Anotace

- Definice vlastních datových typů (výčtové datové typy)
- Konstrukce `case ... of ...`
- Fronta a zásobník,
- Základní třídící algoritmy

Další využití tvorby vlastních typů

Chceme počítat dny v týdnu. Jak to uděláme?

- Očíslujeme si dny takto: Pondělí=1, Úterý=2,...
- Jenže já to přečísluju: Pondělí=0, Úterý=1,...
- Přijde američan a očísluje: Neděle=1, Pondělí=2,...
- ... anebo Neděle=0, Pondělí=1,...
- Proto raději uděláme zvláštní typ indexovaný dny v týdnu a čísla necháme na překladači.

Výčtový datový typ

- Definujeme v sekci `type`,
- jednotlivé hodnoty klademe do závorek a oddělujeme čárkou.
- Příklad: `type dnyvtydnu=(pondeli,utery,streda, ctvrtek,patek, sobota,nedele);`
- Anebo definujeme přímo proměnnou tohoto typu: `var kal:(pondeli,utery,streda,ctvrtek,patek, sobota,nedele);`

Příklad

- Chceme vyrobit jednoduchý "kalendář" na rok 2012, tedy vypsat datum a údaj o dni v týdnu.
- Pro jednoduchost předpokládejme, že každý měsíc má 30 dnů...
- Zdrojový kód je na webu (kam.mff.cuni.cz/~perm/programovani/enum.pas).
- V příkladu vidíme, že funkce `write` neumí vypsat příslušné názvy, bylo by proto pěkné v závislosti na čísle dne v týdnu vypsat příslušný text. Nápady?
- Buďto mnoho klauzulí `if`, nebo: `case proměnná of ...`

Konstrukce `case ... of ...`

- Umožňuje vytvořit mnoho větví programu v závislosti na obsahu jedné proměnné.
- Syntax:

```
case jméno proměnné of
    hodnota1: příkaz nebo blok
    hodnota2: příkaz nebo blok
    else příkaz nebo blok
end;
```

- Provede se jen větev označená aktuální hodnotou proměnné, `else`-větev je pro ostatní (explicitně neuvedené) případy.
- Klauzule `else` nemusí být přítomna!
- Je-li poslední klauzule blok, jde `end` dvakrát po sobě (první uzavře blok posledních příkazů, druhý uzavře blok `case`).

Příklad – kalendář s jmény dnů

je na adrese
`kam.mff.cuni.cz/ perm/programovani/case_of.pas`.

Ordinální datové typy

- Typy, jejichž hodnoty tvoří lineárně uspořádanou množinu (tedy pro každou dvojici je jasné, který prvek je větší).
- Z pascalských typů jsou to: `integer`, `longint`, `byte`, `char`, `word`, `boolean` a `shortint` (a další definované uživatelem, zejména výčtový typ a interval).
- Pro ordinální typy jsou definovány funkce "Ord", "Pred" a "Succ".

Fronta a zásobník

- Fronta je datová struktura osazená operacemi `zařaď` a `vyřaď`,
- `zařaď` zařadí daný prvek na konec fronty,
- `vyřaď` vyřadí daný prvek ze začátku fronty.
- Zásobník je datová struktura osazená operacemi `push` a `pull`.
- `push` přidá na konec zásobníku, `pull` vytáhne z konce zásobníku.

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?
- Vlastně jde o úlohu prohledávání grafu.
- Podobné: Thesseus hledá Mínótaura.
- Rozdíl: Tehdy nám nešlo o nejkratší cestu.

Algoritmus vlny (myšlenky)

- Na šachovnici vysadíme na specifikované políčko mravence,
- mravenci polezou rovnoměrně všemi dostupnými směry,
- mravenci zkusí všechny cesty, rychleji než první mravenec se tam dostat nelze.
- Jiná intuice: Na příslušné poličko vychrstneme vodu,
- voda teče všemi dostupnými cestami, až doteče na cílové políčko.

Algoritmus vlny (implementace)

- Vytvoř prázdnou frontu f.
- Nastav vzdálenost do všech políček kromě startovního na nekonečno, vzdálenost do startovního nastav na 0.
- Přidej do f startovní políčko.
- Dokud není fronta f prázdná, opakuj:
  - a:=vyřad'(f);
  - Pro všechny sousedy z pole a zkus:
  - Pokud vzdálenost do z je větší než vzdálenost do a +1, nastav políčku z vzdálenost $a+1$ a zařad'(z, f).