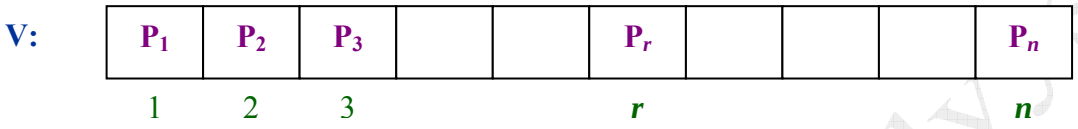
Vektory a matice

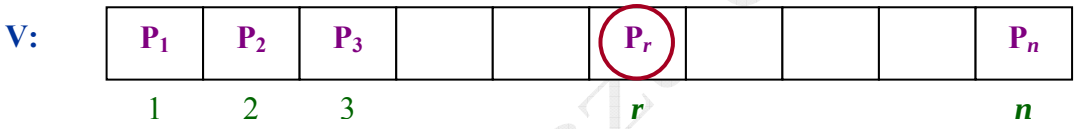
Vektory

- **Vektor** je lineární posloupnost prvků **V**, která obsahuje ***n*** prvků. Každý prvek vektoru **V** je přístupný prostřednictvím **indexu *r*** (*rank*) v rozsahu **[1, *n*]**. Vektor připomíná datový typ pole, ale není to pole.

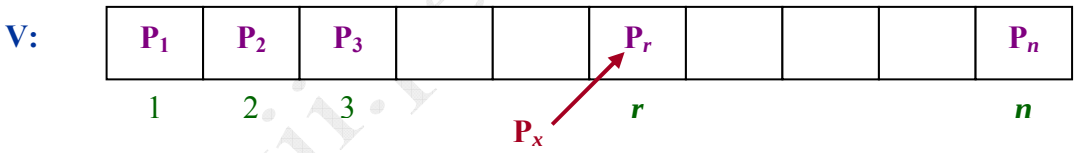


Operace s vektorem

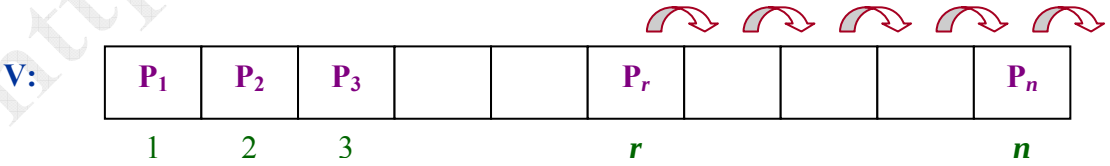
- **Prvek na pozici (*r*)** – operace vrací prvek $\mathbf{P}_r$ vektoru **V** na pozici ***r***.



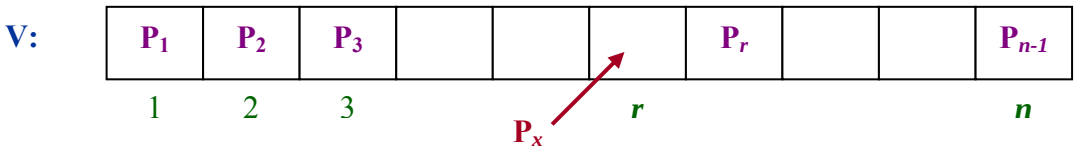
- **Záměna prvku (*r*, P_x)** – operace zamění prvek $\mathbf{P}_r$ vektoru **V** na pozici ***r*** prvkem $\mathbf{P}_x$.



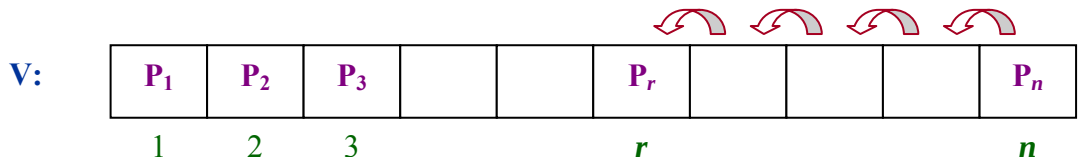
- **Vložení prvku (*r*, P_x)** – operace vloží prvek $\mathbf{P}_x$ na pozici ***r*** vektoru **V**.
  - Nejprve je potřeba vytvořit místo pro nový prvek přesunem prvků na pozicích ***r*** až ***n*** o jednu pozici vpravo.
    - Rozsah vektoru (***n***) se zvýší o jedničku.



  - Poté se vloží prvek $\mathbf{P}_x$ na uvolněnou pozici ***r***.



- **Odstranění prvku (*r*)** – operace odstraní prvek $\mathbf{P}_r$ vektoru **V** na pozici ***r***.
  - Nejprve se odstraní prvek $\mathbf{P}_r$ na pozici ***r***.
  - Poté se posunou všechny prvky na pozicích ***r*+1** až ***n*** o jednu pozici vlevo.
    - Rozsah vektoru (***n***) se sníží o jedničku.



- **Odstranění vektoru** – operace odstraní celý vektor **V**.
  - Opakujeme operaci **odstranění prvku (*r*)** až do snížení rozsahu vektoru (***n***) na nulu (***n*=0**)

Implementace vektoru

- Implementace vektoru se provádí obvykle pomocí **pole** pevné nebo proměnné délky. Implementaci je možné provést i pomocí **spojového seznamu** (viz spojové seznamy).

- **Deklarace vektoru:**

```
type    Prvek = typ prvku;  {integer, real, record, …}

var     Vektor: array [1..MAX] of Prvek;
        n: integer;
```

  - Prvkem může být jakákoliv hodnota, tj. celé číslo (*integer*), reálné číslo (*real*), záznam (*record*) nebo jiná.
  - Hodnota ***MAX*** udává maximální rozsah vektoru. Použití této hodnoty je nutné při deklaraci pole s pevnou délkou.
  - Hodnota ***n*** udává skutečnou velikost vektoru v rozsahu $n \in \langle 1, MAX \rangle$

Implementace operací s vektorem

- **Prvek na pozici (*r*)**

```
function Prvek_na_pozici(r:integer):Prvek;
begin
  Prvek_na_pozici:=Vektor[r]
end;
```

- **Záměna prvku (r, P_x)**

```
procedure Zamena_prvku(r:integer;Px:Prvek);
begin
  Vektor[r]:=Px
end;
```

- **Vložení prvku (r, P_x)**

```
procedure Vlozeni_prvku(r:integer;Px:Prvek);
var i:integer;
begin
   for i:=n downto r do Vektor[i+1]:=Vektor[i];
   n:=n+1;
   Vektor[r]:=Px
end;
```

- **Odstranění prvku (r)**

```
procedure Odstraneni_prvku(r:integer);
var i:integer;
begin
   {zrušení prvku, například: Vektor[r]:=0}
   for i:=r to n do Vektor[i]:=Vektor[i+1];
   n:=n-1
end;
```

  - Instrukce pro zrušení prvku (uvolnění místa v paměti) není nutná u staticky alokovaných prvků, tj. u prvků, kterým je paměťové místo přiřazeno již při deklaraci. U dynamicky alokovaných prvků (paměť přidělena až za běhu programu) je instrukce zrušení prvku nutná.

- **Odstranění vektoru**

```
procedure Odstraneni_vektoru;
var i:integer;
begin
   for i:=1 to n do Odstraneni_prvku(i)
end;
```

  - V případě staticky alokovaných prvků stačí pro zrušení celého vektoru instrukce:

```
n:=0
```

Matice

- **Maticí** typu $m \times n$ nazýváme obdélníkové pole $m \times n$ **prvků matice**, uspořádaných do m řádků a n sloupců. Matice označujeme velkými písmeny, prvky matice odpovídajícími malými písmeny **se dvěma indexy**, přičemž první index udává číslo řádku, druhý číslo sloupce, ve kterém se prvek nachází. Je-li typ matice známý, lze používat stručný zápis $\mathbf{A} = (a_{ij})$.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

- Na základě vlastností (rozměry matice, typ prvků) matice je možné zavést následující pojmy (známé z lineární algebry):

  - Matici typu $(1 \times n)$ budeme nazývat **řádkovým vektorem** (o n složkách) a matici typu $(m \times 1)$ budeme nazývat **sloupcovým vektorem** (o m složkách).

$$A_r = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} \qquad A_s = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}$$

  - Prvky matice se stejnými indexy (a_{ii}) tvoří **hlavní diagonálu** matice.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & a_{ii} & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

  - Má-li matice všechny prvky pod (respektive nad) hlavní diagonálou rovny nule, nazýváme matici **horní** (respektive **dolní**) **trojúhelníkovou maticí**.

$$A_H = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \qquad A_D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

  - Matici, která má stejný počet řádků, jako sloupců, nazýváme **čtvercovou maticí**.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

- Matici, jejíž všechny prvky jsou rovny nule nazýváme **nulovou maticí**.

$$A = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

- Čtvercovou matici, která má všechny prvky mimo hlavní diagonálu rovny nule nazýváme **diagonální maticí**.
  - Diagonální matici, která má na hlavní diagonále všechny prvky rovny jedné nazýváme **jednotkovou maticí** a označujeme ji **E**.

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \qquad E = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

- **Transponovanou maticí** k matici $\mathbf{A} = (a_{ij})$ typu $(m \times n)$ rozumíme matici $\mathbf{A}^T = (b_{ij})$ typu $(n \times m)$, pro jejíž prvky platí $b_{ij} = a_{ji}$.
- Čtvercovou matici **A** nazveme **symetrickou**, jestliže pro ni platí $\mathbf{A} = \mathbf{A}^T$.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \qquad A^T = \begin{pmatrix} b_{11} = a_{11} & b_{12} = a_{21} & \cdots & b_{1m} = a_{m1} \\ b_{21} = a_{12} & b_{22} = a_{22} & \cdots & b_{2m} = a_{m2} \\ b_{31} = a_{13} & b_{32} = a_{23} & \cdots & b_{3m} = a_{m3} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} = a_{1n} & b_{n2} = a_{2n} & \cdots & b_{nm} = a_{mn} \end{pmatrix}$$

Sčítání a násobení matic

- **Součinem celého čísla α a matice** $\mathbf{A} = (a_{ij})$ typu $(m \times n)$ nazýváme matici $\alpha \cdot \mathbf{A} = (\alpha \cdot a_{ij})$ typu $(m \times n)$.

$$\alpha \cdot A = \begin{pmatrix} \alpha \cdot a_{11} & \alpha \cdot a_{12} & \alpha \cdot a_{13} & \cdots & \alpha \cdot a_{1n} \\ \alpha \cdot a_{21} & \alpha \cdot a_{22} & \alpha \cdot a_{23} & \cdots & \alpha \cdot a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha \cdot a_{m1} & \alpha \cdot a_{m2} & \alpha \cdot a_{m3} & \cdots & \alpha \cdot a_{mn} \end{pmatrix}$$

- **Součtem matic** $\mathbf{A} = (a_{ij})$ a $\mathbf{B} = (b_{ij})$ stejného typu $(m \times n)$ nazýváme matici $\mathbf{A} + \mathbf{B} = (a_{ij} + b_{ij})$ typu $(m \times n)$.

$$A + B = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & a_{m3} + b_{m3} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

Rozdílem matic **A** a **B** stejného typu ($m \times n$) nazýváme matici **A** + (–1) · **B** typu ($m \times n$). Označujeme ji **A** – **B**.

$$A - B = \begin{pmatrix} a_{11} - b_{11} & a_{12} - b_{12} & a_{13} - b_{13} & \cdots & a_{1n} - b_{1n} \\ a_{21} - b_{21} & a_{22} - b_{22} & a_{23} - b_{23} & \cdots & a_{2n} - b_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & a_{m3} - b_{m3} & \cdots & a_{mn} - b_{mn} \end{pmatrix}$$

Součinem matic **A** = (a_{ij}) typu ($m \times n$) a **B** = (b_{ij}) typu ($n \times p$) nazýváme matici **C** = (c_{ij}) typu ($m \times p$), pro jejíž prvky platí:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \quad (i = 1,..., m; j = 1,..., p)$$

Označujeme **C** = **A** · **B.**

- Prvek c_{ij} je skalárním součinem i-tého řádku matice **A** a j-tého sloupce matice **B**.
- **Násobení matic není komutativní**, tj. obecně **A** · **B** ≠ **B** · **A**.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ b_{31} & b_{32} & \cdots & b_{3p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

$$C = A \cdot B = \begin{pmatrix} c_{11} = (a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1}) & c_{12} & \cdots & c_{1p} = (a_{11}b_{1p} + a_{12}b_{2p} + \cdots + a_{1n}b_{np}) \\ c_{21} = (a_{21}b_{11} + a_{22}b_{21} + \cdots + a_{2n}b_{n1}) & c_{22} & \cdots & c_{2p} = (a_{21}b_{1p} + a_{22}b_{2p} + \cdots + a_{2n}b_{np}) \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} = (a_{m1}b_{11} + a_{m2}b_{21} + \cdots + a_{mn}b_{n1}) & c_{m2} & \cdots & c_{mp} = (a_{m1}b_{1p} + a_{m2}b_{2p} + \cdots + a_{mn}b_{np}) \end{pmatrix}$$

Implementace matice

- Implementace matice se provádí obvykle pomocí **dvojrozměrného pole** . Implementaci je možné provést i pomocí **spojového seznamu**.

- **Deklarace matice:**

```
type    Prvek = typ prvku;  {integer, real, record, …}
        Matice = array [1..Max_M, 1..Max_N] of Prvek;

var     m, n: integer
```

  - Prvkem může být jakákoliv hodnota, tj. celé číslo (*integer*), reálné číslo (*real*), záznam (*record*) nebo jiná.
  - Hodnoty ***Max_M*** a ***Max_N*** udávají **maximální** velikost matice (počet řádků a sloupců). Použití těchto hodnot je nutné již při deklaraci matice.
  - Hodnoty ***m*** a ***n*** udávají **skutečnou** velikost matice v rozmezí $m \in \langle 1, Max\_M \rangle$, $n \in \langle 1, Max\_N \rangle$

Implementace vybraných operací s maticemi

- **Stanovení transponované matice A^T k matici A**

```
function Transponovana(A:Matice; m,n:integer):Matice;
var i,j:integer;
begin
   for i:=1 to m do for j:=1 to n do
        Transponovana[j,i]:=A[i,j]
end;
```

- **Součin celého čísla α a matice A**

```
function Soucin(alfa:integer; A:Matice; m,n:integer):Matice;
var i,j:integer;
begin
   for i:=1 to m do for j:=1 to n do
        Soucin[i,j]:=A[i,j]*alfa
end;
```

- **Součet a rozdíl matic A a B**

```
function Soucet(A,B:Matice; m,n:integer):Matice;
var i,j:integer;
begin
   for i:=1 to m do for j:=1 to n do
        Soucet[i,j]:=A[i,j]+B[i,j]
end;
```

  - Pro rozdíl matic stačí pozměnit tělo cyklu: `Rozdil[i,j]:=A[i,j]-B[i,j]`

Násobení matic A a B

```
function Nasobeni(A,B:Matice; m,n,p:integer):Matice;
var i,j,k:integer;
    Pom: Prvek;
begin
  for i:=1 to m do for j:=1 to p do
  begin
       Pom:=0;
       for k:=1 to n do
            Pom:=Pom+A[i,k]*B[k,j];
       Nasobeni[i,j]:=Pom
  end;
end;
```

- Při návrhu algoritmu byla uplatněna optimalizační **metoda odstranění opakovaných výpočtů** (použití proměnné **`Pom`** místo opakovaného použití prvku pole **`Nasobeni[I,J]`**), přesto má algoritmus pro násobení matic obvykle **kubickou asymptotickou operační složitost**, tj. operační složitost roste stejně rychle jako funkce x^3.