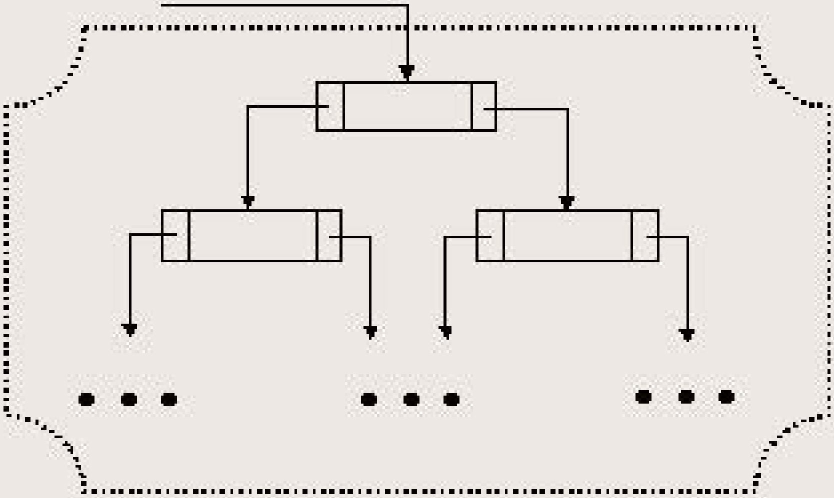


"Вычисленная" структура хранения является *деревом поиска*

Определение 3.7. Связный граф без циклов называется *деревом*.

Определение 3.7'. Структура типа *дерева* (*древовидная структура*) с базовым типом Т – это

- либо пустая структура,
- либо узел (вершина) со значением типа Т, с которым связано конечное число древовидных структур (*поддеревьев*) с базовым типом Т.

Определение 3.7-1. Узел y, который находится непосредственно под узлом x (т.е. есть ребро (x,y)), называется (непосредственным) *потомком* x, а x называется (непосредственным) *предком* y.

Определение 3.7-2. Узел, у которого нет предков, называется *корнем*.

Определение 3.7-3. Узел, у которого нет потомков, называется *листом*.

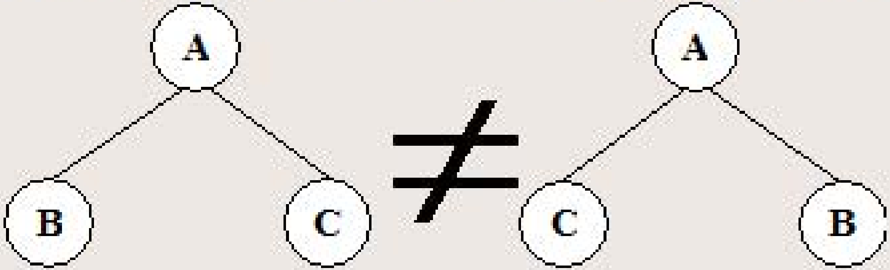
Определение 3.7-4. Число ветвей (ребер), которые нужно пройти, чтобы продвинуться от корня к узлу x,.называется *длиной пути* к x.

Определение 3.7-5. Узлы с одинаковой длиной пути образуют уровень (ярус) дерева. Корень расположен на уровне 1, его потомки на уровне 2 и т.д. (принято изображать узлы дерева одного и того же уровня на одной горизонтальной прямой).

Определение 3.7-6. Максимальный уровень дерева называется его *глубиной*.

Определение 3.7-7. Число непосредственных потомков узла называется *степенью узла*. Максимальная степень всех узлов есть *степень дерева*.

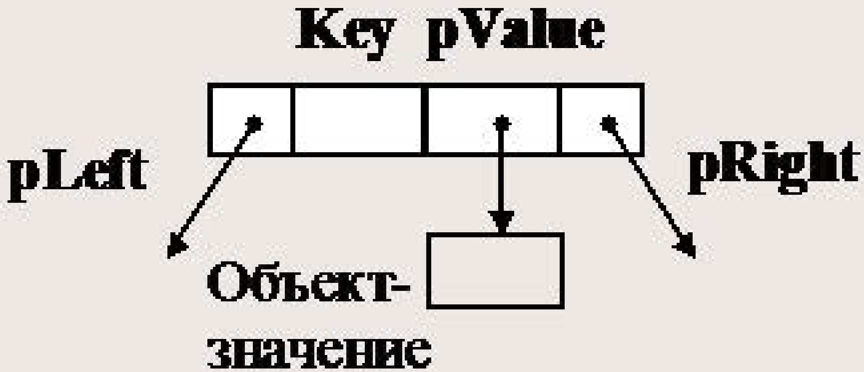
Определение 3.7-8. *Упорядоченное дерево* – это дерево, у которого ветви каждого узла упорядочены.



Определение 3.7-9. Упорядоченное дерево степени 2 называется *бинарным деревом*.

Определение 3.7-10. Если для любой вершины бинарного дерева значения в левом потомке меньше значения узла, а значение в правом потомке больше значения узла, то такое дерево называется ***деревом поиска***.

```
class TTreeNode : public TTabRecord {
  protected: // указатели на поддеревья
    PTTreeNode pLeft, pRight;
  public:
    TTreeNode ( TKey k="", PTDatValue pVal=NULL,
      PTTreeNode pL=NULL, PTTreeNode pR=NULL );
    PTTreeNode  GetLeft  ( void ) const;
    PTTreeNode  GetRight ( void ) const;
};
```

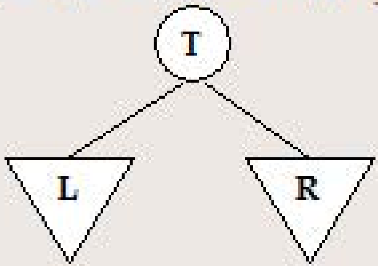



TDataCom
TTable
TTreeTable
Абстрактный базовый класс – спецификации методов таблицы
Класс, обеспечивающий представление таблиц при помощи деревьев поиска

☑ Обработка дерева – выполнение необходимой операции для каждой узла дерева

➮ Реализация подобного типа действий предполагает умение *обхода* (*обхода*) дерева

Представим дерево в общем виде



T – top (корень)

L – left (левое поддерево)

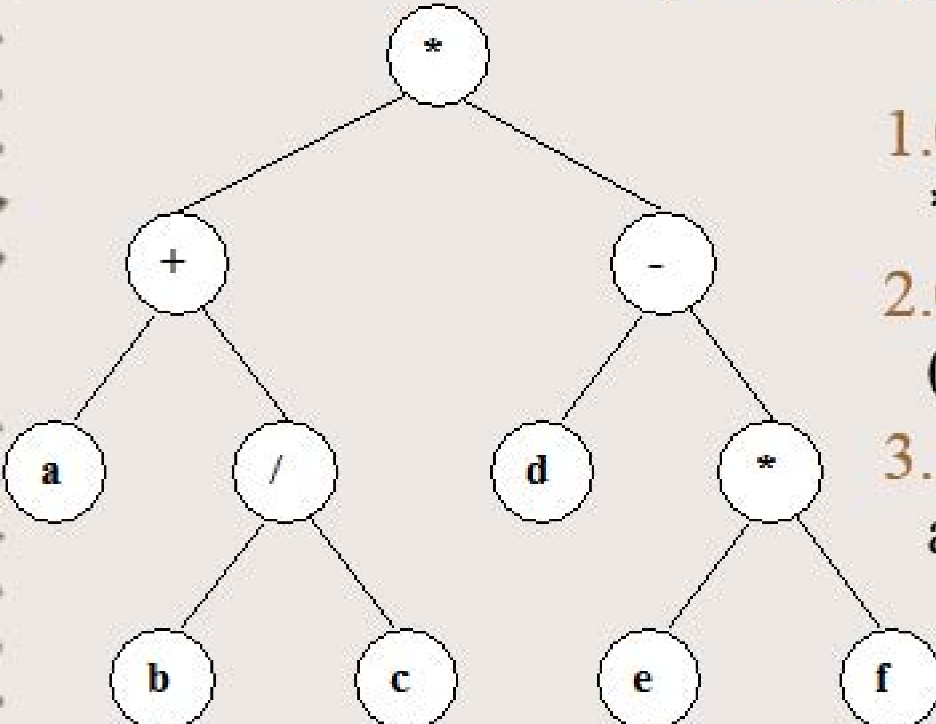
R – right (правое поддерево)

Тогда возможны следующие варианты обхода

1. T, L, R – сверху вниз
2. L, T, R – слева направо
3. L, R, T – снизу вверх
4. T, R, L – сверху вниз
5. R, T, L – справа налево
6. R, L, T – снизу вверх

Пример: Представление арифметического выражения в виде бинарного дерева

(a+b/c)*(d-e*f)



1. Сверху вниз TLR

*+a/bbc-d*ef - префиксная

2. Слева направо LTR

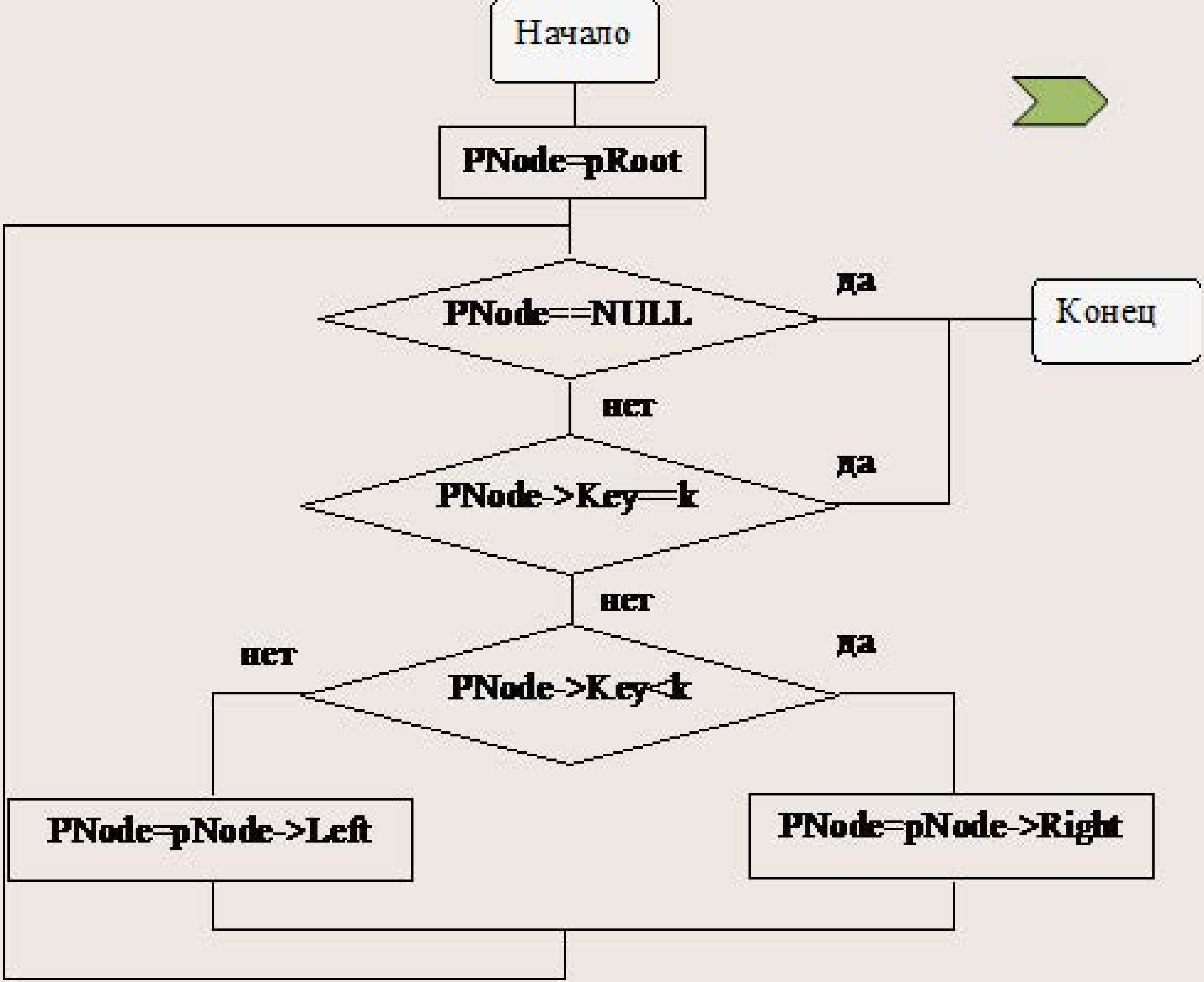
(a+b/c)*(d-e*f) - инфиксная

3. Снизу вверх LRT

abc/+def*-* - постфиксная

Пример: Печать значений дерева поиска (схема **LTR**, рекурсия)

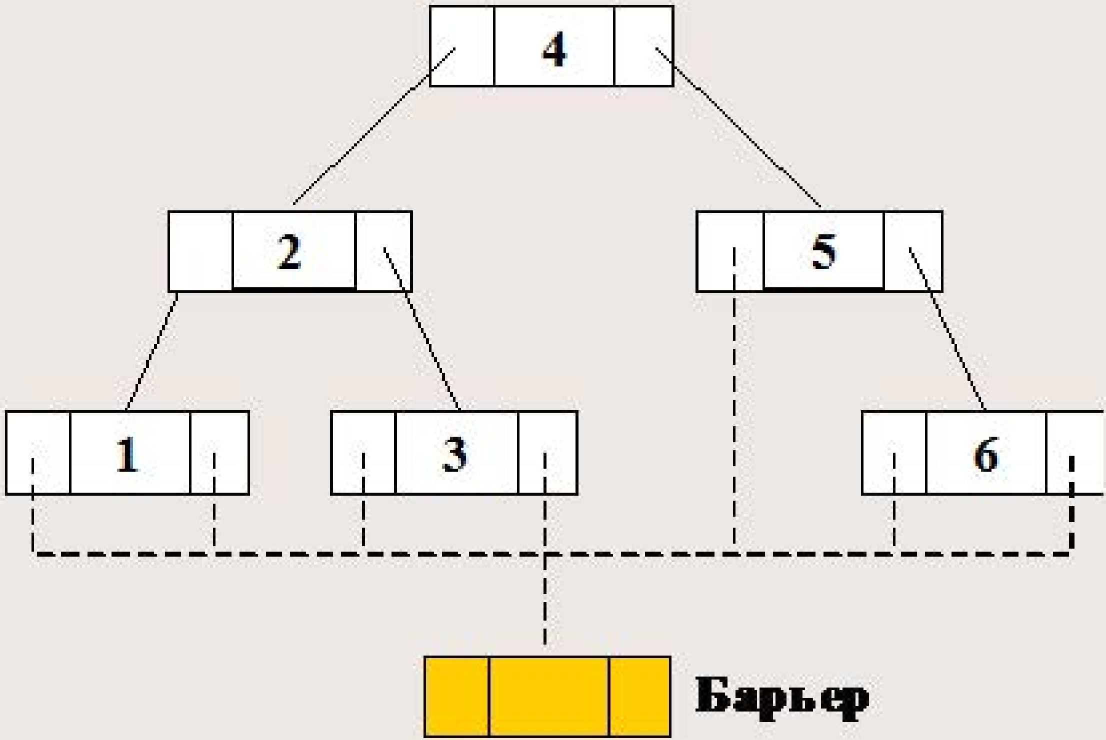
```
void TTreeTable :: PrintTreeTab(ostream
 &os, PTTreeNode pNode) {
  if ( pNode != NULL ) { // печать
 дерева с вершиной pNode
    PrintTreeTab(os,pNode->pLeft);
    pNode->Print(os); os << endl;
    PrintTreeTab(os,pNode->pRight);
  }
}
```



Начало
PNode=pRoot
PNode==NULL
да
Конец
нет
PNode->Key==k
да
нет
PNode->Key<k
нет
да
PNode=pNode->Left
PNode=pNode->Right

```
//  поиск в дереве поиска - рекурсия
PTTreeNode
 FindRecord(TKey k,PTTreeNode pNode){
 if ( pNode != NULL ) { // лист
  if ( pNode->Key < k ) // вправо
    pNode = FindRecord(k,pNode->Right);
  if ( pNode->Key > k ) // влево
    pNode = FindRecord(k,pNode->Left);
 }
 return pNode;
}
```

Введение барьера

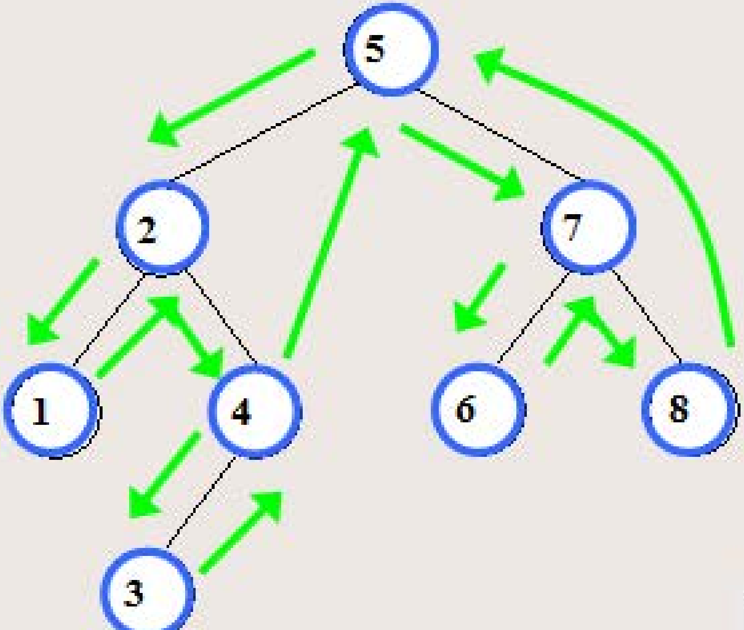


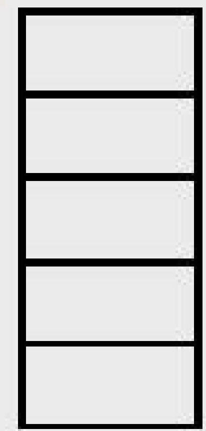
Введение барьера

```
//  поиск в дереве поиска - рекурсия
PTTreeNode
 FindRecord(TKey k,PTTreeNode pNode){
 if ( pNode->Key < k ) // вправо
   pNode = FindRecord(k,pNode->Right);
 if ( pNode->Key > k ) // влево
   pNode = FindRecord(k,pNode->Left);
 return (pNode==pBarrier) ? NULL : pNode;
}
```

☑ Схема обхода LTR, нерекурсивный вариант

☑ Неиспользованные узлы пути до текущей вершины запоминаются в стеке (*фиксатор пути*)





1	2	3	4	5	6	7	8

☑ Инициализация (Reset) – переход к крайнему левому узлу

```
PTTreeNode pNode = pCurrent = pRoot;
CurrPos = 0;
while ( pNode != NULL ) {
  St.push(pNode);
  pCurrent = pNode;
  pNode=pNode->GetLeft();
}
```

☑ Переход к следующему узлу (GoNext) от узла pCurrent (располагается на вершине стека)

- Переход к правому потомку
- Если правый потомок имеется, то переход к крайней левой вершине
- Если правого потомка нет, выборка узла из стека

☑ Переход к следующему узлу (GoNext)

```
if (!IsTabEnded() && (pCurrent != NULL)) {
  pNode = pCurrent = pCurrent->GetRight();
  St.pop(); // исключение из стека
  while ( pNode != NULL ) {
    // переход к крайней левой вершине
    St.push(pNode); // добавить в стек
    pCurrent = pNode;
    pNode=pNode->GetLeft();
  }
  if ( (pCurrent==NULL) && !St.empty() )
    pCurrent = St.top();
  CurrPos++;
}
```

- $T_{min} = 1$
- $T_{max} = \log_2 N$ (при сбалансированном дереве)
- $T_{max} = N$ (при вырожденном дереве)
- $T_{cp} = ?$

Пусть даны N различных ключей со значениями $1,\ldots,N$ и появление любого ключа равновероятно.

Пусть первый ключ равен i. Левое поддерево будет содержать $(i-1)$ узлов, правое поддерево - $(n-i)$ узел.

$a_N = \frac{1}{N}\sum_{i=1}^{N} a_N^i$ - средняя длина пути для дерева с N узлами,

где a_N^i есть средняя длина пути в дереве, в котором корень равен i.

Оценим

$$a_N^i = (a_{i-1}+1)\frac{i-1}{N} + 1\frac{1}{N} + (a_{N-i}+1)\frac{N-i}{N}$$

Тогда

$$a_N = \frac{1}{N}\sum_{i=1}^{\mathrm{N}} \mathrm{a}_N^i = \frac{1}{N}\sum_{i=1}^{N}((a_{i-1}+1)\frac{i-1}{N}+1\frac{1}{N}+(a_{N-i}+1)\frac{N-i}{N}) =$$

$$= \frac{1}{N}(N+\frac{1}{N}\sum_{i=1}^{N}[(i-1)a_{i-1}+(n-i)a_{N-i}]) =$$

$$= 1+\frac{2}{N*N}\sum_{i=1}^{N}(i-1)a_{i-1} = 1+\frac{2}{N*N}\sum_{i=1}^{N-1} i a_i$$

Из последнего выражения следует

$$(1)\quad a_N = 1 + \frac{2}{N*N}\sum_{i=1}^{N-1} i a_i = 1 + \frac{2}{N*N}(N-1)a_{N-1} + \frac{2}{N*N}\sum_{i=1}^{N-2} i a_i$$

$$(2)\quad a_{N-1} = 1 + \frac{2}{(N-1)*(N-1)}\sum_{i=1}^{N-2} i a_i$$ - умножим на $((N-1)/N)^2$

$$(3)\quad \frac{2}{N*N}\sum_{i=1}^{N-2} i a_i = \left(\frac{N-1}{N}\right)^2 (a_{N-1} - 1)$$ - подставим (3) в (1)

$$(4)\quad a_N = \frac{2}{N*N}((N^2-1)a_{N-1} + 2N - 1)$$

Отсюда можно получить (*проверяется подстановкой*)

$$a_N = 2\frac{N+1}{N}H_N - 3, \quad H_N = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$$

$$H_N = \gamma + \ln N + \frac{1}{12N^2} + \ldots \quad \text{(формула Эйлера, } \gamma \cong 0{,}577\text{)}$$

$$(N >> 1) \Rightarrow a_N \cong 2[\ln N + \gamma] - 3 = 2\ln N - c$$

Пусть $a_N^* = \log_2 N$ есть средняя длина пути для идеально сбалансированного дерева

$$\lim_{N\to\infty} \frac{a_N}{a_N^*} = \frac{2\ln N}{\log_2 N} = 2\ln 2 = 1{,}386$$