**Лекция 7**. Машины Поста. Процедурные модели вычисления

**ЦЕЛЬ**: Ознакомить с понятиями: Машины Поста. Ассоциативные исчисления в алфавите. Канонические и нормальные системы Поста. Вычислимость функции по Посту. Процедурные модели вычисления

**ВОПРОСЫ**:

1. Машины Поста.
2. Канонические и нормальные системы Поста. Вычислимость функции по Посту.
3. Процедурные модели вычисления

**ОСНОВНЫЕ ПОНЯТИЯ:** Машины Поста. Ассоциативные исчисления в алфавите. Канонические и нормальные системы Поста. Вычислимость функции по Посту. Процедурные модели вычисления.

**Машина Поста** (МП) — абстрактная вычислительная машина, предложенная [Эмилем Леоном Постом](http://ru.wikipedia.org/wiki/ÐÐ¾ÑÑ,_Ð­Ð¼Ð¸Ð) (Emil L. Post), которая отличается от [машины Тьюринга](http://ru.wikipedia.org/wiki/ÐÐ°ÑÐ¸Ð½Ð°_Ð¢ÑÑÑÐ¸Ð½Ð³Ð°) большей простотой. Обе машины «эквивалентны» и были созданы для уточнения понятия «[алгоритм](http://ru.wikipedia.org/wiki/ÐÐ)».

## Принцип работы. МП состоит из каретки (или считывающей и записывающей головки) и разбитой на секции бесконечной в обе стороны ленты (см. пример ниже). Каждая секция ленты может быть либо пустой — 0, либо помеченной меткой 1. За один шаг каретка может сдвинуться на одну позицию влево или вправо, считать, поставить или уничтожить символ в том месте, где она стоит. Работа МП определяется программой, состоящей из конечного числа строк. Всего команд шесть:

1. N.   →   J сдвиг вправо
2. N.   ←   J сдвиг влево
3. N.   1   J запись метки
4. N.   0   J удаление метки
5. N.   ? J1, J0   условный переход по метке
6. N.   Stop остановка

где N. — номер строки, J — строка на которую переходит управление далее.

Для работы машины нужно задать программу и ее начальное состояние (т. е. состояние ленты и позицию каретки). После запуска возможны варианты:

1. работа может закончиться невыполнимой командой (стирание несуществующей метки или запись в помеченное поле);
2. работа может закончиться командой Stop;
3. работа никогда не закончится.

**Функционирование машины Поста-Успенского.**Что­бы ма­ши­на на­ча­ла ра­бо­тать, не­об­хо­ди­мо за­дать:(1) про­грам­му;(2) на­ча­ль­ное со­сто­яние ма­ши­ны, т.е. ка­ким-то об­ра­зом рас­ста­вить мет­ки по сек­ци­ям лен­ты (в час­тно­сти, мож­но все сек­ции оста­вить пус­ты­ми) и по­ста­вить ка­ре­т­ку про­тив од­ной из сек­ций. *В на­ча­ль­ном со­сто­янии ма­ши­ны По­ста-Успен­ско­го* ка­ре­т­ка ста­ви­т­ся всег­да про­тив сек­ции с но­ме­ром 0. При та­ком со­гла­ше­нии на­ча­ль­ное со­сто­яние ма­ши­ны пол­нос­тью опре­де­ле­но со­сто­яни­ем лен­ты. Ра­бо­та ма­ши­ны на осно­ва­нии за­дан­ной про­грам­мы (и при за­дан­ном на­ча­ль­ном со­сто­янии) про­ис­хо­дит сле­ду­ющим об­ра­зом. Ма­ши­на при­во­дит­ся в на­ча­ль­ное со­сто­яние и при­сту­па­ет к вы­пол­не­нию пер­вой ко­ман­ды про­грам­мы (что озна­ча­ет "*вы­пол­нить ко­ман­ду*", бу­дет объ­яс­не­но ни­же). Эта ко­ман­да вы­пол­ня­ет­ся за один шаг, по­сле че­го ма­ши­на при­сту­па­ет к вы­пол­не­нию той ко­ман­ды, но­мер ко­то­рой (на­зо­вем его*a*) ра­вен от­сыл­ке (од­ной из от­сы­лок, ес­ли их две) пер­вой ко­ман­ды. Эта ко­ман­да так­же вы­пол­ня­ет­ся за один шаг, по­сле че­го на­чи­на­ет­ся вы­пол­не­ние ко­ман­ды, но­мер ко­то­рой ра­вен от­сыл­ке ко­ман­ды с но­ме­ром *a*. Каж­дая ко­ман­да вы­пол­ня­ет­ся за один шаг, а пе­ре­ход от вы­пол­не­ния од­ной ко­ман­ды к дру­гой про­ис­хо­дит по сле­ду­юще­му пра­ви­лу:

пусть на k-м ша­ге вы­пол­ня­лась ко­ман­да с но­ме­ром i, тог­да, ес­ли эта ко­ман­да име­ет един­ст­вен­ную от­сыл­ку j, то на (k+1)-м ша­ге вы­пол­ня­ет­ся ко­ман­да с но­ме­ром j; ес­ли эта ко­ман­да име­ет две от­сыл­ки j1 и j2, то на (k+1)-м ша­ге вы­пол­ня­ет­ся од­на из двух ко­манд - с но­ме­ром j1 или с но­ме­ром j2(ка­кая имен­но, бу­дет ука­за­но ни­же); ес­ли, на­ко­нец, вы­пол­ня­юща­яся на k-м ша­ге ко­ман­да во­все не име­ет от­сыл­ки, то на (k+1)-м ша­ге и на всех по­сле­ду­ющих ша­гах не вы­пол­ня­ет­ся ни­ка­кая ко­ман­да - ма­ши­на оста­нав­ли­ва­ет­ся.

Поскольку в основном алгоритмы записываются на языке, близком к одному из процедурных языков высокого уровня, то наша задача— определить «элементарные» операции такой системы записи алгоритма.

Практически значимые реализации вычислительных алгоритмов включают в себя обычно следующие программные фрагменты:

- диалоговый или файловый ввод исходных данных;

- проверка исходных данных на допустимость;

- собственно решение поставленной задачи;

- представление (вывод) полученных результатов.

Про­грам­ми­ро­ва­ние для ма­ши­ны По­ста предъ­яв­ля­ет та­кие же тре­бо­ва­ния к про­грам­ми­с­ту, как и про­грам­ми­ро­ва­ние для лю­бых дру­гих ма­шин или в иных ал­го­рит­ми­че­с­ких сис­те­мах.

**ПРОЦЕДУРНЫЕ МОДЕЛИ ВЫЧИСЛЕНИЯ.**

В рамках анализа алгоритмов по трудоемкости мы можем считать, что обслуживающие фрагменты программной реализации (ввод, проверка и вывод) являются общими или эквивалентными для разных алгоритмов решения данной задачи. Такой подход приводит к необходимости анализа только непосредственного алгоритма решения. При этом предполагается размещение исходных данных и результатов в «оперативной» памяти, включая в это понятие как собственно оперативную память, так и кэш память, регистры и буфера реального процессора. Таким образом, множество элементарных операций, учитываемых в функции трудоемкости, не включает операции ввода/вывода данных на внешние носители. Для формального процедурного языка высокого уровня такими «элементарными» операциями, коррелированными с основными операциями языков процедурного программирования, будем считать следующие:

1) простое присваивание: а <— b;

2) одномерная индексация: а[г] : (адрес(о) + г • длина элемента);

3) арифметические операции: {\*, /, -, +};

4) операции сравнения: а{ <, >, =, <, > }b;

5) логические операции: (11) {or, and, not} (12);

6) операции адресации в сложных типах данных: (namel.name2).

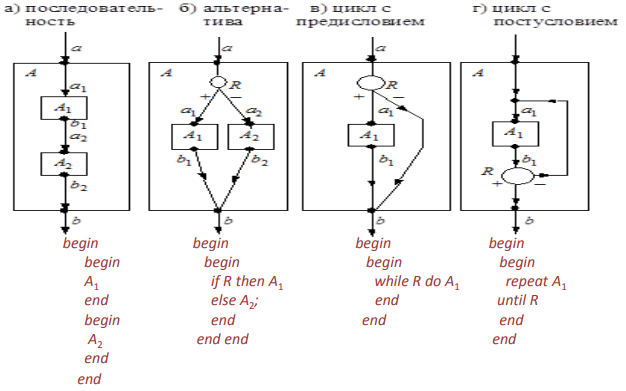
По отношению к введенному набору элементарных операций алгоритмического базиса формального процедурного языка высокого уровня необходимо сделать несколько замечаний.

- Опираясь на идеи структурного программирования, из набора элементарных операций исключается команда перехода, поскольку ее можно считать связанной с операцией сравнения в конструкции ветвления или цикла по условию. Такое исключение оправдано запретом использования оператора перехода на метку в идеологии структурного программирования.

- Операции доступа к простым именованным ячейкам памяти считаются связанными с операциями, операнды которых они хранят.

- Конструкции циклов не рассматриваются, т. к. могут быть сведены к указанному набору элементарных операций.

Структурное (структурированное) программирование - запрет GO TO. В начале 70–х годов Дейкстра предложил принцип последовательного уточнения логической структуры алгоритмов. Внутреннее содержание каждого программного блока в БСА уточняется одним из четырёх способов, показанных на рисунке.

****

Алгоритмические языки делят на две группы. Первую группу образуют языки, которые называются языки операторного, или процедурного типа. Элементарными единицами программы являются здесь операторы, т.е. приказы, выполнение которых сводится к четко определенному изменению четко определенной части памяти машины. Типичным представителем этой группы является язык машины Поста - Фортран, С, JAVA…..

Языки второй группы называются языками сентенциального, или декларативного типа (sentence - высказывание, предложение). Программа на таком языке представляется в виде набора предложений (соотношений, правил, формул), которые машина, понимающая данный язык, умеет каким-то образом применять к обрабатываемой информации. Простейшим примером является язык нормальных алгоритмов Маркова – РЕФАЛ, Snowball, Lisp, Prolog, Python…….

Процедурное (императивное, алгоритмическое) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 1940-х годах. Теоретической моделью процедурного программирования служит алгоритмическая система под названием Машина Тьюринга.

Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. Основным является оператор присваивания, служащий для изменения содержимого областей памяти. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Процедурный язык программирования предоставляет возможность программисту определять каждый шаг в процессе решения задачи. Особенность таких языков программирования состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом. Используя процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.

Условие задачи: Сократ - человек. Все люди смертны.

Найти: Смертен ли Сократ?

Запишем условие в терминах языка пролога (со знака % начинаются комментарии):

% Сократ - человек human(sokrat).

% Платон - тоже человек human(platon).

% Чтобы некто был смертным, он должен быть человеком mortal(Someone) :- human(Someone).

теперь спросим пролог систему, смертен ли Сократ:

?- mortal(sokrat).

Yes % да

?- mortal(stranger).

No % нет, поскольку пролог система не знает ничего о stranger и потому, не может ответить на вопрос % Мы можем спросить какие смертные существа известны системе:

?- mortal(Who).

Who = platon ;

Who = sokrat

На “своих” задачах декларативные (функциональные, динамические) языки позволяют получать выигрыш в быстродействии, примерно в пять раз. При распараллеливании дают выигрыш на порядок, из-за своей особой семантики.

Основные приложения –

- искусственный интеллект,

- машинное обучение,

- поиск и распознавание, военные приложения

……….

Впрочем, в повседневной работе Python (Prolog,Refal и т.д.) опытные программисты ϑ используют в дополнение к JAVA или С.

Динамические (функциональные) языки позволяют более быстро распарсить логи, сгенерировать тестовые данные или sqlскрипт для тестового заполнения базы, админские скрипты для работы с файлами на них - одно удовольствие.

**ВЫВОДЫ:**

Машина Поста (МП) — абстрактная вычислительная машина, которая отличается от [машины Тьюринга](http://ru.wikipedia.org/wiki/ÐÐ°ÑÐ¸Ð½Ð°_Ð¢ÑÑÑÐ¸Ð½Ð³Ð°) большей простотой. Обе машины «эквивалентны» и были созданы для уточнения понятия «[алгоритм](http://ru.wikipedia.org/wiki/ÐÐ)».Поскольку в основном алгоритмы записываются на языке, близком к одному из процедурных языков высокого уровня (см., например, основной текст книги), то наша задача— определить «элементарные» операции такой системы записи алгоритма. В ма­ши­не По­ста вна­ча­ле в за­по­ми­на­ющее устро­йство за­кла­ды­ва­ет­ся ис­хо­дное дан­ное, по­сле че­го ра­бо­та про­ис­хо­дит по про­грам­ме, не на­хо­дя­щей­ся в за­по­ми­на­ющем устро­йстве. В ком­пь­ю­те­ре вна­ча­ле в за­по­ми­на­ющее устро­йство за­кла­ды­ва­ет­ся на­ча­ль­ная ин­фор­ма­ция, со­сто­ящая из про­грам­мы ре­ше­ния дан­ной за­да­чи и ис­хо­дно­го дан­но­го, по­сле че­го ра­бо­та про­ис­хо­дит по Уни­вер­са­ль­ной Про­грам­ме, во­пло­щён­ной в кон­стру­к­ции ма­ши­ны. Та­ким об­ра­зом, про­грам­ми­ро­ва­ние для ма­ши­ны По­ста предъ­яв­ля­ет та­кие же тре­бо­ва­ния к про­грам­ми­с­ту, как и про­грам­ми­ро­ва­ние для лю­бых дру­гих ма­шин или в иных ал­го­рит­ми­че­с­ких сис­те­мах.