Понятие алгоритма. Свойства, способы описания

Лекция3

*Понятие алгоритма

Под алгоритмом понимается «**точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату**» (ГОСТ 19.781-74).

Алгоритм включает систему правил, определяющих содержание и конечную последовательность действий (шагов и операций), выполняемых над некоторыми объектами с целью переработки исходных и промежуточных данных в искомый результат. Это предписание конкретному исполнителю о том, какие действия, над какими объектами и в каком порядке следует выполнять для решения поставленной задачи.

*Способы представления алгоритмов

- словесный (средствами языка человеческого общения)
- структурно-стилизованный (языком псевдокодов)
- графический (схемами из графических блок - символов)
- программный (текстами программ).

Словесный способ записи алгоритмов представляет собой описание последовательности действий по обработке данных на естественном языке человеческого общения. Очевидные преимущества: доступность и простота использования. Недостатки: алгоритмы плохо формализуемы, громоздки и ненаглядны.

Структурно-стилизованный способ записи основан на применении естественного языка в формализованном представлении предписаний, конструкции которого близки к конструкциям структурных языков программирования.

В ***графическом*** представлении алгоритмы изображаются в виде блок-схемы, дополненной элементами словесной или математической записи. Схема алгоритма включает геометрические фигуры (блочные символы), соединенные между собой стрелками (линиями), указывающими порядок выполнения операций.

*Основные свойства алгоритма

- однозначность
- дискретность
- конечность и результативность
- рациональность
- массовость
- корректность

Однозначность. Указания, составляющие алгоритм, должны быть четкими и однозначными, не допускать произвольного или двоякого толкования.

Дискретность. Определяемый алгоритмом вычислительный процесс может быть разделен на отдельные этапы (шаги), представляющие собой упорядоченные элементарные операции. В этой упорядоченной записи выполнение действий очередного предписания допустимо лишь после исполнения предыдущего.

Вычислительный процесс после выполнения заданной алгоритмом конечной последовательности действий должен заканчиваться выдачей результатов или сообщением о невозможности решить задачу. Эти взаимосвязанные свойства алгоритма называются ***конечностью и результативностью***.

Рациональность. Алгоритм вычислительного процесса должен привести к результату не просто за конечное число шагов, но за наименьшее время при минимальном использовании ресурсов компьютера.

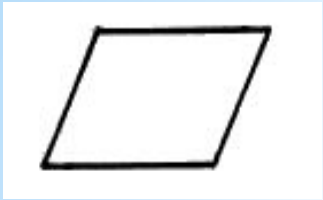
Эти требования определяются свойством алгоритма, подразумевающим многовариантность путей решения любой задачи, из которых следует выбирать самый эффективный.

Массовость. Означает возможность использования алгоритма для решения множества однотипных задач с различными исходными данными.

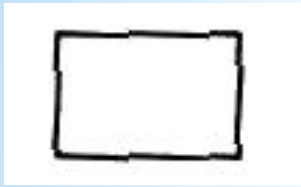
Корректность. Способность алгоритма давать правильные результаты решения задачи при различных исходных данных.

*Графический метод

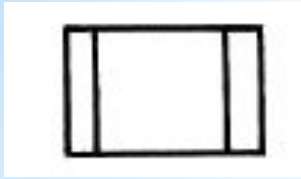
Наиболее распространенным способом представления алгоритма является *графический*. В графическом представлении алгоритмы изображаются в виде блок-схемы, дополненной элементами словесной или математической записи. Схема алгоритма включает геометрические фигуры (блочные символы), соединенные между собой стрелками (линиями), указывающими порядок выполнения операций. Блочные символы стандартизированы и различаются по типу выполняемых действий (ГОСТ 19.701-90)



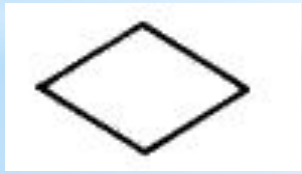
Данные. Символ отображает данные. Носитель данных не определен.(*ввод-вывод информации*)



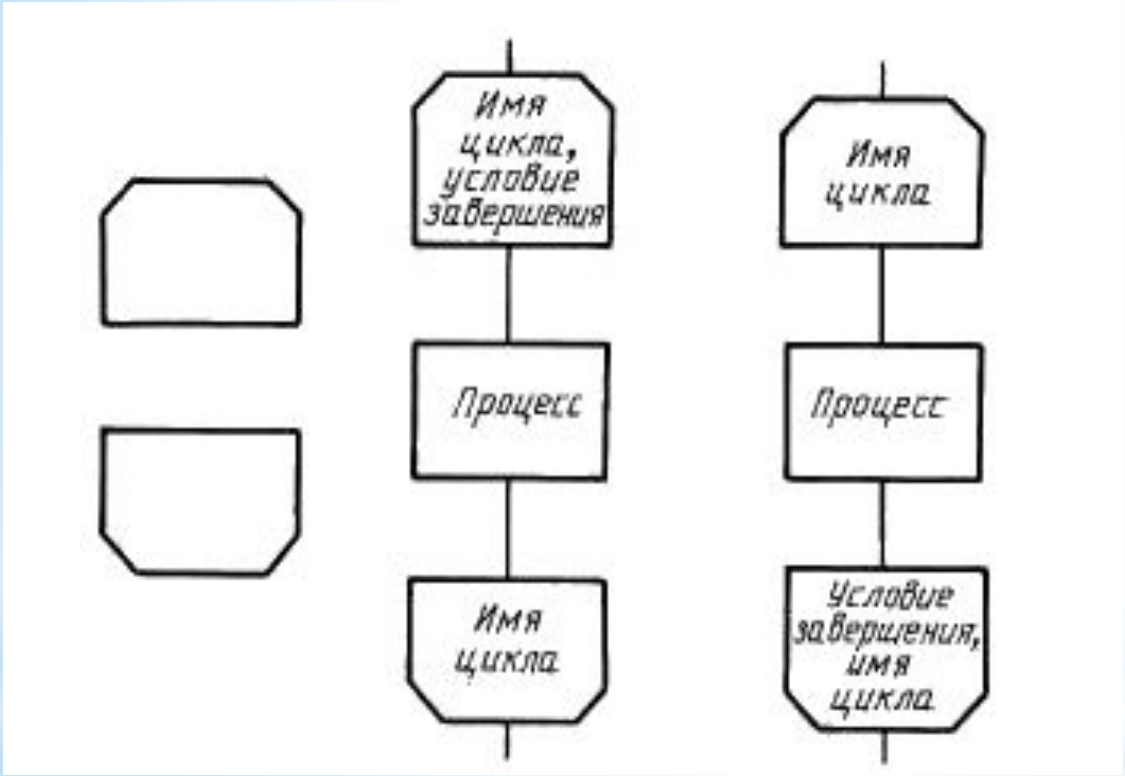
Процесс. Символ отображает обработку данных любого вида.(*оператор присваивания*)



Предопределенный процесс. Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (подпрограмме, модуле).



Решение. Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. (*условный оператор или оператор выбора*)



Граница цикла. Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения, помещаются внутри символа в начале или в конце, в зависимости от расположения операции проверяющей условие. (*операторы цикла*)

Соединитель. Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и тоже уникальное обозначение.

Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы…)

При соединении блоков следует использовать только вертикальные и горизонтальные линии потоков.

Горизонтальные потоки, имеющие направление справа налево, и вертикальные потоки, имеющие направление снизу вверх, должны быть обязательно помечены стрелками. Прочие потоки могут быть помечены или оставлены непомеченными.

Расстояние между параллельными линиями потоков должно быть не менее 3 мм , между остальными элементами схемы – не менее 5 мм .

Горизонтальный и вертикальный размеры блока должны быть кратны 5 мм (делиться на 5 нацело). Отношение горизонтального и вертикального размеров блока 1.5 является основным. При ручном выполнении блока допустимо отношение 2.

Блоки "Начало", "Конец" и "Соединитель" имеют высоту вдвое меньше основной высоты блоков.

*Базовые алгоритмические структуры: следование, ветвление, цикл

Алгоритмическая структура образующая линейную последовательность операций, которые выполняются по очереди в порядке записи называется *следованием*.

Возможность альтернативного выбора при выполнении программы предоставляют *ветвления*, при выполнении которых алгоритм может пойти по одной из двух (или более) возможных ветвей в зависимости от справедливости проверяемого условия.

Цикл представляет собой многократно повторяющуюся последовательность шагов алгоритма.

Структурное программирование

Структу́рное программи́рование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Эдсгером Дейкстрой, разработана и дополнена Николасом Виртом.

В соответствии с данной методологией:

1. Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

последовательное исполнение — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;

ветвление — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;

цикл — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

2. Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде подпрограмм.

В этом случае в тексте основной программы, вместо помещённого в подпрограмму фрагмента, вставляется инструкция **вызова подпрограммы**. При выполнении такой инструкции выполняется вызванная подпрограмма, после чего исполнение программы продолжается с инструкции, следующей за командой вызова подпрограммы.

3. Разработка программы ведётся пошагово, методом «сверху вниз».

Сначала пишется текст основной программы, в котором, вместо каждого связного логического фрагмента текста, вставляется вызов подпрограммы, которая будет выполнять этот фрагмент. Вместо настоящих, работающих подпрограмм, в программу вставляются «заглушки», которые ничего не делают. Полученная программа проверяется и отлаживается. После того, как программист убедится, что подпрограммы вызываются в правильной последовательности, подпрограммы-заглушки последовательно заменяются на реально работающие, причём разработка каждой подпрограммы ведётся тем же методом, что и основной программы.

*Достоинства структурного программирования:

- Структурное программирование позволяет значительно сократить число вариантов построения программы по одной и той же спецификации, что значительно снижает сложность программы и, что ещё важнее, облегчает понимание её другими разработчиками.
- В структурированных программах логически связанные операторы находятся визуально ближе, а слабо связанные — дальше, что позволяет обходиться без блок-схем и других графических форм изображения алгоритмов.
- Сильно упрощается процесс тестирования и отладки структурированных программ.

Домашнее задание

1.Составить опорный конспект лекции по теме «Понятие алгоритма. Свойства, способы описания» на основе презентации.

2.Программирование на языке Pascal. Рапаков Г. Г., Ржеуцкая С. Ю. СПб.: БХВ-Петербург, 2004, стр. 7-13,

Составить

- блок-схему алгоритма определения, является ли число положительным, отрицательным или нулем
- блок-схему алгоритма определения всех делителей натурального числа.