Алгоритмы и структуры данных

Лекция 6
ДЕРЕВЬЯ
(продолжение)

1. Представления и реализации бинарных деревьев (См. учебное пособие «ДСД», п. 3.5)
2. Примеры функций на бинарных деревьях
3. БД с размеченными листьями

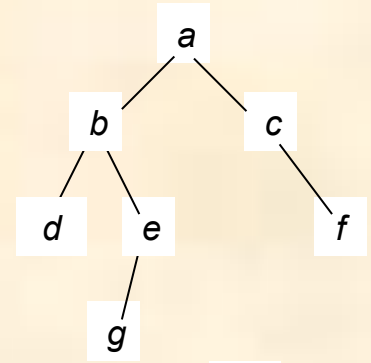
Ссылочная реализация бинарного дерева в связанной памяти

Базовый тип узлов *Elem*.

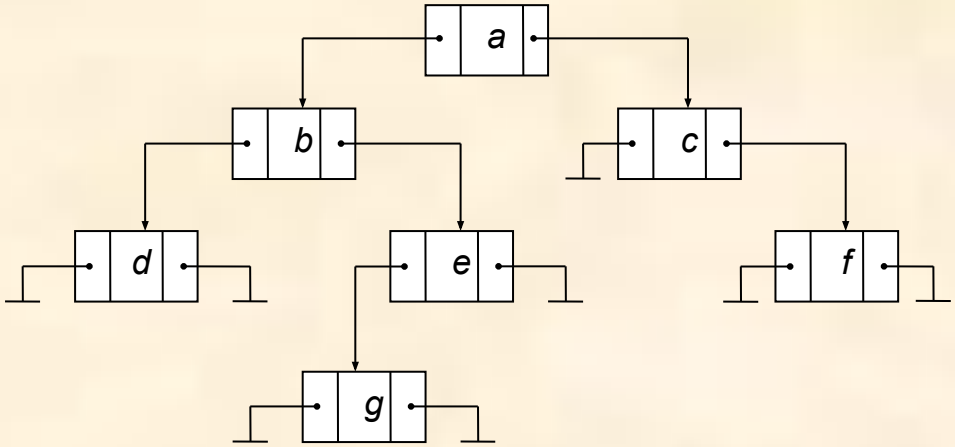
BT (*Elem*) представляетя рекурсивными типами *BinT* и *Node*

```
type
BinT = ^Node;              {представление бинарного
   дерева}
Node = record              {узел: }
Info: Elem;     {− содержимое}
LSub: BinT;     {− левое поддерево}
RSub: BinT      {− правое поддерево}
end {Node}
```

Пример: бинарное дерево (а) и его представление в ссылочной реализации (б)



а



б

Набор основных функций для работы с бинарным деревом на основе ссылочной реализации

CreateBT: *BinT*;

NullBT (*t*: *BinT*): *Boolean*;

RootBT (*t*: *BinT*): *Elem*;

LeftBT (*t*: *BinT*): *BinT*;

RightBT (*t*: *BinT*): *BinT*;

ConsBT (*e*: *Elem*; *LS*, *RS*: *BinT*): *BinT*;

DestroyBT (**var** *b*: *BinT*);

Otkaz (*n*: *Byte*);

Функция *CreateBT* формально специфицируется соотношениями

CreateBT: → *BT*; *Null* (*CreateBT*) = *true*.

Otkaz вызывается при попытке некорректного применения основных функций

В процедурно-модульной парадигме

интерфейс АТД "Бинарное дерево” представлен в файле Btree.h,

реализация основных функций для работы с АТД "Бинарное дерево” - в файле bt_implementanion.cpp,

пример работы с АТД "Бинарное дерево” - в файле work_bt.cpp, где для ввода ВТ из файла используется его КЛП-представление, а для вывода его на экран порядок КЛП, ЛКП и ЛПК.

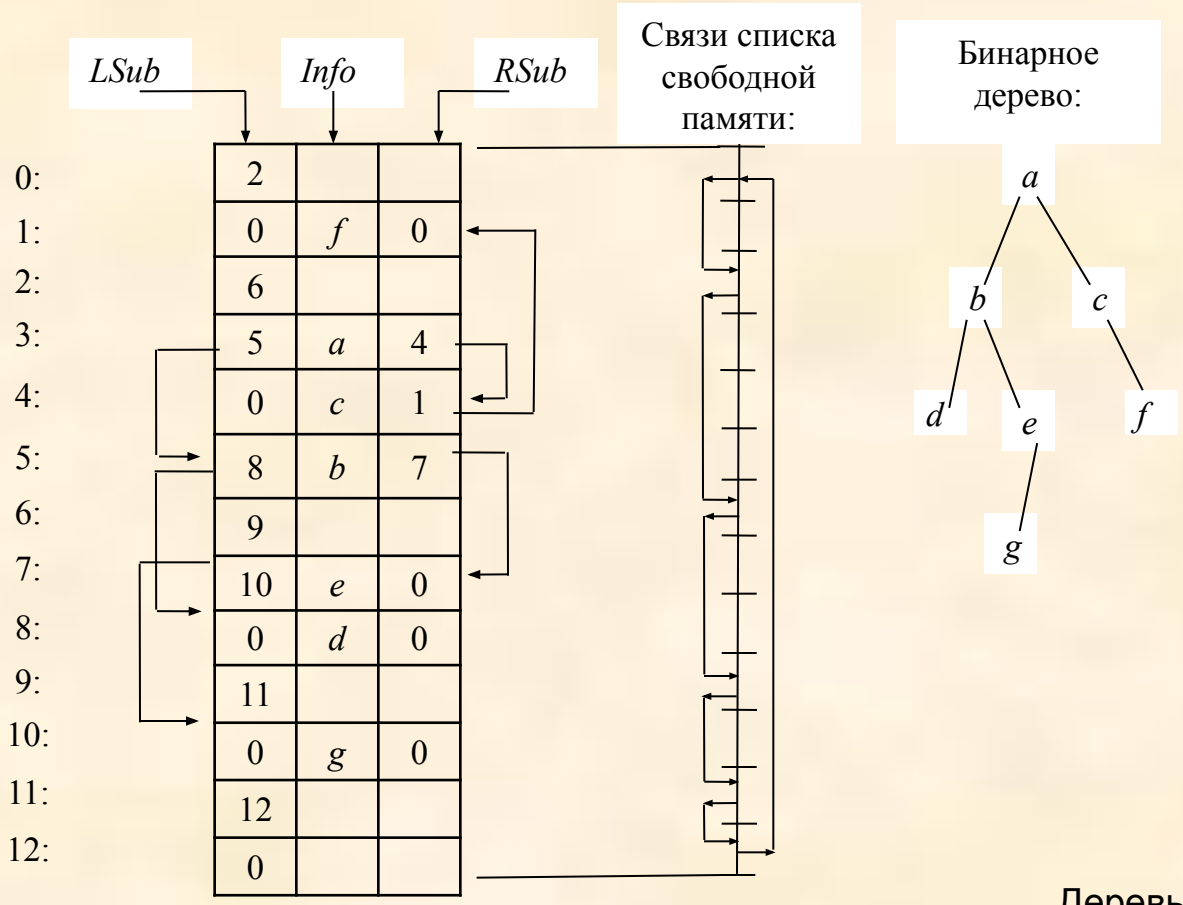
Программа также подсчитывает высоту и количество узлов ВТ.

Ссылочная реализация ограниченного бинарного дерева на базе вектора

```
type Adr = 0 .. MaxAdr;          {диапазон «адресов» в векторе Mem}
     BinT = Adr;                 {представление бинарного дерева}
     Node = record               {узел: }
              Info: Elem;        {– содержимое}
              LSub: BinT;        {– левое  поддерево}
              RSub: BinT         {– правое поддерево}
     end {Node};
     Mem = array [Adr] of Node   {вектор для хранения дерева}
```

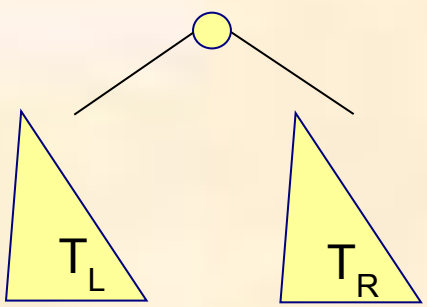
Пример ссылочного представления бинарного дерева на базе вектора

Дерево представляется переменной b: Bin T, значение b = 3 - номер элемента массива, в котором хранится корень.



Примеры функций на бинарных деревьях

T:



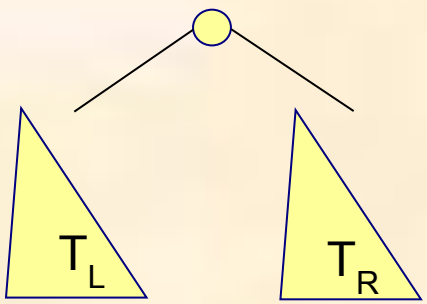
Число узлов БД:

$$Nv(T) = \begin{cases} 0, & \text{при } T = \Lambda \\ Nv(T_L) + Nv(T_R) + 1, & \text{при } T \neq \Lambda \end{cases}$$

```
Nat0 Nv (BinT t)
{
    if (Null(t)) return 0;
    else return (Nv (LeftBT(t)) + Nv (RightBT(t)) +1);
}
```

Примеры функций на бинарных деревьях

T:

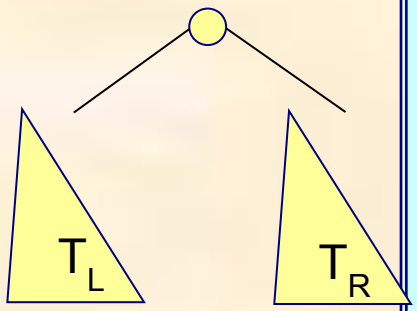


Число листьев БД:

$$NLeaf(T) = \begin{cases} 0, & \text{при } T = \Lambda \\ 1, & \text{при } (T_L = \Lambda) \,\&\, (T_R = \Lambda) \\ NLeaf(T_L) + NLeaf(T_R), & \text{иначе} \end{cases}$$

```
Nat0 NLeaf (BinT  t):
{
   if (Null(t)) return  0;
   else if (Null(LeftBT(t)) && Null(RightBT(t))) return 1;
     else return (NLeaf (LeftBT(t)) + NLeaf (RightBT(t)));
}
```

T:



Высота БД:

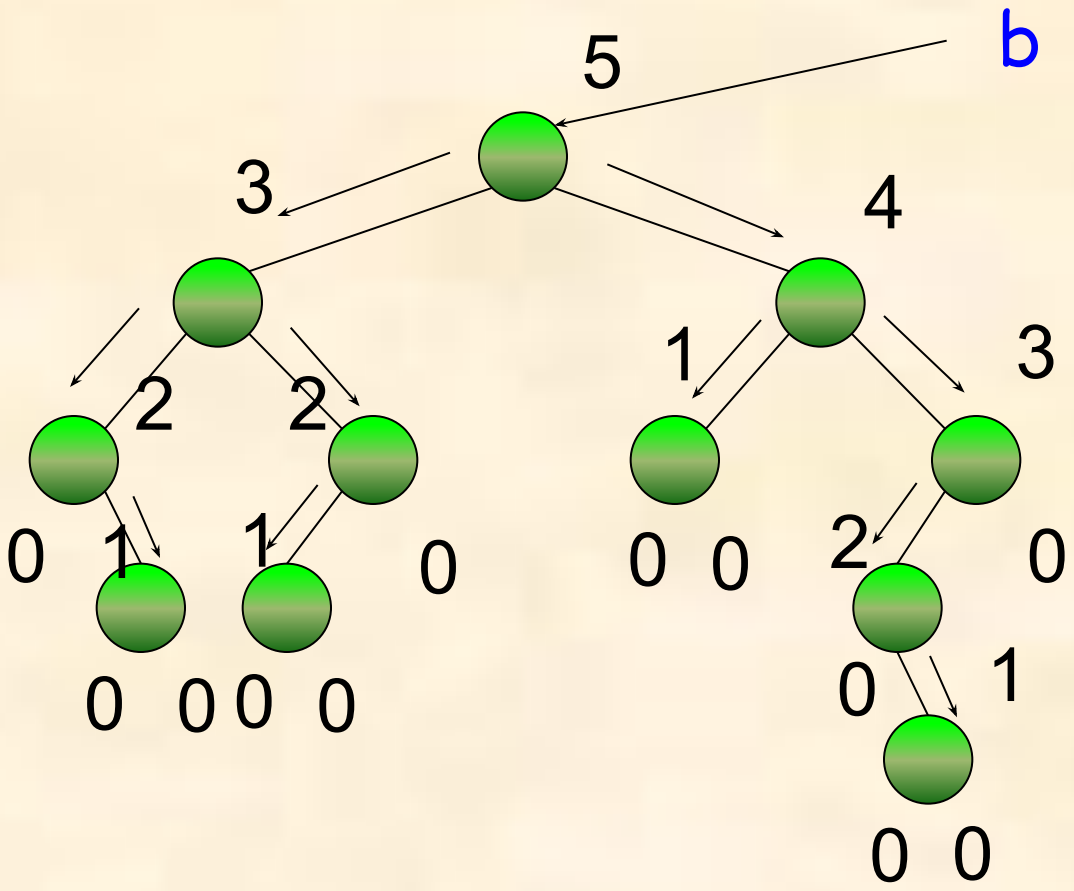
$$H(T) = \begin{cases} 0, & \text{при } T = \Lambda \\ \max(H(T_L), H(T_R)) + 1, & \text{при } T \neq \Lambda \end{cases}$$

```
Nat0 H ( BinT t)
{
    if (Null(t)) return 0;
    else return (max (H(LeftBT(t)), H(RightBT(t))) +1);
}
```

Вычисление H (b)

H(Λ)=0

H( )=1



Проверить свойство дерева:
для каждого узла БД имеем $H(T_L) - H(T_R) = 1$

```
Bool  HF (BinT  t)
{   if (Null(t))  return true;
    else
    {
       HL = H(LeftBT(t));
       HR = H(RightBT(t));
       return (HF(LeftBT(t)) && HF(RightBT(t))
        &&  ( (HL -  HR) == 1));
    }
}
```

!Пояснить неэффективность такой реализации функции HF

!Самостоятельно написать хороший вариант

Бинарные деревья с размеченными листьями (комбинации)

Бинарное дерево с размеченными листьями - это либо знак («атом», атомарное дерево), либо (упорядоченная) пара бинарных деревьев с размеченными листьями.

⟨*comb*(α)⟩ ::= ⟨*atomic*(α)⟩ | ⟨*left*(α)⟩ ⟨*right*(α)⟩

⟨*atomic*(α)⟩ ::= ⟨α⟩

⟨*left*(α)⟩ ::= ⟨*comb*(α)⟩

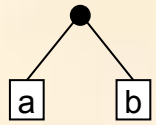
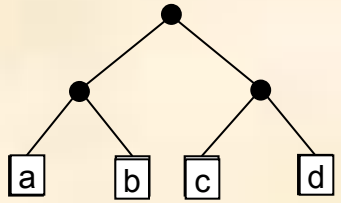
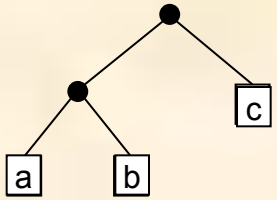
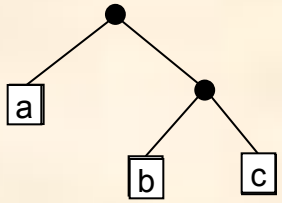
⟨*right*(α)⟩::= ⟨*comb*(α)⟩

Можно ввести понятие «пара» (pair):

⟨*comb*(α)⟩ ::= ⟨*atomic*(α)⟩ |⟨*pair*(α)⟩

⟨*pair*(α)⟩ ::= ⟨*comb*(α)⟩ ⟨*comb*(α)⟩

Примеры скобочной записи и графического изображения комбинаций:

(a.b)	
((a.b).(c.d.))	
((***a***.***b***).***c***)	
(a.(b.c))	

Смешанное бинарное дерево (СБД)

Можно ввести «гибрид» бинарных деревьев и комбинаций.

Определим *смешанное* (*расширенное*, *декорированное*) *бинарное дерево* (*СБД*) над базовыми типами α (внутренние узлы) и β (внешние узлы - листья):

⟨*СБД*(α, β)⟩ ::= ⟨*atomic*(β)⟩ |⟨*root*(α)⟩ ⟨*left*(α, β)⟩ ⟨*right*(α, β)⟩,

⟨*atomic*(β)⟩ ::= ⟨β⟩,

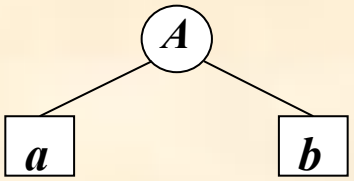
⟨*root*(α)⟩ ::= ⟨α⟩,

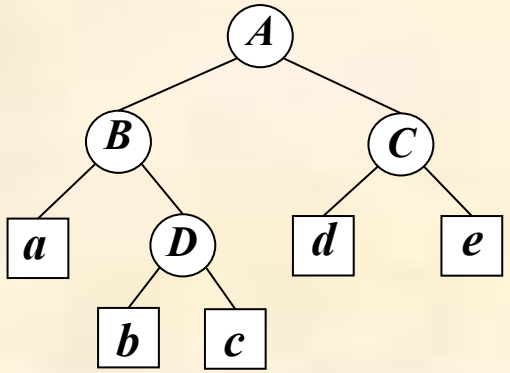
⟨*left*(α, β)⟩ ::= ⟨*СБД*(α, β)⟩,

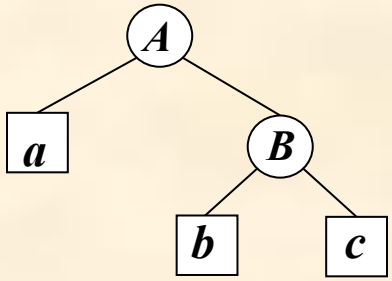
⟨*right*(α,β)⟩ ::= ⟨*СБД*(α, β)⟩.

Примеры СБД

$\alpha = \{A, B, C, \ldots\}$; $\beta = \{a, b, c, \ldots\}$



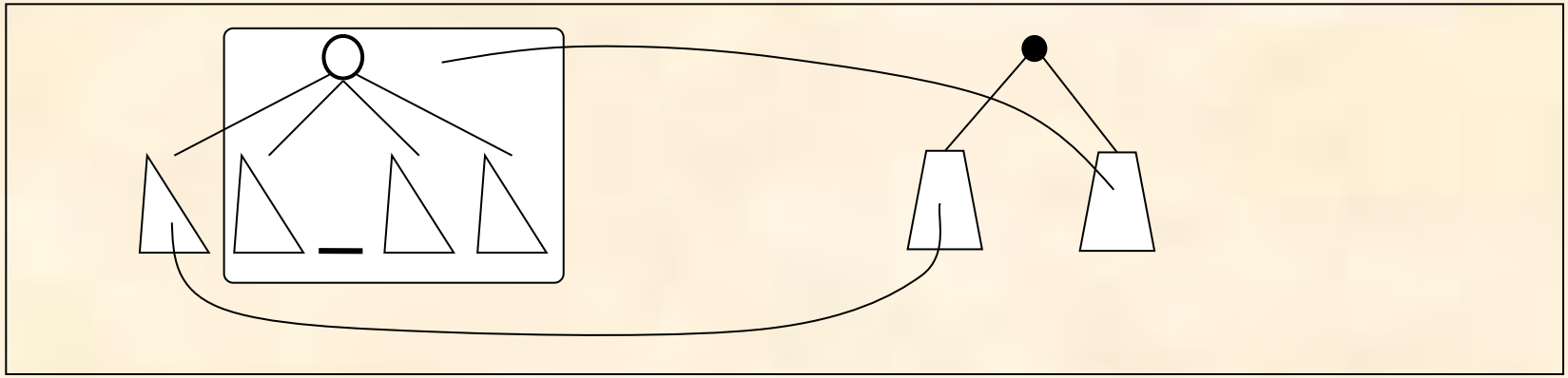




Если α - пусто, то получим *комбинации* над типом β, а если β - пусто (пустое дерево), то получим *бинарные деревья* над типом α.

Соответствие между деревьями (T) и комбинациями (C)

- дерево с нулевым лесом поддеревьев (с пустым *листингом*) соответствует атомарной комбинации;
- дерево с ненулевым лесом поддеревьев (с непустым *листингом*) преобразуется в комбинацию так, что *первое* дерево леса соответствует *левой* части комбинации, а оставшееся дерево, состоящее из корня и остальных деревьев леса, соответствует *правой* части комбинации.



Формально соответствие «дерево ⇒ комбинация» может быть описано функцией

$$C(T) \equiv \textbf{if } Null(Listing(T)) \textbf{ then } Root(T) \textbf{ else } ConsComb\,(\,C(Head(Listing(T))),\, C(\,ConsTree(Root(T),\, Tail(Listing(T))))$$

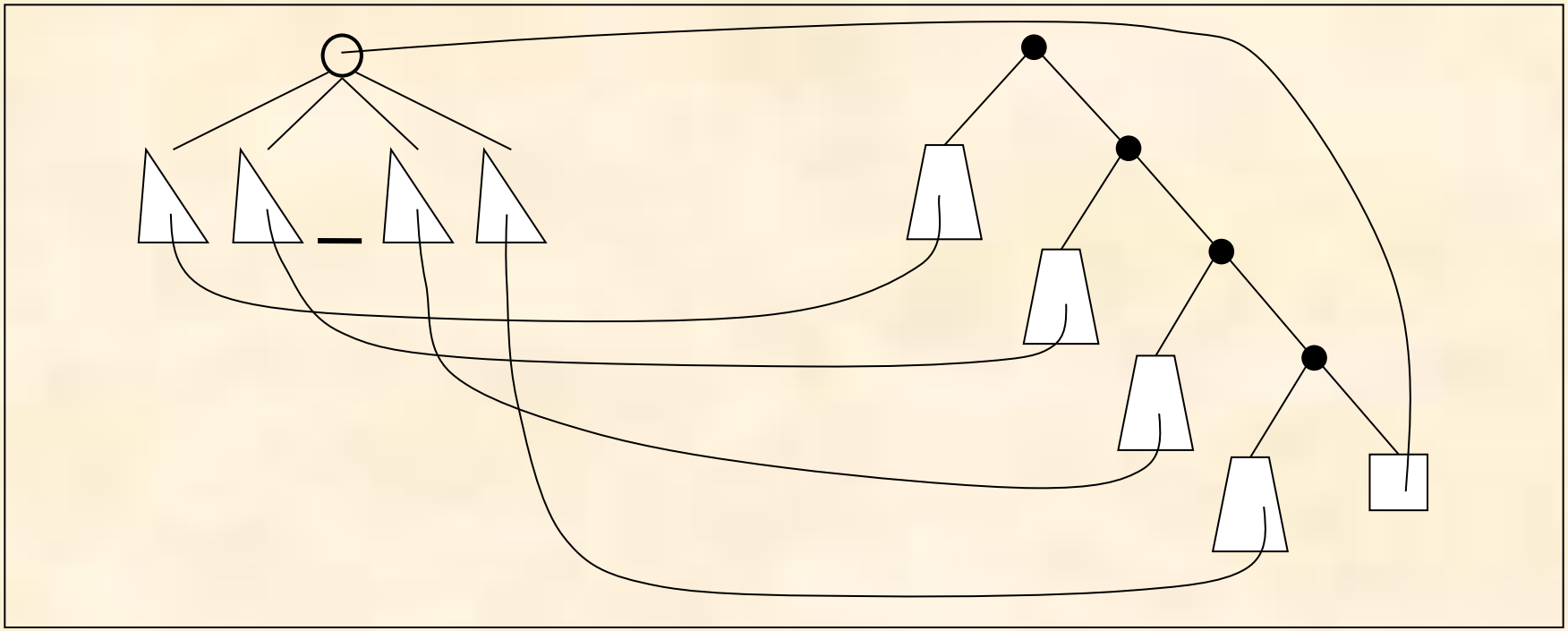
Listing(*T*) – лес поддеревьев исходного дерева,

Head(*Listing*(*T*)) – первое дерево этого леса,

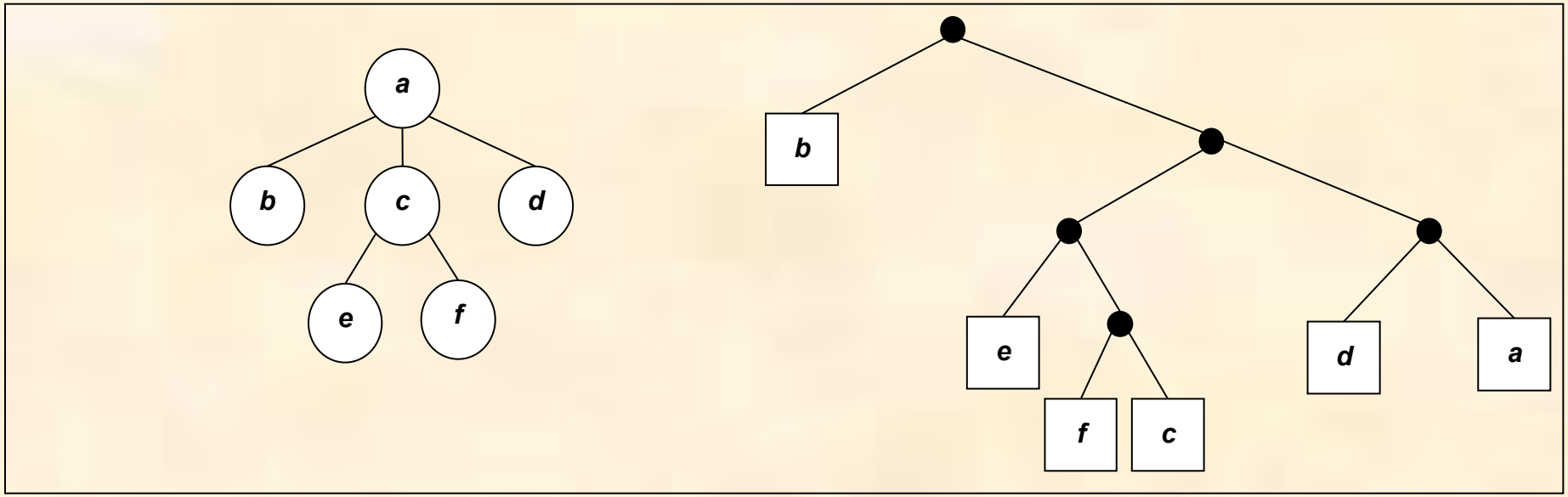
Tail(*Listing*(*T*)) – лес остальных поддеревьев (т.е. за исключением первого),

ConsTree(*Root*(*T*), *Tail*(*Listing*(*T*))) – дерево, состоящее из корня исходного дерева и остальных деревьев леса его поддеревьев.

Преобразование на предыдущем слайде можно развернуть по последовательности поддеревьев:



Пример преобразования «дерево ⇒ комбинация»

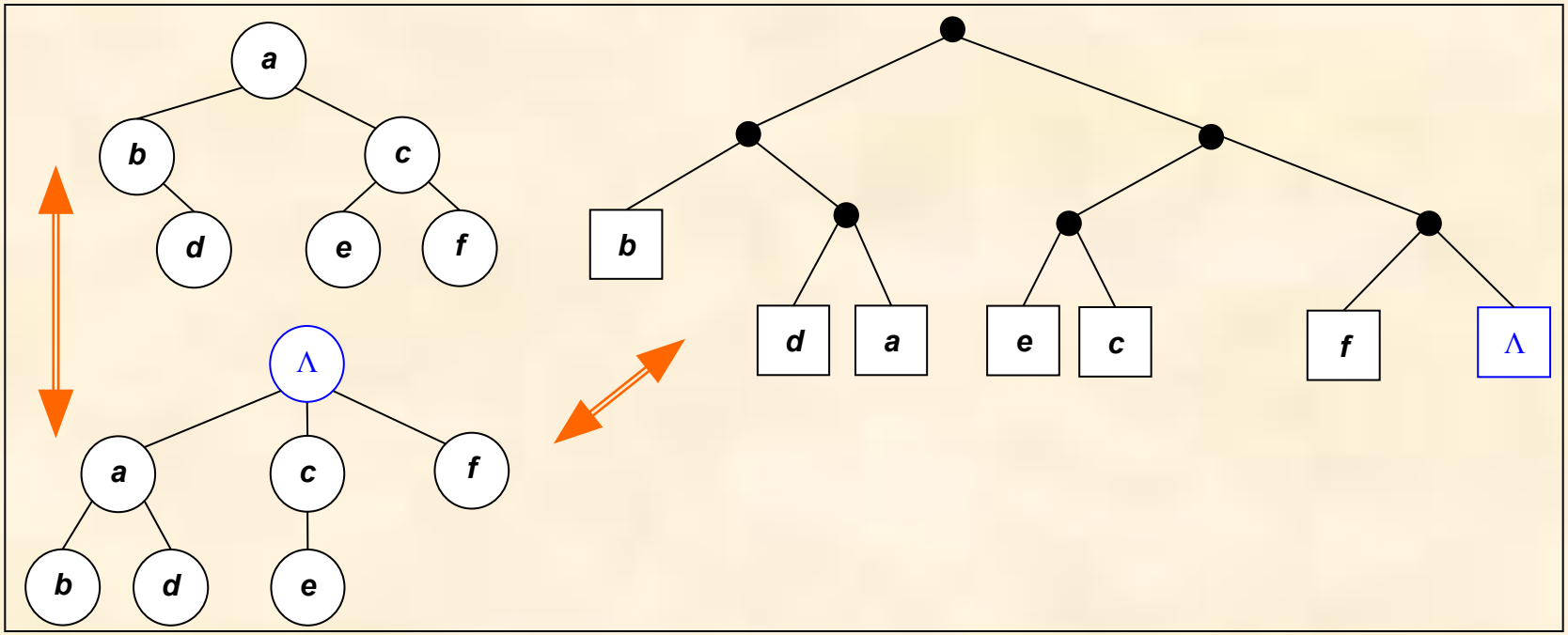


Обратное преобразование из комбинации в дерево получается обращением стрелок на рисунках слайдов 17, 19 и описывается функцией

$T(C) \equiv$ **if** *Atom*(C) **then** *ConsTree*(*Val*(C), **nil**)
else *ConsTree*(*Root*(*T*(*Right*(C))), *Cons*(*T*(*Left*(C)), *Listing*(*T*(*Right*(C))))))

Преобразования «бинарное дерево ⇒ комбинация»

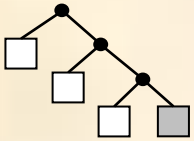
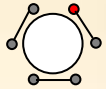
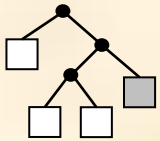
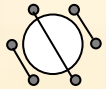
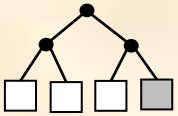
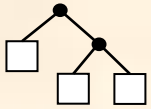
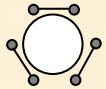
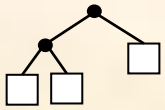
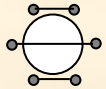
1. бинарное дерево ⇒ лес,
2. лес ⇒ дерево (добавляется фиктивный корень),
3. дерево ⇒ комбинация



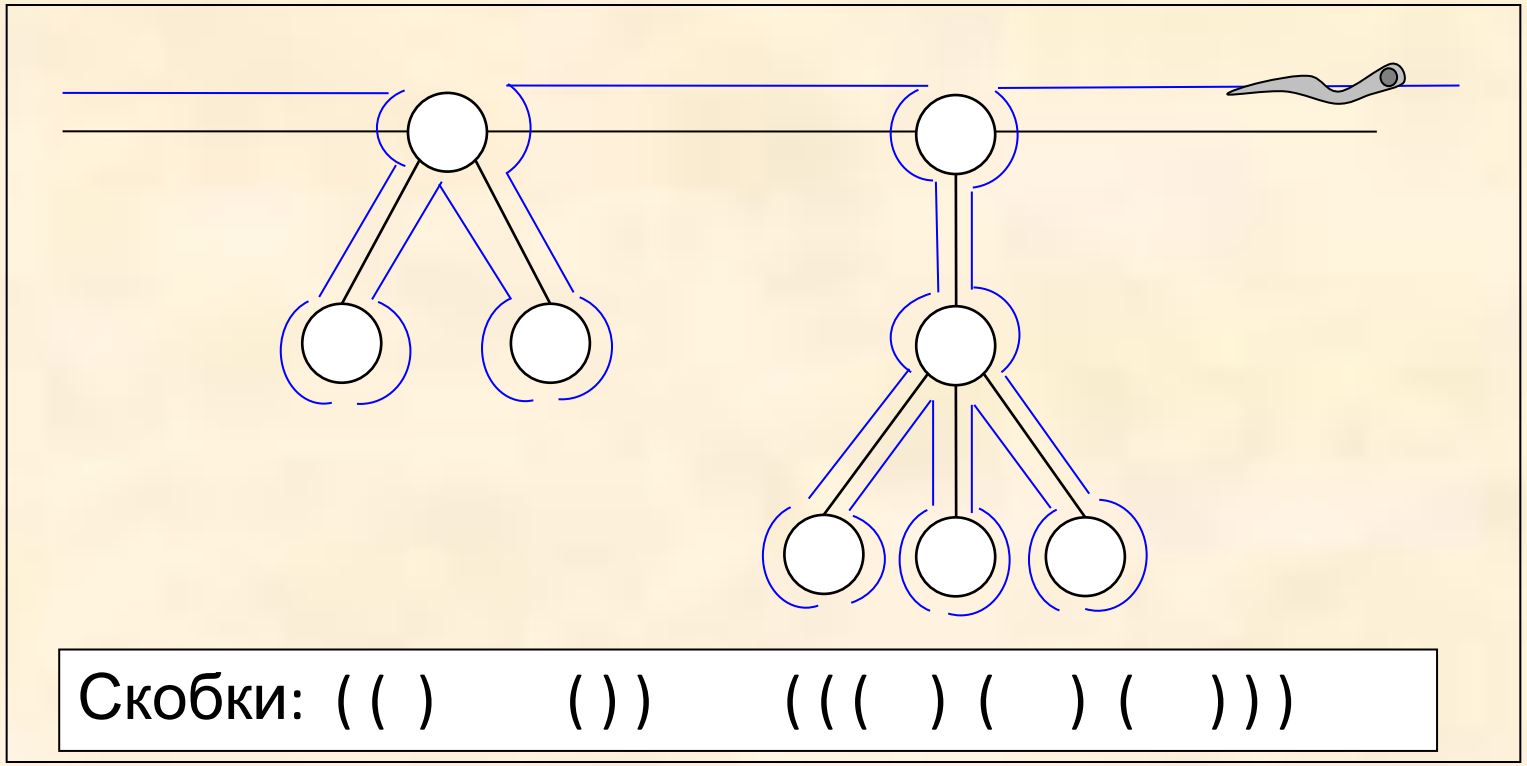
Примеры соответствия комбинаторных объектов (3 узла)

Задание. Представить аналогичную таблицу для случая «Число узлов =4».

Дональд Э. Кнут, Искусство программирования. Том 4. Выпуск 4. Генерация всех деревьев.

№	*Лес*	*Бинарное дерево*	*Комбинация*	*Скобки*	*Рукопожатия*
1				( ) ( ) ( )	
2				( ) (( ))	
3				(( )) ( )	
4				(( ) ( ))	
5				((( )))	

Ползущий червь



КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ