Структури на податоци: Стек и ред

Data structures: Stack and Queue

Version 2.0

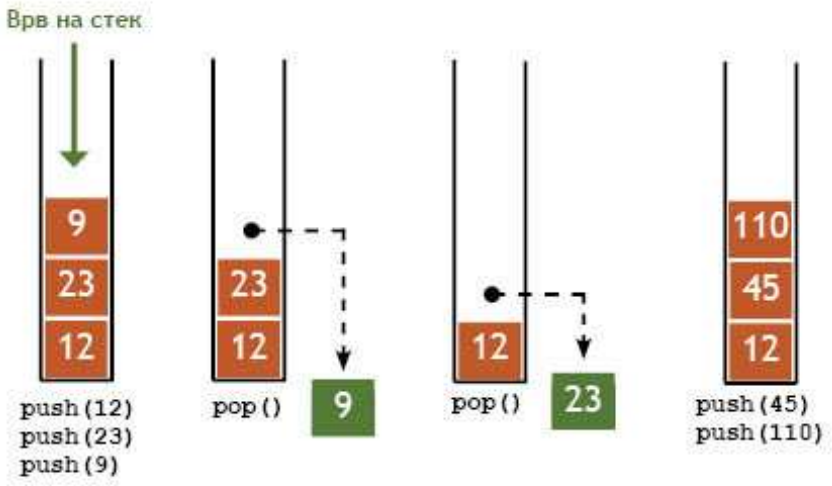
algoritam.blog.com.mk
2007

Податочна структура: Стек

Стекот е една од основните податочни структури што се користи при програмирањето. Оваа структура може да се создаде со помош на низа што исто така претставува посебна и добро позната податочна структура. Стекот може да се сфати како низа со точно утврдени правила на користење на таа низа:

- Поставување елемент на врвот на стекот или процедура ***push(x : element)***
- Вадење на елементот кој се наоѓа на врвот од стекот или функција ***pop()*** која како резултат го враќа оној елемент што го вадиме од врвот на стекот.

Стекот има свој капацитет. При ставање на нов елемент на врвот од стекот, (се проверува) се внимава да не се надмине капацитетот на стекот. Но исто така не можеме да извадиме елемент од стекот доколку нема ниту еден елемент кој е поставен во стекот. Стекот се нарекува **LIFO**(*анг.* Last In First Out) структура.



Слика 1: Приказ на функционирање на стекот

Пример програма за податочната структура стек реализирана во Pascal:

```
program primer_za_stek;
var
   stek:array[1..100]of integer;
   kapacitet:integer;
   tekoven_broj_elementi:integer;

procedure inicijaliziraj_stek;
begin
      kapacitet:=100;
      tekoven_broj_elementi:=0;
end;

procedure push(x:integer);
begin
     {proveruvame dali ima se uste mesto vo stekot}
     if tekoven_broj_elementi<kapacitet
      then
```

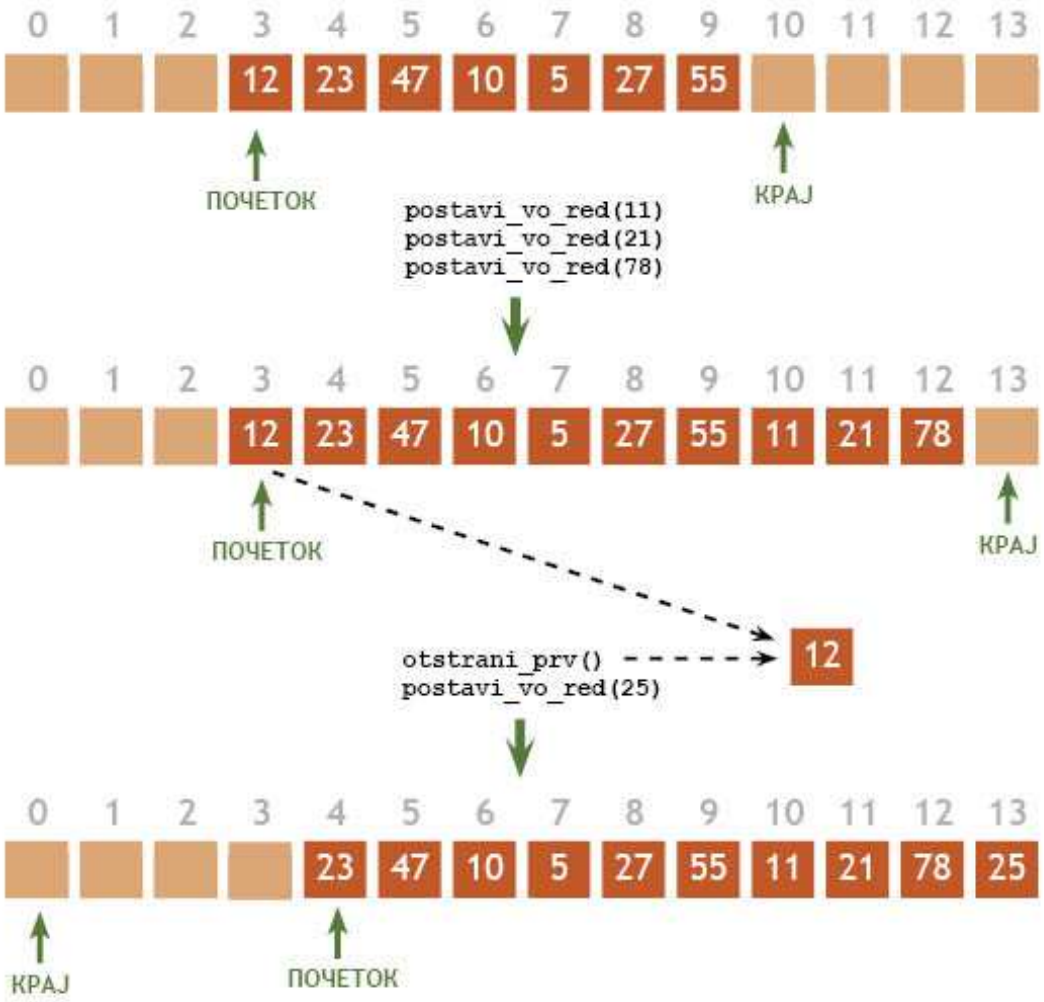
```
            begin
                 {go zgolemuvame brojot na tekovni elementi vo stekot}
                 inc(tekoven_broj_elementi);
                 {go postavuvame elementot <x> na soodvetnoto mesto}
                 stek[tekoven_broj_elementi]:=x;
            end
        else
            writeln('Nema poveke mesto vo stekot !');
end;

function pop():integer;
begin
      {proveruvame dali ima barem eden broj postaven vo stekot}
      if tekoven_broj_elementi>0
        then
            begin
                 {go vrakame brojot od vrvot na stekot kako rezultat}
                 pop:=stek[tekoven_broj_elementi];
                 {go namaluvame brojot na tekovni elementi(broevi) vo
stekot}
                 dec(tekoven_broj_elementi);
            end
        else
            begin
                 {ako stekot e prazen, ispisuvame poraka}
                 writeln('Stekot e prazen !');
                 {se vraka rezultat -1}
                 pop:=-1;
            end;
end;

{procedura za pecatenje na elementite vo stekot}
procedure pecati_stek;
var
    i:integer;
begin
     writeln('Elementite na stekot se, pocnuvajki od dnoto kon vrvot : ');
     for i:=1 to tekoven_broj_elementi do
         begin
              writeln(stek[i]);
         end;
end;

{Glavna programa}
begin
      inicijaliziraj_stek;
      push(12);
      push(23);
      push(9);
      pecati stek;
      writeln('Izvadi element : ',pop());
      pecati_stek;
      writeln('Izvadi element : ',pop());
      pecati_stek;
      push(45);
      push(110);
      pecati_stek;
      readln;
end.
```

Податочна структура: Ред

Редот е уште една податочна структура што доста често се користи. Редот исто така може да се реализира со помош на низа, но со други, посебни правила на работа и употреба на низата:

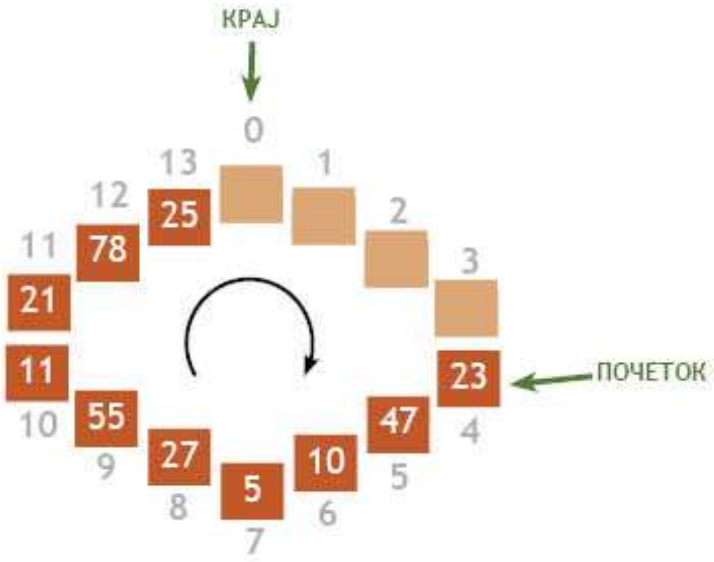
- Поставување елемент на крајот на редот или процедура ***postavi_vo_red(x : element)***
- Отстранување на елемент од почетокот на редот или функција ***otstrani_prv()*** која како резултат го враќа оној елемент што го отстрануваме од почетокот на редот.

Како и стекот, и редот има свој капацитет. При ставање на нов елемент во редот, (се проверува) се внимава да не се надмине капацитетот на стекот. Но исто така не можеме да извадиме елемент од редот доколку нема ниту еден елемент кој е поставен во редот.



Слика 2: Приказ на функционирање на редот

За да ги контролираме почетокот и крајот на редот, користиме помошни променливи за зачувување на индексите на почетниот елемент и крајниот (слободен) елемент. Новите елементи кои ги ставаме во редот, ги ставаме од десна страна, притоа зголемувајќи ја вредноста на **КРАЈ** за 1. Елементи вадиме од левата страна, притоа зголемувајќи ја вредноста на **ПОЧЕТОК** за 1. Доколку **ПОЧЕТОК** или **КРАЈ**, според примерот на сликата, добијат вредност **14** тогаш нивната вредност се менува на **0**. Всушност тоа изгледа како двата краја на редот да се споени, односно елементите со реден број 0 и 13 од низата се соседни. Редот уште се нарекува **FIFO** (*анг.* First In First Out) структура.



Слика 3: Сликата го покажува замисленото поврзување на двата краја на редот (елементи со реден број 0 и 13)

Пример програма за податочната структура ред реализирана во Pascal:

```
program primer_za_red;
var
   red:array[0..99]of integer;
   kapacitet:integer;
   tekoven_broj_elementi:integer;
   pocetok,kraj:integer;

procedure inicijaliziraj_red;
begin
      kapacitet:=100;
      tekoven_broj_elementi:=0;
      pocetok:=0;
      kraj:=0;
end;

procedure postavi_vo_red(x:integer);
begin
     {proveruvame dali ima se uste mesto vo redot}
     if tekoven_broj_elementi<kapacitet
        then
            begin
                 {go zgolemuvame brojot na tekovni elementi}
                 inc(tekoven_broj_elementi);
```

```
                {go postavuvame elementot <x> na soodvetnoto mesto}
                red[kraj]:=x;
                {go zgolemuvame <kraj> za 1}
                inc(kraj);
                {ako <kraj> go nadmine kapacitetot(100) t.e. granicata na
nizata (99), togas <kraj> dobiva vrednost 0}
                if kraj=kapacitet
                   then
                       kraj:=0;
            end
        else
            writeln('Nema poveke mesto vo redot !');
end;

function otstrani_prv():integer;
begin
     {proveruvame dali ima barem eden broj postaven vo redot}
     if tekoven_broj_elementi>0
        then
            begin
                {go namaluvame brojot na tekovni elementi vo redot}
                dec(tekoven_broj_elementi);
                {go vrakame brojot od pocetokot na redot kako rezultat}
                otstrani_prv:=red[pocetok];
                {go zgolemuvame <pocetok> za 1}
                inc(pocetok);
                {ako <pocetok> go nadmine kapacitetot(100) t.e. granicata
na nizata (99), togas <pocetok> dobiva vrednost 0}
                if pocetok=kapacitet
                   then
                       pocetok:=0;
            end
        else
            begin
                {ako redot e prazen, pecatime poraka}
                writeln('Redot e prazen !');
                {se vraka rezultat -1}
                otstrani_prv:=-1;
            end;
end;

{procedura za pecatenje na redot}
procedure pecati_red;
var
    i:integer;
begin
     writeln('Elementite na redot se, pocnuvajki od prviot kon posledniot :
');
     if tekoven_broj_elementi>0
        then
            if (pocetok<kraj)
               then
                   begin
                       for i:=pocetok to kraj-1 do
                           writeln(red[i]);
                   end
               else
                   begin
                       for i:=pocetok to kapacitet-1 do
                           writeln(red[i]);
                       for i:=0 to kraj-1 do
```

```
                              writeln(red[i]);
                    end;
end;

{Glavna programa}
begin
      inicijaliziraj_red;
      postavi_vo_red(12);
      postavi_vo_red(23);
      postavi_vo_red(45);
      pecati_red;
      writeln('Izvadi element : ',otstrani_prv());
      pecati_red;
      writeln('Izvadi element : ',otstrani_prv());
      pecati_red;
      postavi_vo_red(110);
      pecati_stek;
      readln;
end.
```

Во рамки на реализацијата на податочните структури стек и ред, би можело да се креираат помошни функции кои ќе дадат одговор на прашањата *Дали стекот/редот е празен ?* или пак *Дали стекот/редот е пополнет ?*. На тој начин ќе може пред секое ставање или вадење на елемент да се провери состојбата на податочната структура.

Стекот и редот имаат голема практична примена во софтверските системи. Стекот игра голема улога при извршувањето на функциите и процедурите во рамки на една програма (стек меморија), додека пак редот како податочна структура особено онака како што е реализиран во примерот (со фиксен капацитет) се користи како помошен елемент во комуникацијата на два процеса кои работат паралелно. Во таков случај, редот се користи за зачувување на податоците што првиот процес му ќи испрача на вториот процес. Процес е програма којашто се извршува во рамки на компјутерскиот систем.

Слика 4: Два процеси коишто комуницираат преку ред. Редот во ваков случај се поставува затоа што процес 1 генерира податоци побрзо отколку што процес 2 може истите да ги конзумира односно обработи