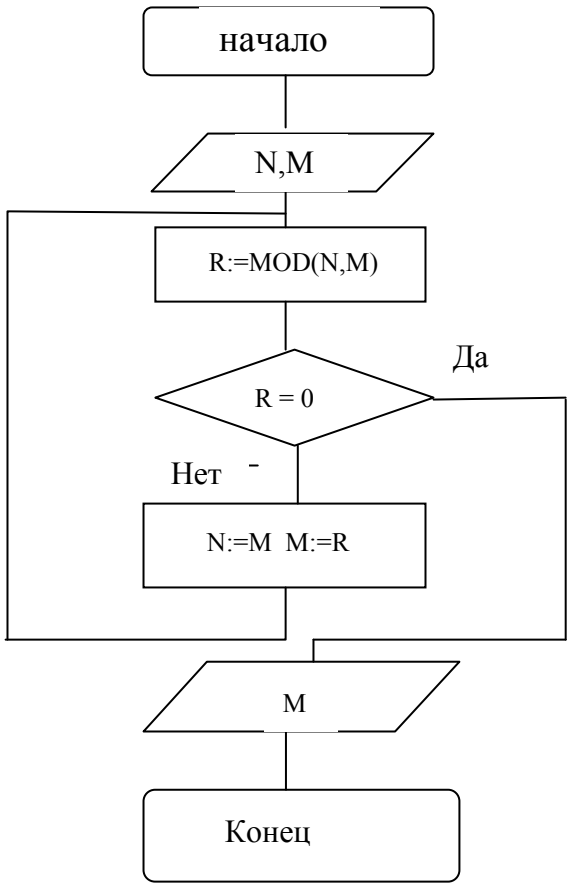


Рис. 1. Блок-схема алгоритма Евклида

Для того чтобы человек и компьютер понимали друг друга, разработаны специальные языки для записей алгоритмов – алгоритмические языки. Алгоритмический язык отличается от машинного тем, что состоит из слов и символов, как естественный язык, но в нем мало слов (обычно 30–40) и очень строгие правила составления предложений. Основные слова языка называют служебными словами. В алгоритмических языках используют слова английского алфавита. Алгоритмический язык легко понимают и человек, и компьютер. Алгоритм, записанный на алгоритмическом языке, – это программа для компьютера. Каждое предложение в программе – оператор.

Совокупность вычислительных процессов, используемых при решении математических, экономических, научно-технических и др. задач на ЭВМ, по характеру связей между выполняемыми в алгоритме операциями в общем виде может быть разделена на три группы: линейные, разветвляющиеся и циклические. Структура алгоритма находится в прямой зависимости от типа отображаемого вычислительного процесса.

Контрольные вопросы и упражнения

1. Какие способы описания схем алгоритмов вы знаете?
2. Что такое переменная, ввод, присваивание?
3. Укажите отличие операций ввода и присваивания.
4. Поясните отличие равенства и присваивания.
5. Укажите достоинства и недостатки словесного и формульно-словесного описания алгоритмов.
6. В чем достоинства графического метода описания алгоритмов?
7. Дать определение блок-схемы.
8. Перечислите известные блоки и укажите их назначение.
9. Перечислите правила построения алгоритмов на языке блок-схем.

Управляющие структуры

В предыдущем описании алгоритма Евклида мы увидели ряд операций различного характера и назначения. Основными являются:

1) последовательное исполнение – исполнение инструкций алгоритма в том порядке, как они представлены в тексте программы (естественный порядок). Линейный алгоритм – это алгоритм, в котором действия выполняются только один раз и строго в том порядке, в котором они записаны;

2) переход – записывается в виде инструкции «перейти к m», где m – метка, указывающая место в программе, куда необходимо передать управление ходом вычислительного процесса;

3) условное исполнение или разветвление – исполнение одной группы действий при выполнении некоторого условия и другой группы действий при его нарушении. Разветвляющийся алгоритм – это алгоритм, в котором то или иное действие выполняется после анализа условия, который на блок-схеме показывают с помощью логического блока (ромб). Логический блок имеет один вход и два выхода (ветвь «да» и ветвь «нет»). Существуют две формы условного исполнения:

а) полное условное предложение «если A, то B, иначе C».

Здесь A – условие, B и C – разные группы действий (рис. 2, а);

б) укороченное условное предложение «если A, то B» (рис. 2, б)

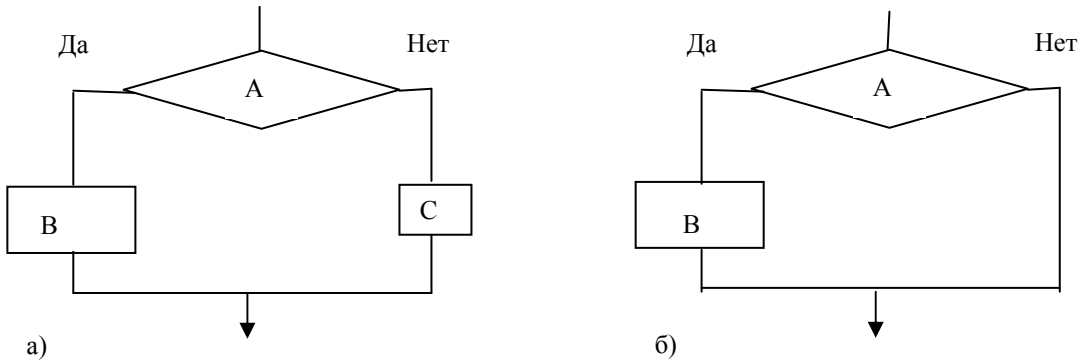


Рис. 2. Блок-схема вариантов разветвлений

4) цикл – многократное исполнение некоторой группы действий при различных значениях входящих параметров. С точки зрения алгоритмизации, цикл – это алгоритм, в котором группа операторов выполняется несколько раз подряд. Циклы бывают с известным (фиксированным, заданным, конечным) числом повторений (другие названия – цикл с параметром, цикл со счетчиком, цикл типа арифметической прогрессии) и с неизвестным числом повторений (по иному – циклы с условием, циклы типа «пока», итерационные циклы). У любого цикла должна производиться проверка окончания, то есть выхода из цикла.

Цикл с предусловием. Проверка окончания осуществляется в начале цикла до исполнения операторов области действия цикла (тела цикла).

«Пока А, повторяй В». Здесь, как и ранее, А – условие, В – операторы тела цикла. Блок-схема изображена на рис. 3, а). Предполагается, естественно, что исполнение группы действий В влияет на выполнение условия А и при каком-то повторе условие А не будет выполнено и произойдет выход из цикла (иначе он некорректно построен).

Цикл с постусловием. Проверка производится в конце цикла после исполнения операторов тела цикла. Блок-схема изображена на рис. 3, б).

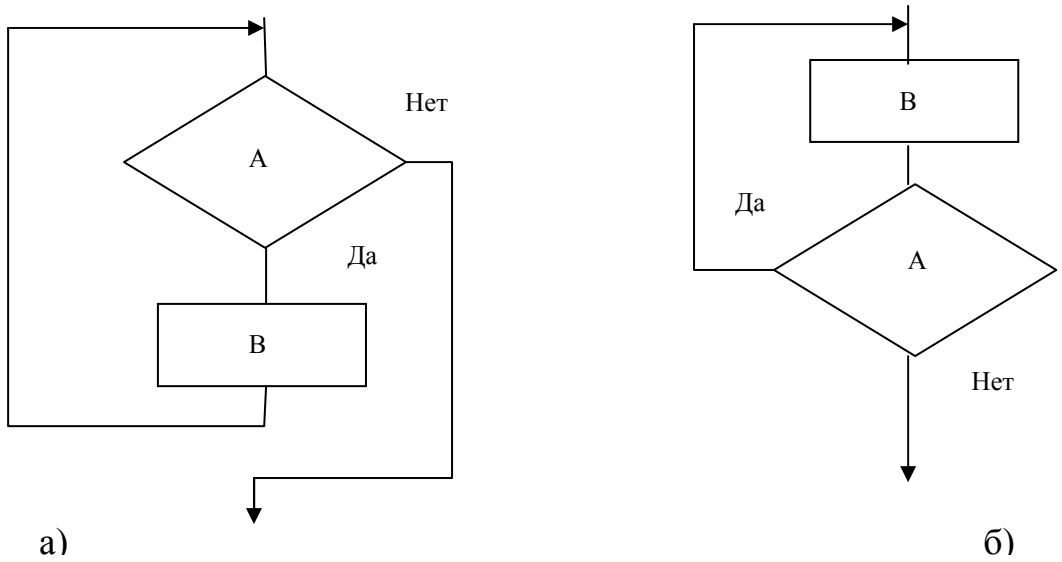


Рис. 3. Блок-схемы циклов с условием

Отличие этих двух циклов в том, что в первом из них группа действий B может быть не выполнена ни разу. Во втором случае это невозможно и тело цикла исполняется хотя бы один раз.

Цикл с параметром. Конструкция цикла выглядит следующим образом.

Для I от M до N шаг H повторяй B,

где I – параметр или индекс цикла, он выполняет роль счетчика, то есть следит за количеством повторений в цикле; M и N – соответственно нижняя и верхняя границы изменения параметра цикла; H – шаг изменения параметра цикла; B – некоторая группа действий.

В процессе работы параметр I принимает последовательные значения M, M+H, M+2H, …, соответствующие членам арифметической прогрессии.

Контрольные вопросы и упражнения

1. Перечислите и охарактеризуйте основные управляющие структуры.
2. В чем отличие полного условного предложения и укороченного условного предложения?
3. Почему цикл с параметром можно называть циклом типа арифметической прогрессии?
4. Дайте определение линейного вычислительного процесса.
5. Какой процесс называется разветвляющимся?

6. Чем определяется выбор ветви вычислений?
7. Какие типы циклов вы знаете?
8. В чем отличие циклов с предусловием и с постусловием?
9. Какие величины задаются в случае цикла с заданным числом повторений?

Типовые задачи программирования

Рассмотрим ряд примеров использования управляющих структур.

1. Тривиальный. Найти наибольшее из двух чисел – max{A,B}.

На неформальном уровне алгоритм решения прост

1. Ввод (A,B)
2. Если A≥B, то MAX:=A, иначе MAX:=B
3. Вывод (MAX)

Решение достигается использованием одного полного условного предложения.

2. Усложняем. Найти наибольшее из трех чисел – max{A,B,C}.

По методу парных сравнений получим следующий алгоритм

1. Ввод (A,B,C)
2. Если A≥B, то {если A≥C, то MAX :=A, иначе MAX:= C},

   иначе {если B≥C, то MAX:=B, иначе MAX:=C}
3. Вывод (M)

Здесь решение получается с использованием трех полных условных предложений, два из которых внутренние и одно внешнее. Графическая иллюстрация решения приведена на рис. 4.

Решение примера 2 для 4, 5,…чисел приводит уже к более громоздким алгоритмам, то есть с увеличением размерности задачи алгоритм, построенный на базе парных сравнений, может стать необозримым.

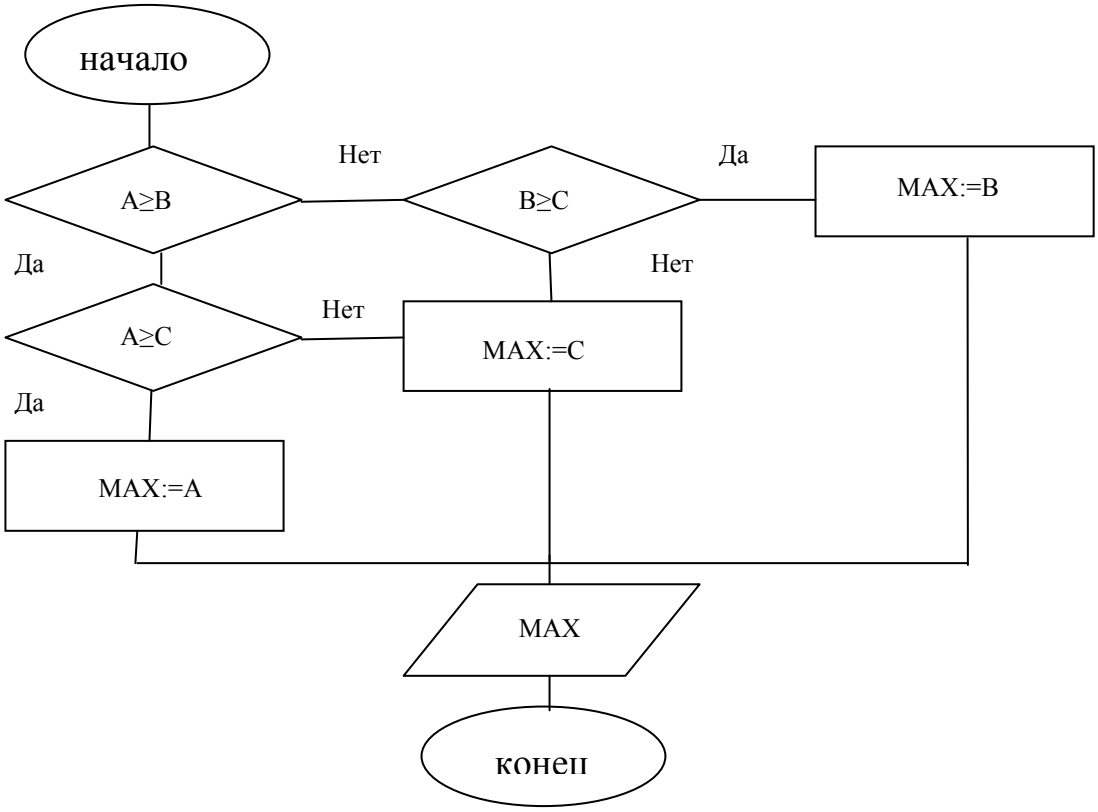


Рис. 4. Блок-схема нахождения наибольшего из трех чисел

3. Переформулируем алгоритм на основе сравнения каждого очередного числа с наибольшим числом, найденным среди предыдущих.

1) Ввод (A,B,C)
2) MAX:=A
3) Если MAX<B, то MAX:=B
4) Если MAX<C, то MAX:=C
5) Вывод (MAX)

В пунктах 3,4 применяются укороченные условные предложения. Теперь после второго этапа ячейка MAX «помнит» наибольшее из одного числа, то есть само это число, после третьего шага – наибольшее из двух чисел, после четвертого – наибольшее из трех чисел и т.д.

4. Обобщаем. Пользуясь последней методикой, найдем наибольшее из n чисел – max $\{a_i, i=1,2,\ldots, n\}$, приведя решение задачи к циклической процедуре с известным числом повторений

1) Ввод (N,A)
2) MAX:=A(1)

3) для I от 2 до N шаг 1 повторяй
если MAX<A(I), то MAX:=A(I)
4) Вывод (MAX)

Здесь A – имя массива – структурного набора однородных данных. Заметим, что шаг, равный единице, настолько популярен, что его можно опустить («по умолчанию»). Записанная конструкция носит название «разветвление в цикле» (рис. 5).

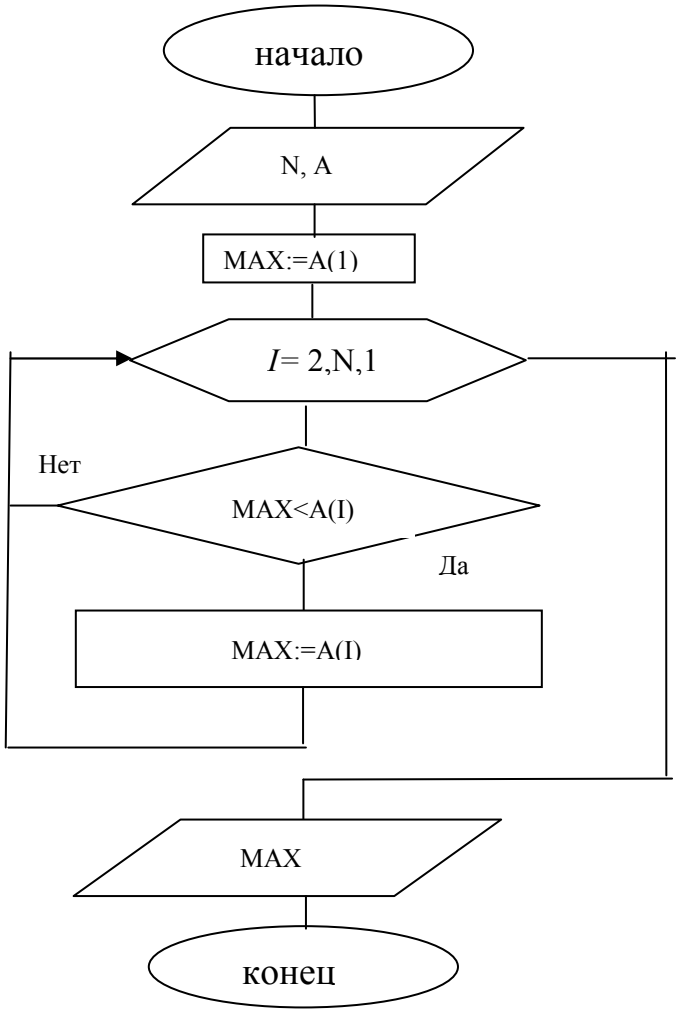


Рис. 5. Блок-схема отыскания наибольшего из N чисел

5. Найти номер первого наибольшего элемента одномерного массива A длины N. Для решения этой задачи наряду с запоминанием наибольшего элемента требуется фиксация и его положения в массиве. Соответствующая блок-схема алгоритма приведена на рис. 6.

Задание. Ответить на вопросы (устно):

– что надо изменить в алгоритме для отыскания минимального элемента в массиве?

– что надо изменить в алгоритме примера 5 для нахождения номера последнего наибольшего элемента в массиве?

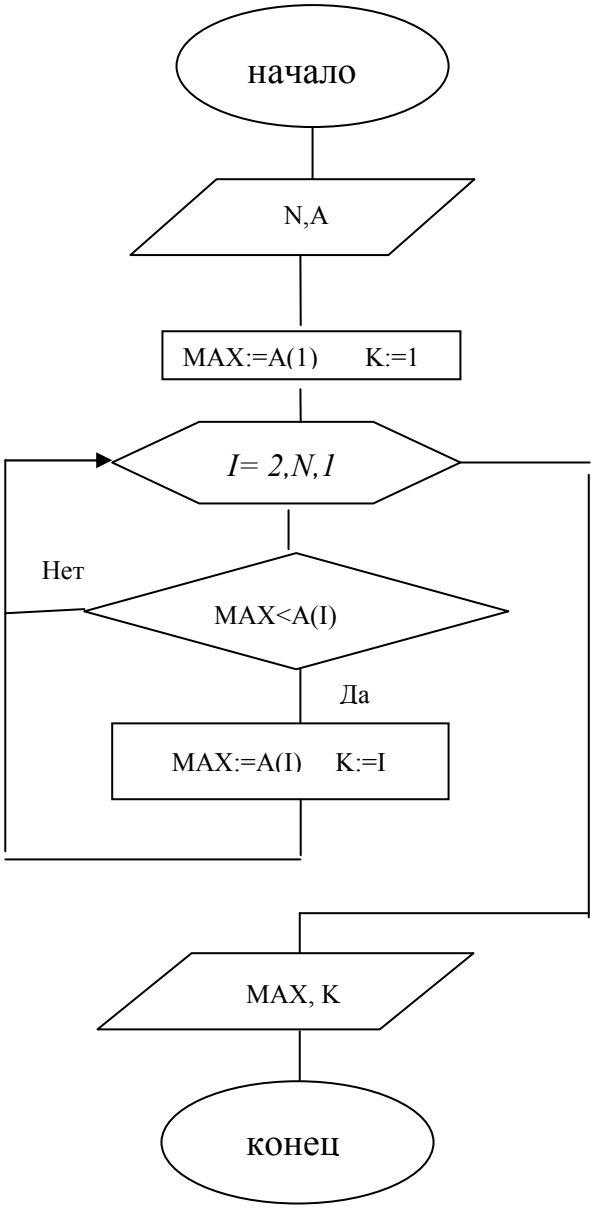


Рис. 6. Определение первого наибольшего элемента в массиве

6. Классическая задача определения суммы и произведения элементов одномерного массива $s = \sum_{i=1}^{n} a_i$ и $p = \prod_{i=1}^{n} a_i$ методом накопления имеет следующее решение

1) Ввод (N,A)*)*
2) S:=0 P:=1
3) Для I от 1 до N шаг 1 повторяй

S:=S+A(I); P:=P*A(I)

4) Вывод (S,P).

Задание. Объяснить необходимость этапа 2). Нарисовать соответствующую блок-схему решения.

7. Дан одномерный массив A длины N. Определить количество его элементов, равных единице.

1) Ввод (N,A)
2) K:=0
3) Для I от 1 до N шаг 1 повторяй
если A(I)=1, то K:=K+1
4) Вывод (K)

Блок-схема представлена на рис. 7.

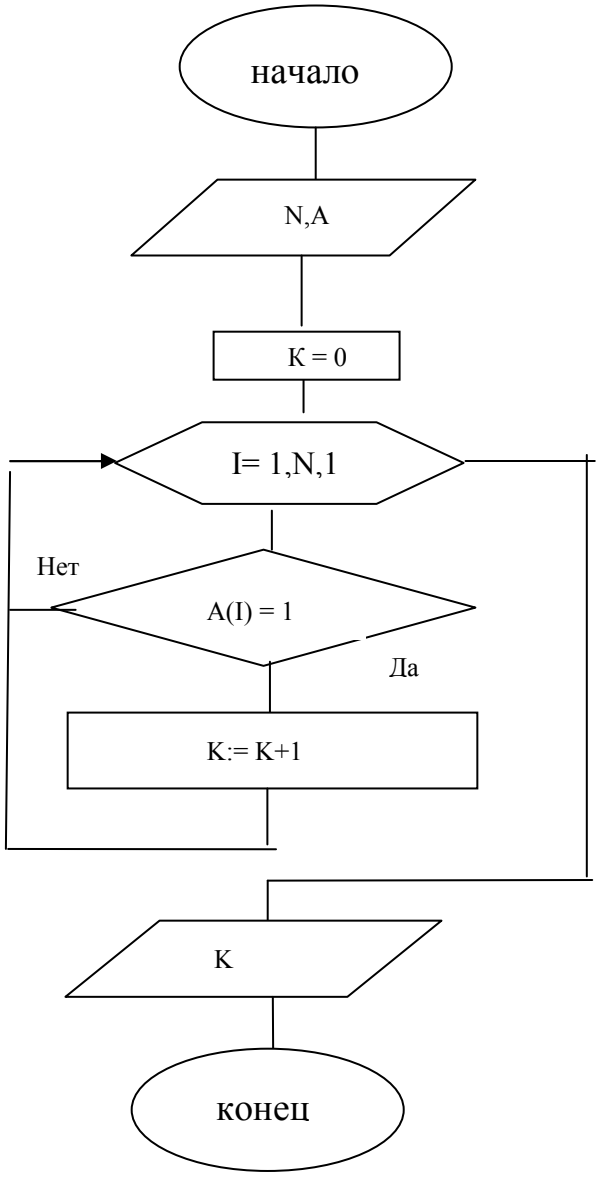


Рис. 7. Подсчет числа элементов, равных единице

8. Вычислить скалярное произведение двух n-мерных векторов $(XY) = \sum_{i=1}^{n} x_i y_i$. Решим эту задачу, используя цикл с предусловием

1) Ввод (N,X,Y)
2) S:=0 I:=0
3) I:=I+1
4) S:=S+X(I)*Y(I)
5) Если I ≤ N, то перейти к 3), иначе перейти к 6)
6) Вывод (S)

Задание. Нарисовать соответствующую блок-схему.

9. Вычислить сумму $s = \sum_{i=1}^{n} \frac{x^i}{i!} = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}.$

Факториал непосредственно компьютер считать не умеет. Поэтому в общем виде на неформальном уровне алгоритм решения задачи можно представить

1) Ввод (N,X)
2) S:=0
3) Для I от 1 до N шаг 1 повторяй
   { вычисли I-ое слагаемое, пополни сумму I-ым слагаемым}
4) Вывод (S).

Это простой пример так называемого нисходящего проектирования программ – последовательная детализация. Сначала для сложной программы разрабатывается обобщенная структура, чтобы не потерялся общий сюжет задания, создается укрупненный эскиз программы, затем отдельные части последовательно детализируются.

Конкретизируем алгоритм. Любое I-е слагаемое можно записать как

$a_i = \frac{x^i}{i!} = \frac{x^{i-1} \cdot x}{(i-1)! \cdot i} = a_{i-1} \cdot \frac{x}{i}$, и алгоритм принимает вид

1) Ввод (N,X)
2) S:=0 A:=1
3) Для I от 1 до N шаг 1 повторяй
   {A := A*X / I S := S+A }
4) Вывод (S)

10. Вычислить сумму бесконечного ряда с точностью до ε

$$s = \sum_{i=1}^{\infty} \frac{x^i}{i!} = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

В этой задаче число повторений в цикле заранее неизвестно, оно зависит от требуемой точности и скорости сходимости ряда. Поэтому здесь могут использоваться только циклы типа «пока», а суммировать надо до тех пор, пока последнее прибавленное слагаемое не станет меньше ε.

1) Ввод (X, ε)
2) S:=0 A:=1 I:=1
3) Повторяй
   A:=A*X / I
   S:=S+A
   I:=I+1
   пока A≥ε
4) Вывод (S)

Задание. Нарисовать блок-схему решения задачи 10, используя цикл с предусловием.

11. Дан набор чисел, его нужно упорядочить, например, в порядке возрастания.

Можно, как это делалось в предыдущих задачах, найти самый маленький элемент, записать его первым, потом среди оставшихся элементов (в укороченном массиве) опять искать минимальный элемент, записать его следующим и далее повторять подобную процедуру, но это неэффективно.

Лучше использовать идею «борьбы с беспорядком» – попарное сравнение двух соседних элементов и, если предыдущий элемент больше соседнего последующего, их меняют местами.

После первого прохождения массива справа окажется самое большое из имеющихся чисел (оно «всплывает» – «метод пузырька»). Снова проход, но уже на один шаг короче и так n раз, но порядок может быть достигнут раньше (в частном случае вообще можно предположить, что исходный массив был уже упорядочен). Если на каком-то проходе перестановок не было, то порядок достигнут и просматривать элементы дальше уже не нужно. Алгоритм (с учетом комментариев в фигурных скобках) может быть записан следующим образом:

1) Ввод (N,A)
2) M:=N {M:=N - сначала весь массив }
P:=1 {P – признак: P=1 – надо проходить; P=0 – не надо }
3) Пока P=1, повторяй
<< P:=0; {если ни одной перестановки, то останется P=0}
< Для I от 1 до M-1 шаг 1 повторяй, {M-1, т.к. считаем пары чисел}
если A(I) > A(I+1) , то
< C:=A(I); A(I):= A(I+1); A(I+1):=C; { Обмен данными через вспомогательную ячейку}
P:=1>
M:=M-1 >>
4) Вывод (A) {это уже упорядоченный набор чисел}

Здесь мы имеем ситуацию с так называемыми «вложенными циклами». Для наглядности приняты обозначения:

Внутренний цикл: < Для I ……. P:=1>

Внешний цикл: << P:=0; < Внутренний цикл >; M:=M-1 >>.

Задание. Нарисовать блок-схему последнего алгоритма.

Перечисленными выше структурами в начальном изучении ограничивается знакомство с неформальным программированием.

Контрольные вопросы и упражнения

1. Записать формульно-словесный алгоритм вычисления следующих выражений:

а) $y = \begin{cases} A^2 + B^2, & \text{если} \quad A \le B, \\ (A+B)^2, & \text{если} \quad A > B \end{cases}$ б) $y = \begin{cases} \dfrac{x}{A}, & \text{если} \quad x < 0, \\ A\sqrt{x}, & \text{если} \quad x > 0, \\ A, & \text{если} \quad x = 0 \end{cases}$

2. Какие типы алгоритмов вы знаете? Подберите пример алгоритма для каждого типа.

3. В каких случаях используются циклы с условием?

4. Что такое «тело цикла с условием»?

5. Может ли тело цикла с условием не выполниться ни разу?

6. Может ли тело цикла с условием выполняться бесконечное число раз?

7. В каких случаях используется цикл с параметром? Как он оформляется? Что происходит при его выполнении?

8. Может ли тело цикла с параметром не выполнится ни разу?

9. Чему равно количество повторений тела цикла с параметром, если параметр цикла принимает:

а) все целые значения от 1 до 10?

б) все целые значения от a до b?

в) все нечетные значения от 1 до 20?

г) все значения от 10 до 100 с шагом 7?

10. Как поменять местами содержимое двух ячеек памяти?

11. Есть 27 монет. Известно, что одна монета фальшивая (ее вес меньше). На чашечных весах можно сравнивать вес монет (весы показывают, какие монеты весят больше, меньше или вес одинаковый). Найдите фальшивую монету. Составить алгоритм решения этой задачи.

12. Нарисовать блок-схему алгоритма прямого вычисления выражения $4x^3 + 3x^2 + 2x + 1$ по заданному значению x.

13. Нарисуйте блок-схему с использованием циклического вычисления для выражения $4x^3 + 3x^2 + 2x + 1$ записанного в виде $x(x(4x + 3) + 2) + 1$.

14. В упражнениях 12, 13 использованы разные алгоритмы вычисления тождественных выражений. Почему алгоритм из упражнения 13 более рационален?

15. Решите задачу 8, используя цикл с параметром.

16. Решите задачу 9, используя циклы с условием.

17. Нарисовать блок-схему для вычисления выражения y

$$y = \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \ldots + \frac{1}{15^2}.$$

18. В чем заключается «нисходящее проектирование программ»?

19. Изложите суть методики построения алгоритма для вычисления сумм бесконечных рядов.

20. Для чего в алгоритме упорядочения массива (задача 11) использовано понятие признака?

21. Что такое «вложенные циклы»?

Пример контрольного теста по алгоритмизации

1. Строго определенная последовательность действий, необходимых для решения поставленной задачи, – это …

а) метод решения;

б) алгоритм;

в) блок – схема.

2. Алгоритм является …

а) последовательностью команд, которую может выполнить исполнитель;

б) системой команд исполнителя;

в) математической моделью;

г) информационной моделью.

3. Ниже перечислены основные свойства алгоритма. Некоторые из этих понятий *не* относятся к основным свойствам. Укажите, какие именно.

а) дискретность;

б) определенность;

в) актуальность;

г) результативность;

д) массовость;

е) строгость;

ж) секретность.

4. Свойство, означающее, что решение задачи, записанное в виде алгоритма, разбито на отдельные простейшие команды, которые расположены в порядке их выполнения, – это …

а) дискретность;

б) определенность;

в) результативность.

5. Массовость алгоритма – это свойство заключается в том, что каждый алгоритм, разработанный для решения некоторой задачи, должен быть применен для решения задач данного типа при всех допустимых значениях исходных данных. Верно ли данное высказывание?

6. Какие Вы знаете способы записи алгоритмов:

а) описание с помощью слов;

б) описание с помощью формул;

в) математическое описание;

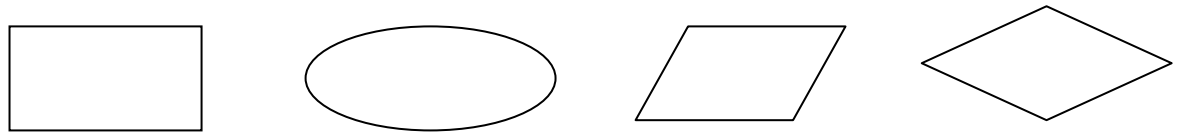
г) описание в виде блок – схем;

д) описание с помощью графических схем;

е) с использованием алгоритмического языка.

7. Графическое описание алгоритмов как последовательности действий называется … . Вставьте пропущенное словосочетание.

8. Какая фигура в блок-схеме обозначает выполнение операции или группы операций?



1 2 3 4

9. Команда алгоритма, в которой делается выбор: выполнять или не выполнять какую либо группу команд, называется … . Вставьте слово.

10. В зависимости от особенностей своего построения алгоритмы делятся на несколько основных групп:

а) линейные;

б) вычислительные;

в) разветвляющиеся;

г) сложные;

д) обработки данных;

е) циклические;

ж) структурные.

Некоторые из этих понятий не относятся к основным группам алгоритмов. Укажите, какие именно.

11. «Линейным называется алгоритм, в котором все этапы выполняются строго последовательно». Верно ли такое высказывание?

12. Укажите правильный вариант ответа. Циклом называется:

а) участок решения задачи, выполняемый строго последовательно;

б) последовательность действий, выполняемых многократно, каждый раз при новых значениях параметров;

в) выбор одного из нескольких вариантов вычислительного процесса.

Заключение

Приведенные материалы будут способствовать формированию у учащихся компетенций, необходимых при решении задач на конструирование алгоритмов на практических занятиях, а в дальнейшем при написании программ, подготовке и сдаче лабораторных работ, компьютерных тестов и зачетов по рассмотренным темам.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Могилев, А.В. Информатика: учебное пособие/ А.В.Могилев, Н.И. Пак, Е.К. Хеннер. - М.: Академия, 2004. – 848 с.
2. Информатика: учебник / под ред. Н.В. Макаровой. – М: Финансы и статистика, 2008. – 766 с.
3. Окулов С. М. Основы программирования/ С. М. Окулов. - М.: БИНОМ, 2013. – 440 с.
4. Кудинов, Ю. И. Основы современной информатики/ Ю. И. Кудинов, Ф. Ф Пащенко. – М.: Лань, 2010. – 256 с.
5. Основы алгоритмизации вычислительных процессов: метод. указания по курсу «Информатика» / ВГАСУ; В. П. Авдеев, В. И. Гильмутдинов, А. Д. Кононов, А. А. Кононов. – Воронеж. –, 2005. – 41 с.
6. Информатика: учебное пособие / В. И. Гильмутдинов, А. Д. Кононов, А. А.Кононов, - Воронеж, 2010. – 56 с.
7. Хлебников, А. А. Информатика: учебник/ А. А. Хлебников. – М.: Феникс, 2013. – 443 с.
8. Информатика: учебное пособие / А. Д. Кононов, А. А. Кононов; ВГАСУ. – Воронеж, 2016. – 53 с.
9. Основы программирования на языке Паскаль: учебное пособие по курсу «Информатика» / А.Д.Кононов, А.А. Кононов; ВГТУ. – Воронеж, 2017. – 53 с.
10. Информатика: учебное пособие / А. Д. Кононов, А. А. Кононов. – Воронеж: Изд-во ВГТУ. - 2018. – 100 с.

Оглавление

ИНФОРМАТИКА

Неформальное программирование и основы алгоритмизации вычислительных процессов

Методические указания к изучению дисциплины
«Информатика» для студентов направления 08.03.01
«Строительство» всех форм обучения

Составители: **Кононов** Александр Давыдович
Кононов Андрей Александрович

Рисунок на обложке : https://yandex.ru/images/search?text= …

Редактор Аграновская Н.Н.

Подписано в печать 18.03. 2020. Формат 60×84 1/16. Бумага для множительных аппаратов.
Усл.-печ. л. 1,4. Тираж 305 экз. Заказ № 41.

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский проспект, 14

Участок оперативной полиграфии издательства ВГТУ
394026 Воронеж, Московский проспект, 14