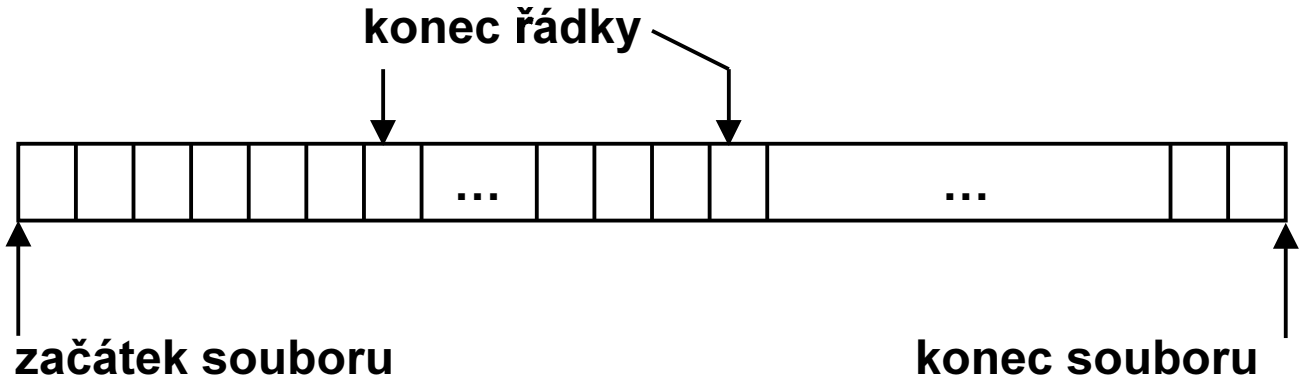
# Textový soubor

- **soubor, který obsahuje pouze znaky, je čitelný a vytvořitelný libovolným textovým editorem.**
- **textový soubor je složen z řádků, popř. stránek vzájemně oddělených řídícím znakem konec řádku ( dvojice znaků *cr (cariage return) - #13* a *lf (line feed) - #10* ), popř. řídícím znakem znakem konec stránky *ff ( form feed - #12)*.**




## Práce s textovým souborem

1. **deklarace souboru**

***var F: text;***

**F - identifikátor virtuálního souboru ( v programu pracujeme pouze s tímto identifikátorem, ne se jménem fyzického souboru)**

2. **Vytvoření vazby mezi fyzickým a virtuálním souborem**

   **Assign(F,'*jméno_souboru*');**

   ***jméno_souboru* - jméno fyzického souboru (pokus.txt). Pod tímto jménem je soubor uložen na disku (pásce)**

3. **Otevření souboru**

   **Reset( *F* )**

   **Standardní procedura, otevírá soubor *F* pouze pro čtení. Soubor musí existovat, jinak dojde k chybě (Run time error) !!!**
   **Soubor se otevírá na začátku - aktuální se stává první položka.**

   **Rewrite( *F* )**

   **Standardní procedura, otevírá soubor *F* pro zápis (přepis).**
   **Pokud soubor neexistuje, tak se vytvoří nový soubor, jinak se soubor otevírá na začátku - aktuální je první položka, ostatní položky nejsou přístupné.**
   **Z takto otevřeného souboru nelze číst !!!**

   **Append( F )**

   **Procedura, která otevírá soubor pro doplňování.**
   **Soubor musí existovat, jinak dojde k chybě (Run time error) !!!**

**Soubor se otevře se na konci (tj. použitím procedury write/ writeln se na konec existujícího souboru doplňují nové údaje).**
**Z takto otevřeného souboru nelze číst !!!**

4. **Práce se souborem - lze pracovat pouze s otevřeným souborem !!!! Práce s textovým souborem je pouze sekvenční (nelze použít proceduru *seek* )**

   ***eof(F)* - logická funkce, vrací *true* pokud se nacházíme na konci souboru (neexistuje další složka)**

   ***eoln(F)* - logická funkce, vrací true, pokud se stal aktuální položkou znak konce řádku**

   ***read(F,X)* - *X* může být typu *real, integer , char.***
   - **Jeli X typu *integer* ze souboru F se přečte posloupnost znaků odpovídajících celému číslu a hodnota čísla se přiřadí do X, čtení se ukončí, je-li hodnota přečteného znaku odlišná od přípustných znaků zápisu čísla.**
   - **Je-li X typu *char*, ze souboru se přečte jeden znak a ten se uloží do proměnné X.**
   - **Je-li X typu *real* ze souboru F se přečte posloupnost znaků odpovídajících reálnému číslu (v desetinném nebo semilogaritmickém tvaru) a hodnota čísla se přiřadí do proměnné X.**

   ***readln(F,X)* - pro X platí totéž jako u procedury *read,* po přečtení odpovídajících údajů se přeskočí na další řádek.**

***write(F,X)* - procedura nejprve vyčíslí hodnotu proměnné (výrazu) *X,* získaná hodnota se převede na posloupnost znaků a zapíše se do souboru *F*. Výraz může být typu *char*, *string*, *integer*, *real* a *boolean.***

***writeln(F,X)* - pro X platí totéž jako u write, po výpisu všech hodnot se do souboru zapíše znak nový řádek (dvojice *cr*, *lf*)**

***SeekEof(F,N)* – logická funkce, vrací hodnotu true, zbývají-li do konce souboru pouze mezery, tabulátory či konce řádků.**

***SeekEoln(F,N)* – logická funkce, vrací hodnotu true, zbývají-li do konce řádku pouze mezery či tabulátory.**

**5. Uzavření souboru**

***Close(F)* - zápis obsahu vyrovnávací paměti do souboru na disku**

# Soubory bez udaného typu

- **jsou používány tehdy, jedná-li se o soubory vytvořené pomocí jiných programových systémů nebo s informacemi, které nelze do posloupnosti identických složek efektivně rozdělit.**
- **u tohoto typu souborů se pracuje s bloky o délce N bytů**

## Práce se souborem bez udaného typu

1. **deklarace souboru**

***var F: file;***

**F - identifikátor virtuálního souboru ( v programu pracujeme pouze s tímto identifikátorem, ne se jménem fyzického souboru)**

2. **Vytvoření vazby mezi fyzickým a virtuálním souborem**

**Assign(F,'*jméno_souboru*');**

***jméno_souboru* - jméno fyzického souboru (pokus.dat). Pod tímto jménem je soubor uložen na disku (pásce)**

3. **Otevření souboru**

**Reset( *F, N* )**

**Standardní procedura, otevírá soubor *F* pro čtení po blocích s délkou N. Soubor musí existovat, jinak dojde k chybě (Run time error) !!!**
**Soubor se otevírá na začátku - aktivní se stává první blok.**

**Rewrite( *F,N* )**

**Standardní procedura, otevírá soubor *F* pro zápis (přepis) po blocích o délce N.**
**Pokud soubor neexistuje, tak se vytvoří nový soubor, jinak se soubor otevírá na začátku - aktuální je první položka, ostatní položky nejsou přístupné.**
**Z takto otevřeného souboru nelze číst !!!**

4. **Práce se souborem - lze pracovat pouze s otevřeným souborem !!!!**

***eof(F)* - logická funkce, vrací *true* pokud se nacházíme na konci souboru (neexistuje další blok)**

***blockread(F,B,N[,RESULT])* - čtení *N* bloků souboru *F* do proměnné *B.* POZOR !!! Není kontrolováno, zda rozsah bloku nepřesahuje velikost proměnné B. Nepovinná proměnná *RESULT* obsahuje skutečný počet přečtených bloků. Pokud není tato proměnná a v souboru není obsažen odpovídající počet bloků dojde k chybě. Aktuální pozice v souboru se posune o *RESULT* bloků.**

***blockwrite(F,B,N[,RESULT])*** **– zápis N bloků z paměti počínaje adresou proměnné B libovolného typu. Nepovinná proměnná RESULT obsahuje počet skutečně zapsaných bloků do souboru. Pokus není tento parametr uveden a zapíše se méně bloků než je uvedeno v A, dojde k chybě**

***Seek(F,N)*** **– změna aktivního bloku na N ( N je typu *longint).***

## 6. Uzavření souboru

***Close(F)*** **- zápis obsahu vyrovnávací paměti do souboru na disku**

# Datový typ množina

- **Strukturovaný datový typ, statická homogenní struktura nad bázovým typem. Bázovým typem je ordinální typ.**
- **Množina hodnot datového typu množina jsou libovolné podmnožiny prvků bázového typu, včetně množiny prázdné.**

**Deklarace:**

***type S = set of T;***

***var M : S;***

***T* - bázový typ**

**Konkrétní prvky typu množina se zapisují obdobně jako v matematice uvedením seznamu jednotlivých prvků. Na rozdíl od matematiky se však seznam uzavírá do hranatých závorek.**

**Př. Prvky typu množina**

**[ ] prázdná množina**
**[prvek1], [prvek2], [prvek3]**
**[prvek1, prvek2, prvek3]**

**Př. Definice typu množina**

```
type MNOZINA_CISLIC   = set of 0 .. 9;
type MNOZINA_ZNAKU    = set of char;
type MNOZINA_PISMEN   = set of 'A' .. 'Z';
```

**Pro datový typ množina jsou definovány výrazy typu množina tj. hodnotou výrazu může být množina. Výraz vyjadřující konstrukci množiny – konstruktor.**

**<u>Operace s datovým typem množina</u>**

- **vytvoření prázdné množiny**
  **[ ]**

- **vytvoření (konstruktor) neprázdné množiny**
  **[ $v_1$, $v_2$, …, $v_N$ ]**

  **$v_i$ je výraz stejného ordinálního typu**

  **Seznam vi definuje prvky, které patří do množin. Hodnoty, které patří do množiny je možné definovat i pomocí intervalu**

  **[ dmi .. hmi ]**

**dmi - dolní mez intervalu**
**hmi - horní mez intervalu**

**Př.**

```
var M1 : MNOZINA_ZNAKU;
var M2 : MNOZINA_PISMEN;
var M3 : MNOZINA_CISEL;


M1:= [ '1', '2' ];
M2:= [ ];
M3:= [1,10 ];
M1:=M2;
```

**Při přiřazování množin je nutné zachovat pravidla typové kompatibility. Kompatibilní jsou takové množiny, jejichž bázové typy jsou kompatibilní.**
**Je-li bázovým typem interval, je kontrolován rozsah hodnot přiřazované množiny (žádný prvek nesmí být mimo interval).**

– **množinové operace**

**+ sjednocení**

*** průnik**

**- rozdíl**

**Operandy množinových operací jsou množiny s kompatibilními bázovými typy, výsledek je typu množina s bázovým typem kompatibilním s typem operandů.**

**Př.**

```
var  A : set of  1..8;
var  B : set of 15 .. 30;
var  C : set of 0 .. 30 ;
var  D : set of 'A' .. 'Z' ;
```

**A:= [1, 4, 6 .. 8 ]; B:= [ 20, 25 .. 30 ]; C:=[ 3..7, 15..27];**
**D:= [‘A’ .. ‘Z’ ] ;**

| Operace | výsledek |
|---|---|
| A*C | [ 4, 6, 7 ] |
| B*C | [20, 25, 26, 27] |
| A*D | nekompatibilní bázové typy |
| A+B+C | [1, 3 .. 8, 15 .. 30] |
| D + [ ‘K’ .. ‘Z’] | [ ‘A’ .. ‘Z’ ] |
| A-B | A |
| B-A | B |
| A-C | [ 1.. 8 ] |

- **Relační operace ( operandy jsou typu množina, výsledek je typu boolean )**

**=** **rovnost – množiny prvního i druhého operandu obsahují totožné prvky**

**<>** **nerovnost – množiny prvního i druhého operandu neobsahují všechny totožné prvky**

**<=** **inkluze c “ je obsažena” – všechny prvky prvního operandu jsou obsaženy ve druhém operandu.**

**>=** **inkluze b “obsahuje” - množina prvků prvního operandu obsahuje všechny prvky množiny druhého operandu**

**Př. následující výrazy nabývají hodnoty *true***

**A<>B A*B=[ ] [ 6 .. 8 ] <=A**

– **Relační operátor *in***

***V in S***

**V - výraz, jehož typ je kompatibilní s bázovým typem množiny S**

**Výsledkem je hodnota typu boolean ( *true* - pokud hodnota výrazu patří do množiny, *false* - pokud hodnota nepatří do množiny ).**

## Implementace datového typu množina

**Datový typ množina - řetězec logických hodnot (tzv. charakteristický vektor)**

**Př. set of 1 .. 8 [ 1, 3, 4, 5 ]**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| I | 0 | I | I | I | 0 | 0 | 0 |

**patří do množiny** **nepatří do množiny**

**Každá množina je v paměti zobrazena na celistvém počtu bytů. Velikost paměti zabírané objektem množina lze spočítat ze vztahu**

***V = ( N div 8 ) - ( M div 8 ) + 1***

***N* - ordinální číslo největšího prvku množiny**
***M* - ordinální číslo nejmenšího prvku množiny**

**Adresa bytu ve kterém je uložen bit s informací o prvku X s ordinálním číslem E se vyčíslí podle vztahu**

$$A = ( E \text{ div } 8 ) - ( M \text{ div } 8 ) + PA$$

***PA* - počáteční adresa množiny reprezentovaná identifikátorem proměnné**

**Pozice bitu uvnitř určeného bytu je dána vztahem**

$$B = ( E \bmod 8 )$$

# Programové jednotky TPU

**Programová jednotka - knihovna procedur a funkcí**

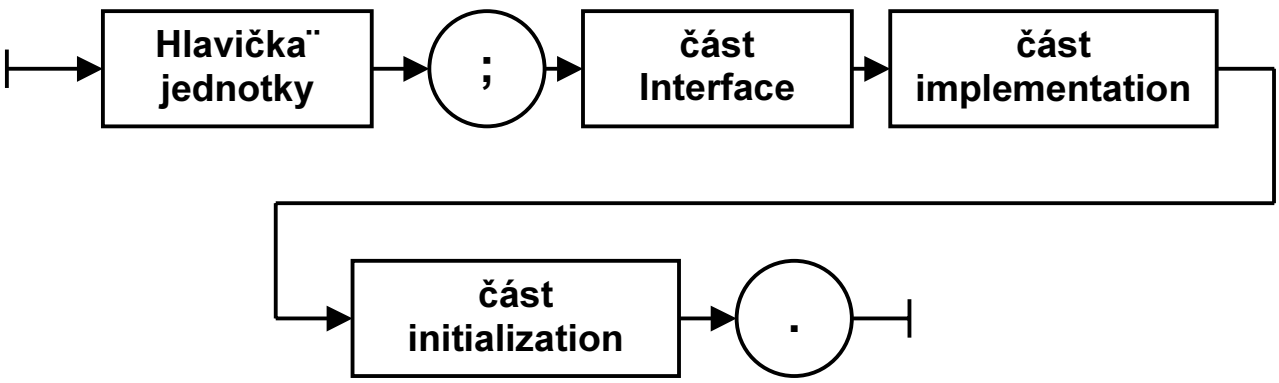
**Důvody zavedení TPU :**

- **umožňuje vytvářet knihovny**
- **u starších verzí Pascalu - maximální velikost programu = velikost segmentu ( 64 KB), větší programy - rozdělit na jednotky, každá jednotka je umístěna v samostatném segmentu tj. maximální velikost každé jednotky je 64 KB.**

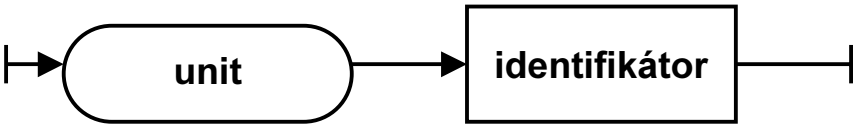
**Jednotky v Turbo Pascalu :¨**

- **standardní - jsou součástí distribuce TP**
- **uživatelsky definované**

## Uživatelsky definované jednotky



## Hlavička jednotky




## Část interface

- **obsahuje deklarace konstant, typů, proměnných a hlavičky procedur a funkcí, které jsou veřejné tj. jsou dostupné v programu ( v jednotce), ke kterému je jednotka připojena.**