Рекурсивные алгоритмы.

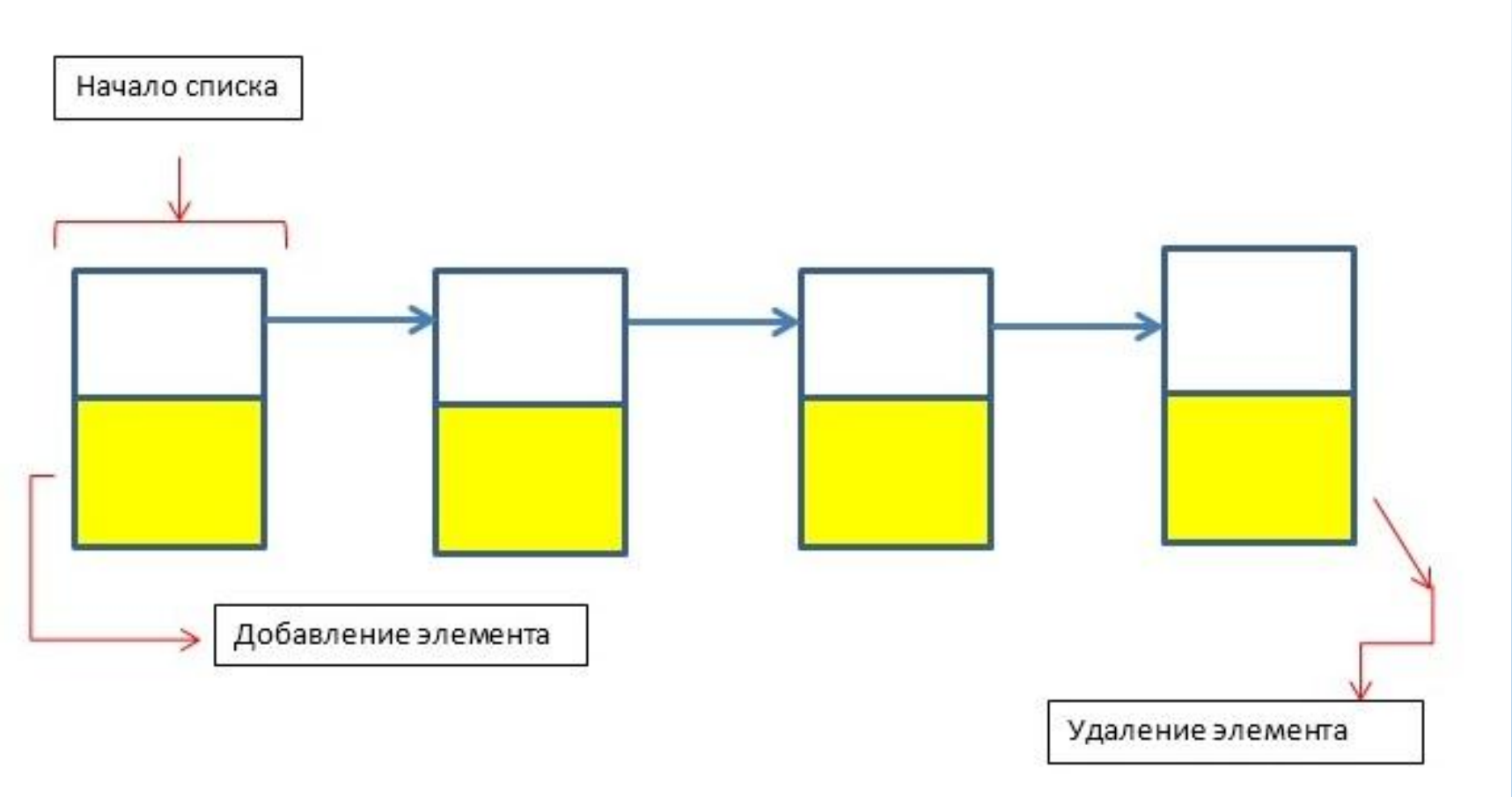
Очередь

Учитель Информатики и ИКТ, ГБОУ СОШ № 1056, Маркова А.И.

Данная структурная единица используется очень широко: при моделировании, при диспетчеризации задач в операционной системе, при буферизованном вводе-выводе и т.д.

Очередь, как и стек, является частным случаем списка. Доступ возможен только к первому и последнему элементам очереди.

<u>Очередь</u> – динамическая структура данных, добавление элементов в которую выполняется в один конец, а выборка - из другого конца.



Элемент, который был добавлен в очередь первым, первым достигнет её начала. Обозначение такой структуры данных – FIFO, т.е. First In First Out – «первый вошёл – первый вышел».

Набор операций для очереди:

- создание очереди (create_que);
- включение элемента в очередь (push_que);
- исключение элемента из очереди (popque);
- проверка пустоты очереди (TestEmptyQue);
- очистка очереди без освобождения памяти для нее (initque);
- удаление очереди (delete_que).

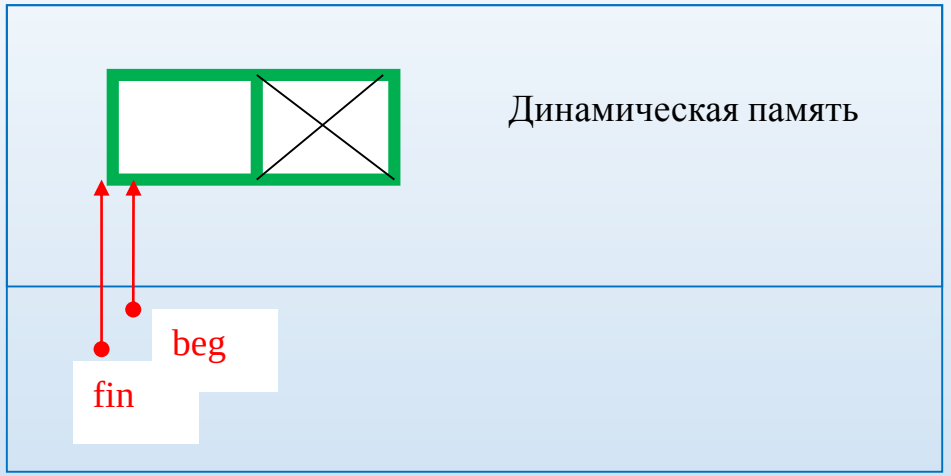
Для работы с очередью используются указатели на её начало и конец, а так же вспомогательный указатель:

```
var beg, fin, p: pnod;
```

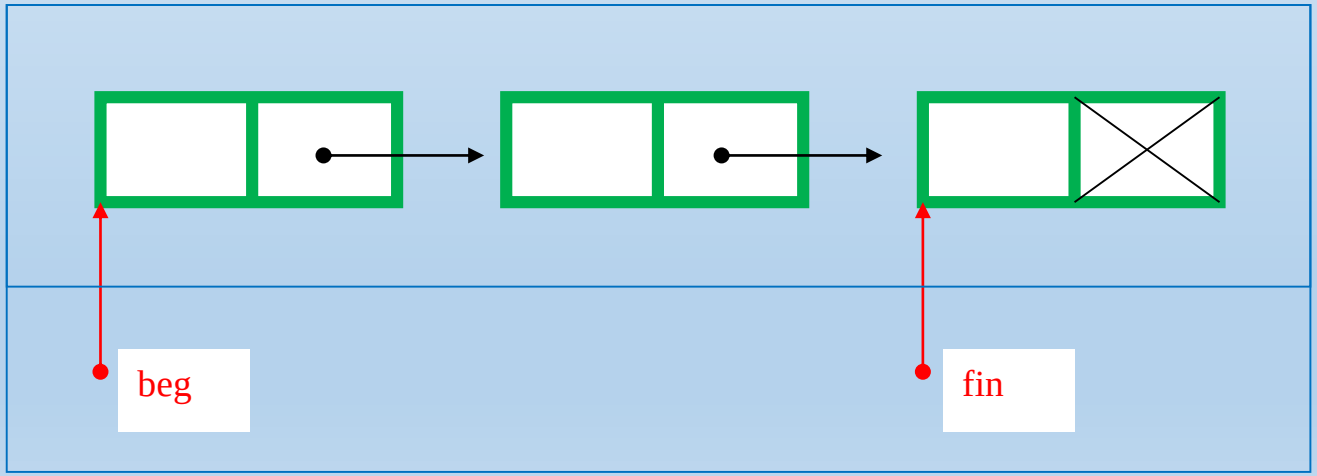
Тип указателя должен соответствовать типу элементов, из которых состоит очередь:

```
new(beg);
beg^.d := 100;
beg^.s := ‘Ваня’ ;
beg^.p : nil;
fin := beg;
```

Начальное формирование очереди *– это создание её первого элемента и установка на него обоих указателей* (рис 2.)



Очередь, состоящая из трех элементов, изображена со всей возможной тщательностью (рис 3)



Добавление элемента в конец очереди выполняется с помощью вспомогательного указателя:

```
new (p);                                                    {1}

p^.d := 10;    p^.s := ‘Слава’ ;   p^.p := nil;             {2}

fin^.p := p;                                                {3}

fin := p;                                                   {4}
```

В операторах {1} и {2} создаётся новый элемент и заполняется его информационная часть.

Оператор {3} добавляет в последний элемент очереди ссылку на новый элемент.

Оператор {4} устанавливает новое значение указателя на конец очереди.

with beg^.do writeln (d, s);	{1}	Выборка элемента выполняется из начала очереди
p := beg; beg := beg^.p;	{2}	указатель на начало очереди сдвигается на следующий элемент
dispose (p);	{3}	при этом выбранный элемент удаляется

Данная динамическая структурная единица - очередь, эффективна, если невозможно определить, сколько памяти необходимо для хранения, когда память следует распределять во время выполнения программы по мере необходимости отдельными блоками.

Для решения задач сортировки так же эффективнее и экономнее использовать динамические структуры, поскольку упорядочивание не требует перестановки элементов, а сводиться к изменению указателей на эти элементы.

Теперь, на основании выше изложенного, напишем программу.

Задача. Программа формирует очередь из пяти целых чисел и их текстового представления и выводит её на экран.

Решение:

Операции с очередью оформим в виде процедур.

Процедура начального формирования называется first,

помещения в конец очереди – add,

выборки – get.

Начало программы

```
program ochered;
const n = 5;
type pnde = ^node;
        node = record
          d : word;
           s : string;
           p : pnode;
         end;
var    beg, fin : pnode; {указатели на начало и конец очереди};
        i : word;
        s : string;
const text: array [1..n] of string = (‘one’, ‘two’, ‘three’, ‘four’, ‘five’);
                                    {--------- начальное формирование очереди ----------}
procedure first (var beg, fin : pnode;   d : word;   const s : string);
begin
    new (beg);
    beg^.d := d;  beg^.s := s;  beg^.p := nil;
    fin := beg;
end;
{--------- добавление элемента в конец ----------}
```

Продолжение программы

```
procedure add (var fin : pnode;  d : word;    const s : string);
var  p: pnode;
begin
     new (p);
     p^.d := d;  p^.s := s;   p^.p := nil;
     fin^.p := p;
end;
                                {--------- выборка элемента из начала ----------------}
procedure get (var beg : pnode;  var d : word;  var s : string);
var p : pnode;
begin
    d := beg^.d;  s := beg^.s;
    p := beg;    beg := beg^.p;

     dispose (p);
end;
                            {--------- главная программа ----------------}
```

Продолжение программы

```
begin
        { занесение в очередь :}
        first (beg, fin, 1, text[1]);
        for  i := 2 to 5 do add (fin, i, text[i]);
          { выборка из очереди :}
          while beg <> nil do begin
               get (beg, i, s);
              writeln (i:2, s);
           end;
end.
```

Конец программы