15. СТЕК

15.1. Общо описание

Стекът е структуриран тип данни, чийто елементи образуват линейна последователност. В процеса на използването на стека ту се добавят нови елементи към него, ту се отнемат (изваждат) елементи от него.

Често тази линейна последователност се сравнява с купчина чинии по следните причини:

- както купчината чинии, така и стекът имат основа и връх;
- нова чиния и нов елемент се добавят, като се разполагат на върха;
- чиния и елемент се вземат само от върха.

Прието е да се казва, че достъпът до елементите на стека е от тип LIFO (Last In, First Out - последен влязал, пръв излязал). В програмирането стековете са полезни при създаване на приложни програми (приложения), които обработват елементите в ред, обратен на реда на постъпването им.

Основни операции са:

1. Инициализация на сткек - привеждане на стека в състояние “празен стек”.
2. Добавене на елемент.
3. Отнемане на елемент.
4. Копиране на върхния елемент.

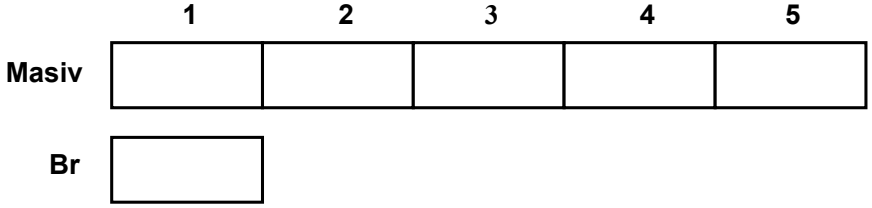
Типът на елементите може да е стандартен или дефиниран от потребителя, но най-често е запис.

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

15.2. Стек, използващ масив

15.2.1. Кратко описание

За да се организира такъв стек са нужни един едномерен масив и една целочислена променлива (Фиг.15.1). Всеки добавен към стека елемент става елемент на масива. Поради това типът на елементите на масива трябва да съвпада с типа на елементите на стека. А целочислената променлива е едновременно индекс на последния зает елемент от масива, номер на върхния елемент и брояч на елементите в стека. Очевидно, при ползването на стека ту ще се добавят, ту ще се отнемат елементи, т.е. броят на елементите в стека е променлив в границите от нула (празен стек) до максимално допустимия, който се определя при дефиниране на масива.



Фиг.15.1

15.2.2. Дефиниране на тип стек, използващ масив

```
Const N=50;       {Максимален брой на елементите в стека}
Type
  TipInf=record   {Тип на елемент от стека}
     Ime:string[20];
     EGN:string[10]
     end;
  TipStr=record   {Тип на структурата стек}
     Masiv:array[1..N] of TipInf; {Масив, използван от стека}
     Br:0..N       {Броячът на елементите в стека}
     end;
```

15.2.3. Дефиниране на стекове (променливи от тип стек)

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

```
Var
  Stek1,Stek2,StekX:TipStr;
```

са дефинирани три стека с имена съответно Stek1, Stek2 и StekX и всеки от тях има свой масив за елементите и свой брояч на елементите.

15.2.4. Инициализиране на стек

Под инициализиране на стека се разбира обявяване на стека за празен, а това се постига като се присвои нулева стойност на брояча на елементите. Инициализация на стек от декларирания тип може да се направи чрез следната процедура:

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

15.2.5. Добавяне на нова компонента към стека

Към стек 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 ОП за нов елемент}
      then begin
         Br:=Br+1;
         Masiv[Br]:=X;
         DobEl:=true
       end
    else DobEl:=false
  End;
```

Тази функция придобива стойност true, когато се справи с добавянето на нов елемент (в стека има място за нов елемент), и стойност false, когато добавянето е невъзможно, поради това, че стекът е пълен (заети са всички елементи в стека).

15.2.6. Копиране и отстраняване на компонента от стека

От стек от декларирания тип може да се копира и отстрани компонента

например чрез следната функция:

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

Тази функция придобива стойност true, когато се справи с копирането и отстраняването на компонентата (стекът не е празен) и стойност false, когато отнемането е невъзможно, поради това че стекът е празен (няма нито един елемент в стека).

15.3. Стек, използващ указатели

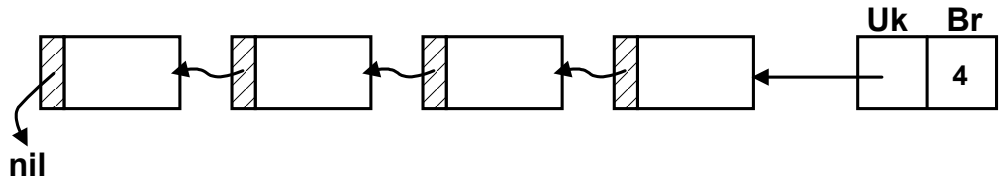
15.3.1. Кратко описание

За да се организира такъв стек е нужна една променлива от тип указател, която да указва върха на стека. С цел да се създадат известни удобства обикновено се ползва и една целочислена променлива за брояч на компонентите в стека. поради това ще разглеждаме стека като променлива от тип запис с две полета: указател към върха и брояч на компонентите.

Компонентите на стека са динамични променливи от тип запис с две полета: информационно поле Inf и указателно поле, указващо предходната компонента на стека. Информационното поле обикновено също е от тип запис с полета, определени от конкретното предназначение на стека. Тук ще приемем, че то има само две полета: Ime и EGN.

Стек, реализиран чрез указател, има две съществени предимства:

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



Фиг.15.2 Стек, използващ указатели

15.3.2. Дефиниране на стек, използващ указатели

```
Type
  TipInf=record     {Тип на информац. поле на компонента от стека}
           Ime:string[25];
           EGN:string[10]
         end;
  TipUk=^TipEl;     {Тип на указател към компонента от стека}
```

```
TipElem=record {Базов тип или тип на компонента от стека}
            Inf:TipInf;
            Uk:TipUk
    end;
TipStr=record    {Тип на структурата стек}
          Uk:TipUk;
          Br:integer
    end;
```

15.3.3. Дефиниране на стекове (променливи от тип стек)

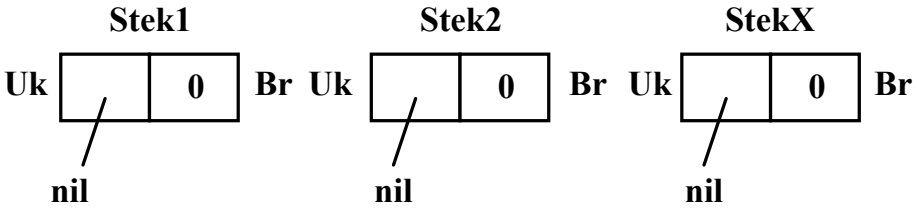
Стековете, нужни на програмата, се дефинират в раздела **Var**. Например с оператора

```
Var
  Stek1,Stek2,StekX:TipStr;
```

са дефинирани три срека с имена съответно Stek1, Stek2 и StekX.

15.3.4. Инициализиране на стек

Под инициализиране на стека се разбира обявяване на стека за празен, а това се постига като се присвои нулева стойност на брояча на елементите и стойност nil на указателя към върха на стека.



Фиг.15.3 Инициализирани стекове

Инициализация на стек от декларирания тип може да се направи чрез следната процедура, където Str е фиктивен параметър, който се замества с името на инициализирания стек:

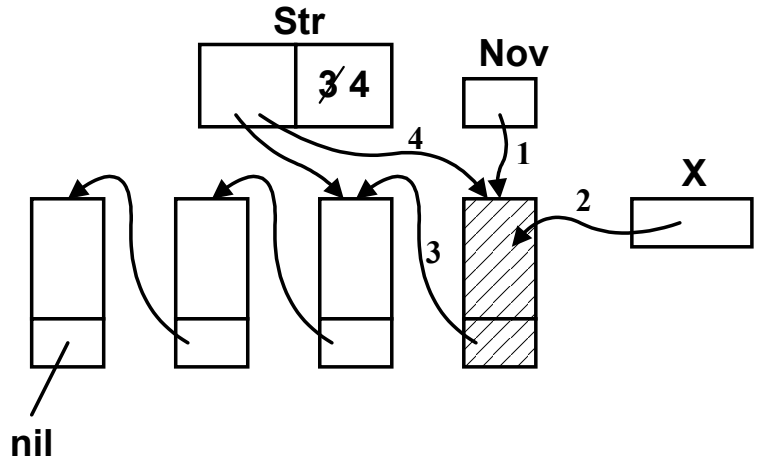
```
Procedure Init(Var Str:TipStr);
  Begin
    Str.Uk:=nil; Str.Br:=0;
  End;
```

15.3.5. Добавяне на нови компоненти към стека

Както е показано на Фиг.15.4, за да се добави нова компонента към стека, трябва да се изпълнят следните операции, като се използва допълнителен указател NowEl

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. Записване адреса на досегашната върхова компонента в указателното поле на новата компонента;
4. Записване адреса на новата компонента (вече върхова компонента) в указателното поле на стека.

Фиг.15.4 Добавяне на компонента



Към стек от декларирания тип може да се добави нова компонента например чрез функцията, дадена по-долу, където Str и X са фиктивни параметри. При извикване на функцията те се заместват съответно с името на стека 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)      {Има място в ОП за нови компоненти}
      then begin
        New(Nov);
        Nov^.Inf:=X;
        Nov^.Uk:=Str.Uk;
        Str.Uk:=Nov;
        Str.Br:=Str.Br+1;
        DobEl:=true
      end
      else DobEl:=false
  End;
```

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

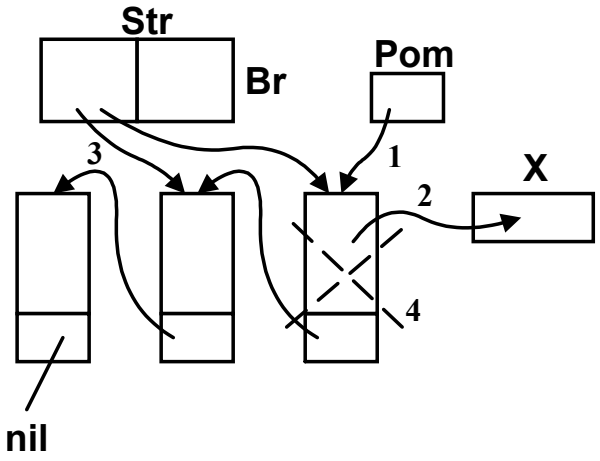
15.3.6. Копиране и отнемане на компоненти от стека

Както е показано на Фиг.15.5, за да се отнеме компонента от стека трябва да се изпълнят поредица операции, като се използва помощен указател Pom.

От стек от декларирания тип може да се отнеме компонента например чрез функцията, дадена по-долу, където Str и X са фиктивни параметри. При извикване на функцията те се заместват съответно с името на стека 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
    If Str.Uk<>Nil {Стекът не е празен}
       then begin
          Pom:=Str.Uk;
          X:=Str.Uk^.Inf;
          Str.Uk:=Str.Uk^.Uk;
          Dispose(Pom);
          Str.Br:=Str.Br-1;
```

```
        OtnEl:=true;
      end
    else OtnEl:=false
End;
```



Фиг.15.5 Копиране и отнемане на компонента