Typ char (1)

- Je konečná uspořádaná množina znaků, které umí vstupní zařízení akceptovat a výstupní zařízení publikovat
- Jejími hodnotami jsou znaky uzavřené mezi apostrofy. Např. ’*’, ’A’, ’B’, ’’’’ (apostrof musí být zapsán dvakrát).
- Na dvojice hodnot typu char lze aplikovat relační operátory (=, < >, <, <=, >=, >) které dávají hodnotu typu Boolean

Typ char (2)

- Musí splňovat následující požadavky:
  - podmnožina dekadických číslic ’0’ až ’9’ musí být numericky uspořádaná a souvislá, tj.:
    - succ (’0’) = ’1’ až succ (’8’) = ’9’
    - pred (’1’) = ’0’ až pred (’9’) = ’8’
  - podmnožina velkých písmen ’A’ až ’Z’ a podmnožina malých písmen ’a’ až ’z’ musí být abecedně uspořádané (nemusí být souvislé), tj.
    - ’A’ < ’B’, ’B’ < ’C’, … , ’Y’ < ’Z’
    - ’a’ < ’b’, ’b’ < ’c’, … , ’y’ < ’z’

Typ char (3)

- Dnes se používá jako typ char rozšířená množina znaků ASCII (American Standard Code for Information Interchange):
  - platí: $X < Y$ pro $\forall X \in \langle$'A'; 'Z'$\rangle$, $\forall Y \in \langle$'a'; 'z'$\rangle$ neplatí tedy např. 'a' < 'Z'
  - pokud rozšířená část ASCII obsahuje znaky národní abecedy, tak jejich množina není souvislá ani uspořádaná

Typ char (4)

- Standardní funkce:
  - ord (z): vrací ordinální číslo znaku v uspořádané množině znaků
  - chr (i): vrací znak, jehož ordinální číslo je i
  - platí ord (chr (i)) = i a chr (ord (z)) = z
- Uspořádání je definováno takto:

$$z_1 < z_2 \Leftrightarrow \text{ord}(z_1) < \text{ord}(z_2)$$

Typ char (5)

- Poznámka:
  ordinální číslo odpovídající číslici nepředstavuje jeho numerickou hodnotu. Tuto hodnotu můžeme vypočítat jako:
  ord (z) - ord (’0’), kde z ∈ ⟨’0’; ’9’⟩

Výčtové typy (1)

- Výčtový typ se definuje seznamem identifikátorů reprezentujících hodnoty tohoto typu
- Výskyt identifikátoru představuje jeho zavedení jako identifikátoru konstanty příslušného typu

Výčtové typy (2)

- Vlastní definice výčtového typu se provádí v části definic typů, která předchází části deklarací proměnných a je uvozena klíčovým slovem type:
  - obecný tvar definice typu:
    type identifikátor = popis typu;
    kde identifikátor značí identifikátor nově definovaného typu

Výčtové typy (3)

- pro výčtové typy má tvar:
  type vyctovy_typ = (id_1, id_2, ... , id_n);
- např.:
  type predmet = (matematika, fyzika, chemie);
  barva = (cervena, zluta, zelena, modra);

- Žádný identifikátor se nesmí vyskytovat v seznamu prvků více než u jednoho výčtového typu

Výčtové typy (4)

- Každý výčtový typ je ordinální a proto na operandy (téhož) výčtového typu mohou být aplikovány relační operátory
- Standardní funkce ord (x), succ (x) a pred (x) mohou mít argumenty výčtových typů

Výčtové typy (5)

- Uspořádání hodnot výčtového typu je dáno pořadím jednotlivých identifikátorů v definici typu:
  - první identifikátor má ordinální číslo 0
  - ordinální čísla dalších se zvyšují po jedné

Výčtové typy (6)

- Př.: type Den = (Po, Ut, St, Ct, Pa, So, Ne);
  Platí:
  - ord (Po) = 0, ord (Ut) = 1, ... , ord (Ne) = 6
  - succ (Po) = Ut, ... , succ (So) = Ne
  - pred (Ut) = Po, ... , pred (Ne) = So
  - Po < Ut, Ut < St, ... , So < Ne
- Pozn.: příkladem výčtového typu je i typ Boolean, který je definován takto:
  type Boolean = (false, true);

Výčtové typy (7)

- Pozor:
  proměnné výčtových typů nemohou být uvedeny jako argumenty příkazů vstupu a výstupů (toto omezení neplatí pro typ Boolean)

Typ interval (1)

- Typ, který specifikuje neprázdnou souvislou podmnožinu nějakého ordinálního typu
- Dolní a horní mez podmnožiny udávají dvě konstanty (v TP a BP konstantní výrazy) ordinálního typu
- Horní mez musí být větší nebo rovna dolní mezi

Typ interval (2)

- Např. popisem 1..10 se definuje interval, kterému přísluší hodnoty i typu integer (v TP a BP typu byte) pro něž platí:

$$1 \leq i \leq 10$$

- Interval, který specifikuje nějakou podmnožinu ordinálního typu T nazýváme intervalem z T a typ T hostitelským typem tohoto intervalu

Typ interval (3)

- Příklady typu interval:

```
type cislice = '0'..'9';
     prirozena = 1..maxint;
     pracovni_dny = Po..Pa;
nebo:
const x = 50;
      y = 10;
type rozsah = 2*(x-y)..2*(x+y);
```

Typ interval (4)

- Typ interval je typ, který je kompatibilní se svým hostitelským typem. To znamená, že pro objekty typu interval jsou definovány tytéž operátory jako pro objekty hostitelského typu.
- Zavedením intervalu jako typu proměnné se tedy neomezuje množina operací aplikovatelných na tuto proměnnou, omezuje se pouze množina přípust. hodnot proměnné

Typ interval (5)

- Proměnné typu interval lze přiřadit pouze takovou hodnotu typu T, která patří do příslušného intervalu, jinak dojde k chybě

Příkaz case (1)

- Umožňuje větvení výpočtu do několika různých větví v závislosti na hodnotě ordinálního typu. Obecný tvar:

case α of

$k_{11}, k_{12}, \ldots, k_{1n_1}$: P_1; (* $n_1 \geq 1$ *)

$k_{21}, k_{22}, \ldots, k_{2n_2}$: P_2; (* $n_2 \geq 1$ *)

$k_{m1}, k_{m2}, \ldots, k_{mn_m}$: P_m; (* $n_m \geq 1$ *)

else P_{m+1}; (* nepovinná část *)

end;

Příkaz case (2)

- Výraz α příkazu case musí být ordinálního typu. Téhož typu musí být konstanty u příkazů P_1 až P_m
- Žádná konstanta nesmí být více než u jednoho příkazu
- V případě, že výraz α nabude některé z hodnot k_{ij} $(1 \leq j \leq n_i, 1 \leq i \leq m)$ je vybrán a proveden příkaz P_i $(1 \leq i \leq m)$

Příkaz case (3)

- Jestliže výraz α nabude hodnoty různé od všech hodnot k_{ij} a je uvedena větev else, provede se příkaz P_{m+1}. Pokud větev else uvedena není, je příkaz case ekvivalentní prázdnému příkazu (tj. neprovede se nic)

Příkaz case (4)

- Poznámky:
  - TP a BP vyžadují, aby ordinální typ výrazu α byl byte-sized (uložen na 8 bitech) nebo nebo word-sized (uložen na 16 bitech), takže není možné použít typ longint
  - Konstanty k_{ij} mohou být zadány i formou intervalu:
    Př.: ’a’, ’b’, ’c’, ’d’, ’e’, ’f’ odp. ’a’..’f’

Příkaz for (1)

- Příkaz pro cyklus s předem známým počtem opakování
- Obecný tvar:

  for i := dm to hm do P

  kde
  - i je proměnná ordinálního typu, tzv. řídící proměnná cyklu
  - dm, hm jsou výrazy ordinálního typu, který musí být kompatibilní vzhledem k přiřazení s typem proměnné i

Příkaz for (2)

- Výrazy dm a hm reprezentují dolní a horní mez hodnot, kterých bude v průběhu provádění cyklu postupně nabývat řídící proměnná i
- Hodnoty dm a hm nesmí být v průběhu cyklu měněny (pokud ano, nemá to význam)
- Hodnota řídící proměnné i by v průběhu cyklu neměla být měněna

Příkaz for (3)

- Příkaz for je ekvivalentní:

```
if dm <= hm then
   begin
      i := dm;
      P;
     while i < hm do begin i := succ(i);
                                P;
                          end;
   end;
```

Příkaz for (4)

- Je zřejmé, že pro dm > hm je příkaz for ekvivalentní prázdnému příkazu
- Příkaz for může mít také tvar:

  for i := hm downto dm do P
- Sémantika je analogická:

  místo:

  – i := dm;
  – while i < hm do
  – i := succ(i);

  i := hm;
  while i > dm do
  i := pred(i);