Соловьев С.Ю.

soloviev@glossary.ru

Алгоритмы и Алгоритмические языки

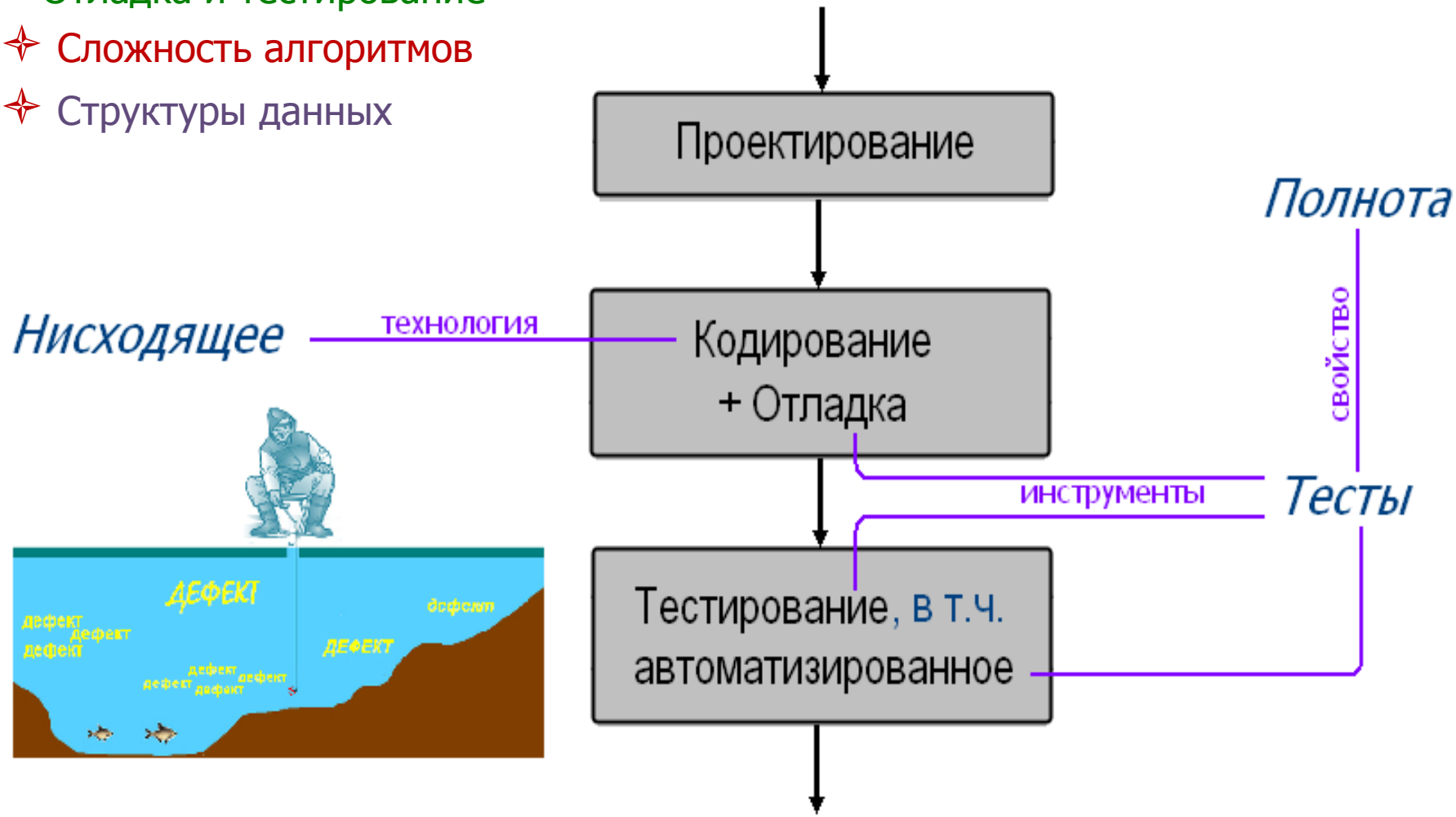
www.park.glossary.ru/pascal/

Лекция No. 12

2022

Напоминание

- ✓ Нисходящее программирование
- ✓ Отладка и тестирование
- ✧ Сложность алгоритмов
- ✧ Структуры данных



Сложность алгоритмов

Асимптотический анализ > Асимптотические обозначения

$$\frac{2^n}{\sqrt{2}}\left(\prod_{i=1}^{n-1}\cos\frac{\pi i}{4n}\right)^2 \quad \textit{почти} \quad E^n \qquad (E = 1.604956...)$$

Пусть $f, g : N \to N$

$f(n) = \mathcal{O}(g(n))$ *если* $\exists(C,L)$: $f(n) < Cg(n)$ $\forall n > L$

$f(n) = \Omega(g(n))$ *если* $\exists(c,L)$: $cg(n) < f(n)$ $\forall n > L$

$f(n) = \Theta(g(n))$ *если* $\exists(c,C,L)$: $cg(n) < f(n) < Cg(n)$ $\forall n > L$

$f(n) = o(g(n))$ *если* $\forall c\ \exists\ L$: $f(n) < cg(n)$ $\forall n > L$

$f(n) = \omega(g(n))$ *если* $\forall C\ \exists\ L$: $f(n) > Cg(n)$ $\forall n > L$

Сложность алгоритмов

Пусть **A** – алгоритм,
n – параметр, характеризующий объем исх.данных и зависящий от конкретной задачи.

Обозначим

T(n) = max{ время выполнения **A** для исх.данных объема **n** }

~ к-во операций

Рассмотрим $\mathbf{f : N \to N}$. Говорят: при $\mathbf{n \to \infty}$

алгоритм **A** имеет сложность **O(f(n))**, "величина порядка f(n)"

если $\left|\frac{T(n)}{f(n)}\right| < C$ при $n \to \infty$ для нек. $C > 0$.

Модификации: max → среднее; время → память M(n).

Сложность алгоритмов / Пример

$$\vec{a} = (a_1, \ldots, a_n)$$
$$\vec{b} = (b_1, \ldots, b_n)$$

$$(\vec{a}, \vec{b}) = \sum_{i=1}^{n} a_i \cdot b_i$$

Пусть
```
const   DIM = 20;
type  BEKTOP = array [1..DIM] of real;
```

```
function DotProduct(var A,B : BEKTOP) : real;
   var I : integer;
       S : real;
begin  S:=0;
       for I:=1 to DIM do S:=S+A[I]*B[I];
       DotProduct:=S
end;
```

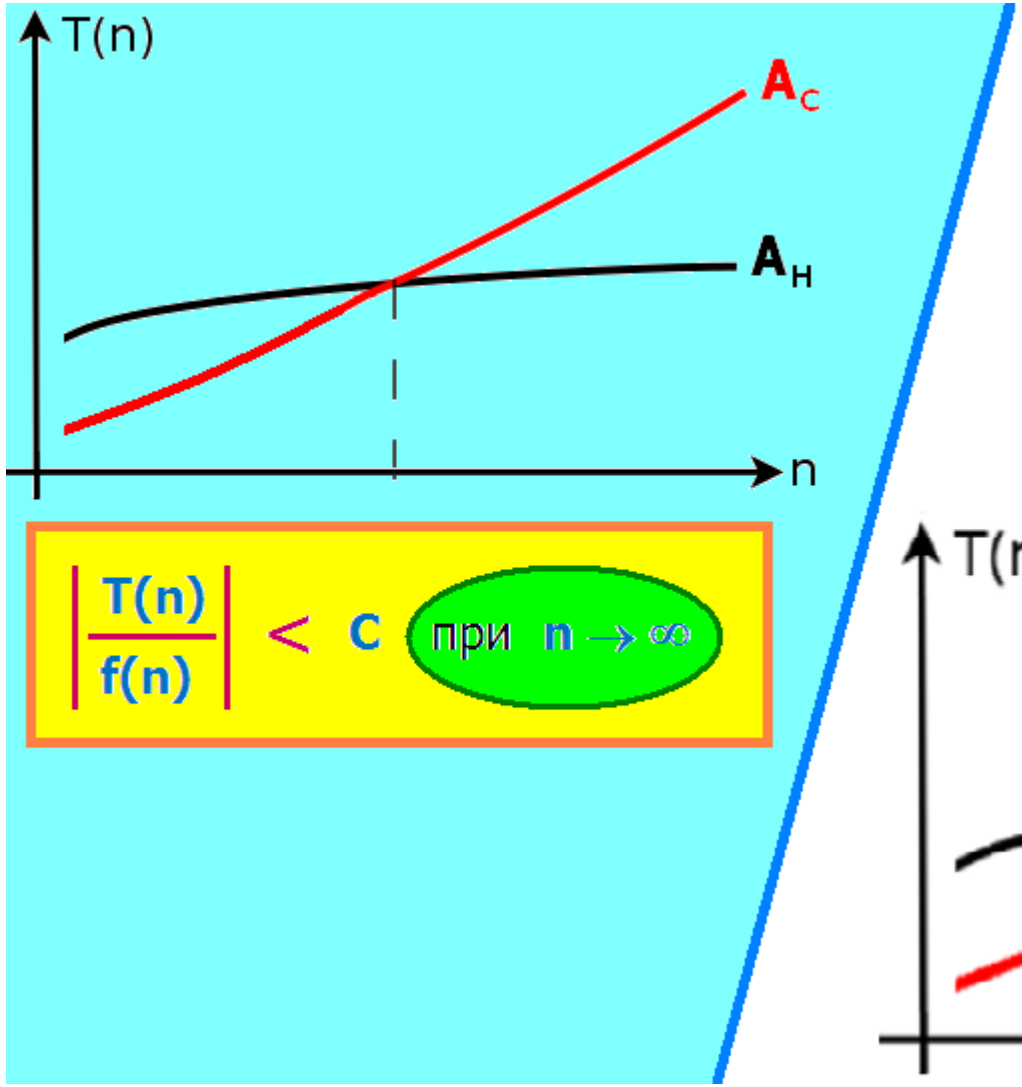
Сложность **O(n)** n = DIM

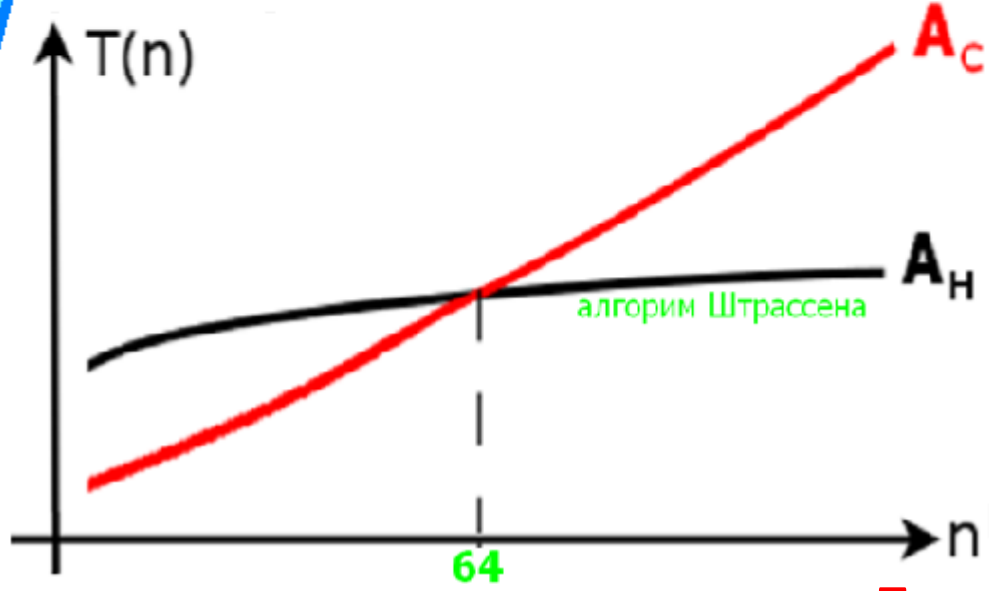
Сложность алгоритмов / Открытия

A_c – "старый" алгоритм	задача	$A_н$ – "новый" алгоритм	
	поиск		
$O(n)$	AVL/1962	$O(\log n)$	
	выч.мат.'		
$O(n^2)$	БПФ	$O(n \log n)$	
	лин.прогр.		
$O(2^n)$	Хачиян/1979	$O(n^8)$	**P vs. NP**
	умножение		
$O(n^2)$	Карацуба/1960	$O(n^{1.585})$	$\log_2 3$
	умножение матриц		
$O(n^3)$	Штрассен/1969	$O(n^{2.81})$	$\log_2 7$

Сложность алгоритмов / Соотношения

A_c – "старый" алгоритм $A_н$ – "новый" алгоритм





Структуры данных

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

Типичная структура данных поддерживает:

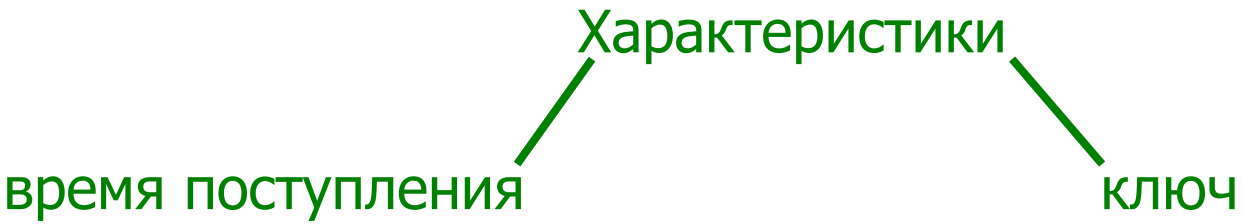
-1- включение в структуру нового элемента с заданными характеристиками;

-2- удаление из структуры элемента с заданными характеристиками;

-3- поиск в структуре элемента с заданными характеристиками;

-4- порождение пустой структуры;

-5- проверка структуры на наличие/отсутствие элементов.



Структуры данных

строительный материал

- массивы
- ссылки + записи

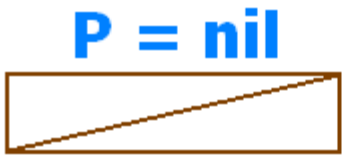
Линейные списки

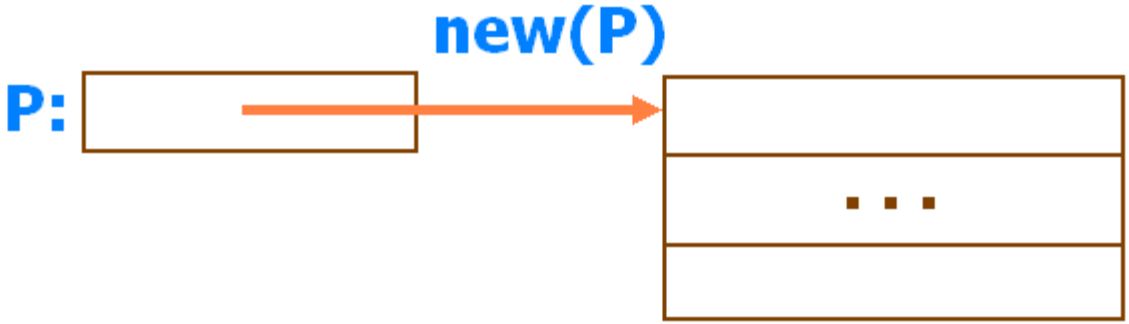
Деревья, графы

виды

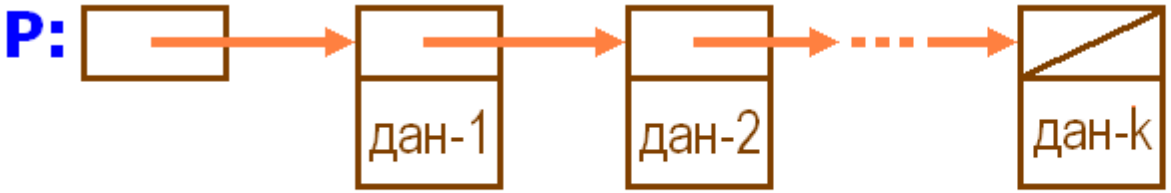
- очереди
- стеки
- таблицы
- деревья поиска

Пусть **P** – указатель на запись record … end;

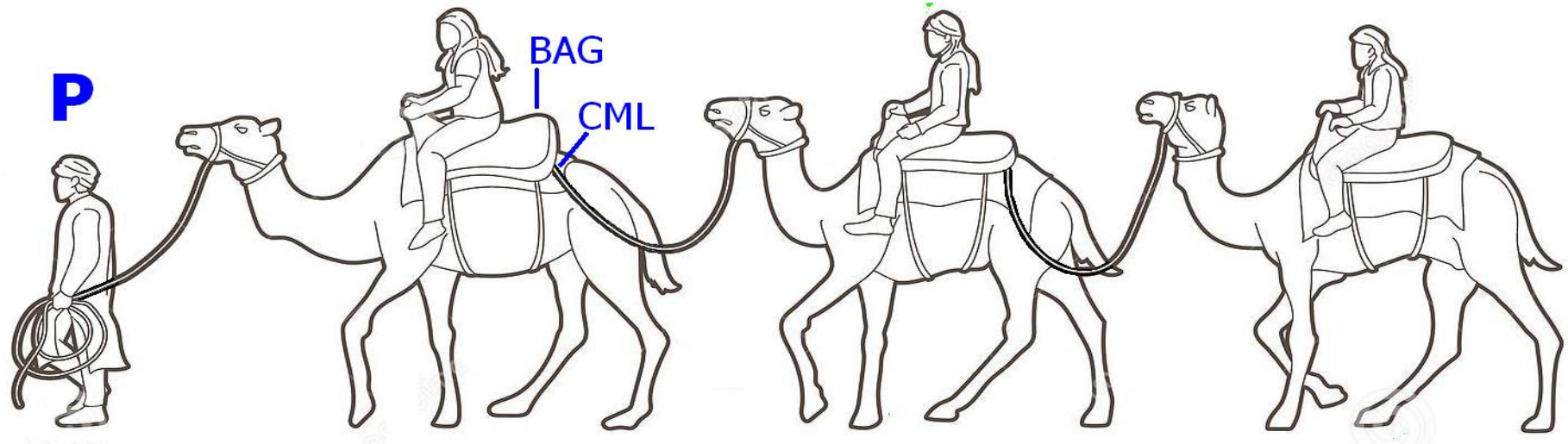




Однонаправленные списки



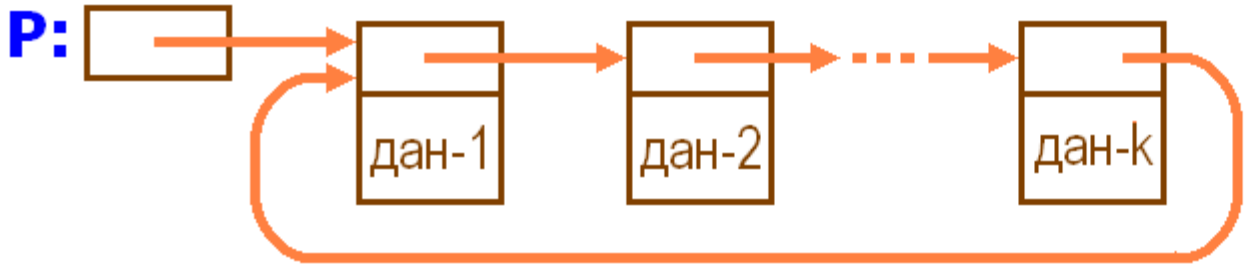

```
type Camel = record
               CML : ^Camel;
               BAG : ...   end;
var P : ^Camel;
```



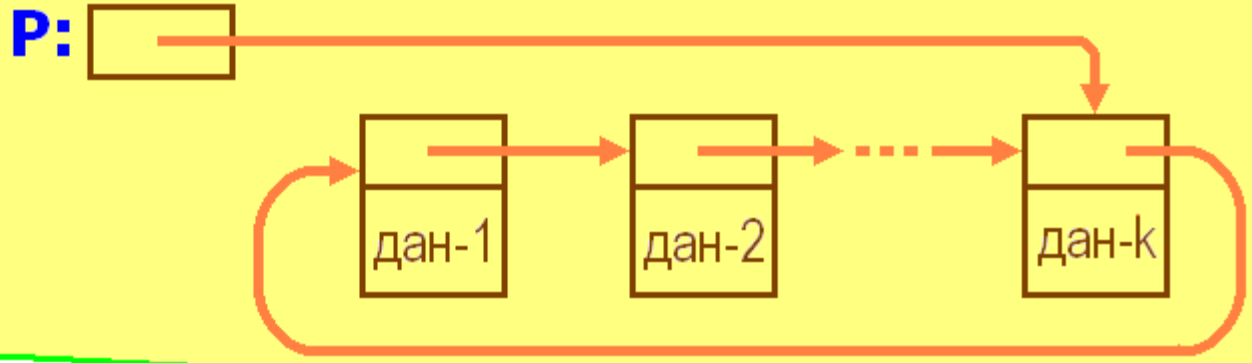
Однонаправленный список с заголовком:

Виды списков

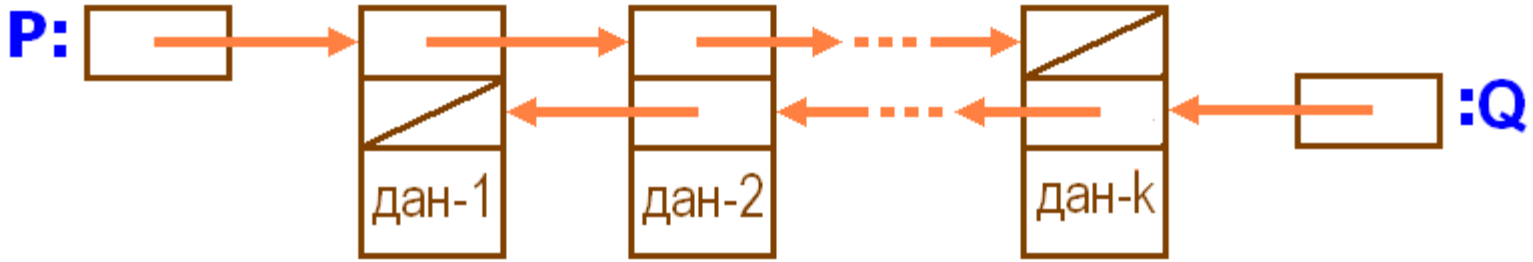
Циклический список:



Предпочтительно:

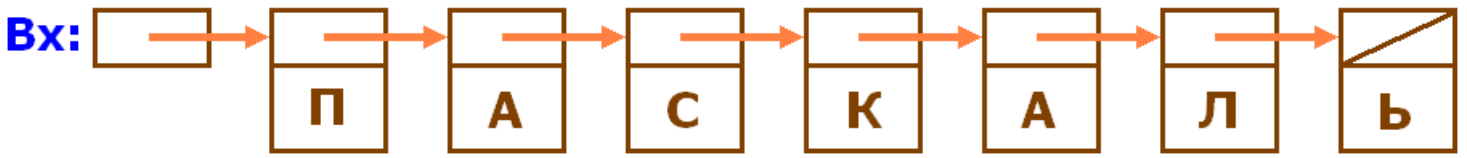


Двунаправленный список:



Пример использования списка – 1

```
program Oct03;
type pOneChar = ^OneChar;
        OneChar = record NEXT : pOneChar;
                         LETT : char;
                  end;
var  Bx,P,Q : pOneChar;
       S  : packed array [1..7] of char;
       I  : integer;
begin  S:='ПАСКАЛЬ';
       Bx:=nil;
       for I:=7 downto 1 do begin
          new(P); P^.NEXT:=Bx;
                  P^.LETT:=S[I];
          Bx:=P
       end;
end.
```



Пример использования списка – 2

```
program Oct03;
type pOneChar = ^OneChar;
        OneChar = record NEXT : pOneChar;
                         LETT : char;
                  end;
var  Bx,P,Q : pOneChar;
       S  : packed array [1..7] of char;
       I  : integer;
begin  S:='ПАСКАЛЬ';
       Bx:=nil;
       for I:=7 downto 1 do begin
          new(P); P^.NEXT:=Bx;
                  P^.LETT:=S[I];
          Bx:=P
       end;
```

```
P:=Bx;
while P <> nil do begin
     write(P^.LETT,'+');
     P:=P^.NEXT
end;
writeln;
```

```
end.
```

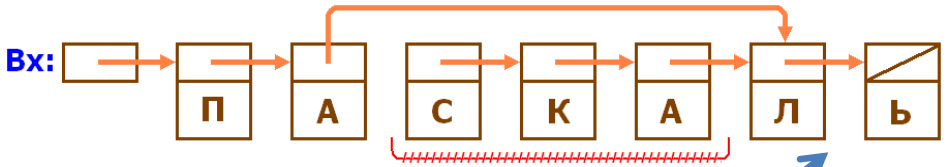
П+А+С+К+А+Л+Ь+

Пример использования списка – 3

```
program Oct03;
type pOneChar = ^OneChar;
       OneChar = record NEXT : pOneChar;
                        LETT : char;
                  end;
var  Bx,P,Q : pOneChar;
          S  : packed array [1..7] of char;
          I   : integer;
```

```
function FIND(A : char) : pOneChar;
   var P : pOneChar;
begin   P:=Bx;
        while P <> nil do
        if P^.LETT = A then begin
            FIND:=P;
            Exit
        end              else P:=P^.NEXT;
        FIND:=nil                      end;
```

```
begin   S:='ПАСКАЛЬ';
        Bx:=nil;
         for I:=7 downto 1 do begin
            new(P); P^.NEXT:=Bx;
                    P^.LETT:=S[I];
            Bx:=P
         end;
```




```
P:=FIND('А');
P^.NEXT:=FIND('Л');

if  FIND('К') = nil
then  writeln('нет')
else   writeln('да');
```

```
end.
```

НЕТ

Очереди / FIFO

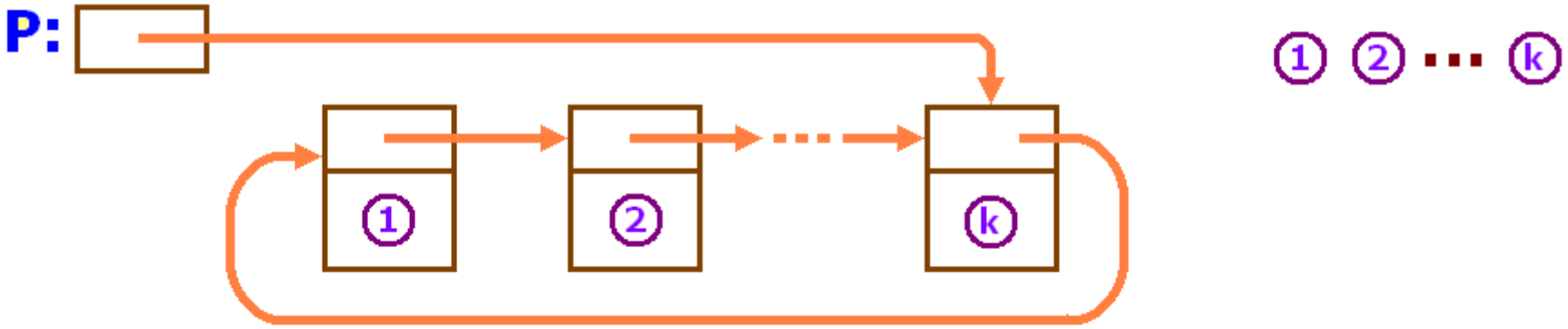
Очередь – последовательность (однотипных) элементов данных, в которой:

- включение элементов выполняется на одном конце последовательности; а
- выбор элементов – на другом конце.

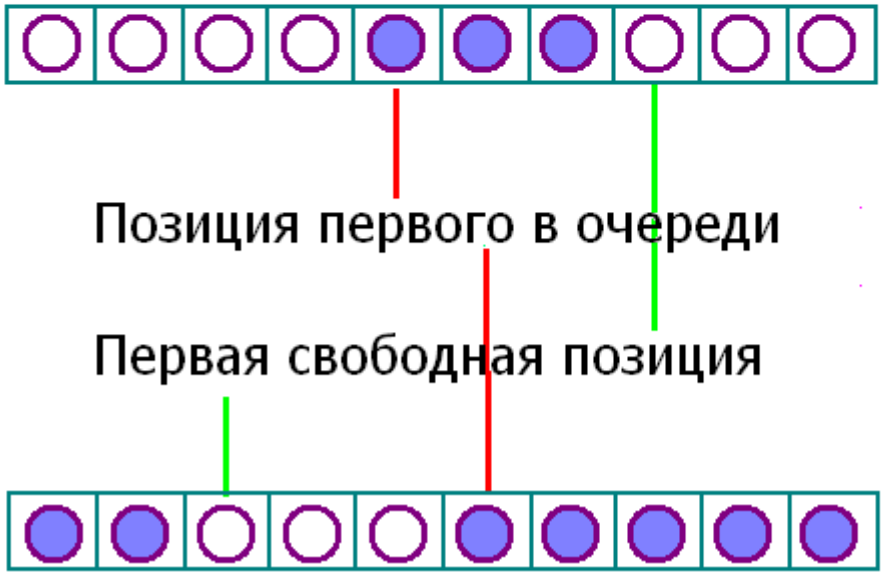
Основные операции:

- включить элемент в очередь — procedure PUSH
- выбрать элемент из очереди — procedure POP
- породить пустую очередь;
- проверить очередь на наличие элементов.

Очереди / реализация



Буфер клавиатуры:



Стек/Магазин/ LIFO

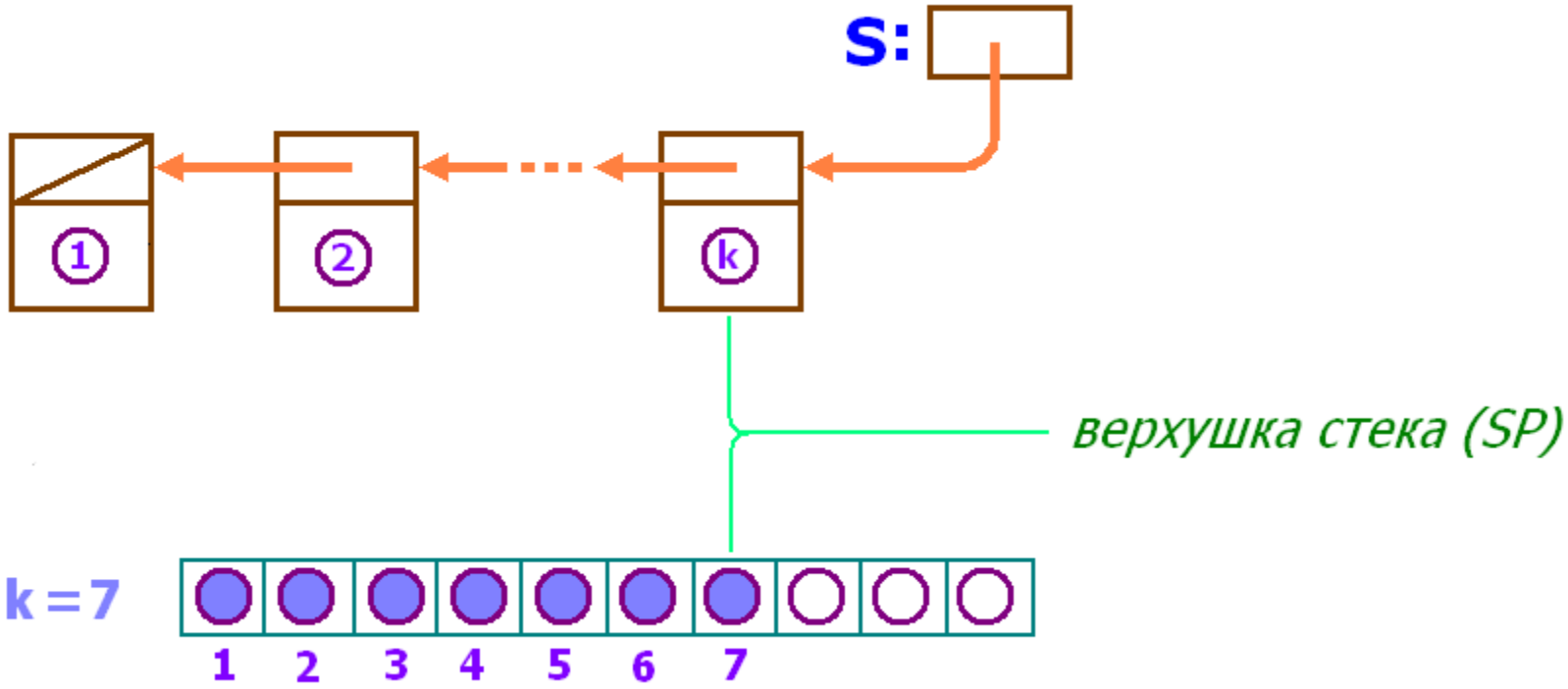
Стек – последовательность (однотипных) элементов данных, в которой:

- включение и выбор элементов выполняется на одном конце последовательности.

Основные операции:

- включить элемент в стек — procedure PUSH
- выбрать элемент из стека — function POP
- породить пустой стек;
- проверить стек на наличие элементов.

Стеки / реализация



Пример стека

Проверить строку на баланс скобок: **()**, **[]**, **{}**: (b)*{1-2*3}, но {[k}

Пусть type CTP = packed array [1..80] of char;

```
function BALS(var S : CTP) : boolean;
   var I,SP : integer;
          SK : CTP;
begin    BALS:=false;
          SP:=0;                {породить пустой стек}
          for I:=1 to 80 do
            case S[I] of
         '(' : PUSH(')');
         '[' : PUSH(']');
         '{' : PUSH('}');
'}', ']', ')' :       if   Empty      then Exit
               else if POP <> S[I] then Exit
             end;
          BALS:=Empty
end;
```

```
procedure PUSH(A : char);
begin       SP:=SP+1;
          SK[SP]:=A
end;
function POP : char;
begin   POP:=SK[SP];
                    SP:=SP-1
end;
function Empty : boolean;
begin     Empty:=(SP <= 0)
end;
```

Обзор лекции No.12

Сложность алгоритмов

Структуры данных

Списки, деревья, графы

Однонаправленные списки

Циклические списки

Двунаправленные списки

Очереди

Стеки

--- Конец лекции No. 12 ---