Лекция № 2

Алгоритмы. Свойства и формы представления

Вопросы лекции:

1. Понятие алгоритма.
2. Свойства алгоритма.
3. Формы представления алгоритмов.
4. Основные алгоритмические структуры.

Понятие алгоритма

Алгоритмизация – это процесс построения алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения.

Подготовка задачи для решения на ЭВМ состоит из нескольких этапов:

1.Постановка задачи
2.Формализация задачи
3.Построение алгоритма
4.Составление программы на языке программирования
5.Отладка и тестирование программы
6.Проведение расчетов и анализ полученных результатов

Алгоритм - это система правил, описывающая последовательность действий, которые необходимо выполнить, чтобы решить задачу.

Алгоритм - некоторая последовательность предписаний (правил), однозначно определяющих процесс преобразования исходных и промежуточных данных в результат решения задачи.

Свойства алгоритма:

Дискретность означает, что выполнение алгоритма разбивается на последовательность законченных действий - шагов. Каждое действие должно быть завершено исполнителем прежде, чем он перейдет к выполнению следующего. Значения величин в каждом шаге алгоритма получаются по определенным правилам из значения величин, определенных на предшествующем шаге.

Определенность предполагает то обстоятельство, что каждое правило алгоритма настолько четко и однозначно, что значения величин, получаемые на каком-либо шаге, однозначно определяются значениями величин, полученными на предыдущем шаге, и при этом точно известно, какой шаг будет выполнен следующим.

Результативность (и **конечность**) алгоритма предполагает, что его исполнение сводится к выполнению конечного числа действий и всегда приводит к некоторому результату. В качестве одного из возможных результатов является установление того факта, что задача не имеет решений.

Основные характеристики алгоритма:

Массовость понимается, что алгоритм решения задачи разрабатывается в общем виде так, чтобы его можно было применить для целого класса задач, различающихся лишь наборами исходных данных. В этом свойстве и заключена основная практическая ценность алгоритма.

Под **эффективностью** алгоритма будем понимать такое его свойство (качество), которое позволяет решить задачу за приемлемое для разработчика время. К параметру, характеризующему эффективность алгоритма, следует отнести также объем памяти компьютера, необходимый для решения задачи.

Формы представления алгоритмов

1. ***Словесный*** – содержание этапов вычислений задается на естественном языке в произвольной форме с требуемой детализацией.

Словесное описание имеет минимум ограничений и является наименее формализованным. Однако при этом алгоритм получается и наименее строгим, допускающим появление неопределенностей. Также в этой форме алгоритм может оказаться очень объемным и трудным для восприятия человеком.

ПРИМЕР. Пусть задан массив чисел. Требуется проверить, все ли числа принадлежат заданному интервалу. Интервал задается границами A и B.

п.1 Берем первое число. На п.2.
п.2 Сравниваем: выбранное число принадлежит интервалу;
если да, то на **п.3, если нет – на п.6.**
п.3 Все элементы массива просмотрены? Если да, то на п.5,
если нет – то на **п.4.**
п.4 Выбираем следующий элемент. На п.2.
п.5 Печать сообщения: все элементы принадлежат интерва-
лу. На **п.7.**
п.6 Печать сообщения: не все элементы принадлежат интер-
валу. На **п.7.**
п.7 Конец.

При этом способе отсутствует наглядность вычислительного процесса, т. к. нет достаточной формализации.

Формы представления алгоритмов

- **2. *Формульно-словесный*** – задание инструкций с использованием математических символов и выражений в сочетании со словесными пояснениями.

Например, требуется написать алгоритм вычисления площади треугольника по трем сторонам.

п.1 – вычислить полупериметр треугольника

$p=(a+b+c)/2.$ К п.2.

п.2 – вычислить $S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$

К п.3.

п.3 – вывести S , как искомый результат и прекратить вычисления.

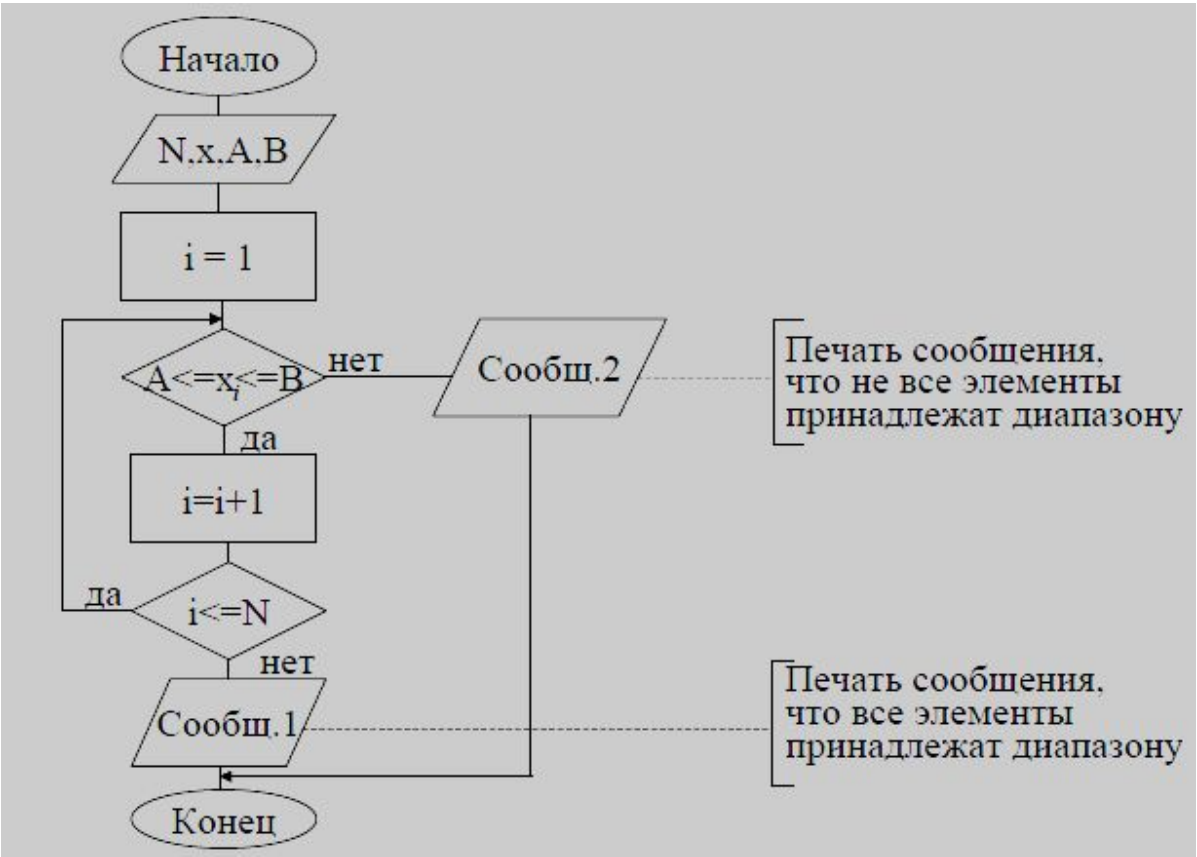
При использовании этого способа может быть достигнута любая степень детализации, более наглядно, но не строго формально.

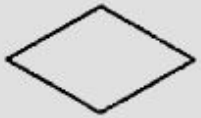
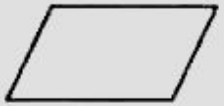
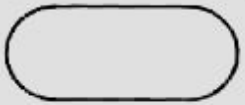
Формы представления алгоритмов

3. *Блок - схемный* – это графическое изображение логической структуры алгоритма, в котором каждый этап процесса переработки данных представляется в виде геометрических фигур (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций.

Внутри блоков указывается поясняющая информация, характеризующая выполняемые ими действия. Конфигурацию и размер блоков, а также порядок построения схем определяет ГОСТ 19002 и ГОСТ19003.

Блок-схемы могут быть традиционные и структурированные.



Название символа	Обозначение	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Блок проверки условия, имеющий один вход и ряд альтернативных выходов, один из которых может быть активизирован после выполнения условий
Модификация		Модификация команды или группы команд с целью воздействия на некоторую последующую функцию
Предопределенный процесс		Процесс, состоящий из одной или нескольких операций (шагов) подпрограммы или модуля
Ввод-вывод		Ввод-вывод информации, при котором отсутствует необходимость в описании фактического носителя данных
Пуск-останов		Начало или конец алгоритма, вход (выход) подпрограммы
Линии потока данных		Отображает поток данных или управления (при необходимости могут быть добавлены стрелки-указатели)
Соединитель		Используется для обрыва линии и продолжения ее в другом месте блок-схемы
Комментарий		Символ используют для добавления комментариев или пояснительных записей

Формы представления алгоритмов

4. Псевдокод **-** позволяет формально изображать логику программы, не заботясь при этом о синтаксических особенностях конкретного языка программирования. Обычно представляет собой смесь операторов языка программирования и естественного языка. Является средством представления логики программы, которое можно применять вместо блок-схемы. Запись алгоритма в виде псевдокода:

Выбираем первый элемент ($i=1$)

IF $A>x_i$ или $x_i>B$ **THEN**

печать сообщения и переход на конец

ELSE

переход к следующему элементу($i=i+1$)

IF массив не кончился ($i <= n$) **THEN**

переход на проверку интервала

ELSE

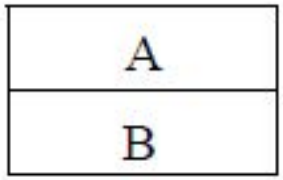
печать сообщения, что все элементы входят в интервал

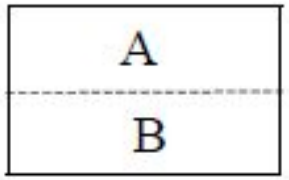
Конец

Формы представления алгоритмов

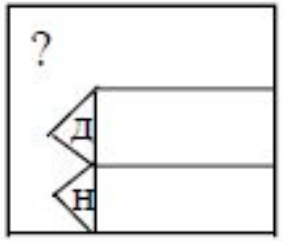
5. Структурные диаграммы - могут использоваться в качестве структурных блок-схем, для показа межмодульных связей, для отображения структур данных, программ и систем обработки данных. Существуют различные структурные диаграммы: диаграммы Насси-Шнейдермана, диаграммы Варнье, Джексона, МЭСИД и др.

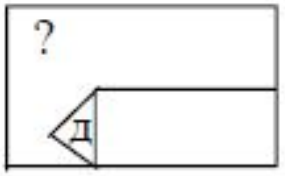
Основные элементы МЭСИД:



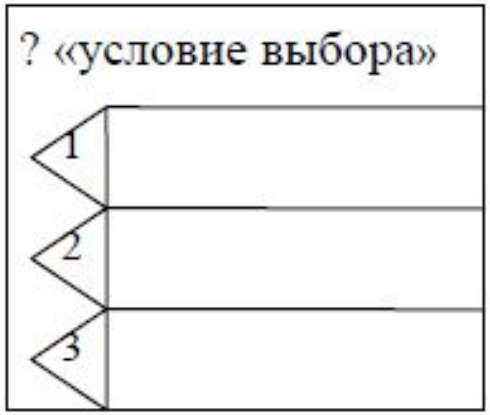


- следование

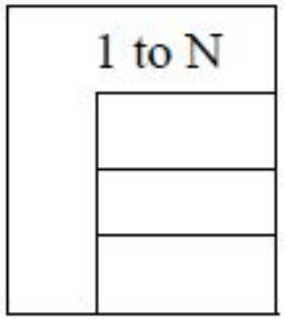


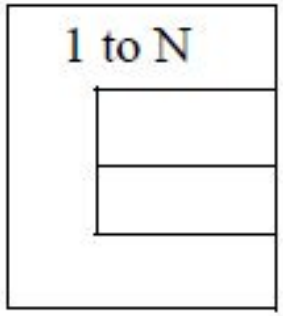


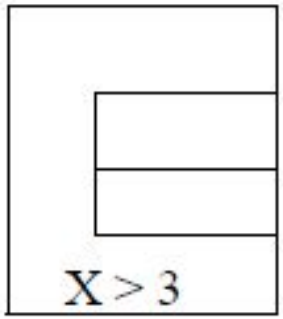
- развилка



- выбор







- повторение

Формы представления алгоритмов

6. Языки программирования - изобразительные средства для непосредственной реализации программы на ЭВМ.

Программа – алгоритм, записанный в форме, воспринимаемой ЭВМ. Каждая машина имеет свой собственный язык (машинный язык) и может выполнять программы только на этом языке. Это последовательность машинных команд. Писать программы на машинном языке очень сложно и утомительно. Для повышения производительности труда программистов применяются искусственные языки программирования. При этом требуется перевод программы, написанной на таком языке, на машинный язык. Этот перевод выполняет ***транслятор***. Наиболее часто встречающимся транслятором интерпретирующего типа является транслятор с языка Бейсик, где команды читаются, преобразуются и выполняются сразу. Итогом работы такого транслятора являются требуемые результаты.

Текст программы на исходном языке сначала переводится в текст на машинном языке и получается так называемый объектный модуль. Затем объектный модуль должен быть обработан программой Редактором межпрограммных связей и только после этого программа будет готова к выполнению.

Виды алгоритмов и их реализация

Алгоритмы в зависимости от цели, начальных условий задачи, путей ее решения, определения действий разработчика **подразделяются на**:

- **механические**, или **детерминированные** (жесткие);
- **гибкие**, или **стохастические** (вероятностные и эвристические).

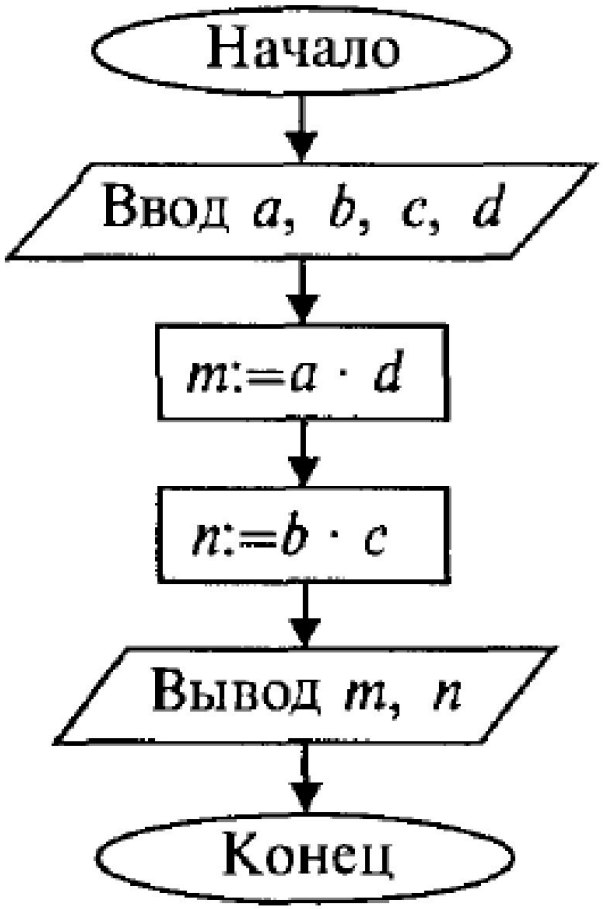
Механический алгоритм задает определенные действия, обозначая их в единственной последовательности, обеспечивающей однозначный требуемый (искомый) результат в том случае, если выполняются условия процесса, для которых разработан алгоритм. К таким алгоритмам относятся алгоритмы работы машин, станков, двигателей и т. п.

Вероятностный (стохастический) алгоритм предлагает программу решения задачи несколькими путями или способами, приводящими к достижению результата.

Эвристический алгоритм (от греческого слова «эврика») — это такой алгоритм, в котором достижение конечного результата однозначно не определено, так же как не обозначена вся последовательность действий. В этих алгоритмах используются универсальные логические процедуры и способы принятия решений, основанные на аналогиях, ассоциациях и прошлом опыте решения похожих задач. При реализации эвристических алгоритмов большую роль играет интуиция разработчика.

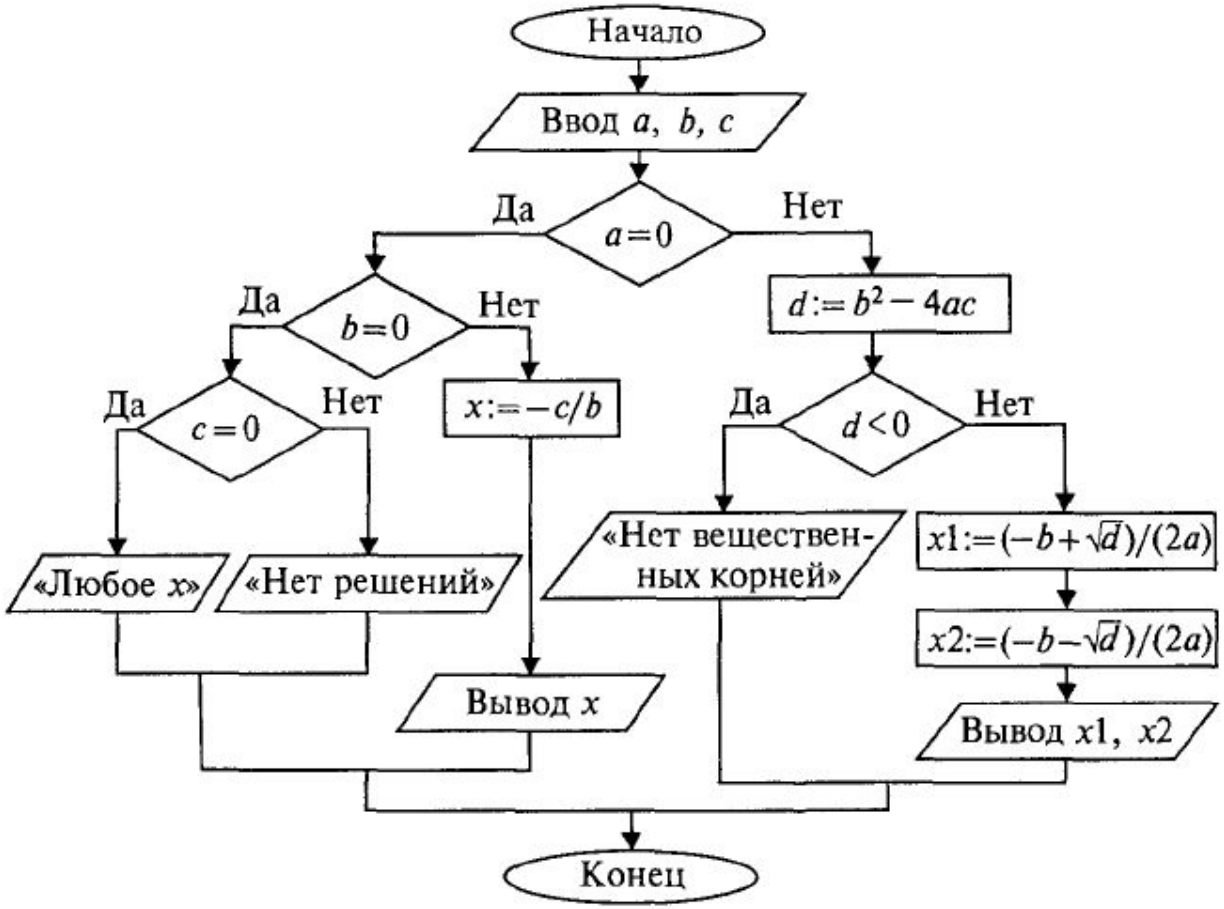
Основные алгоритмические структуры

Линейный вычислительный процесс - это процесс, блоки которого выполняются последовательно один за другим (порядок выполнения блоков естественный).



Основные алгоритмические структуры

Разветвляющаяся структура используется тогда, когда возникает необходимость в зависимости от исходных данных или от полученных промежуточных результатов осуществлять вычисление по одним или другим формулам, то есть в зависимости от выполнения какого-то логического условия вычислительный процесс должен идти по одной или другой ветви. Такой процесс называют **разветвляющимся**.



Примеры на языке VB

Выбор в программе

Выбор в программе выполняют условные операторы. Условные операторы помогают осуществить «ветвление» программы, т.е. передать управление по условию, на ту или иную «ветку» - это может быть фрагмент текста, процедура, подпрограмма или даже другой модуль.

Виды условных операторов Visual Basic:

If ... Then ... Else ...

Select ... End Select

Ветвление If ... Then ... Else

Этот оператор обычно используется, когда проверяется одно или два условия в программе. Конструкцию If ... Then ... Else ...(Если ... Тогда ... Иначе ...) можно записать в одном из двух форматов - блочном (в несколько строк) и линейном (в одной строке).

Примеры на языке VB

Линейный формат:

If *логич_выражение* **Then** *блок_команд_1* **[Else** *блок_команд_2***]**

Если *логич_выражение* принимает значение «истина», выполняется *блок_команд_1*, иначе выполняется *блок_команд_2*. **Else** можно опустить, в этом случае, , если *логич_выражение* ложно, *блок_команд_1* просто пропускается.

Пример 1:

Если значение переменной A больше нуля, то вычислить A^2, иначе - вычислить A^3

If A>0 Then S= A^2 Else S=A^3

Пример 2:

Если значения переменных A и B равны между собой , то прервать выполнение процедуры

If A=B Then Exit Sub

Примеры на языке VB

Блочный формат:

If *логич_выражение_1* **Then**
блок_команд_1
[ElseIf *логич_выражение_2* **Then**
блок_команд_2
.....
[Else
блок_команд_n
End If

Если *логич_выражение_1* принимает значение «истина», выполняется ***блок_команд_1***, иначе, если *логич_выражение_2* истинно, выполняется ***блок_команд_2*** и т.д. Если ни одно условие не удовлетворяется, то выполняется ***блок_команд_n***, следующий за **Else**.

Примеры на языке VB

Пример 1:

Если значения переменных A и B положительны, то вывести их сумму и произведение, в противном случае - вывести сообщение об ошибке.

```
If  A>0 And B>0 Then
Print A+B
Print A*B
Else
Print «ошибка»
End  If
```

Пример 2:

Если значение переменной B меньше 10, то увеличить его в 2 раза, если значение от 10 до 20, то увеличить в 3 раза, в остальных случаях - уменьшить в 10 раз.

```
If  B<10 Then
        B=B*2
ElseIf  B>=10 And B<=20  Then
B=B*3
Else
B=B/10
End  If
```

Примеры на языке VB

Оператор множественного выбора Select Case

Используется при необходимости осуществить проверку более сложных условий.

Формат команды

Select Case арифм_выражение или симв_выражение
Case условие 1
блок команд 1
Case условие 2
блок команд 2
.....
Case Else
блок команд n
End Select

В поле операндов Select Case записывается произвольное арифметическое выражение или символьное выражение, которое в процессе выполнения программы принимает то или иное числовое, логическое или символьное *значение*.

Примеры на языке VB

В поле операндов каждого оператора Case надо указать условие в одном из трех форматов:

Case константа_1, константа_2,...

Case Is знак_отношения константа

Case константа_1 **To** константа_2

Алгоритм множественного выбора заключается в следующем. Сначала вычисляется значение выражения, записанного в **Select Case**. Далее проверяется, удовлетворяет ли это значение одному из указанных в **Case** условий. Если значение удовлетворяет какому-то условию, выполняется блок команд, следующий за данным **Case**. Если ни одно условие не удовлетворяется, выполняется блок команд, следующий за **Case Else**. При выполнении того или иного блока команд управление передается команде, следующей за **End Select**.

Примеры на языке VB

Пример:

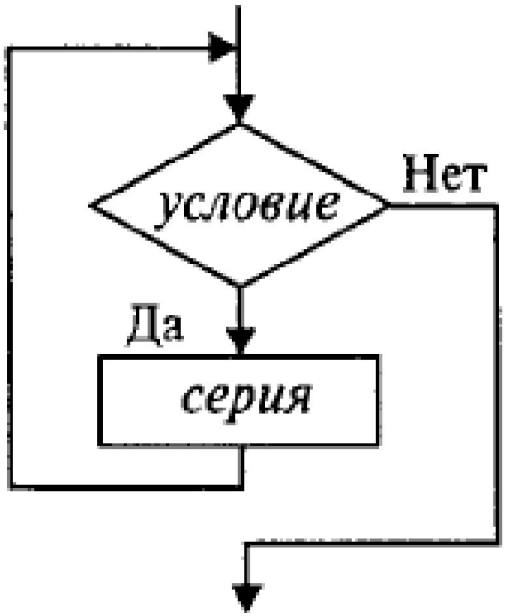
```
Input A
Select Case A
Case 1,5
Print «A равно 1 или 5»
Case Is >5
Print «A больше 5»
Case -8 TO 2.5
Print «A не меньше -8, но не больше 2.5»
Case Else
Print «Ни одно условие не выполняется»
End Select
```

Основные алгоритмические структуры

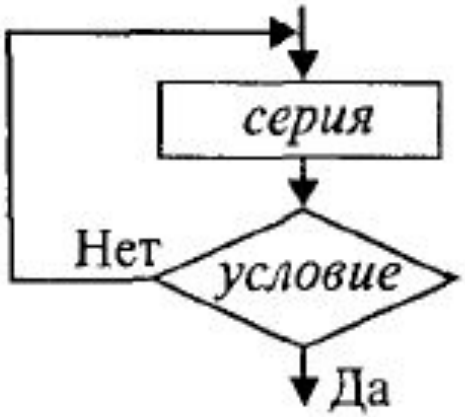
Циклическая структура. Очень часто встречаются процессы, когда решение задачи сводится к многократному вычислению по одним и тем же математическим зависимостям при различных входящих в них величинах. Многократно повторяющиеся участки этого вычислительного процесса называют **циклами**, а сам процесс - **циклическим**.

Цикл - это многократно повторяемая часть программы.

Для окончания циклического вычисления необходима проверка некоторого логического условия.



цикл с предусловием



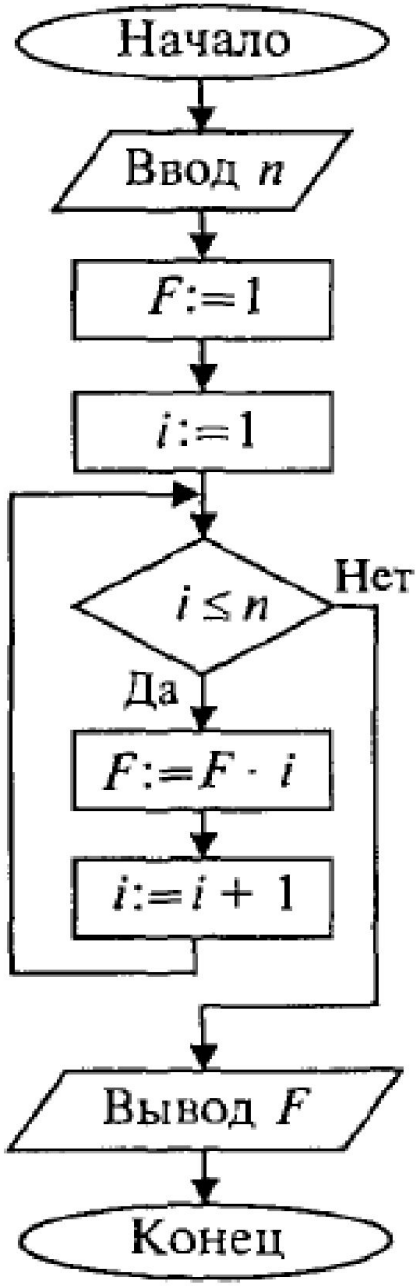
цикл с постусловием

Циклические алгоритмы

В качестве примера рассмотрим алгоритм нахождения факториала натурального числа:

$$n! = \begin{cases} 1, & \text{если } n = 0; \\ 1 \cdot 2 \ldots n, & \text{если } n \geq 1. \end{cases}$$

- Как правило в создаваемых программах присутствуют алгоритмические участки всех видов: линейные, ветвящиеся, циклические.
- Отдельные часто употребляемые алгоритмы можно выносить за пределы основной программы, такие алгоритмы называются **вспомогательными**.
- Согласно теоремам структурного программирования любой алгоритм может быть представлен при помощи операций присваивания, ветвления и цикла.



Примеры на языке VB

В языке Visual Basic предусмотрено два основных способа организации циклов:

- повторение блока команд заданное количество раз (*цикл со счетчиком*);
- циклическое повторение блока команд, пока выполняется (или не выполняется) некоторое условие.

Цикл со счетчиком For ...Next (Для ... Следующий)

Цикл For ... Next - это цикл с заранее заданным количеством повторений.

For переменная_цикла=нач_значение **To** конеч_значение **[Step** шаг**]**
блок_команд
Next переменная_цикла

В поле операндов оператора For указываются:

- числовая переменная, которая называется *счетчиком* или *переменной цикла*.
- начальное и конечное значение счетчика.

Если необходимо, укажите шаг изменения счетчика по окончании каждого цикла (по умолчанию этот шаг равен 1).
Конструкция завершается оператором Next.

Примеры на языке VB

Принцип работы оператора :

Переменной цикла присваивается начальное значение, после этого первый раз выполняется блок команд (тело цикла). Оператор NEXT увеличивает текущее значение переменной цикла на величину шага , и, если новое значение переменной цикла не превышает заданное конечное значение, в очередной раз выполняется блок команд.

Пример 1: *Вывести 10 раз текст «привет»*

```
For I=1 To 10
Print «привет»
Next I
```

Пример 2: *Вывести все четные натуральные числа от 10 до 20.*

```
For K=10 To 20 Step 2
Print K
Next K
```

Можно выйти из цикла не дожидаясь выполнения всех повторений при помощи оператора **Exit For**. Управление будет передано на оператор, стоящий после **Next**.

С помощью For ... Next можно организовывать вложенные циклы - каждый со своим For, Next и счетчиком.

Примеры на языке VB

Универсальный цикл Do ...Loop (Делать ... Цикл)

Наиболее гибкий и универсальный способ организации цикла по условию обеспечивает конструкция Do ... Loop. Конструкция имеет четыре формата:

Циклы с предусловием

1. **Do While** логич_выражение
   блок_команд
   Loop

Блок_команд выполняется до тех пор, пока значение логич_выражения **истинно.**

2. **Do Until** логич_выражение
   блок_команд
   Loop

Блок_команд выполняется до тех пор, пока значение логич_выражения **ложно**.

Примеры на языке VB

Циклы с постусловием

(при первом входе условие не проверяется, поэтому блок_команд будет выполнен хотя бы один раз).

3. **Do**
   блок_команд
   Loop While логич_выражение

Блок_команд выполняется до тех пор, пока значение логич_выражения **истинно.**

4. **Do**
   блок_команд
   Loop Until логич_выражение

Блок_команд выполняется до тех пор, пока значение логич_выражения **ложно**.

Можно выйти из цикла не дожидаясь выполнения всех повторений при помощи оператора **Exit Do**. Управление будет передано на оператор, стоящий после **Loop**.

Примеры на языке VB

Программирование графики

На форме или в графическом поле (PictureBox) можно рисовать различные графические примитивы с использованием графических методов:

Scale – позволяет задать систему координат и масштаб для формы или графического окна:

Object.Scale (X1,Y1) - (X2,Y2)

Pset – установка точки с заданными координатами и цветом:

Object.Pset (X,Y) [,Color]

Line – рисование линии, прямоугольника или закрашенного прямоугольника заданного цвета:

Object.Line (X1,Y1) - (X2,Y2) [,Color][,B][F]

Circle – рисование окружности, овала или дуги с заданными координатами центра, радиусом, цветом, начальным и конечным углом дуги и коэффициентом сжатия:

Object.Circle (X,Y),Radius [,Color, Start, End, Aspect]

Литература:

1. Каймин В.А. Информатика: учебник. – 5-е изд. – М.: ИНФРА-М, 2012. -285 с.
2. Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие / Под ред.проф. Л. Г. Гагариной. – М.: ИД «Форум»: ИНФРА-М, 2011. -416 с.
3. Браун С. Visual Basic 6: учебный курс. СПб.: Питер, 2001. – 576 с.
4. Глушаков С.В., Сурядный А.С. Программирование на Visual Basic 6.0: Учебный курс. – М.: Изд. «Фолио», 2002. – 497 с.