16. ОПАШКА

16.1. Общо описание

Опашката е последователност от компоненти, които чакат да бъдат взети за обработване. В процеса на използването на опашката ту се добавят към нея нови компоненти, ту се вземат от нея компоненти за обработка. Вземането на компоненти от опашката става по реда, в който са постъпили в нея, т.е. най-напред се взема и обработва компонентата, наредила се (постъпила) в опашката първа, а най-накрая - компонентата, наредила се последна. Поради това е прието да се казва, че опашката е структура от тип FIFO (First In, First Out - първи влязал, първи излязал).

В програмирането опашките са полезни при създаване на приложни програми (приложения), които обработват компонентите в реда на постъпването им.

От програмистка гледна точка опашката е структура от данни, чийто компоненти образуват линейна последователност.

Основни операции са:

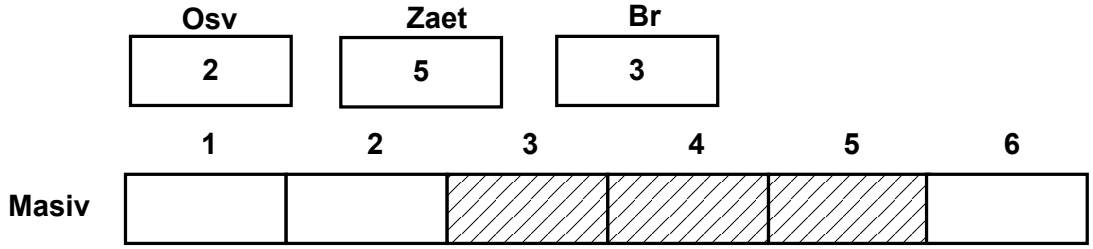
1. Инициализация на опашка - привеждане на опашката в състояние “празна опашка”.
2. Добавяне на компонента.
3. Отнемане на компонента.

Типът на компонентите може да е стандартен или дефиниран, но най-често е запис.

Опашка може да се създаде по два начина: като се използва масив или като се използват указатели.

16.2. Опашка, използваща масив

16.2.1. Кратко описание



Фиг.16.1

За да се организира такава опашка са нужни един едномерен масив и три целочислени променливи (Фиг.16.1). Всяка добавена към опашката компонента става елемент на масива. Поради това типът на елементите на масива трябва да съвпада с типа на компонентите на опашката. Едната целочислена променлива (Zaet) е индекс на последния зает елемент от масива (края на опашката), втората целочислена променлива (Osv) е индекс на последния освободен елемент от масива (началото на опашката), а третата (Br) е брой на компонентите в опашката.

Очевидно при ползването на опашката ту се добавят, ту се отнемат компоненти, т.е. броят на компонентите в опашката е променлив в границите от нула (празна опашка) до максимално допустимия, който се определя при дефиниране на масива. Разполагането на компоненти е по-особено. Когато е зает и последният елемент на масива, но в него има свободни елементи, то те са в началото на масива, т.е. следващата добавена компонента ще попадне в първия елемент на масива.

16.2.2. Дефиниране на тип опашка, използваща масив

```
Const N=50;                {Максимален брой компоненти в опашката}
Type
  TipInf=record     {Тип на компонента от опашката}
      Ime:string[20];
      EGN:string[10]
    end;
  TipStr=record     {Тип на структурата опашка}
      Masiv:array[1..N] of TipInf; {Масив, използван от опашката}
      Osv,          {Индекс на последния освободен елемент от масива}
      Zaet,         {Индекс на последния зает елемент от масива}
      Br:0..N       {Брояч на компонентите в опашката}
    end;
```

16.2.3. Дефиниране на опашка (променливи от тип опашка)

Опашките, нужни на програмата, се дефинират в раздела Var. Например с оператора:

```
Var
  Opash1, Opash2, OpashX:TipStr;
```

са дефинирани три опашки с имена съответно Opash1, Opash2 и OpashX.

16.2.4. Инициализиране на опашка

Под инициализиране на опашка се разбира обявяване на опашката за празна, а това се постига като се присвоят нулеви стойности на трите целочислени променливи. Инициализация на опашка може да се направи чрез следната процедура:

```
Procedure Init(Var Str:TipStr);
  Begin
    Str.Osv:=0; Str.Zaet:=0; Str.Br:=0;
  End;
```

16.2.5. Добавяне на нова компонента към опашка

Към опашка от декларирания тип може да се добави нова компонента например чрез следната функция:

```
Function DobEl(Var Str:TipStr;X:TipInf):boolean;
  Begin
    With Str do
      If Br<N          {Има място в опашката за още елементи}
        then begin
          If Zaet<N
```

```
            then Zaet:=Zaet+1
          else Zaet:=1;
          Masiv[Zaet]:=X;
          Br:=Br+1;
          DobEl:=true
        end
        else DobEl:=false      {Няма място в опашката за още елементи}
    End;
```

Тази функция придобива стойност true, когато се справи с добавянето на нова компонента (в опашката има място за нова компонента), и стойност false, когато добавянето е невъзможно, поради това, че в опашката няма място за нов елемент.

16.2.6. Отнемане на компонента от опашка

От опашка може да се отнеме компонента например чрез следната функция:

```
Function OtnEl(Var Str:TipStr;Var X:TipInf):boolean;
  Begin
    With Str do
      If Br>0 {Опашката не е празна}
        then begin
             If Osv<N
               then Osv:=Osv+1
             else Osv:=1;
          X:=Masiv[Osv];
          Br:=Br-1;
          OtnEl:=true
        end
        else OtnEl:=false     {Опашката е празна}
  End;
```

Тази функция придобива стойност true, когато се справи с отнемането на компонента (опашката не е празна) и стойност false, когато отнемането е невъзможно, поради това, че в опашката няма нито един елемент.

16.3. Опашка, използваща указатели

16.3.1. Кратко описание

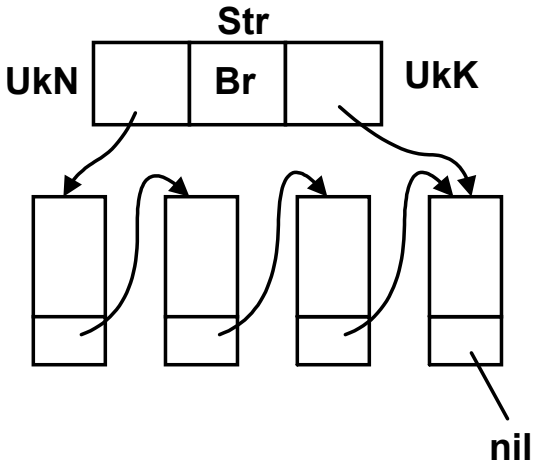
За да се организира такава опашка са нужни две променлива от тип указател - UkN, и UkK, указващи началото и края на опашка. С цел да се създадат известни удобства обикновено се ползва и една целочислена променлива за брояч на компонентите в опашка. Поради това ще разглеждаме опашката като променлива от тип запис с три полета: указател към началото, указател към края и брояч на компонентите.

Компонентите на опашка са динамични променливи от тип запис с две полета:

- информационно поле Inf - обикновено е от тип запис с полета, определени от конкретното предназначение на опашка. Тук ще приемем, че то има само две полета: Ime и EGN;
- указателно поле - указва следващата компонента в опашката.

Тази опашка има две съществени предимства:

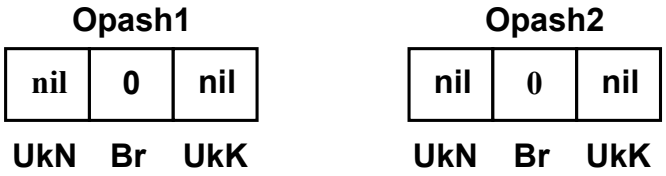
- във всеки момент от изпълнението на програмата, тя заема само толкова място колкото й е необходимо;
- тя не се препълва докато в ОП на компютъра все още има свободно място.



Фиг.16.2 Опашка, използваща указатели

16.3.2. Дефиниране на тип опашка, използваща указатели

```
Type
  TipInf=record     {Тип на информац. поле на комп. от опашката}
      Ime:string[25];
      EGN:string[10]
      end;
  TipUk=^TipEl;     {Тип на указател към комп. от опашката}
  TipEl=record      {Базов тип или тип на комп. от опашката}
      Inf:TipInf;   {Информац. поле на комп. от опашката}
      Uk:TipUk      {Указател към следв. комп от опашката}
      end;
  TipStr=record     {Тип на структурата опашка}
      UkN:TipUk;    {Указател към началото на опашката}
      UkK:TipUk;    {Указател към края на опашката}
      Br:integer    {Брой на компонентите в опашката}
      end;
```



Фиг.16.3 Инициализирани опашки

16.3.3. Дефиниране на опашки (променливи от тип опашка)

Опашките, нужни на програмата, се дефинират в раздела **Var**. Например с оператора

```
Var
  Opash1, Opash 2, Opash X:TipStr;
```

са дефинирани три опашки с имена съответно Opash1, Opash2 и OpashX.

16.3.4. Инициализиране на опашка

Под инициализиране на опашка се разбира обявяване на опашката за празна, а това се постига като се присвои нулева стойност на брояча на компонентите и стойност **nil** на указателите към началото и към края на опашката (Фиг.16.3).

Опашки от декларирания тип могат да се инициализират чрез следната процедура, където Str е фиктивен параметър, който се замества с името на инициализираната опашка.

```
Procedure Init(Var Str:TipStr);
  Begin
    Str.UkN:=nil;Str.UkK:=nil;Str.Br:=0;
  End;
```

16.3.5. Добавяне на нова компонента към опашка

Както е показано на Фиг.16.4, за да се добави нова компонента към опашка, трябва да се изпълнят следните операции, като се използва допълнителен указател Nov:

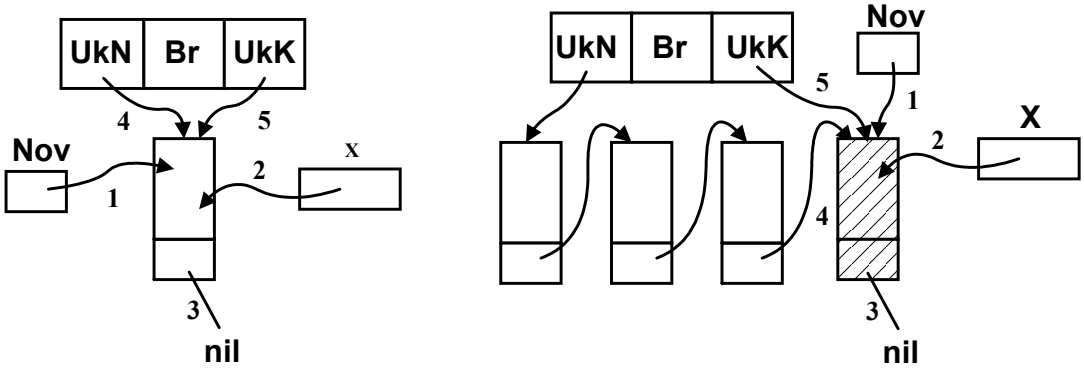
1. Създаване на нова празна компонента - това става с помощта на процедурата New.
2. Записване в информационното поле на новата компонента информационното съдържание на новата компонента.
3. Присвояване стойност nil на указателното поле на новата компонента.
4. Записване адреса на новата компонента в указателното поле на досегашната последна компонента или в указателя UkN, ако опашката е празна.
5. Записване адреса на новата компонента (вече последна компонента) в указателното поле на опашка UkK.

Към опашка от декларирания тип може да се добави нова компонента например чрез функцията, дадена по-долу, където Str и X са фиктивни параметри, който се заместват съответно с името на опашката и променливата, чиято стойност трябва да запълни информационното поле на новата компонента.

```
Function DobEl(Var Str:TipStr;X:TipInf):boolean;
  Var
    Nov:TipUk;
  Begin
    If MaxAvail>=SizeOf(TipEl)
      then With Str do
             begin
               New(Nov);
               Nov^.Inf:=X;
               Nov^.Uk:=Nil;
```

```
            If UkK=Nil              {Празна ли е опашката}
               thenUkN:=Nov         {Опашката е празна}
               else UkK^.Uk:=Nov;  {Опашката не е празна}
            UkK:=Nov;
            Br:=Br+1;
            DobEl:=true
          end
      else DobEl:=false
    End;
```

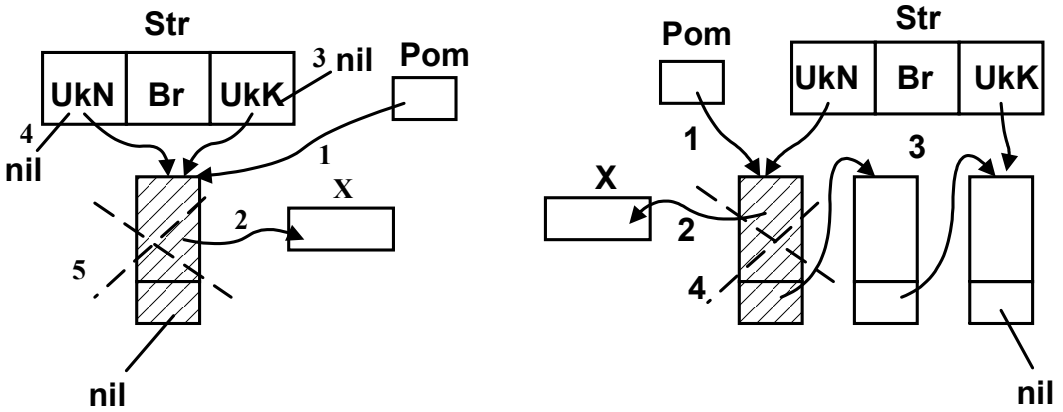
Тази функция придобива стойност true, когато се справи с добавянето на нова компонента (в ОП има място за нова компонента), и стойност false, когато добавянето е невъзможно, поради това, че в ОП няма място за нова компонента.



Фиг.16.4 Добавяне на компонента към опашка

16.3.6. Копиране и отнемане на компонента от опашка

Както е показано на Фиг.16.5, за да се отнеме компонента от опашка трябва да се изпълнят поредица операции, като се използва помощен указател Pom.



Фиг.16.5 Копиране и отнемане на компонента

От опашка от декларирания тип може да се отнеме компонента например чрез функцията, дадена по-долу, където Str и X са фиктивни параметри. При обръщение

към функцията те се заместват съответно с името на опашката и променливата, която трябва да присвои стойността на информационното поле на отнеманата компонента:

```
Function OtnEl(Var Str:TipStr;Var X:TipInf):boolean;
  Var
    Pom:TipUk;
  Begin
    With Str do
      If UkN<>nil
    then begin
           Pom:=UkN;
           X:=UkN^.Inf;
           If UkN=UkK
             then UkK:=Nil;
           UkN:=UkN^.Uk;
           Dispose(Pom);
           Br:=Br-1;
           OtnEl:=true
         end
    else OtnEl:=false
End;
```