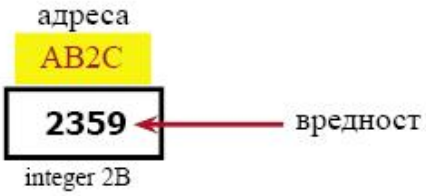
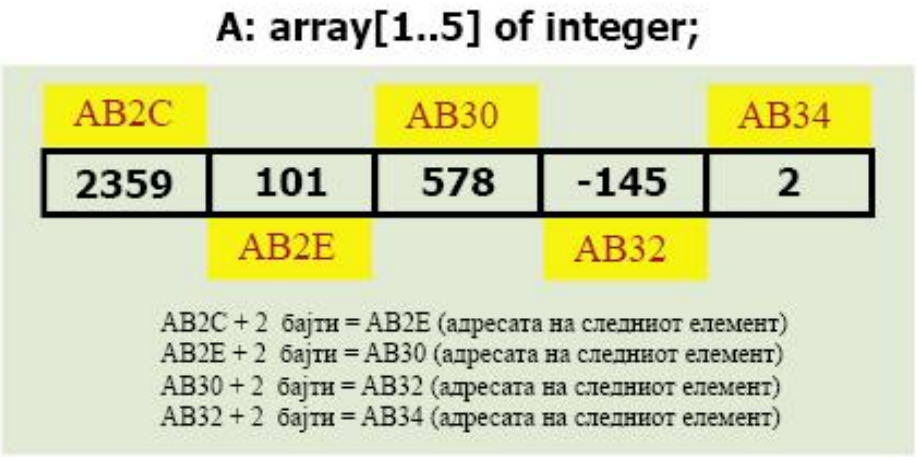
Покажувачи во Pascal

Секој податок, секоја променлива која ја декларирате во рамки на програмата има своја адреса. Познавајќи ја таа адреса, може да се пристапи до вредноста на променливата. Но, при програмирање, адресите не ги знаеме.



Низата како еден вид на податочна структура има своја поставеност во меморијата. Нејзините елементи се последователно поставени во меморија. За да можеме да ја лоцираме низата(и нејзините елементи) во меморијата доволно е да знаеме која е почетната адреса, односно адресата на првиот елемент, типот на елементите во низата, како и бројот на елементи.

На пример нека декларираме низа од 5 елементи од тип integer, податочен тип кој во програмскиот јазик Pascal зафаќа 2B. Рапоредот на елементите во меморија би изгледал на следниов начин:



Во програмскиот јазик Pascal, покрај директното пристапување кон променливи, можеме и индиректно да пристапиме кон променливите преку т.н. ***покажувачи***.

Покажувачите не се ништо друго, туку адреси, кои ако ги знаеме можеме да пристапиме до вредноста на некоја мемориска локација. Има повеќе типови на покажувачи, колку што има и типови на податоци (а нив ги има бесконечно!).

На пример сакаме да декларираме покажувач од тип integer. Тоа во Pascal би се извело на следниов начин:

```
var
pokazuvac: ^integer;
```

Но за "вистински" да одвоиме мемориска локација кон која ќе покажува pokazuvac, ќе ни треба наредбата `new(pokazuvac)`. Со оваа наредба прво се одделува мемориска локација за вредноста, а потоа адресата на истата мемориска локација се сместува во pokazuvac.

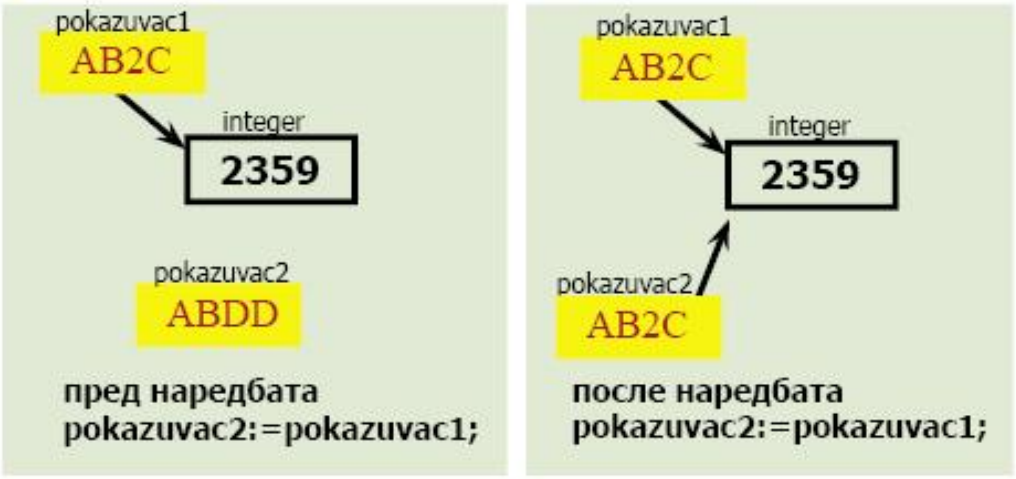
За да ја сместиме вредноста 23 на местото каде покажува `pokazuvac` ќе треба да го испишеме следниов код:

```
new(pokazuvac);
pokazuvac^:=23;
```

Обично, до сега, директно игравме со вредностите на променливите, без да менуваме некакви адреси. Интересното со покажувачите е тоа што е возможно две адреси, односно два покажувачи да покажуваат на една иста мемориска локација.

На пример, нека е даден следниов код:

```
var
      pokazuvac1:^integer;
      pokazuvac2:^integer;
begin
      new(pokazuvac1);
      new(pokazuvac2);
      pokazuvac1^:=2359;
      pokazuvac2:=pokazuvac1;{pokazuvac2 нека покажува таму каде што
ќе покажува pokazuvac1, односно pokazuvac1 и pokazuvac2 ќе содржат
исти адреси}
      writeln(pokazuvac1^);
      writeln(pokazuvac2^);
      …
```



Бидејќи и двата покажувачи покажуваат на иста мемориска локација, при извршувањето на наредбата `writeln` за двата покажувачи, двапати ќе се испечати 2359. Ако на пример, после претходно зададениот код ја извршиме наредбата `pokazuvac2^:=101`, тогаш при повторно печатење на вредностите на местата каде покажуваат `pokazuvac1` и `pokazuvac2`, двапати ќе се испечати бројот 101. Тоа е затоа што и двата покажувачи покажуваат кон иста мемориска локација, и доколку содржината на мемориската локација биде изменета преку `pokazuvac2` , истата промена ќе важи и за `pokazuvac1`. Во овој случај како да имаме два можни начини на пристап кон една мемориска локација и тие два начина(влезови) се подеднакво приоритетни.

Но откако сме ги искористиле покажувачите, треба меморијата што ја зафаќа мемориската локација кон која покажува покажувачот да ја ослободиме. За таа цел се користи наредбата `dispose(pokazuvac)`. Откако ќе се изврши оваа наредба, вредноста на покажувачот(адресата) станува недефинирана, па ако сакаме да пристапиме кон мемориската локација кон која покажува покажувачот(???), при извршување на програмата ќе се јави `run-time` грешка[1].

```
var
      pokazuvac1:^integer;
      pokazuvac2:^integer;
begin
      new(pokazuvac1);
      new(pokazuvac2);
      pokazuvac1^:=2359;
      pokazuvac2:=pokazuvac1;{pokazuvac2 нека покажува таму каде што
ќе покажува pokazuvac1, односно pokazuvac1 и pokazuvac2 ќе содржат
исти адреси}
      writeln(pokazuvac1^);
      writeln(pokazuvac2^);
      pokazuvac1^:=101;
      writeln(pokazuvac1^);
      writeln(pokazuvac2^);
      dispose(pokazuvac1);
      dispose(pokazuvac2);
end.
```

Понекогаш сакаме еден покажувач да не покажува кон ниту една мемориска локација. За таа цел, во програмскиот јазик Pascal ќе ни послужи резервираниот збор **NIL.**

```
      pokazuvac:=NIL;
      …
      if (pokazuvac<>NIL) then …
```

[1] Run-time грешка се јавува при извршување на програмата, односно кога програмата е веќе компјалирана(преведена). За разлика од овие грешки, постојат и грешки кои се јавуваат при компајлирање, а такви се на пример синтаксните грешки (на пример: сте декларирале променлива од тип integer, а ја користите за чување на податок од тип string)