Соловьев С.Ю.

soloviev@glossary.ru

Алгоритмы и Алгоритмические языки

www.park.glossary.ru/pascal/

Лекция No. 13

2022

Напоминание

Структура данных – способ организации (однотипных) элементов данных, удобный для решения конкретной задачи.

Таблицы

Таблица – набор однотипных элементов, каждый из которых однозначно определяется значением ключа.

Соглашения:

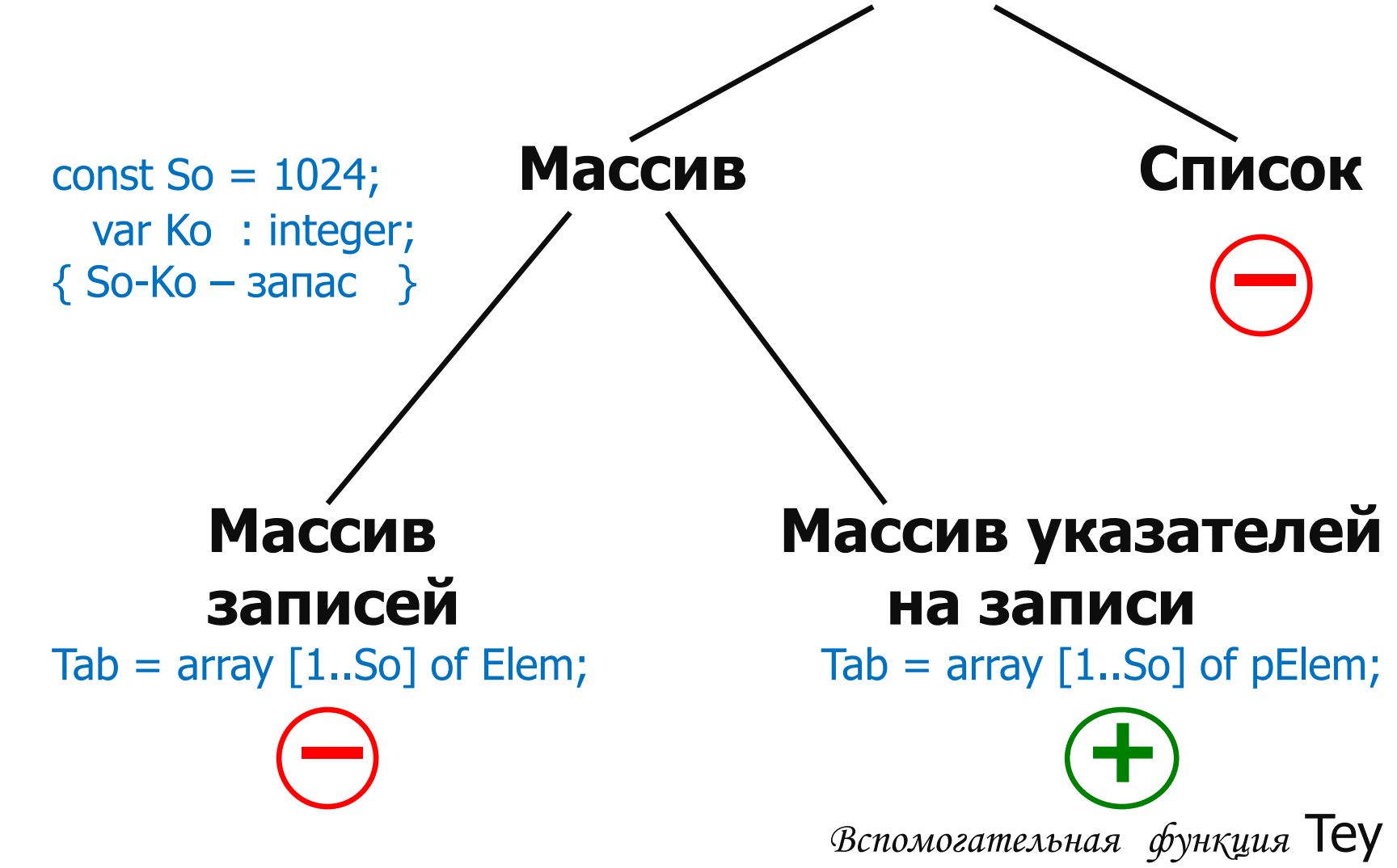
1. Ключ – целое число
2. Elem = record Key : integer; … end;
   pElem = ^Elem;

Операции:

- поиск по ключу — FIND
- включение нового элемента — FORM
- удаление элемента с заданным ключом — KILL
- породить пустую таблицу;
- проверить таблицу на наличие элементов.

Таблицы
- с неупорядоченными элементами: FindTno, FormTno, KillTno
- с упорядоченными элементами: FindTor, FormTor, KillTor

Подходы к реализации таблиц



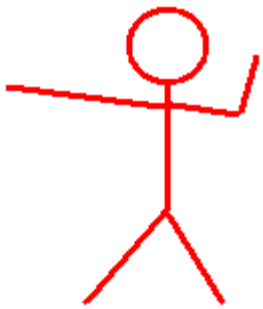

```
function Tey(N : integer) : integer;  begin   Tey:=T[N]^.Key   end;
```

Пусть

```
program … ;
const  So = 1024;
       …
type     Elem = record Key : integer; …  end;
       pElem = ^Elem;
       …
var      T : array [1..So] of pElem;
       Ko : integer;     { занято в T }
            function Tey(N : integer) : integer;
             begin    Tey:=T[N]^.Key       end;
       (*   Find  …  Form …   Kill  …  *)

begin  Ko:=0; { породить пустую таблицу }
       …
end.
```

{ ! }



Поиск в неупорядоченной таблице записи с заданным ключом

FindTno(K) = { позиция K в T ; 0, если K в T нет }

```
function FindTno(K : integer) : integer;
   var I : integer;
begin   for I:=1 to Ko do
        if Tey(I) = K then begin
           FindTno:=I;
           Exit
         end;
         FindTno:=0
end;
```

Сложность **O(n)** n = Ko

Включить в неупорядоченную таблицу запись с заданным ключом

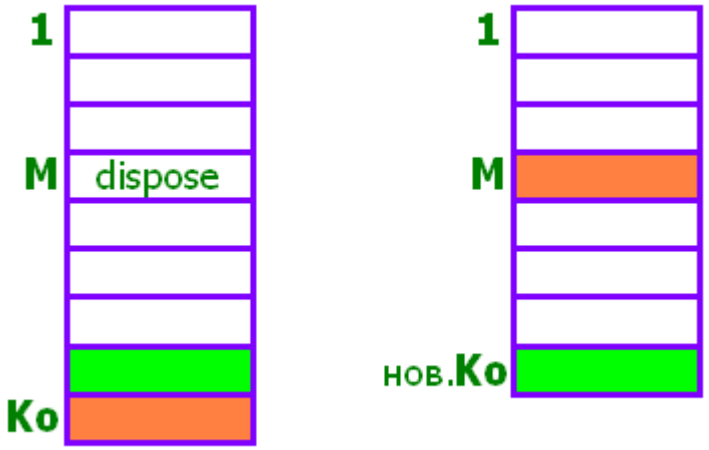
FormTno(K) = { позиция в T, если T[позиция] = nil, то K - новый ключ {A}
если T[позиция] ≠ nil, то K - не новый {Б}
0, если в T нет места {В}

```
function FormTno(K : integer) : integer;
    var M : integer;
begin   M:=FindTno(K);
 {Б}          if   0 <   M   then FormTno:=M
 {B}     else if So <= Ko then FormTno:=0
 {A}     else begin
           Ko:=Ko+1; T[Ko]:=nil; FindTno:=Ko
         end
end;
```

Сложность **O(n)**

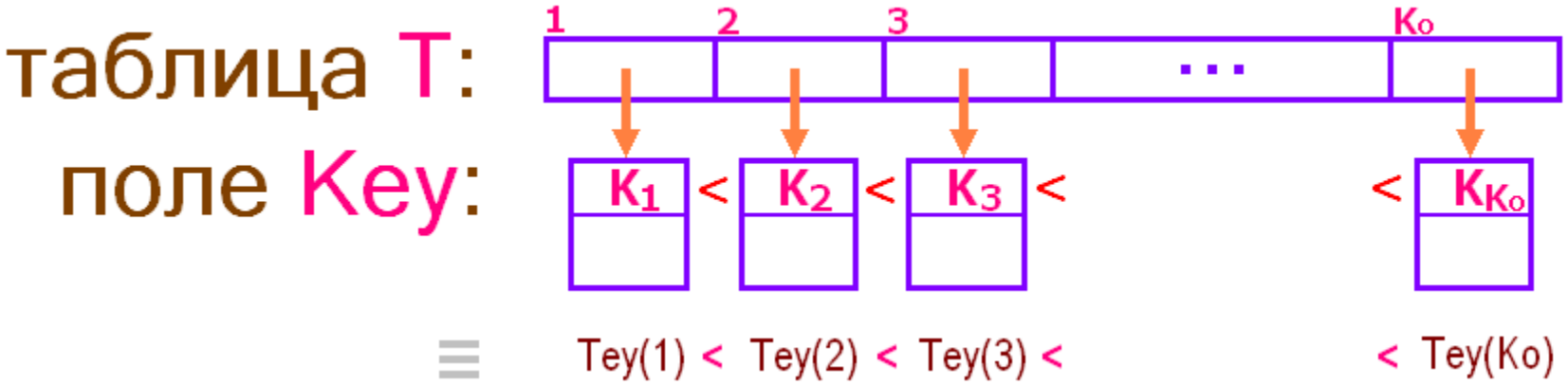
Удалить из неупорядоченной таблицы запись с заданным ключом

```
procedure KillTno(K : integer);
   var M : integer;
begin   M:=FindTno(K);
        if  0 < M  then begin
            dispose(T[M]);
            T[M]:=T[Ko];
            Ko:=Ko-1
         end
end;
```



Сложность **O(n)**

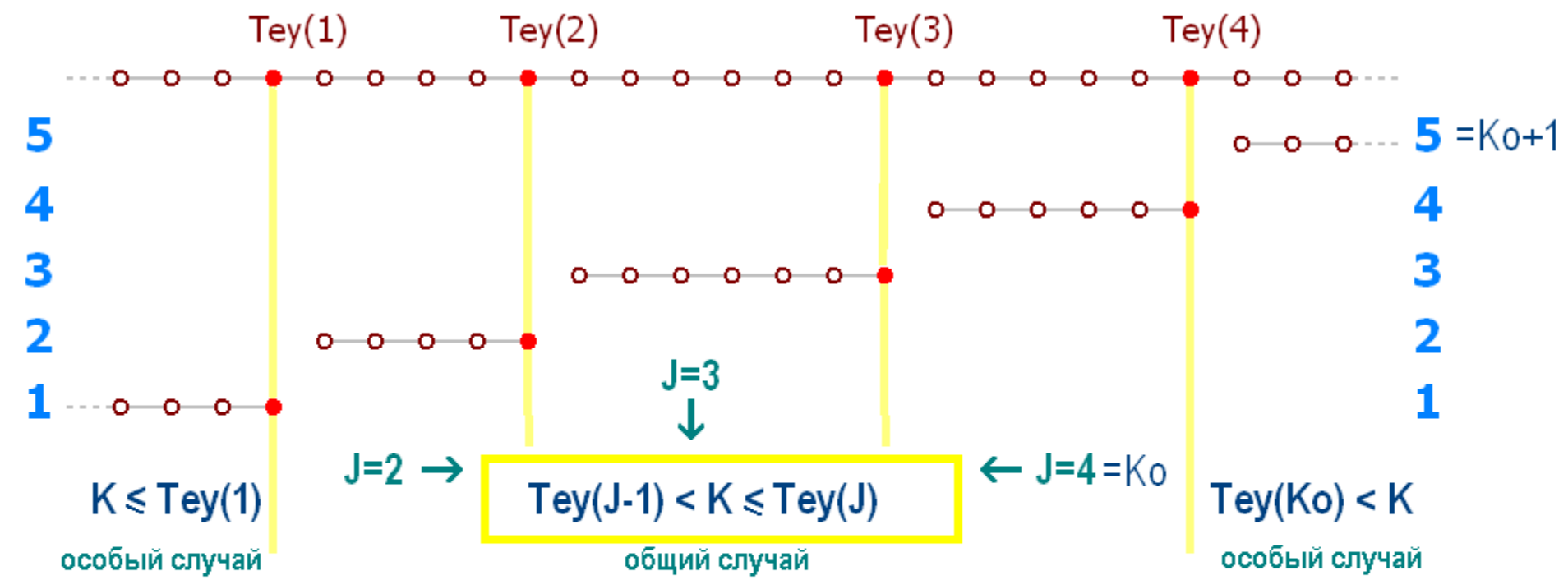
Таблицы с упорядоченными элементами

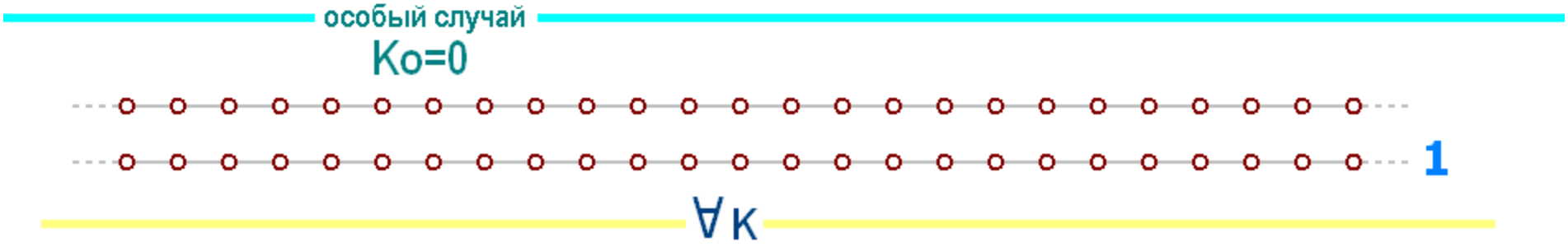


Ko упорядоченных ключей порождают разбиение множества целых чисел на Ko+1 непересекающихся подмножеств.

Вспомогательная функция BFO. По заданному ключу вычислить номер содержащего его подмножества.

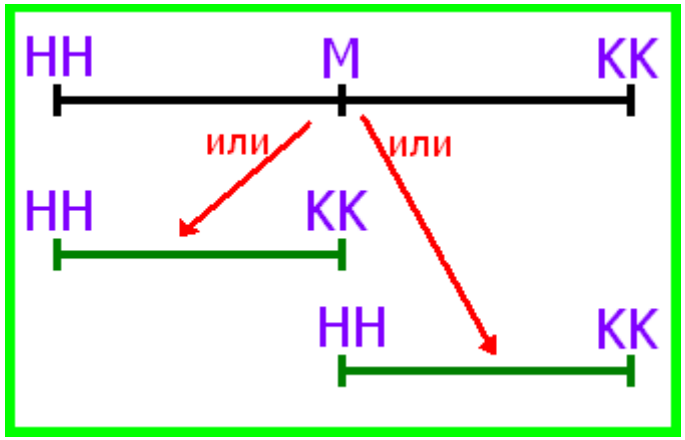
Разбиение множества целых чисел





Функция BFO

```
function  BFO(K : integer) : integer;
   var HH,KK,M : integer;
begin    if  Ko <       1  then begin BFO:=1;       Exit end;  { особый сл. }
         if  K <= Tey(1) then begin BFO:=1;       Exit end;  { особый сл. }
         if  Tey(Ko) < K  then begin BFO:=Ko+1; Exit end;  { особый сл. }
         HH:=1;
         KK:=Ko;
         {  стало: Tey(HH) < K <= Tey(KK) }
         while  HH+1 < KK do begin
            M:=(HH+KK) div 2;
            if Tey(M) < K then HH:=M
                          else   KK:=M
         end;
         BFO:=KK
end;
```



Сложность **O(log n)**

Поиск в упорядоченной таблице элемента с заданным ключом

$$\text{FindTor}(K) = \begin{cases} \text{позиция } K \text{ в } T \\ 0, \text{ если } K \text{ нет в } T \end{cases}$$

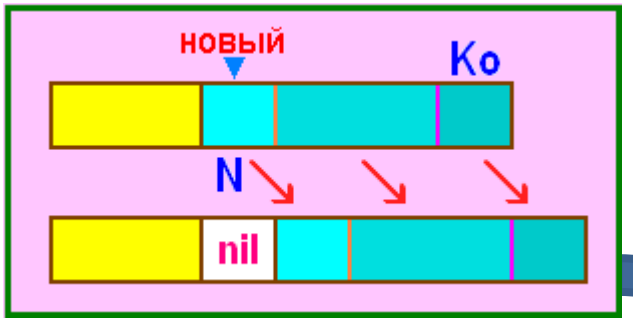
```
function FindTor(K : integer) : integer;
   var  N : integer;
begin  N:=BFO(K);
       FindTor:=0;
       if       N <= Ko then                    { проверить п/мн-во}
       if  Tey(N)  = K   then FindTor:=N
end;
```

O(log n) n = Ko

Включить в упорядоченную таблицу элемент с заданным ключом

FormTor(K) = { позиция K в T, если T[номер] = nil, то K - новый; если T[номер] ≠ nil, то K был; 0, если K в T нет }

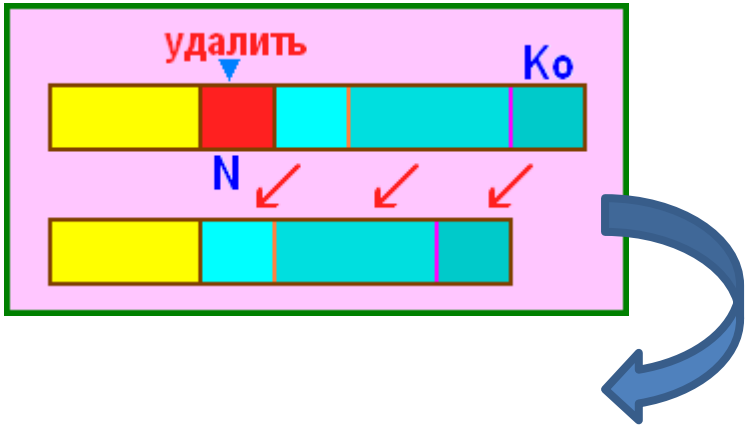
```
function FormTor(K : integer) : integer;
   var I,N : integer;
begin  N:=BFO(K);
       FormTor:=N;
       if       N <= Ko  then
       if  Tey(N)  = K   then Exit;
       if       So <= Ko then begin FormTor:=0; Exit end;
       for I:=Ko downto N do T[I+1]:=T[I];
       T[N]:=nil;
       Ko:=Ko+1
end;
```



Сложность **O(n)**

Удалить из упорядоченной таблицы элемент с заданным ключом

```
procedure KillTor(K : integer);
   var I,N : integer;
begin   N:=FindTor(K);
        if  N <> 0  then begin
              dispose(T[N]);
              Ko:=Ko-1;
              for I:=N to Ko do T[I]:=T[I+1]
            end
end;
```



Сложность **O(n)**

Замечание

```
procedure KillTor(K : integer);
   var I,N : integer;
begin   N:=FindTor(K);
        if  N <> 0  then begin
              dispose(T[N]);
              Ko:=Ko-1;
              for I:=N to Ko do T[I]:=T[I+1]
end     end;
```

$$FindTor(K) = \begin{cases} \text{позиция} \\ 0 \end{cases}$$

```
procedure KillTor(K : integer);
   var I,N : integer;
begin   N:=FindTor(K);
        if (1 <= N) and (N <= Ko) then begin
              dispose(T[N]);
              Ko:=Ko-1;
              for I:=N to Ko do T[I]:=T[I+1]
end     end;
```

Таблицы

		неупорядоченные	упорядоченные
FIND	поиск	**O(n)**	**O(log n)**
FORM	включение	**O(n)**	**O(n)**
KILL	удаление	**O(n)**	**O(n)**

Коды валют на 01.01.2020

1	008	ALL	Лек, Албания	
2	012	DZD	Алжирский динар	
…				
5	048	BHD	Бахрейнский динар	
…				
55	**398**	**RZT**	**Тенге, Казахстан**	**≈1%**
…				
92	**643**	**RUB**	**Российский рубль**	**≈90%**
…				
114	826	GBP	Фунт стерлингов	
115	834	TZS	Танзанийский шиллинг	
116	**840**	**USD**	**Доллар США**	**≈3%**
…				
126	**933**	**BYN**	**Белорусский рубль**	**≈1%**
…				
149	**978**	**EUR**	**Евро**	**≈4%**
…				
153	986	BRL	Бразильский реал	

$log_2(153) \approx 7.2574$

Хеш–таблицы

Хеш-таблица – таблица, в которой доступ к записям определяется хеш-функцией.

Номера строк в хеш-таблице: 0, 1, …, So-1.

Хеш-функция (Функция расстановки) – функция, которая вычисляет **п**о заданному ключу

номер позиции в таблице,

начиная с которого следует искать/размещать запись.

$$h : \{ \text{Ключи} \} \rightarrow \{ 0,1, \ldots, So-1 \}$$

h(ключ) = позиция

Примеры хеш–таблиц

```
{ K ⩾ 0 }
function hash(K : integer) : integer;
begin    hash:=K  mod  So     end;
```

So = 8

0 **0**	8 **0**	16 **0**	24 **0**	32 **0**	40 **0**	48 **0**	56 **0**
1 **1**	9 **1**	17 **1**	25 **1**	33 **1**	41 **1**	49 **1**	57 **1**
2 **2**	10 **2**	18 **2**	26 **2**	34 **2**	42 **2**	50 **2**	58 **2**
3 **3**	11 **3**	19 **3**	27 **3**	35 **3**	43 **3**	51 **3**	59 **3**
4 **4**	12 **4**	20 **4**	28 **4**	36 **4**	44 **4**	52 **4**	60 **4**
5 **5**	13 **5**	21 **5**	29 **5**	37 **5**	45 **5**	53 **5**	61 **5**
6 **6**	14 **6**	22 **6**	30 **6**	38 **6**	46 **6**	54 **6**	62 **6**
7 **7**	15 **7**	23 **7**	31 **7**	39 **7**	47 **7**	55 **7**	63 **7**

$$hash(K) = sign(K) \cdot \left[So \cdot \left\{ \frac{\sqrt{5}-1}{2} \cdot |K| \right\} \right]$$

So = 8

0 **0**	8 **7**	16 **7**	24 **6**	32 **6**	40 **5**	48 **5**	56 **4**
1 **4**	9 **4**	17 **4**	25 **3**	33 **3**	41 **2**	49 **2**	57 **1**
2 **1**	10 **1**	18 **0**	26 **0**	34 **0**	42 **7**	50 **7**	58 **6**
3 **6**	11 **6**	19 **5**	27 **5**	35 **5**	43 **4**	51 **4**	59 **3**
4 **3**	12 **3**	20 **2**	28 **2**	36 **1**	44 **1**	52 **1**	60 **0**
5 **0**	13 **0**	21 **7**	29 **7**	37 **6**	45 **6**	53 **6**	61 **5**
6 **5**	14 **5**	22 **4**	30 **4**	38 **3**	46 **3**	54 **2**	62 **2**
7 **2**	15 **2**	23 **1**	31 **1**	39 **0**	47 **0**	55 **7**	63 **7**

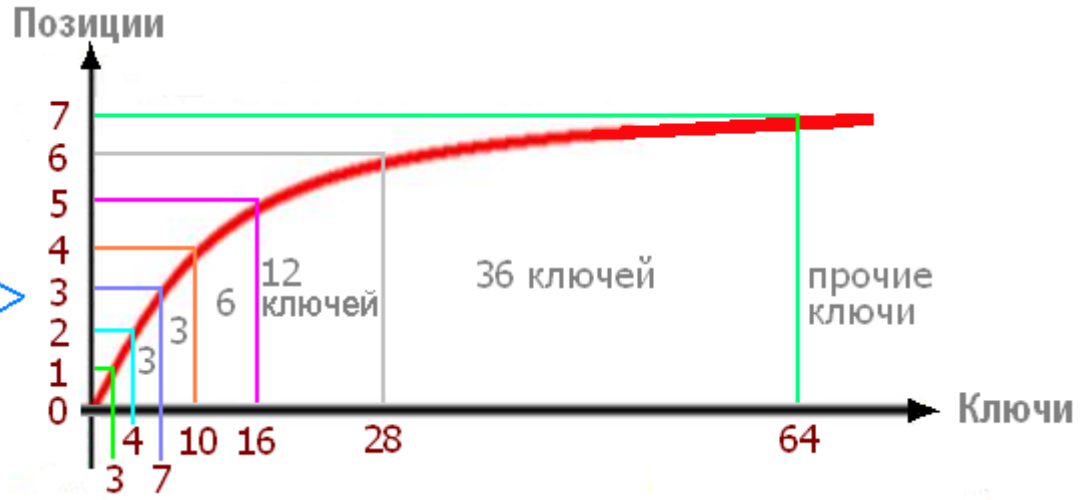
Коллизия – наличие в хеш-таблице записей с ключами K1 и K2 такими, что hash(K1) = hash(K2), но K1 ≠ K2.

Примеры хеш–таблиц

$0 < K$

$$hash(K) = \left[So \frac{K-1}{K+So} \right]$$

So = 8



Хеш-функция должна
- допускать эффективную реализацию и
- минимизировать число коллизий.

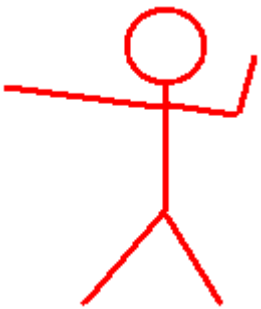
Методы разрешения коллизий

Открытое хеширование

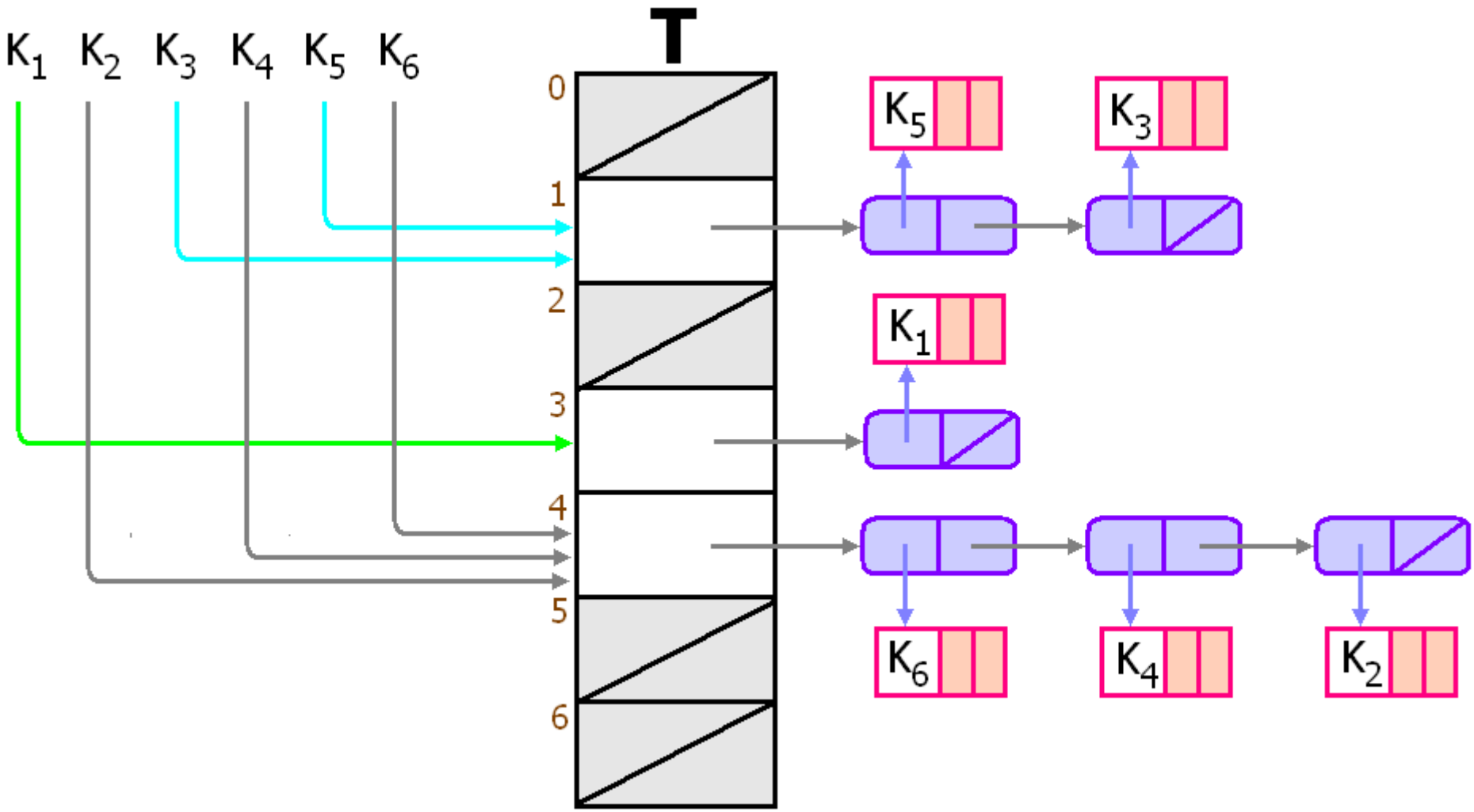
Закрытое хеширование

Открытое хеширование.
Пусть

```
program … ;
const  So = 1024;
       Sx = 1023;            { Sx = So-1 }
       …
type     Elem = record Key : integer; …  end;
        pElem = ^Elem;
        pCamel = ^Camel;
         Camel = record  CML : pCamel;  BAG : pElem end;
var      T : array [0..Sx] of pCamel;
       function hash(K : integer) : integer;  begin . . . end;
       (*    FindHTC      FormHTC      KillHTC    *)
begin   { породить пустую таблицу }
       for I:=0 to Sx do T[I]:=nil;
       …
end.
```

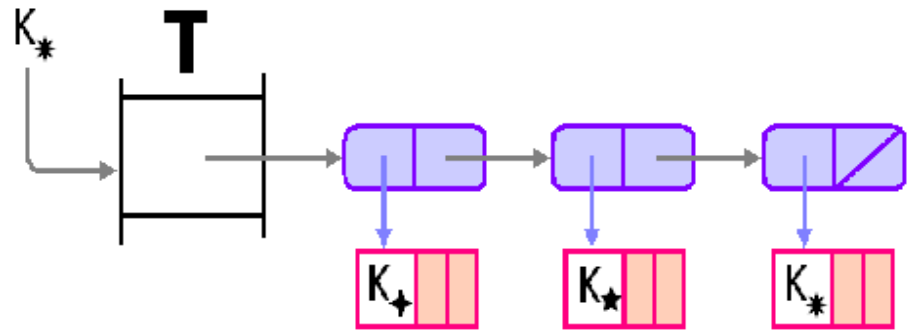


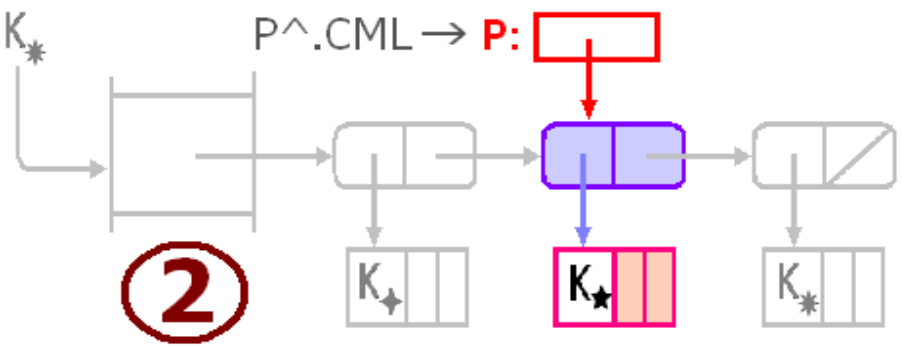
Подход

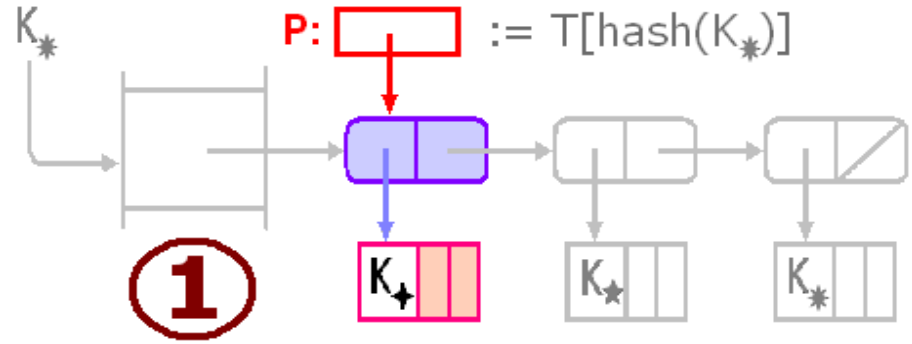


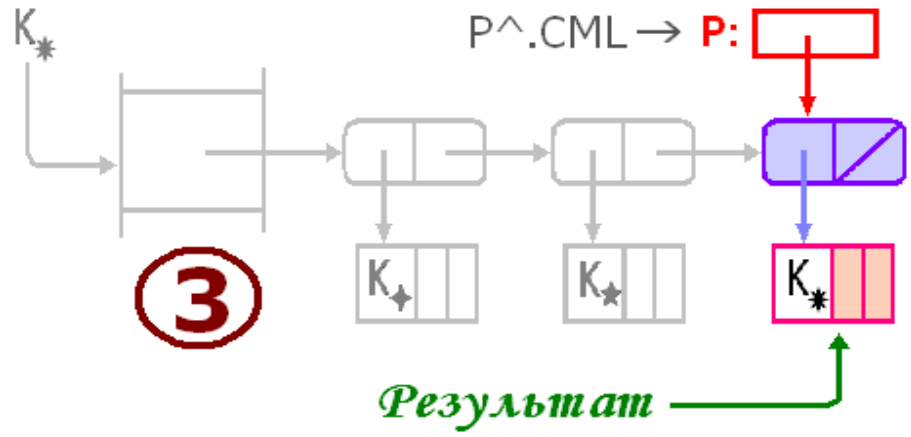
Поиск в хеш-таблице записи с заданным ключом / Подход

$$FindHTC(K) = \begin{cases} pElem & \textit{с ключом } K \\ nil & \textit{иначе} \end{cases}$$









Поиск в хеш-таблице записи с заданным ключом

$$FindHTC(K) = \begin{cases} pElem & \textit{с ключом } K \\ nil & \textit{иначе} \end{cases}$$

```
function FindHTC(K : integer) : pElem;
    var  P : pCamel;
begin  P:=T[hash(K)];
       while P <> nil do
       if      P^.BAG^.Key = K
       then begin  FindHTC:=P^.BAG;  Exit  end
       else                    P:=P^.CML;
                        FindHTC:=nil
end;
```

Включить в хеш-таблицу новую запись с заданным ключом

на существовавшую ранее (Old=true)

FormHTC(K,Old) – указатель или запись с ключом K.

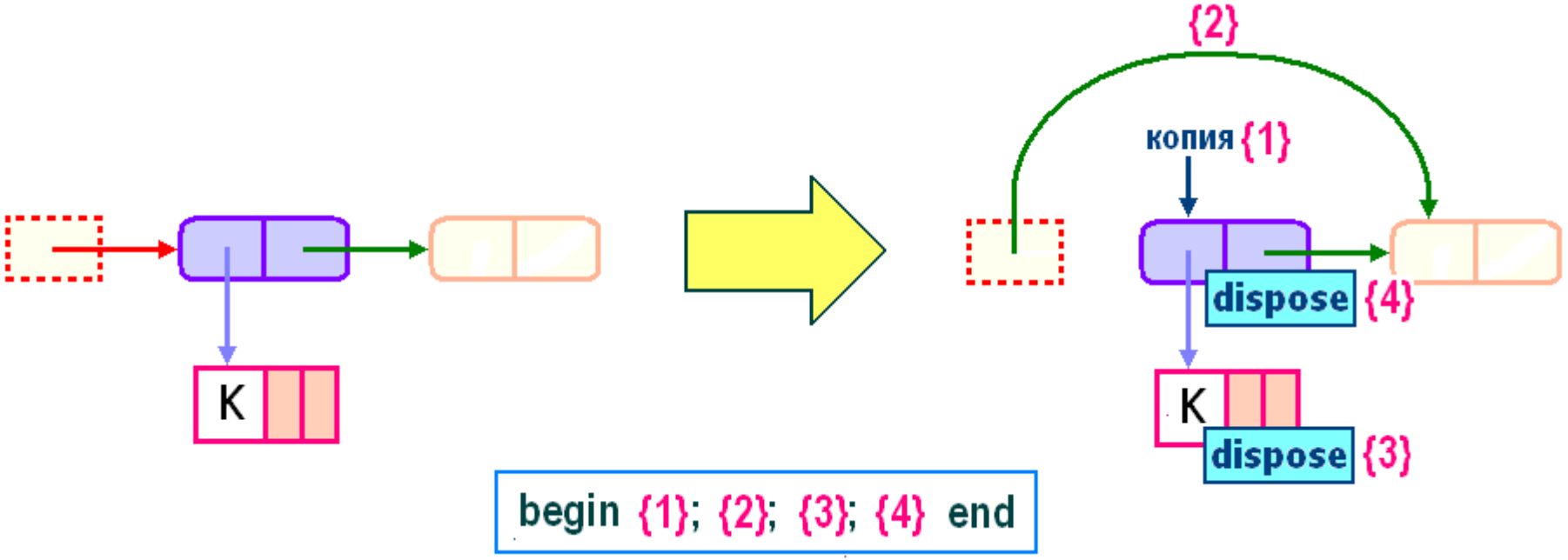
на вновь созданную (Old= false)

```
function FormHTC(K : integer; var Old : boolean) : pElem;
   var   H : integer;
         C  : pCamel;
         E   : pElem;
begin                        E:=FindHTC(K);
                  Old:=(E <> nil);
          if  not Old  then begin
              new(E);  E^.Key:=K;                  { Создать и заполнить}
                       H:=hash(K);
               C:=T[H];
              new(T[H]);   T[H]^.CML:=C;           { Создать и }
                           T[H]^.BAG:=E            { заполнить }
          end;
          FormHTC:=E                    end;
```

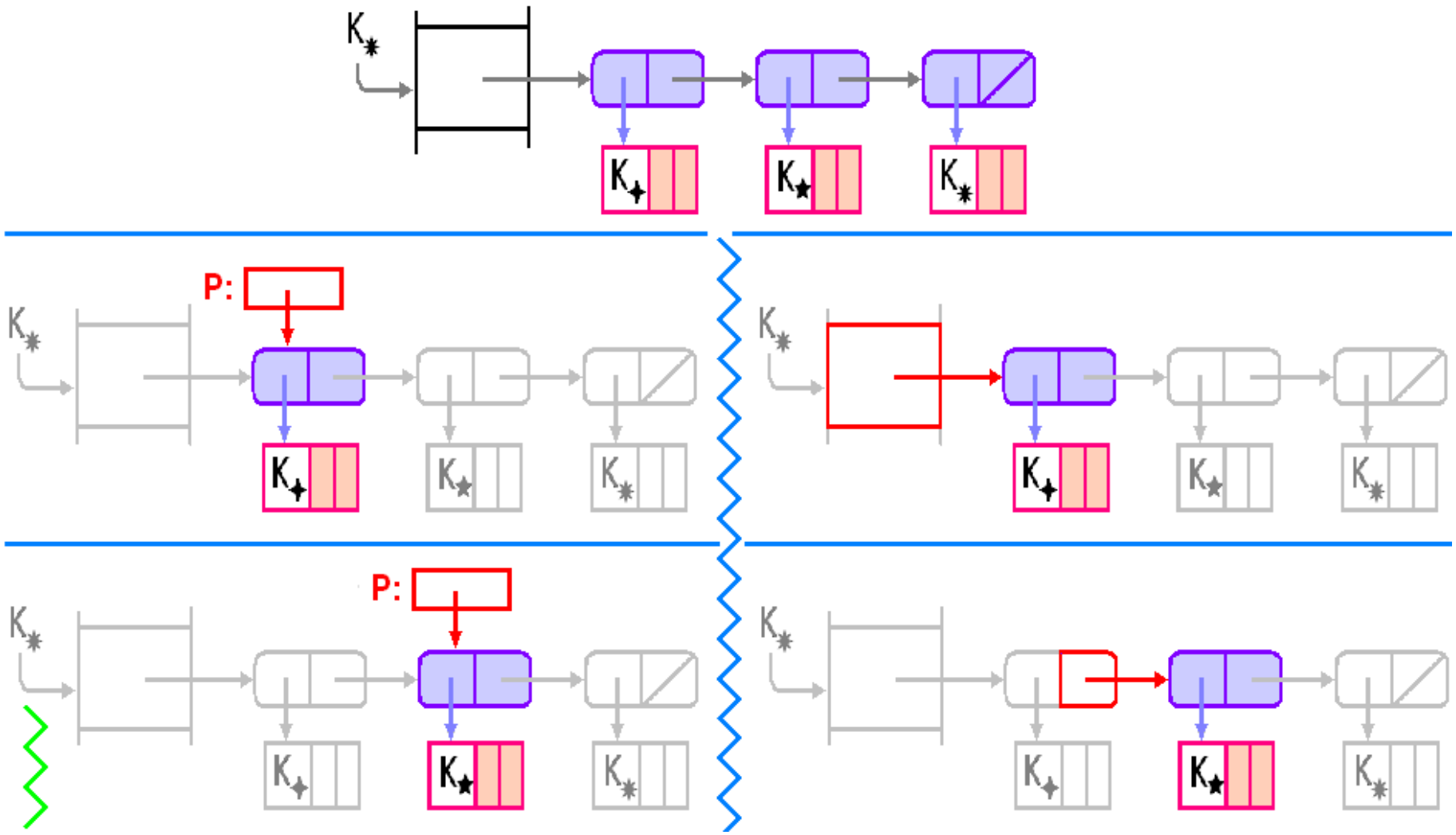
Удалить из хеш-таблицы запись с заданным ключом / Введение

KillHTC(K)

Операция непосредственного удаления записи



FindHTC vs. KillHTC



Удалить из хеш-таблицы запись с заданным ключом

```
procedure KillHTC(K : integer);
    var  H : integer;
         V : pCamel;

begin                H:=hash(K);
          V:=T[H];
        KILL(T[H]);
        while  V <> nil  do begin
            KILL(V^.CML);
            if      V <> nil
            then  V:=V^.CML
        end
end;
```

```
procedure KILL(var C : pCamel);
   var  U : pCamel;
begin    if     C = nil    then Exit;
         if     C^.BAG^.Key <> K
         then Exit;
 {1}     U:=C;
 {2}          C:=C^.CML;
 {3}     dispose(U^.BAG);
 {4}     dispose(U);
         V:=nil                { стоп }
end;
```

Удалить из хеш-таблицы запись с заданным ключом / Рекурсивная реализация

```
procedure RillHTC(K : integer);
    procedure RILL(var C : pCamel);
       var  U : pCamel;
    begin    if   C = nil    then Exit;
             if   C^.BAG^.Key = K    then begin
      {1}          U:=C;
      {2}               C:=C^.CML;
      {3}          dispose(U^.BAG);
      {4}          dispose(U);
                   Exit
             end;
             RILL(C^.CML)
    end;
begin  RILL(T[hash(K)])    end;
```

Обзор лекции No.13

- Таблицы
- Таблицы с неупорядоченными элементами
- Операции поиска, вставки и удаления элементов в таблицах с неупорядоченными элементами
- Таблицы с упорядоченными элементами
- Операции поиска, вставки и удаления элементов в таблицах с неупорядоченными элементами
- Оценки сложности операций поиска, вставки и удаления элементов в таблицах с упорядоченными и неупорядоченными элементами
- Хеш-таблицы
- Хеш-функции
- Открытое хеширование
- Операции поиска, вставки и удаления элементов при открытом хешировании