

Програмирање 1

Групно спремање – питалице

Синтаксне нотације

Увод

Синтаксне нотације користе се за описивање синтаксе („правописа“) неког језика (најчешће програмских).

Пример:

`<indeks> ::= <godina>/<broj>`

`<godina> ::= <cifra><cifra>`

`<broj> ::= <cifra><cifra><cifra><cifra>`

`<cifra> ::= 0|1|2|3|4|5|6|7|8|9`

БНФ нотација

`<neterminal>`

објекат који се описује, служи да сложене описе поделимо на више простих

`terminal`

симбол који се очекује на излазу – азбука синтаксе

оператор `|` значи избор између више опција, на пример `<x> ::= a | b` значи да `x` може бити „a“ или „b“

`{x}` значи да се „x“ понавља од 1 до ∞ пута, број понављања се може ограничити на овај начин: $\{x\}_3^{10}$

ЕБНФ нотација

`nonterminal`

симбол који се описује

`"terminal"`

симбол који се појављује на излазу (у језику)

$(a|b)$

избор између a и b

$\{x\}$

понављање од 0 до ∞ пута (или се може ограничити: $\{x\}_3^{10}$)

$[x]$

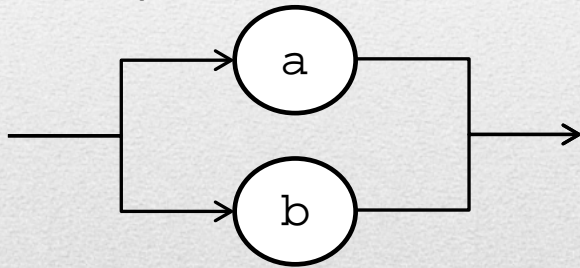
опција – може се појавити али не мора, понављање од 0 до 1

Синтаксни дијаграм

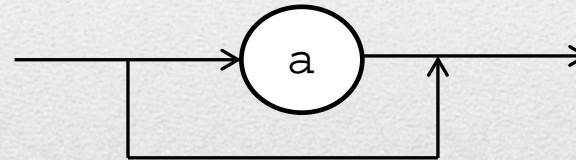
terminal

terminal

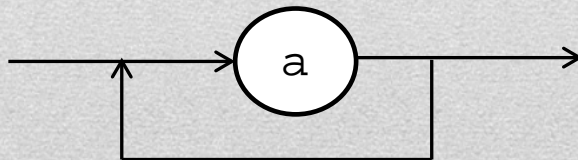
избор



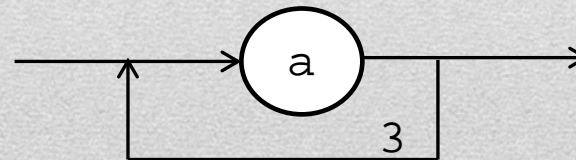
опција



понављање



ограничено понављање



Пример (број мобилног телефона)

БНФ

$\langle \text{telefon} \rangle ::= +381 \langle \text{pozivni} \rangle \langle \text{broj} \rangle$

$\langle \text{pozivni} \rangle ::= 60 \mid 61 \mid 62 \mid 63 \mid 64 \mid 65 \mid 66$

$\langle \text{broj} \rangle ::= \{ \langle \text{cifra} \rangle \}_6^7$

$\langle \text{cifra} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

ЕБНФ

$\text{telefon} = "+381" \text{ pozivni broj}.$

$\text{pozivni} = "6" ("0" \mid "1" \mid "2" \mid "3" \mid "4" \mid "5" \mid "6").$

$\text{broj} = \{ \text{cifra} \}_6^7.$

$\text{cifra} = ("0" \mid "1" \mid "2" \mid "3" \mid "4" \mid "5" \mid "6" \mid "7" \mid "8" \mid "9").$

Јануар 2012. (5. питалица)

Који од понуђених одговора задовољава дату синтаксну дефиницију која описује један термин за полагање испита у БНФ нотацији. Претпоставити да су <слово>, <цифра> и <blanko> већ дефинисани.

```
<ispit> ::= <naziv> , TERMIN: <vreme> , <sala> , <broj>  
<vreme> ::= <sati> : <minuti>  
<sati> ::= 0 <cifra> | 1 <cifra>  
<minuti> ::= <sati> | 2 <cifra> | 3 <cifra> | 4 <cifra> | 5 <cifra>  
<naziv> ::= <rec> | <naziv> <blanko> <rec>  
<rec> ::= <слово> | <rec> <слово>  
<sala> ::= <naziv>  
<broj> ::= <цифра> | <broj> <цифра>
```

A) Programiranje 1,TERMIN:18:30,56,50

B) Osnovi elektrotehnike,TERMIN:11:30,M216,130

C) Fizika,TERMIN:00:13,VMA,100

Јун 2012. (4. питалица)

Који од понуђених одговора задовољава дату синтаксну дефиницију која описује резултат једне одбојкашке утакмице у ЕБНФ нотацији? Претпоставити да су `slovo` и `blanko` већ дефинисани.

```
utakmica = tim "-" tim blanko set ":" set blanko rezultat.  
set = ("0" | "1" | "2" | "3").  
rezultat = "(" poeni {"," blanko poeni} ")"  
poeni = broj ":" broj.  
tim = {slovo}.  
broj = [set](set | "4" | "5" | "6" | "7" | "8" | "9").
```

A) Srbija-Južna Koreja 3:1 (25:23, 25:22, 17:25, 25:12)

B) Kina-Srbija 2:3 (33:31, 25:18, 41:43, 22:25, 13:15)

☒ C) Srbija-Japan 3:0 (25:19, 25:23)

Октобар 2011. (4. питалица)

Који од понуђених израза одговарају синтаксној дефиницији START?

$$\langle \text{START} \rangle ::= 101\langle A \rangle \mid \langle B \rangle$$
$$\langle A \rangle ::= \langle A \rangle 11 \mid \langle A \rangle 0 \mid \langle C \rangle$$
$$\langle B \rangle ::= 100\langle B \rangle \mid \langle B \rangle 01 \mid 11$$
$$\langle C \rangle ::= 11 \mid 00$$

A) 101110110011

B) 1001001101100

C) 10111001110

Сложность алгоритма

Увод

- Сложеност алгоритма описује колико је времена потребно програму да одради посао у зависности од (броја) података са којима ради.
- Функцију сложености добијамо када пребројимо колико се наредби изврши у програму – веома тешко.
- Ред функције сложености је највећи члан те функције (нпр. ако је функција: $f(n) = 3n^2 - 2n + 3$, онда је ред сложености $O(n^2)$ – квадратна сложеност).
- Не морамо да бројимо све наредбе, интересује нас најсложенији део кода – гледамо петље.

Константна – $O(1)$

```
readln(a,b);
```

```
c:=a+b;
```

```
writeln(c);
```

Функција сложености:

$$f(n) = 3$$

Ред сложености :

$$O(1)$$

Линеарна – $O(n)$

```
for i := 1 to n do  
    writeln(i);
```

Функција сложености :

$$f(n) = 2n + 1$$

Ред сложености :

$$O(n)$$

Логоритамска – $O(\log_2 n)$

```
k:=n;  
while k>0 do begin  
    writeln(k);  
    k:=k div 2;  
end;
```

Функција сложености :

$$f(n) = 3\log_2 n + 5$$

Ред сложености :

$$O(\log_2 n)$$

Квадратна – $O(n^2)$

```
for i:= 1 to n do  
  for j:= 1 to n do  
    writeln(i,j);
```

Функција сложености :

$$f(n) = 2n^2 + 2n + 1$$

Ред сложености:

$$O(n^2)$$

Одређивање сложености

```
for i := 1 to n do begin
  k := n;
  while k > 0 do begin
    writeln(k);
    k := k div 2;
  end;
end;
```

$O(n \cdot \log(n))$

Септембар 2011. (2. питалица)

```
for i:=1 to n do
  for j:=1 to i*i do begin
    k:=1; m:=n;
    while m>=k do begin
      k:=k*3;
      m:=m/2
    end
  end;
end;
```

- A) n^3
- ☒ B) $n^3 \log(n)$
- C) $n^2 \log(n)$

Увод у Паскал

Општи пример програма

```
PROGRAM ime(input,output);

CONST pi = 3.14;
      max = 100;
TYPE niz = array [0..max] of integer;
      ceo = integer;
VAR i:integer;
    x,y,z:ceo;

PROCEDURE p();
BEGIN
    ...
END;

FUNCTION f();
BEGIN
    ...
END;

BEGIN
    ... {komentar}
END.
```


Типови података (прости)

Основни (предефинисани):

`integer` – цео број (2, 350, -17)

`real` – реални број (3.125, 18e10, 1.2e-7)

`char` – знак ('a', 'R', '0', '?')

`boolean` – логички тип (false, true)

Типови `integer`, `char` и `boolean` су набројиви (могуће их је пребројати).

Опсег (део неког од набројивих типова):

0..100 – цели бројеви од 0 до 100

'a'..'z' – знаци од 'a' до 'z'

Типови података (прости)

Набројани тип:

$(\underline{ident1}, \underline{ident2}, \underline{ident3}, \dots, \underline{identN})$

Набраја се низ идентификатора (имена) који представљају могуће вредности овог типа.

Пример:

```
VAR dan: (pon, uto, sre, cet, pet, sub, ned);
```

Операције:

```
dan := uto;
```

```
dan := succ(sre); {sledbenik -> cet}
```

```
dan := pred(sre); {prethodnik -> uto}
```

```
ord(sre); {redni broj -> 2 (počinje od 0)}
```

не могу се читати нити писати помоћу read и write

Типови података (сложени)

`string` – низ знакова, текст ('ovo je string')

Пример:

```
VAR s,s1,s2:string[30];
```

Операције:

```
s1:='string';
```

```
s2:=''; {prazan string}
```

```
s:=s1 + s2; {nadovezivanje stringova}
```

```
read(s); {citanje}
```

```
write(s); {pisanje}
```

```
s[3]; {treći znak u stringu}
```

Типови података (сложени)

Низови:

`array [opseg] of tip` – дефинише низ задатог типа

Пример:

```
VAR niz: array [1..100] of integer;
```

Операције:

```
niz[3] := 3; {pristup elementu niza}
```

морају се читати и писати елемент по елемент
(не могу одједном)

Типови података (сложени)

Скупови:

set of opseg – скуп елемената из задатог опсега (опсег не сме бити већи од 255 елемената)

Пример:

```
VAR skup,s1,s2: set of 1..100;
```

Operacije:

```
s1:=[1,2,4..7,9]; {skup sadrži:1,2,4,5,6,7,9}
```

```
s2:=[ ]; {prazan skup}
```

```
skup:=s1+s2; {uniја}
```

```
skup:=s1-s2; {razlika}
```

```
skup:=s1*s2; {presek}
```

```
broj IN skup; {da li je broj u skupu}
```

```
skup:=skup+[broj]; {dodavanje broja u skup}
```

Типови података (сложени)

Записи (Record):

RECORD deklaracije_polja END; – сложени тип који се састоји од више „поља“; поља су променљиве које су груписане унутар податка овог типа

Пример:

```
VAR komplBr: RECORD  
    r, i: real;  
END;
```

Операције:

```
komplBr.r:=10; {pristup polju zapisa}  
komplBr.i:=-3; {komplBr je 10-3i}
```


Дефинисање типа

Могуће је увести нова имена за било који тип податка.
Имена се уводе у TYPE делу.

```
TYPE novoIme = opisTipa;
```

Пример:

```
TYPE ceo = integer;  
      dan = (pon,uto,sre,cet,pet,sub,ned);  
      ocena = 5..10;
```

```
VAR o:ocena; {umesto o:5..10}  
    d:dan;
```

Фебруар 2011. (2. питалица)

```
program bojanka (output);  
type boje = (plavo, crveno, zuto, zeleno, crno, belo);  
var s1, s2, s3: set of boje;  
    i: boje;  
begin  
    s1:=[plavo, zuto, zeleno, crno];  
    s2:=[zeleno, belo, zuto];  
    s3:=(s1*s2)+(s2-[crveno, crno])+[belo, zuto];  
    for i:=succ(zeleno) downto plavo do  
        if (i in s3) then  
            write(output, ord(i):2);  
end.
```

A) 6 3 2

B) 3 2 0

C) 3 2

Процедуре и функције

Дефинисање процедуре и функције

```
PROCEDURE ime(a,b:integer; VAR c:integer);  
VAR ...  
BEGIN  
    ...  
END;
```

```
FUNCTION ime(a,b:integer; VAR c:integer):povratniTip;  
VAR ...  
BEGIN  
    ...  
    ime:=povratnaVrednost;  
    ...  
END;
```


Позив процедуре и функције

```
PROGRAM prog(output);  
PROCEDURE p(x:integer); {ispisuje prosleđenu vrednost}  
BEGIN  
    writeln('x=', x);  
END;  
FUNCTION f(n:integer):integer; {računa faktoriјel}  
VAR i:integer;  
BEGIN  
    f:=1;  
    for i:=2 to n do f:=f*i;  
END;  
VAR a,b:integer;  
BEGIN  
    a:=3; b:=3;  
    p(a); {ispisuje „x=3“}  
    p(7); {ispisuje „x=7“}  
    b:=p(7); {GREŠKA: p је procedura, nema povratnu vrednost}  
    b:=f(4); {upisuje 4!=24 u promenljivu b}  
    writeln(f(a)); {ispisuje „6“}  
    f(5); {računa vrednost 5!, ali se sa tom vrednošću ništa ne radi}  
END.
```

Прослеђивање параметара

По вредности (без VAR) – приликом прослеђивања, прослеђена вредност се копира и прави се локална копија прослеђеног податка. Мењање копије не утиче на оригинал.

По референци (са VAR) – приликом прослеђивања додаје се ново име већ прослеђеној променљивој (референца), помоћу кога јој се може приступати из процедуре. Измена параметра повлачи и измену прослеђене променљиве.

Прослеђивање параметара

```
PROGRAM prog(output);
```

```
PROCEDURE p(x:integer; VAR y:integer);
```

```
BEGIN
```

```
  x:=10;
```

```
  y:=10;
```

```
  writeln(x,y);    ispisuje: 10 10
```

```
END;
```

```
VAR a,b:integer;
```

```
BEGIN
```

```
  a:=3;
```

```
  b:=3;
```

```
  p(a,b);
```

```
  writeln(a,b);    ispisuje: 3 10
```

```
END.
```

Живот променљивих и опсег важења

```

PROGRAM prog;
VAR x,y,z:integer;

PROCEDURE p();
VAR x,a,b,c:integer;
    PROCEDURE pp();
    VAR y,a,t:integer;
    BEGIN ... END;
BEGIN ... END;

PROCEDURE q();
VAR x,y,a,s:integer;
BEGIN ... END;

BEGIN ... END.

```

Опсези важења

(видљивост променљивих):

prog: x, y, z

p(): x, a, b, c, *, y, z

pp(): y, a, t, x, a, b, c, *, y, z

q(): x, y, a, s, *, y, z

Живот променљивих:

Променљиве живе од тренутка када њихова процедура почне са радом, до њеног краја. Редослед приступа променљивој **у**век се одређује на основу права опсега важења, а не на основу тога која је променљива најмлађа (најкасније створена).

Фебруар 2011. (4. питалица)

```
PROGRAM xy(output);  
VAR a, b:INTEGER;  
    PROCEDURE P1 (VAR a: INTEGER)  
    BEGIN  
        a:=a+1; b:=b+1  
    END;  
    PROCEDURE P2 (VAR b: INTEGER; c: INTEGER)  
        VAR a: INTEGER;  
    BEGIN  
        a:=0; c:=c+1; P1(b)  
    END;  
BEGIN  
    a:= 1; b:= 2;  
    P2(a, b);  
    writeln(output, a:2, b:2)  
END.
```

- ☒ A) 2 3 B) 0 3 C) 2 1

Октобар 2005. (5. питалица)

```
PROGRAM Okt05(Input, Output);  
  VAR a, b, c, d, e, f : INTEGER;  
  
  PROCEDURE p(a:INTEGER;VAR b:INTEGER;c:INTEGER;VAR d:INTEGER);  
    PROCEDURE q(e:INTEGER; VAR f:INTEGER);  
      BEGIN  
        e:=e*2; f:=f*3  
      END;  
    BEGIN  
      a:=a+1; b:=b+1; q(b, c); c:=c+1; d:=d+1; e:=e+1; f:=f+1  
    END;  
  
  BEGIN  
    a:=1; b:=1; c:=1; d:=1; e:=1; f:=1;  
    p(a, b, b, f);  
    WRITELN(Output, a, b, c, d, e, f)  
  END.
```

A) 124342

B) 124114

C) 121123

Фебруар 2012. (3. питалица)

```
program zad(input, output);
var a, b, c, x, y:integer;

function f(var a:integer; b, c: integer):integer;
begin
    x:= a + b; a:= b - c; f:= x + c
end;

procedure p(var x, y: integer);
begin
    a:= b - c;
    x:= x + f(y, x, y);
    y:= y - x
end;

begin
    a:= 1; b:= 2; c:= 3; x:= 4; y:= 5;
    p(a, c);
    write(a, ' ', c)
end.
```

A) 4 -8 B) -3 2 C) 0 0

Септембар 2010. (4. питалица)

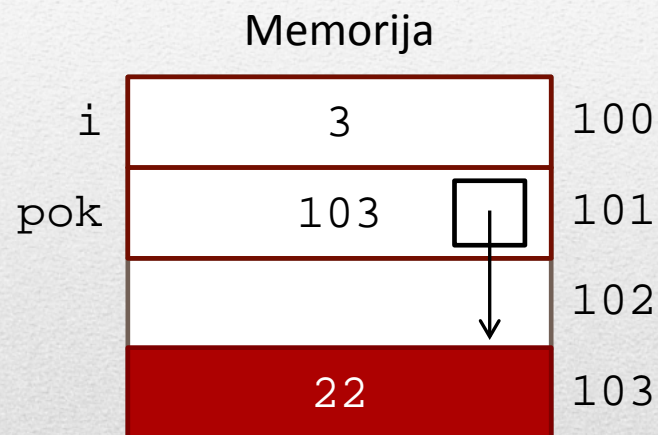
```
PROGRAM Rekurz(Output);  
VAR k, m: INTEGER;  
  
FUNCTION f (n, m:INTEGER):INTEGER;  
BEGIN  
    WRITE(Output, m);  
    IF n>3 THEN f:=n-3  
    ELSE IF m < 6 THEN BEGIN  
        f:=f(n, m+1);  
        WRITE(Output, n)  
    END ELSE f:=f(n+2, m)  
END;  
  
BEGIN  
    k:=1; m:=4;  
    k:=f(k, m); WRITE(Output, k)  
END.
```

- A) 45666112 B) 41516662 C) 41516661

Показивачи

Дефинисање показивача

```
VAR i:integer;  
    pok:^integer; {pokazivač}
```

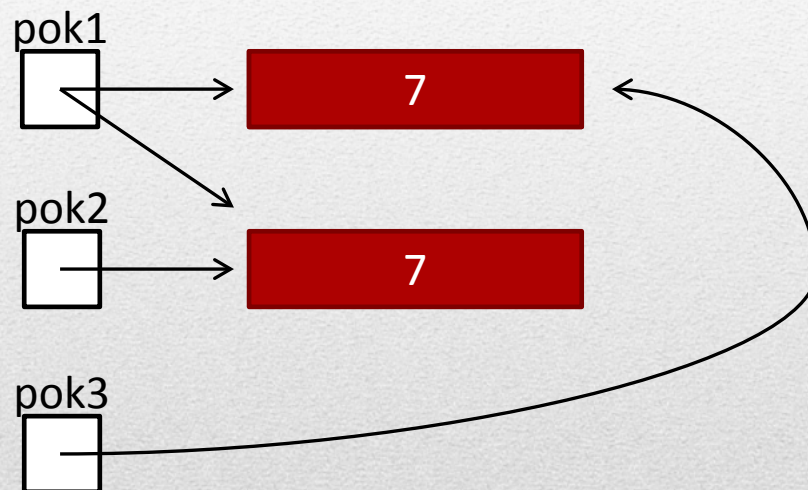


```
i:=3;  
pok:=3; {GREŠKA: 3 nije pokazivač}  
new(pok); {kreira dinamičku promenljivu}  
pok^:=22; {pok^ - ono na šta pokazuje pok}  
dispose(pok); {briše dinamičku promenljivu}
```


Промена вредности показивача

```
VAR pok1, pok2, pok3: ^integer;
```

```
new(pok1);  
new(pok2);  
pok1^:=3;  
pok2^:=7;  
pok3:=pok1;  
pok1:=pok2;  
pok3^:=pok1^;  
dispose(pok1);  
dispose(pok3);
```



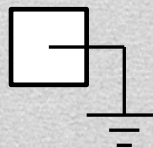
Празан показивач

Ни једна променљива никада не може бити празна.
Увек има неку вредност (која понекада није позната).

Када креирамо неки показивач, он није празан него показује на неку недефинисано локацију.



Да бисмо нагласили да показивач не показује нигде стављамо га да буде једнак `NIL` – нула показивач.



Јун 2005. (5. питалица)

```
program xzy(output);  
type pok = ^integer;  
var x:integer;  
  
procedure p1(x:pok);  
begin x^ := 123; end;  
  
procedure p2(var y:integer);  
var p:pok;  
begin  
    y := 234; new(p); p^ := x; p1(p);  
    y := p^+x; dispose(p)  
end;  
  
begin  
    x := 345; p2(x); writeln(x)  
end.
```

A) 357

B) 345

C) 579

Јун 2006. (5. питалица)

```
PROGRAM Prog(Output);  
TYPE num = ^integer;  
VAR p, q, r, s, t: num;  
PROCEDURE rotate(x, y:num; VAR z:num);  
BEGIN  
    t:=x; x:=y; y:=z; z:=t  
END;  
BEGIN  
    new(p); p^:= 1;  
    new(q); q^:= 2;  
    new(r); r^:= 3;  
    new(s); s^:= 4;  
    new(t); t^:= 5;  
    rotate(r, p, q); rotate(r, s, t);  
    writeln(p^:1, q^:1, r^:1, s^:1, t^:1);  
END.
```

A) 13343

B) 12141

C) 13333

Уланчане листе

Шта је листа

Имамо само једну статичку променљиву (показивач на почетак) помоћу које можемо да приступимо неограниченом броју динамичких променљивих.

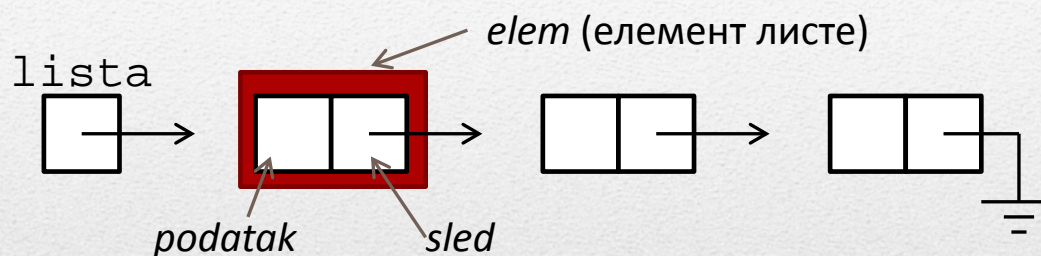
Листа је само „омотач“ за податке. Начин решења проблема не зависи од тога да ли за проблем користимо листу или низ.

За имплементацију листе користимо записе (RECORD). Сваки елемент листе поред податка садржи и показивач на следећи елемент.

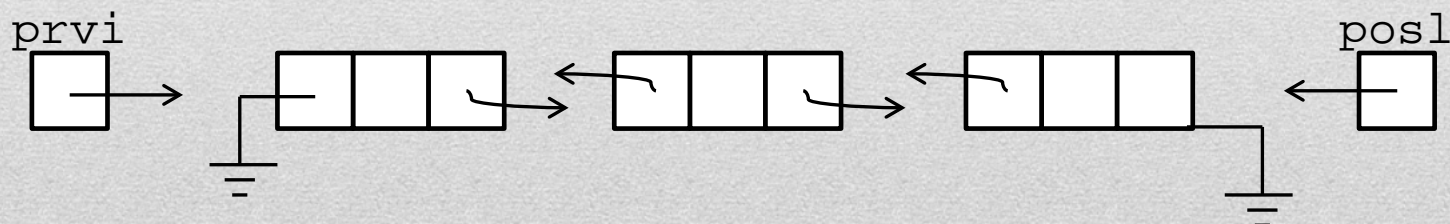
Ефикасније од низа за операције избацавања и уметања, и нема ограничење дужине. За непознат број података боље искоришћење меморије код листе него код низа.

Како листа изгледа

Једноструко уланчана



Двоструко уланчана



Како се дефинише

Једноструко уланчана:

```
TYPE pelem = ^elem;  
    elem = RECORD  
        podatak: TIP;  
        sled: pelem;  
    END;  
VAR lista: pelem;
```

Двоструко уланчана:

```
TYPE pelem = ^elem;  
    elem = RECORD  
        podatak: TIP;  
        pret, sled: pelem;  
    END;  
VAR prvi, posl: pelem;
```


Јануар 2013. (4. питалица)

Који од наведених програмских сегмената на програмском језику Паскал на исправан начин у двоструко уланчану листу испред елемента на који показује показивач `pok`, а иза њему претходног елемента, убацује елемент на који указује показивач `pom`? Претпоставити да су `pok`, `pom` и `pok^.sled` и `pok^.pred` различити од `nil`. Поље `pred` указује на претходни, а поље `sled` на следећи елемент листе.

A) `pom^.pred := pok;`
`pom^.sled := pok^.sled;`
`pok^.sled := pom;`
`pom^.sled^.pred := pom;`

B) `pok^.pred := pom;`
`pom^.sled := pok;`
`pom^.pred := pok^.pred;`
`pom^.pred^.sled := pom;`

C) `pom^.pred := pok^.pred;`
`pom^.sled := pok;`
`pom^.sled^.pred := pom;`
`pom^.pred^.sled := pom;`

Јануар 2012. (1. питалица)

Шта исписује следећи програм на програмском језику Паскал уколико једноструко уланчана листа садржи редом бројеве 7 8 2 5 4 6? Сматрати да функција `ucitaj` исправно формира листу и враћа показивач на њен почетак, а процедура `pisi` исправно исписује садржај једноструко уланчане листе, на чији почетак показује показивач `glava`.

```
program lista(input, output);
type pok = ^elem;
      elem = record
        broj: integer; sled: pok
      end;
var glava: pok;
procedure obrada (var glava: pok);
var tek : pok;
begin
  tek := glava;
  while (tek^.sled <> nil) do
    tek := tek^.sled
```

```
  while (tek^.broj < glava^.broj) do
  begin
    tek^.sled := glava;
    glava := glava^.sled;
    tek := tek^.sled;
    tek^.sled := nil
  end
end;
begin
  glava := ucitaj;
  obrada(glava);
  pisi(glava)
end.
```

A) 2 5 4 6 7 8

B) 6 4 5 2 8 7

C) 6 7 8 2 5 4

Октобар 2011. (3. питалица)

Шта ради процедура `obrada()`, ако јој се проследи показивач на почетак листе целих бројева? Елемент листе садржи цео број у пољу `v` и показивач на следећи елемент у пољу `sled`.

А) Сваки пут када наиђе на секвенцу истих елемената у листи, из секвенце избаци само први елемент.

В) Сваки пут када наиђе на секвенцу истих елемената у листи, из секвенце избаци све елементе осим првог.

С) Сваки пут када наиђе на секвенцу истих елемената у листи, из секвенце избаци само последњи елемент.

```
procedure obrada(var lst: PElement);
var tek, sl: PElement;
begin
    tek := lst;
    if lst <> nil then begin
        sl := lst^.sled;
        while sl <> nil do begin
            if tek^.v = sl^.v then begin
                tek^.sled := sl^.sled;
                dispose(sl);
                sl := tek^.sled
            end
        else begin
            tek := sl;
            sl := tek^.sled
        end
    end
end
end;
```

Срећно на испиту!

Аутор: Игор Лукић – Змајчики

Приредила: Студентска унија Електротехничког факултета
фебруар 2013.