Лекция 7. Списки.

Системная шина

служит для обмена командами и данными между компонентами ЭВМ, расп-ми на мат. плате. ПУ подкл-тся к шине через контроллеры (открытая архитектура). Передача инф-ции по сист. шине осущ-тся по тактам.
Сист. шина включает в себя:

- кодовую шину данных для параллельной (//-льной) передачи всех разрядов числового кода (машинного слова) операнда из ОЗУ в МПП и обратно; (64 разряда);
- кодовую шину адреса для //-льной передачи всех разрядов адреса ячейки ОЗУ; (32 разр.);
- кодовую шину инструкций (команд и упр-щих сигналов, импульсов) во все блоки ЭВМ; (32 разр.);
- шину питания для подключения блоков ЭВМ к системе энергопитания.

Сист. шина обеспечивает 3 направления передачи инф-ции:

1. между МП и ОЗУ;
2. между МП и контроллерами устройств;
3. между ОЗУ и внешними устройствами (ВЗУ и ПУ), в режиме прямого доступа к памяти (DMA - Direct Memory Access).

Все уст-тва подключаются к сист. шине через контроллеры - уст-тва, обесп-щие взаимодействие ВУ и сист. шины.
Для освобождения МП от управления обменом информацией между ОЗУ и ВУ, предусмотрен режим прямого доступа к памяти (DMA)

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).

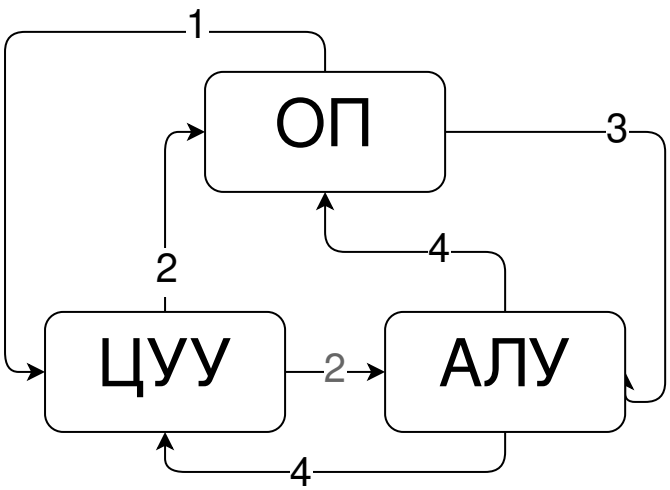
Последовательность действий:

1. ЦУУ считывает команду из ОП (в которой^записаны исходные данные и программа);
2. ЦУУ передаёт сигнал

- в ОП об адресах операндов (B и C) и результата (A)
- в АЛУ о коде операции (сложение);

3. Из ОП в АЛУ передаются значения операндов B и C;
4. АЛУ

- вычисляет сумму
- передаёт её значение в ОП
- передаёт сигнал в ЦУУ о выполнении команды, на основании которого происходит считывание следующей команды



Классификация видов компьютерной памяти

Постоянное и оперативное ЗУ

ЗУ в ЭВМ, состоят из посл-ти ячеек, каждая из кот. содержит значение одного байта и имеет собственный номер (адрес), по кот. происходит обращение к ее содержимому. Все данные в ЭВМ хранятся в двоичном виде (0,1).
ЗУ характеризуются 2-мя параметрами:

- объем памяти - размер в байтах, доступных для хранения информации;
- время доступа к ячейкам памяти - средний временной интервал, в течение кот. находится требуемая ячейка памяти и из нее извлекаются данные.

ОЗУ (RAM - Random Access Memory) предназначено для оперативной записи, хранения и чтения инф-ции (программ и данных), т.е. в процессе, выполняемом ЭВМ в текущий период времени. ОЗУ - энергозависимо. (В ЭВМ на базе процессоров Intel Pentium 32-разрядная адресация, т.е. число адресов = 232, т.е. адресное пространство составляет 4,3 Гбайт), (время доступа 40-80 нсек)

ПЗУ (ROM - Read Only Memory) хранит неизменяемую (постоянную) информацию: программы, выполняемые во время загрузки системы, и постоянные параметры ЭВМ. МП обращается по спец, стартовому адресу из ПЗУ, за своей первой командой. Осн. назначение программ из ПЗУ: проверить состав и работоспособность системы, обеспечить взаим-вие с клавиатурой, монитором, жестким диском. Объем ПЗУ 128-256 Кбайт, время доступа 0,035- 0,1 мкс. Т. к. объем ПЗУ небольшой, но время доступа больше, чем у ОЗУ, при запуске все содержимое ПЗУ считывается в спец, выделенную область ОЗУ.

Материнская плата (шина)

На системной (материнской) плате размещаются: микропроцессор, ОЗУ, ПЗУ, контроллеры и прочие устройства.

ЛОГИЧЕСКАЯ СТРУКТУРА ОПЕРАТИВНОМ ПАМЯТИ

Оперативная память представляет собок множество ячеек.
Каждая ячейка имеет свой уникальный ад| Нумерация ячеек начинается с нуля.
Каждая ячейка памяти имеет объем 1 бай
Максимальный объем адресуемой памя равен произведению количества ячеек N на 1 байт.
Для процессоров Pentium 4 (разрядность шины адреса = 36 бит) максимальный объем адресуемой памяти равен:

$$N \times 1(\text{байт}) = 2^l \times 1(\text{байт}) = 2^{36} \times 1(\text{байт}) = 68719476736(\text{байт}) =$$

$$= 67108864(\text{Кбайт}) - 65536(\text{Мбайт}) - 64(\text{Гбайт})$$

Объем памяти	Ячейки	Десятичный адрес ячейки	Шестнадцатеричный адрес ячейки
64 Гбайт	10101010	68 719 476 735	FFFFFFFFF
...	...	...	...
4 Гбайт	10101010	4 294 967 295	FFFFFFFF
...	...	...	...

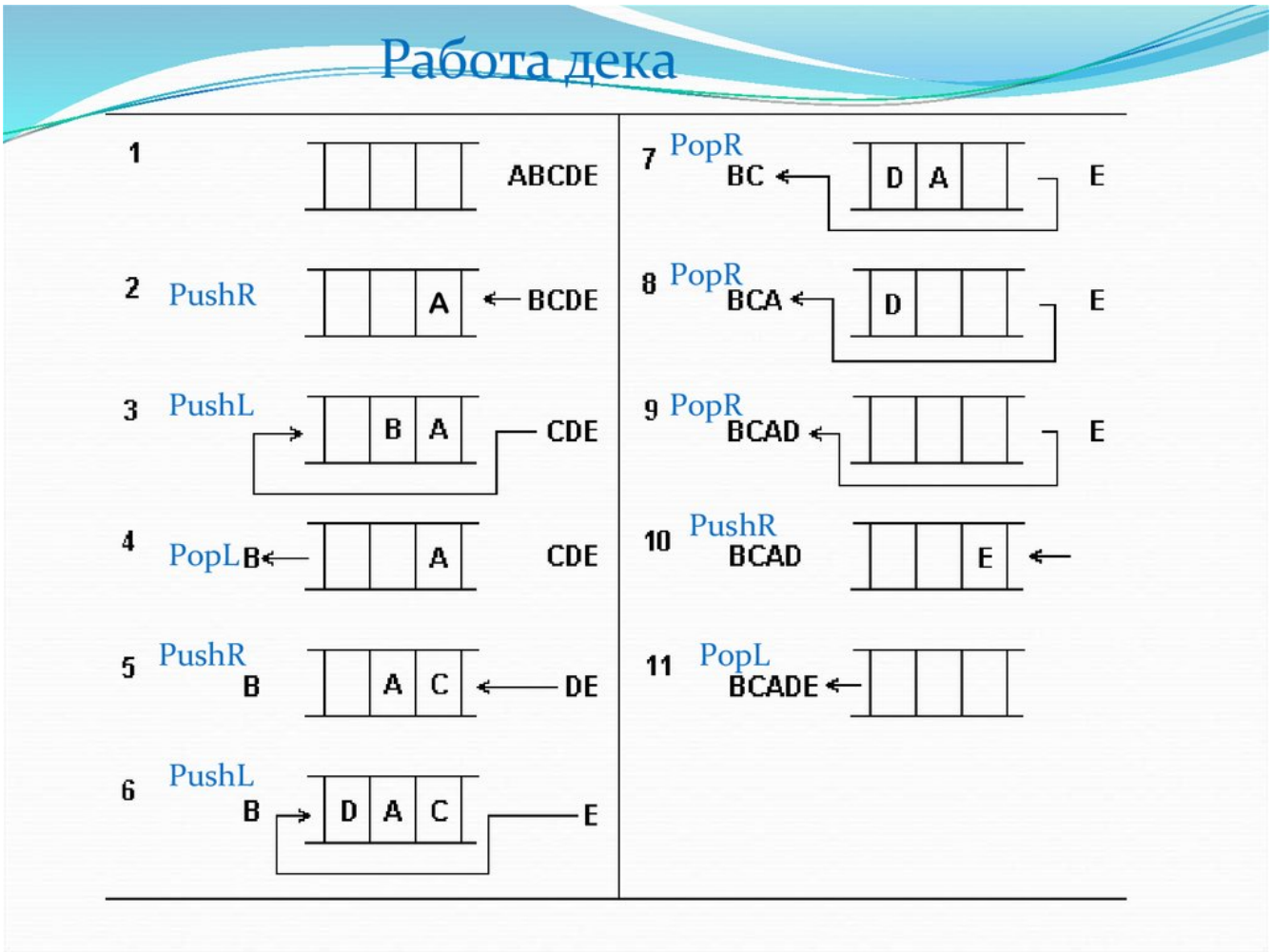
Деки

Дек (от англ, deque double ended *queen*, т. е., очередь с двумя концами) - это последовательный список, в кот. включение и исключение элементов возможны с любого конца.

Логическая и физическая структуры дека соответствуют структуре обычной очереди, но применительно к деку говорят вместо хвоста и головы о левом (l) и правом (R) концах дека.

Операции включения (Push) и исключения (Pop) элементов представляют собой обобщения аналогичных операций в очереди, при этом добавляется параметр,

указывающий, с какого конца (левого или правого) должна выполняться соответствующая операция.



Включение элемента с адресом N в дек справа:

```
If K = 0 Then PL <- N Else F(PR) <- N
PR <- N {теперь он справа}
D(PR) <- Data
F(PR) <- Nil {и он последний}
K <- K + 1
```

Включение элемента с адресом N в дек слева:

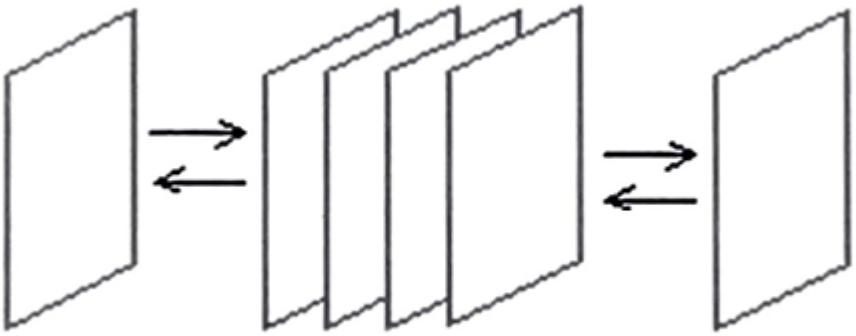
```
If K = 0 Then PR <- N Else F (N) <- PL
PL <- N  {теперь он слева}
D(PL) <- Data
K <- K + 1
```

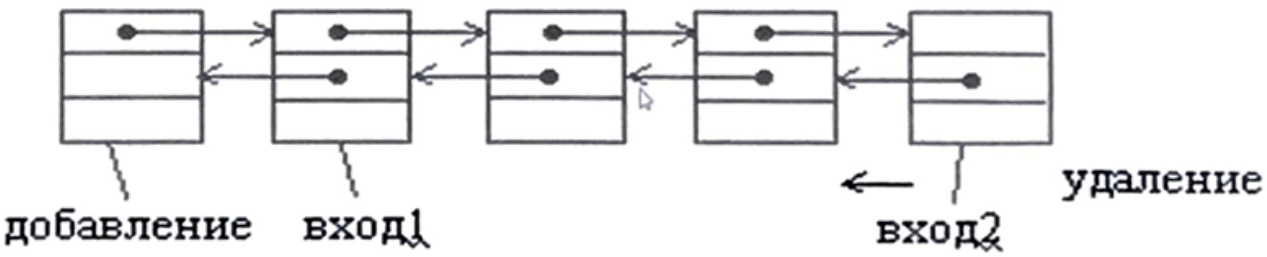
Исключение элемента из дека слева:

```
If K <> 0
Then
  P <- F(PL)
  F(PL) <- E {пополнение списка свободных элементов}
  E <- PL
  PL <- P
  K = K - 1
Else  {отказ от исключения - дек пуст}
```

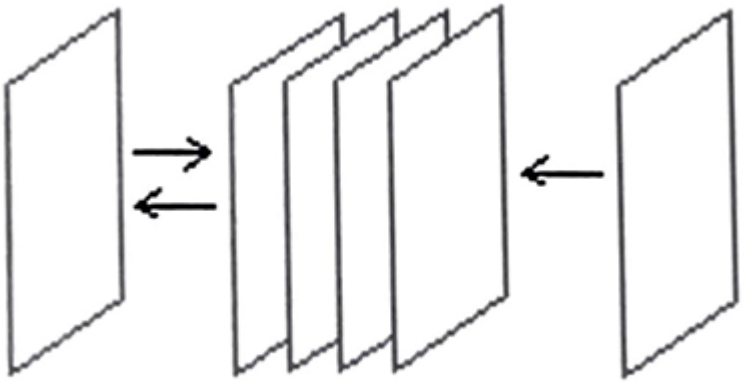
Исключение элемента из дека справа:
Последовательный проход по деку слева
При реализации дека двусвязным списком этого можно избежать:

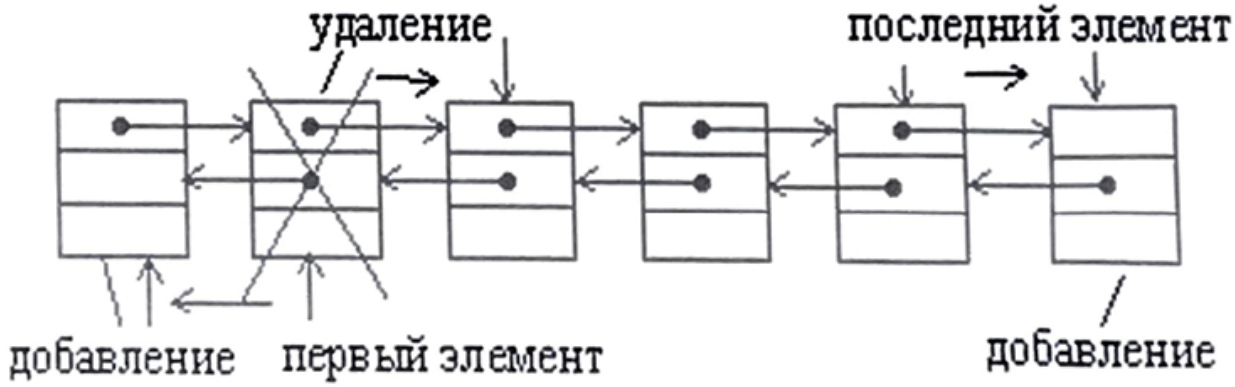
Организация дека в виде двусвязного ациклического списка





Организация дека с ограниченным выходом на основе двусвязного линейного списка





Пример системной поддержки - организация буфера ввода в языке REXX (REstructured extended eXecutor) — интерпретируемый язык программирования, (IBM).

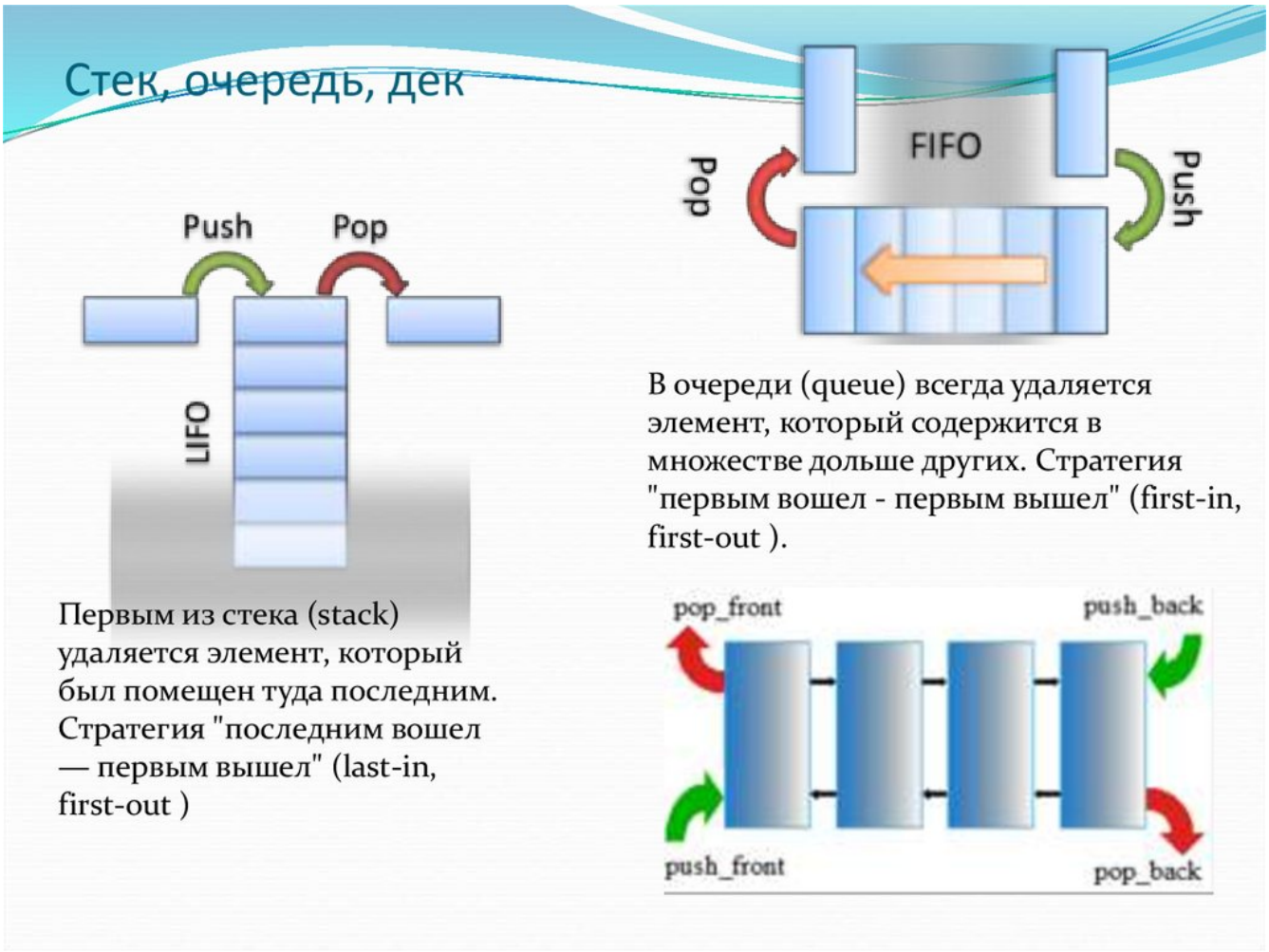
В обычном режиме буфер ввода связанно клавиатурой и работает как FIFO-очередь. Но, в REXX имеется возможность назначить в качестве буфера ввода программный буфер и направить в него вывод программ и системных утилит. В распоряжении программиста есть операции:

- QUEUE - запись строки в конец буфера
- PULL - выборка строки из начала буфера.

Дополнительная операция PUSH - запись строки в начало буфера - превращает буфер в дек с ограниченным выходом. Такая структура буфера ввода позволяет программировать на REXX весьма гибкую конвейерную обработку с направлением выхода одной программы на вход другой и модификацией перенаправляемого потока.

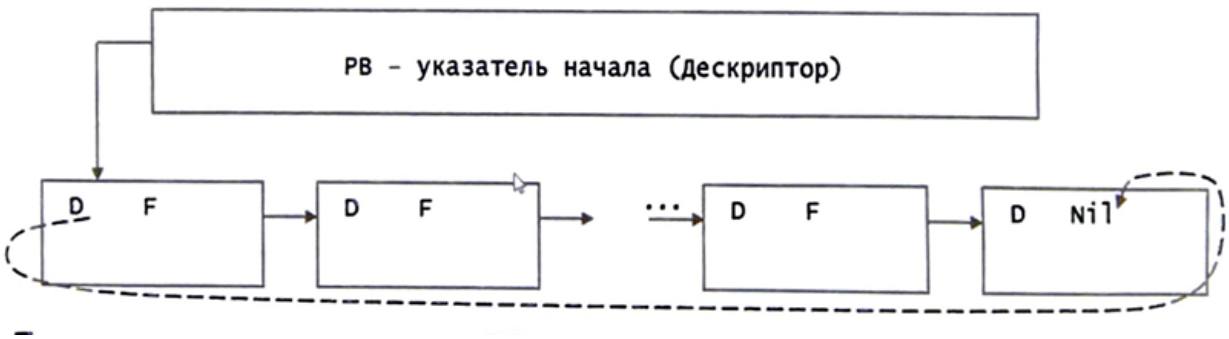
В очереди (queue) всегда удаляется элемент, который содержится в множестве дольше других. Стратегия "первым вошел - первым вышел" (first-in, first-out).

Первым из стека (stack) удаляется элемент, который был помещен туда последним. Стратегия "последним вошел — первым вышел" (last-in, first-out)



Односвязный линейный список

- это совокупность элементов, содержащих два поля:
  - поле D - запись с данными поле F - указатель «вперед», т. е., адрес следующего элемента, (адрес последнего элемента - нулевой).
  - PB - указатель начала (Дескриптор)



Дескриптор: имя списка PB, количество элементов, описание структуры элементов и т.д.

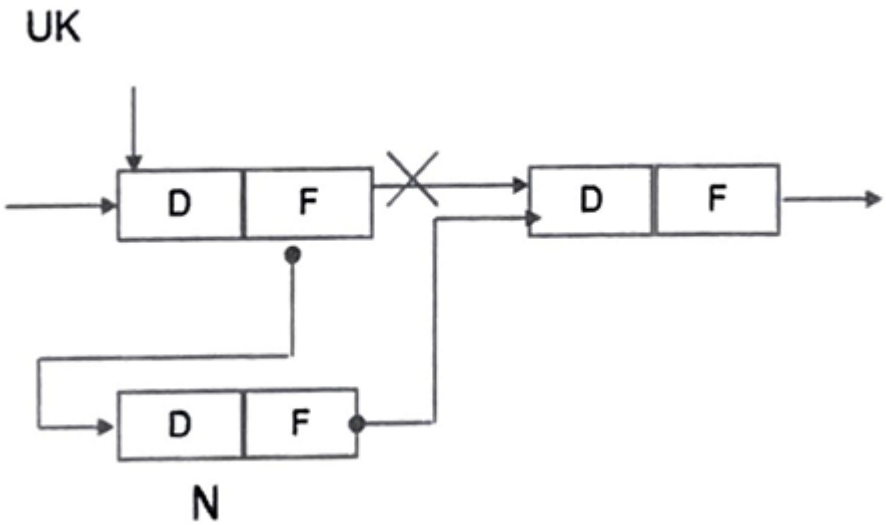
Доступ к элементу списка - просмотр списка с головы (последовательный) - медленно.

Кольцевой - просматривать можно с любого элемента, после доступа к нужному элементу в РВ заносится адрес его последователя.

Главные операции над списками

- Вставка элемента в список;
- Исключение элемента из списка.

Вставка элемента с адресом N в односвязный список после элемента UK:



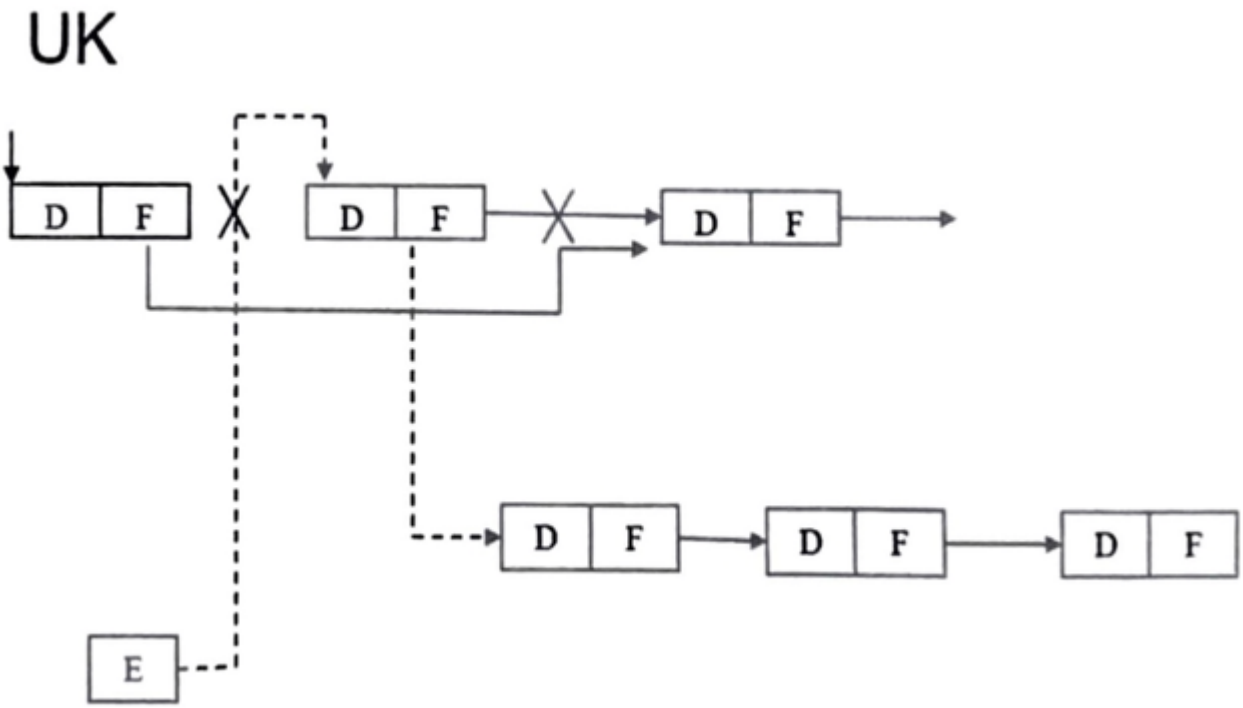

```
F(N) <- F(UK);
F(UK) <- N;
```

Исключение элемента, стоящего после элемента UK из односвязного списка:

```
IF F(UK) <> Nil
THEN
  P <- F(UK)    {запоминаем адрес исключаемого элемента}
  F(UK) <- F(P) {изменяем адрес указателя элемента с адресом UK}
ELSE   {исключение невозможно, так как список пуст}
```

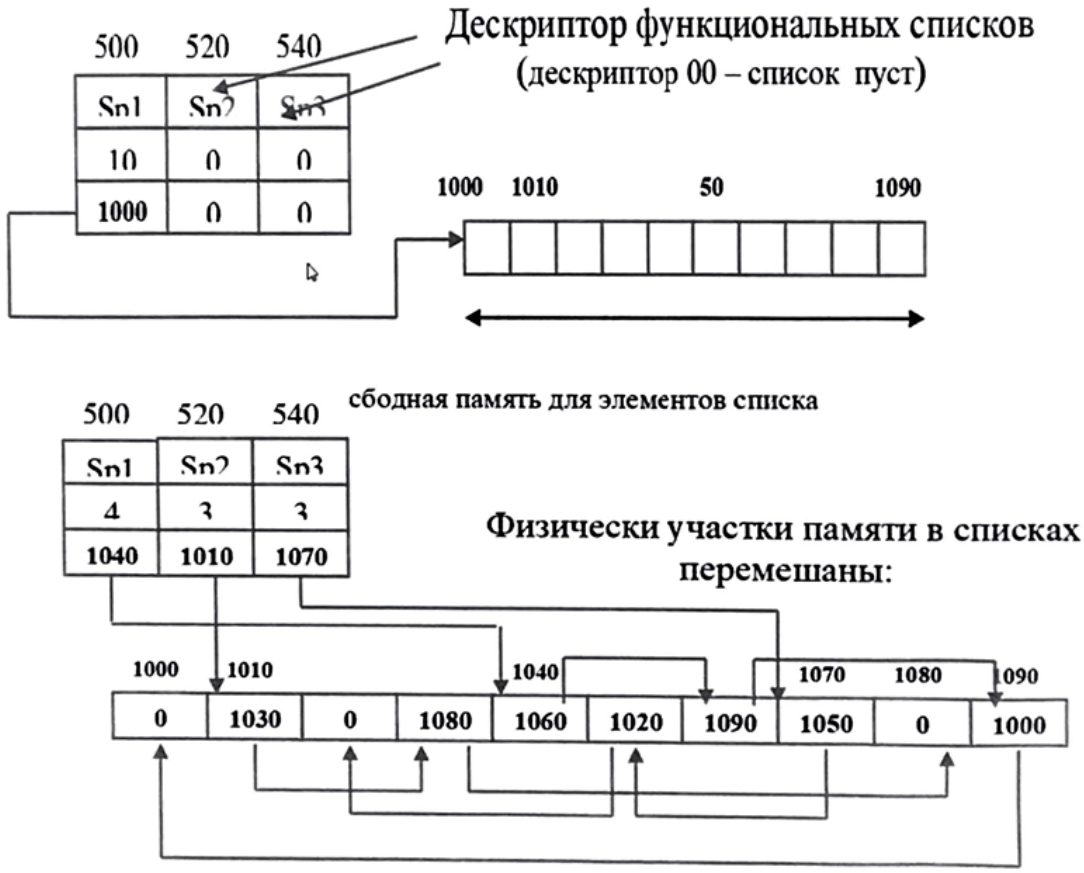
удаленный элемент включается в начало списка свободных элементов:

```
F(P) <- E;
E <-P;
```



При использовании нескольких списковых структур в системе создается не менее 2-х списков:

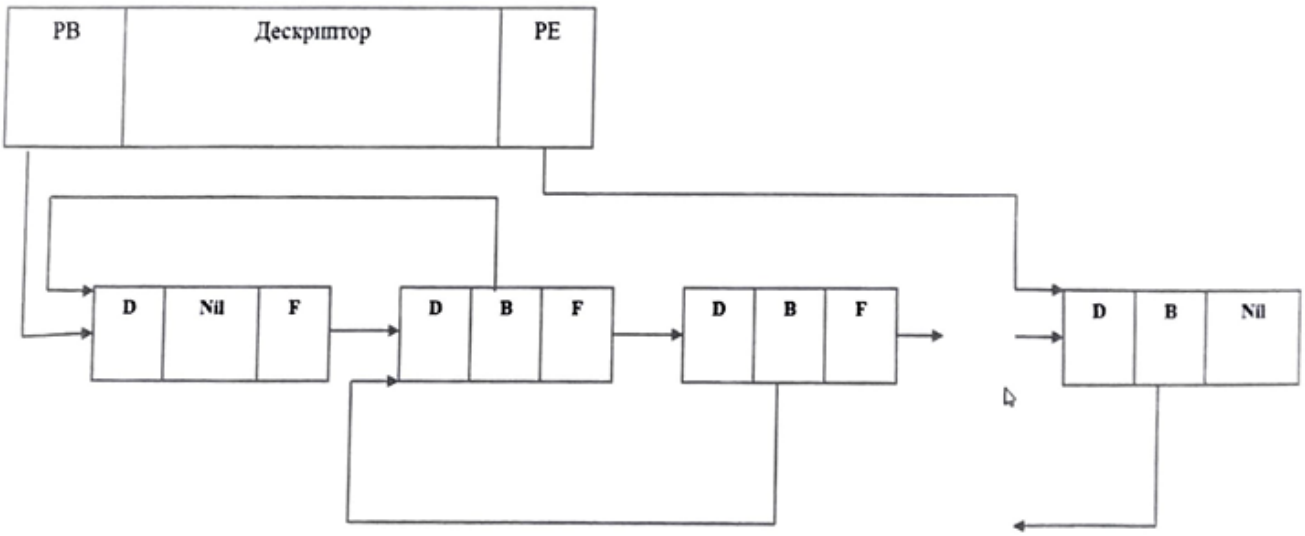
- включенных элементов (функциональный список с информацией) и
- исключенных (свободных элементов), причем второй - один на все существующие в программе списки.



Своб. обл-ти: 1080, 1000, 1020

Двусвязные линейные списки

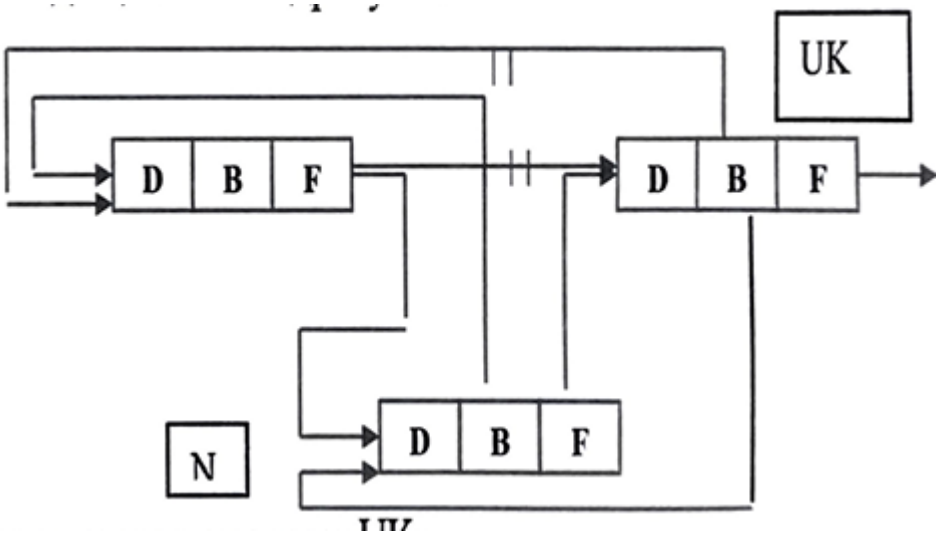
Двусвязный список, в котором каждый элемент имеет два указателя: вперед (F) и назад (B). В структуру этого списка добавляется указатель конца (PE). Начало и конец в этом списке логически эквивалентны, так как доступ к элементу списка имеется с любого конца.



Кольцевой двусвязный список:

объединить два пустых указателя, указатель конца не нужен, а указатель начала м. б. в любом месте.

Включение элемента с адресом N в двусвязный список



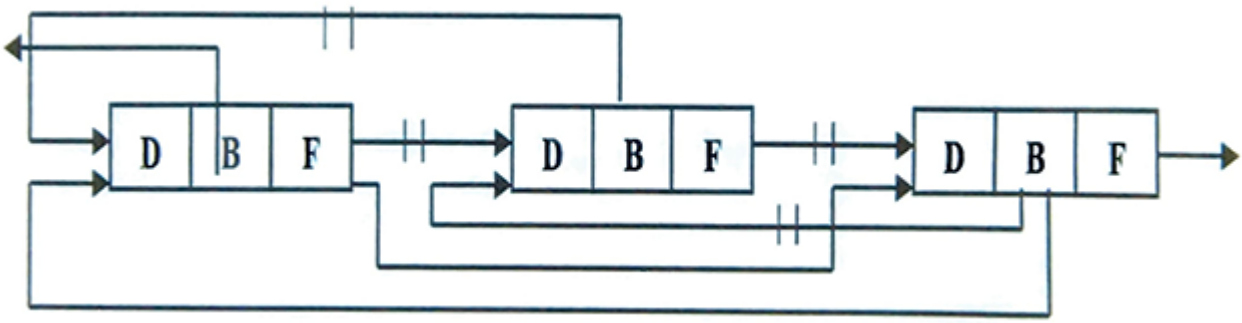
а) перед элементом, находящимся по адресу UK:

```
B(N) <- B(UK)
B(UK) <- N
F(N) <- UK
F(B(N)N
```

б) после элемента, находящегося по адресу UK:

```
F(N) <- F(UK)
B(N) <- UK
B(F(UK)<- N;
F(UK) <- N;
```

Удаление элемента по адресу UK в двусвязном списке:



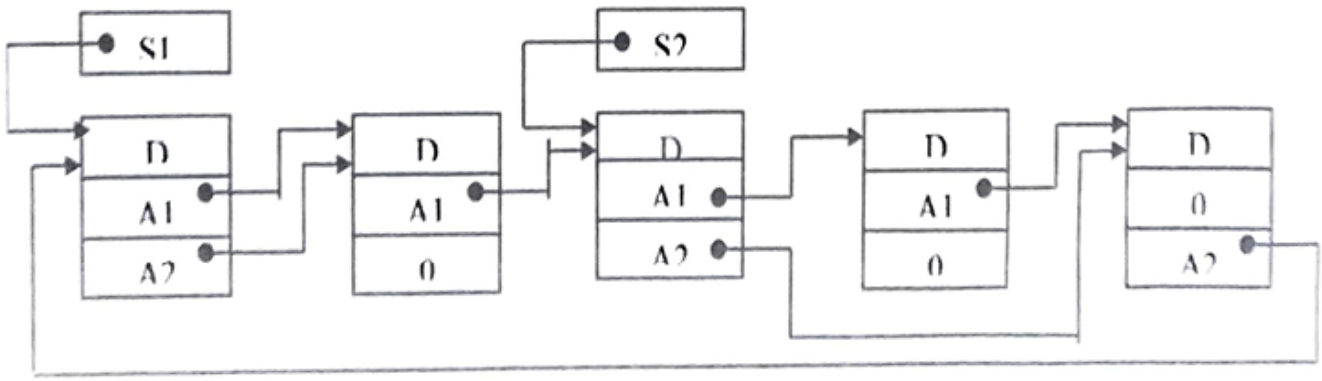

```
B(F(UK)) <- B(UK)
F(B(UK)) <- F(UK)
```

Операции с линейными списками

1. Получить доступ к k-му узлу списка.
2. Объединить два (или более) линейных списка в один список.
3. Разбить линейный список на два (или более)списка.
4. Сделать копию линейного списка.
5. Определить количество узлов в списке.
6. Выполнить сортировку узлов списка по некоторым полям в узлах.
7. Найти узел с заданным значением в некотором поле.

Нелинейные списки

Двусвязный список может быть и нелинейным, т. е. второй указатель двусвязного списка задает произвольный порядок следования. Т.о., каждый элемент этого списка содержится в двух односвязных списках, при этом переменные S1 и S2 явл-ся указателями начала двух разных односвязных списков, например:



Многосвязные списки

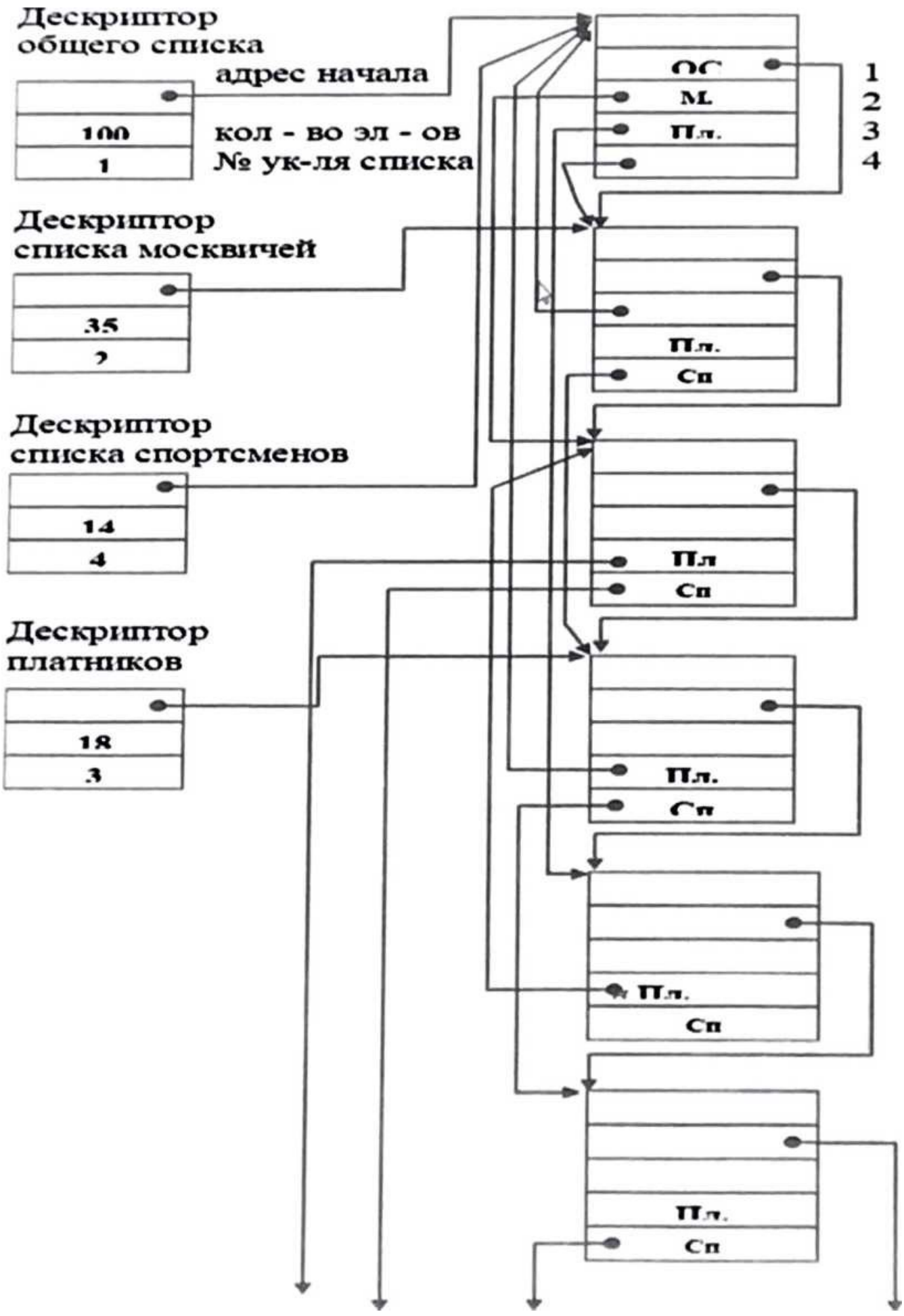
В более общем случае каждый элемент связного списка может содержать произвольное конечное число связок, причем, различное в различных элементах. В этом случае получается многосвязный список, каждый элемент которого входит одновременно в несколько разных односвязных списков.

Такие списки еще называют прошитыми.

Например, есть списки абитуриентов, содержащие общие сведения: фамилию, имя и отчество, год рождения и т.д. Приемную комиссию дополнительно интересует:

- Москвич или нет (нуждается ли в общежитии);
- Спортсмены;
- Платники.

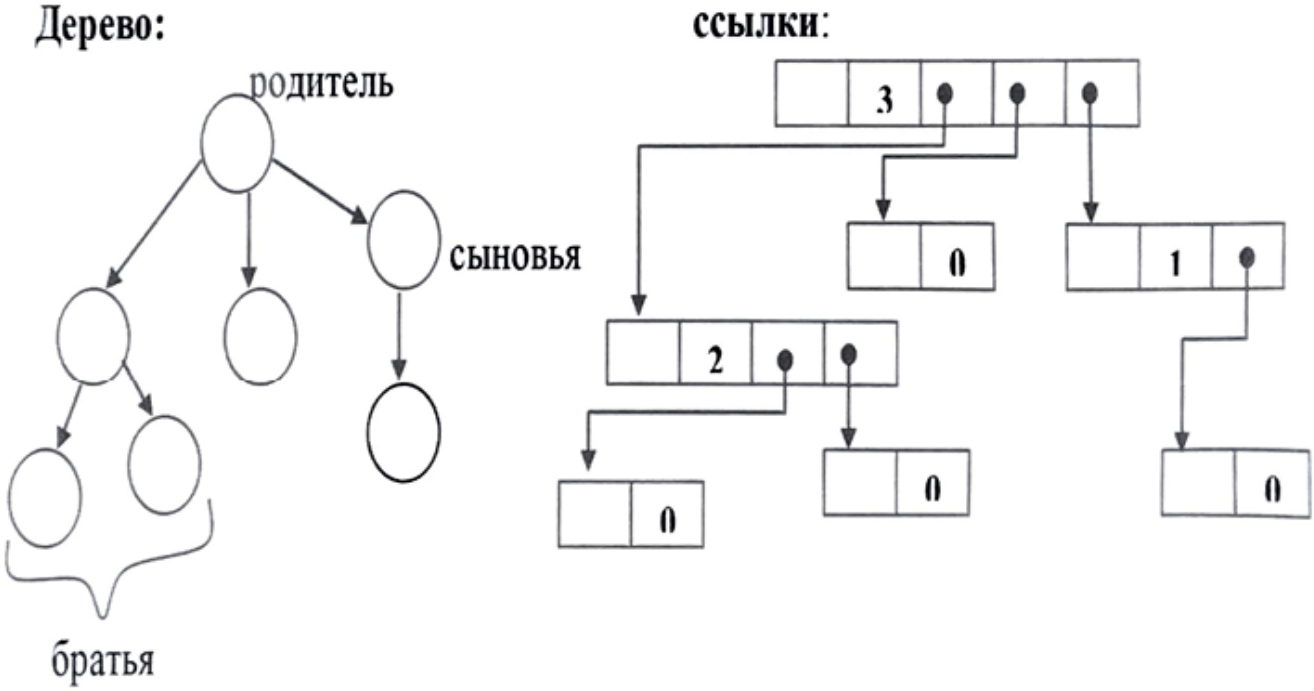
Пример многосвязного списка



Наиболее общий вид многосвязной структуры характеризуется следующими свойствами:

- Каждый элемент структуры содержит произвольное число направленных связок с другими элементами (ссылок на др. элементы);
- С каждым элементом может связываться произвольное число других элементов;
- Каждая связка имеет не только направление, но и вес.
- Такую структуру называют сетью, логически она эквивалентна взвешенному орграфу общего вида.

Пример различного представления связного списка (т.е. различными структурами)



Преобразование в бинарные