Лекция 6. Динамические структуры. Стеки, очереди, деки (продолжение)

Даны пример применения стека.

Обратная польская нотация

(RPN - Reverse Polish Notation) - постфиксная нотация, в кот. аргументы или параметры команды д.б. записаны перед самой командой.

(Обратная польская нотация разработана австралийским ученым Чарльзом Хэмблином в середине 50-х годов прошлого столетия на основе польской нотации, которая была предложена в 1920 году польским математиком Яном Лукасевичем.)

Эта нотация лежит в основе организации вычислений для арифметических выражений. Известный ученый Эдсгер Дейкстра предложил алгоритм для перевода выражений из инфиксной в постфиксную форму.

Пример: `2 + 3`

- `2 3+` - обратная польская запись
- `+ 2 3` - префиксная польская
- `2 + 3` - инфиксная

Пример использования стека

Использование: посимвольный разбор выражения в строке в цикле:

(если текущий символ - знак, то извлекаем 2 эл-та, производим действия, рез-т - в стек)

```
For 1<- начало To длина(строки) Do
    Case строка[1] Of
        Цифра: Push(cTpoKa[1],PS)
    Знак оп-ции: Операнд1 <- Pop(PS)
                 Операнд2 <- Pop(PS)
                 Case строка[1] Of
                    Опер.1
                    Опер.2 Push(рез-т оп-ции,PS)
```

Арифметическое выражение, например: `(20+5)/(0.1+0.4*6)-5.3*4` . Может быть

записано в указанной форме, называемой **инфиксной**, где знаки операций стоят между операндами, порядок действий определяется расстановкой скобок и приоритетом операций.

Постфиксная (или обратная польская) форма записи не содержит скобок, а знаки операций следуют после соответствующих операндов. Для приведённого примера постфиксная форма будет иметь вид:
`20 5 + 0.1 0.4 6 * + / 5.3 4 *` -

Применяя к водимой строке вышеописаный алгоритм

```
stak: Puch(20), Puch(5)
+ : Pop(5), Pop(20) = Puch(25)
stak: 25, Puch(0.1), Puch(0.4), Puch(6)
*: Pop(6), Pop(0.4) = Puch(2.4)
stak: 25, 0.1, 2.4
+ : Pop(2.4), Pop(0.1) = Puch(2.5)
stak: 25, 2.5
/ : Pop(2.5), Pop(25) = Puch(10)
stak: 10, Puch(5.3) ,Puch(4)
* : Pop(4), Pop(5.3) = Puch(21.2)
stak: 10,21.2
- : Pop(21.2), Pop(1) = Puch(-11.2)
```

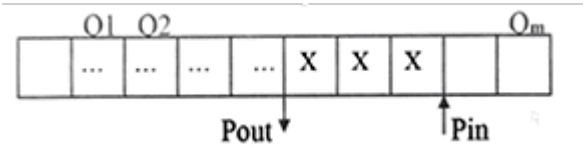
> Пример для понимания, не из лекции
>
> 
>

Очереди

Очередь - это последовательный список переменной длины, включение в кот. идет с одной стороны (с хвоста), а исключение - с другой стороны (с головы). Принцип работы очереди: первым пришел - первым вышел, то есть, First In - First Out (FIFO).

Для хвоста и головы очереди используют два указателя `PIN` и `POUT` (хвост и голова). То есть, исключается элемент с адресом `Pou` t и включается элемент по адресу `Pin`. При моделировании простейшей линейной очереди на основе вектора выделяется последовательность `m` мест длиной `L` под каждый элемент очереди.

Очередь на основе вектора:



В каждый момент времени занята лишь часть выделенной памяти. Колиество элементов в очереди:

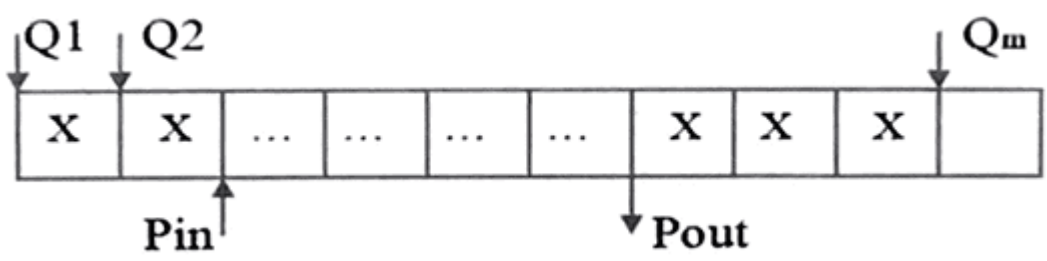
$$K = \frac{Pin - Pout}{L}$$

При включении элемента он располагается по адресу `Pin`, а сам указатель `Pin` перемещается на «пустое» место. Очистка очереди: `Pin = Q1`, `Pout = Q1`.

Переполнение очереди наступает при `Pin = Qm + 1`, независимо от состояния `Pout`.

Для исключения переполнения производят сдвиг эл-тов к голове при каждом исключении из очереди, тогда переполнение будет только, если все m мест заняты. Но на эти последовательные сдвиги тратится время.

кольцевая очередь



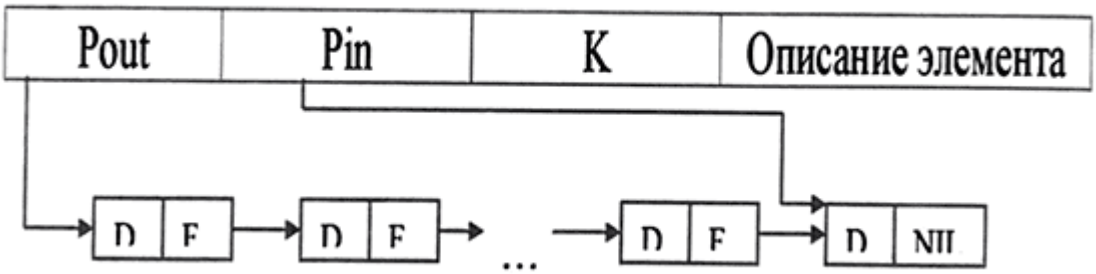
Здесь, если `Pin = Qm + 1`, то производят коррекцию: `Pin = Q1`, если `Pout > Q1`, Т. е., здесь может быть `Pin < Pout`, `Pin > Pout` и

`Pin = Pout` - пустая (но и заполненная).

В кольцевой структуре нет необходимости сдвигать элементы, но сложнее алгоритмы включения и исключения элементов, а радикально проблема переполнения все равно не решается, т. к. при заполнении очереди следующий элемент будет затирать первый, то есть необходимо проверять заполнение очереди.

Очередь на основе списка

Большинство перечисленных проблем устраняется при реализации очереди на основе односвязных линейных списков.



Исходное состояние очереди:

`Pout = Nil`, `Pin = Nil`, `K = 0`.

Операции с очередью

- включение (вставка) элемента в очередь;
- исключение элемента из очереди;
- очистка очереди (с освобождением памяти (список) и без освобождения (массив))

Вставка элемента с адресом N в очередь:

```
If K = 0 Then Pout <- N
         Else F (Pin) <- N
Pin <- N         {теперь он последний}
D(Pin) <- Data
F(Pin) <- Nil  {следующего - нет}
K + 1
```

Исключение элемента из очереди:

```
If K <> 0  {если очередь не пуста}
    Then
        P<- F (Pout)
```

```
        F (Pout) <- E {пополнение списка свободных элементов}
        E <- Pout
        Pout <- P
        K = K-1
        If K = 0 Then Pin <- Nil
    Else  {отказ от исключения - очередь пуста}
```

Очистка очереди:

```
F(Pin) <- E {пополнение списка свободных элементов}
E <- Pout
Pin <- Nil
Pout <- Nil
K <- 0
```

Очереди с приоритетами

Очередь, когда порядок выборки элементов определяется приоритетами элементов очереди. Приоритет м. б. представлен числовым значением, кот. вычисляется либо на основании значений к.-л. полей элемента, либо на основании внешних факторов. Так, и FIFO, и LIFO могут трактоваться как приоритетные очереди, в кот. приоритет элемента зависит от времени его включения в очередь. При выборке элемента всякий раз выбирается элемент с наибольшим приоритетом.

Очереди с приоритетами м. б. реализованы на линейных списковых структурах - в смежном или связном представлении. Возможны очереди с приоритетным включением - в кот. послед-ность элементов очереди все время поддерживается упорядоченной, т.е. место включения эл-та определяется его приоритетом, а при исключении всегда выбирается элемент из начала. Возможны очереди с приоритетным исключением - новый элемент включается всегда в конец очереди, а при исключении в очереди ищется (этот поиск может быть только линейным) элемент с максимальным приоритетом и после выборки удаляется из последовательности. И в том, и в другом варианте требуется поиск, а если очередь размещается в статической памяти - еще и перемещение элементов.

Очереди в вычислительных системах

Кольцевая очередь в ВС - буфер клавиатуры в BIOS IBM PC. При нажатии на любую клавишу генерируется прерывание 9. Обработчик этого прерывания читает код нажатой клавиши и помещает его в буфер клавиатуры - в конец очереди. Коды нажатых клавиш могут накапливаться в буфере клавиатуры, прежде чем они будут прочитаны программой. Программа при вводе данные с клавиатуры обращается к прерыванию 16H. Обработчик этого прерывания выбирает код клавиши из буфера - из

начала очереди - и передает в программу

Очередь - одно из ключевых понятий в многозадачных ОС (Windows NT, Unix, OS/2, ЕС и др.). Ресурсы ВС (процессор, ОП, ВУ и т.п.) используются всеми задачами, одновременно выполняемыми в среде такой ОС. Многие виды ресурсов не допускают одновременного использования разными задачами, такие ресурсы предоставляются задачам поочередно. Т. о., задачи, претендующие на использование того или иного ресурса, выстраиваются в очередь к этому ресурсу. Эти очереди обычно приоритетные, но, часто применяются и FIFO- очереди, т. к. это единственная логич. организация очереди, кот. гарантированно не допускает постоянного вытеснения задачи более приоритетными. Стеки обычно используются ОС для учета свободных ресурсов.

Распределение ресурсов ЭВМ между процессами

Процесс - это программа в момент ее выполнения.

После запуска программы создается соответствующий ей процесс, которому выделяются ресурсы ЭВМ. Каждый процесс получает адресное пространство в ОЗУ, содержащее стек, регистры, счетчик команд и др. необх. элементы. Также ресурсами являются время процессора и доступ к устройствам ввода-вывода.

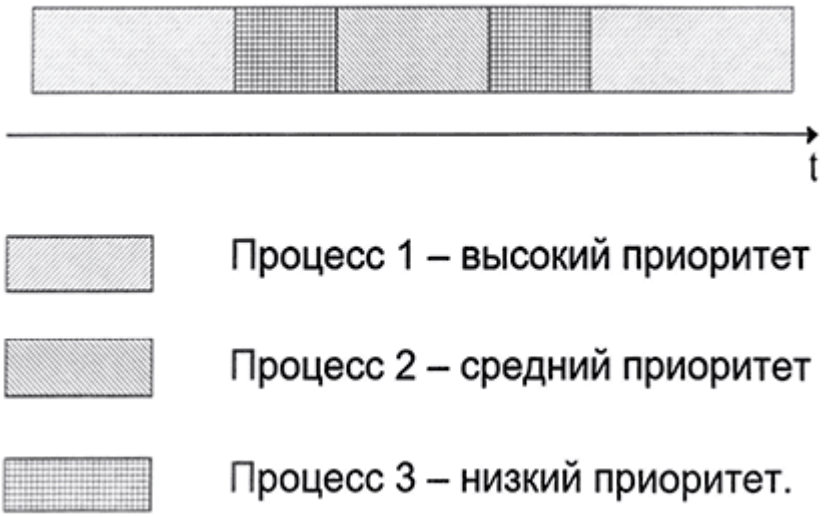
Состояния процесса

- **создание** - подготовка условий для исполнения процессором;
- **выполнение** - непосредственное исполнение процессором;
- **ожидание** по причине занятости к.-л. требуемого ресурса;
- **готовность** - процесс не исполняется, но всё необходимое для его выполнения, кроме времени процессора, предоставлено;
- **завершение** - нормальное или аварийное окончание работы процесса, после кот. время процессора и другие ресурсы ему не предоставляются.

Распределение ресурсов между процессами

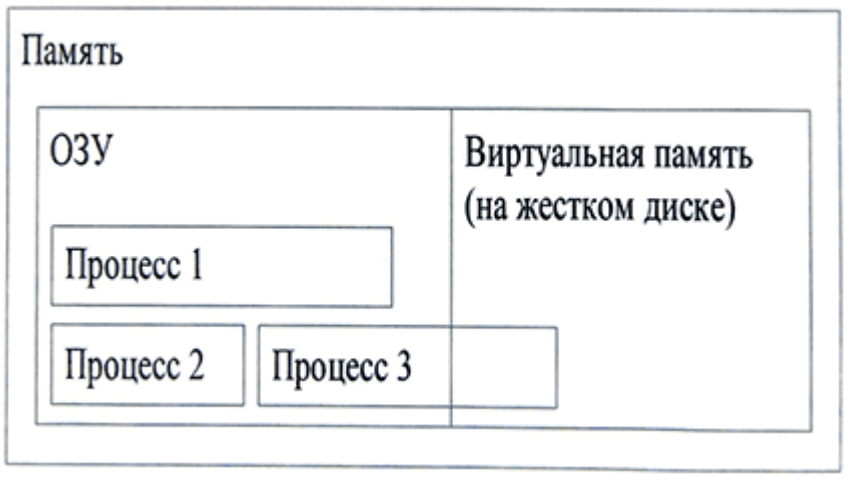
Панировщик, являющийся частью ОС, распределяет ресурсы ЭВМ между процессами. Т. о., процессы конкурируют за ресурсы. Каждый процесс имеет приоритет, в соответствии с которым он получает ресурсы ЭВМ. Наибольший приоритет имеют компоненты ОС, наименьший - программы пользователя. Приоритет процесса зависит также от частоты запроса процессом ресурсов. Чем более требователен процесс к ресурсам, тем он имеет более высокий приоритет.

Пример распределения времени процессора между процессами



- Процесс 1 - высокий приоритет
- Процесс 2 - средний приоритет
- Процесс 3 - низкий приоритет.

Пример распределения ОП между процессами



Потоки

В рамках одного процесса могут создаваться потоки, кот. сообща используют ресурсы, выделяемые для процесса, прежде всего, объем ОЗУ. По существу, потоки вып-тся в рамках одного процесса точно так же, как процессы вып-тся на одном компьютере. В каждый отдельный момент вып-тся один процесс и один поток, только переключение между ними осуществляется очень быстро

Осн. причиной появления потоков явл-тся возможность разделения функций процесса между потоками и выполнение их параллельно

Кр. того, создание и удаление потоков осущ-тся намного быстрее, чем созд. и удал, процессов, что ускоряет работу процесса в целом.

В современных ОП одним из средств взаимодействия между `||`-льно выполняемыми задачами являются очереди сообщений (почтовые ящики). Каждая задача имеет свою очередь - почтовый ящик, и все сообщения, отправляемые ей от других задач, попадают в эту очередь. Задача-владелец очереди выбирает из нее сообщения, причем может управлять порядком выборки - FIFO, LIFO или по приоритету.

В соответствии с основными принципами выделяются пять базовых элементов компьютера:

- арифметико- логическое устройство;
- устройство управления;
- запоминающее устройство;
- система ввода информации;
- система вывода информации.

ЦУУ - центральное устройство управления управляет аппаратными и программными ресурсами

- Производит чтение команд из основной памяти
- определяет адреса операндов команд, тип операции S передаёт сигнал в ОП и АЛУ.

АЛУ- арифметико-логическое устройство

- Выполняет арифметические и логические операции над данными
- вырабатывает различные условия, влияющие на ход вычислительного процесса.

ЦУУ + АЛУ=Процессор

Процессор + Основная Память = Центральные устройства (ядро) ЭВМ

Остальные устройства являются внешними устройствами ЭВМ.

Взаимодействие устройств

Интерфейс (сопряжение) - совокупность линий связи между устройствами, а также вид и порядок сигналов, проходящих по этим линиям.

Типы взаимодействия:

- множественный интерфейс - каждое устройство компьютера соединено отдельными линиями связи с другими устройствами;
- единый интерфейс (общая шина) - в этом случае на одну линию связи (шину) параллельно подключены все устройства компьютера. Их взаимодействие происходит в режиме разделённого по времени интерфейса (по очереди).

Шина - линии связи + устройства синхронизации и усиления сигналов.

Важная характеристика шины - пропускная способность. Зависит она от разрядности шины и от тактовой частоты компьютера.

Разрядность (количество проводов шины) определяет количество бит информации, обрабатываемой одновременно.

Тактовая частота задает скорость выполнения операций.

Существуют шины трёх типов:

- Шины данных;
- Шины адресов;
- Шины команд.

Микропроцессор (CPU - Central Processing Unit)

Управляет работой всех компонент ЭВМ и выполняет операции над информацией. Операции производятся в памяти МП - регистрах.

Основные функции МП:

- выполнение команд программы, расположенной в ОЗУ; команда состоит из кодами операндов, над которыми эта команда осуществляется;
- управление пересылкой информации между микропроцессорной памятью, ОЗУ и периферийными устройствами;
- обработка прерываний;
- управление компонентами ЭВМ.

Схема микропроцессора

Назначение блоков МП

Устройство управления (УУ) выполняет команды, поступающие в МП в следующей последовательности:

1. выборка из регистра-счетчика адреса ячейки ОЗУ, где хранится очередная команда программы;
2. выборка из ячеек ОЗУ кода очередной команды и приема считанной команды в регистр команд;
3. расшифровка кода команды дешифратором команды
4. формирование полных адресов операндов;
5. выборка операндов из ОЗУ или МПП и выполнение заданной команды обработки этих операндов;
6. запись результатов команды в память;
7. формирование адреса следующей команды программы.

Работа центральных устройств ЭВМ

В соответствии с принципом фон Неймана, компьютер работает под управлением программы, загруженной в основную память. **Программа** - совокупность команд, которые выполняются в определённой последовательности.

Типовые команды: арифметическое действие, запись, считывание и пересылка данных.

Пример: `A = B + C`

Последовательность двоичных сигналов:

010	1000	1001	0110
Код операции	Адрес B	Адрес C	Адрес A

(используем 1 - для сигнала высокого уровня, 0 - для сигнала низкого уровня).

Действия выполняются параллельно: один блок выбирает команду, второй дешифрует, третий выполняет и т. д. -> конвейер команд.

Команды, поступающие в УУ, временно хранятся в кэш-памяти первого уровня, освобождая шину для выполнения других операции. Размер кэш-памяти первого уровня 8-32 Кбайт на каждое ядро.
Арифметико-логическое устройство (АЛУ) выполняет все арифметические и логические операции над целыми двоичными числами и символьной информацией.

Устройство синхронизации (УС) определяет дискретные интервалы времени - такты работы МП между выборками очередной команды. Частота, с которой осуществляется выборка команд, называется тактовой частотой.

Интерфейс МП (ИМП) предназначен для связи и согласования МП с системной шиной ЭВМ. Принятые команды и данные временно помещаются в кэш-память второго уровня. Размер кэш-памяти второго уровня - 256 Кбайт на ядро Микропроцессорная память (МПП) включает 14 основных двухбайтовых запоминающих регистров и множество (до 256) дополнительных регистров. Регистры - это быстродействующие ячейки памяти различного размера.