Programování 2 (NMIN102)

Spojové seznamy

RNDr. Michal Žemlička, Ph.D.

Dynamicky alokované struktury

- někdy se hodí, abychom mohli vytvořit strukturu dle aktuálních potřeb
- k danému účelu slouží specifická oblast paměti (paměťového prostoru aplikace) – halda
- než takové místo použijeme, musíme o něj požádat
- když proměnnou v dynamicky alokované paměti už nepotřebujeme, je vhdoné ji uvolnit (někdy hned, někdy celý blok – záleží na použitém modelu)

Dynamicky alokované struktury (2)

- Dynamicky alokujeme velké proměnné (například pole, záznamy) nebo struktury (seznamy, stromy, ...)
- Může se stát, že paměti nebude dost, případně že nebude k dispozici dostatečně velký kus. Dnes se stává spíše až když aplikace běží déle – většinou to znamená, že někde došlo k chybě (například že se paměť alokuje, ale neuvolňuje).

Varianty spojových seznamů

- jednoduchý spojový seznam (jednosměrný)
- obousměrný spojový seznam
- spojový seznam s hlavou

Jednoduchý spojový seznam

- vyžaduje nejméně místa
- je třeba hlídat, zda nejsme na konci a zda jsme nenalezli hledaný prvek
- dá se efektivně procházet jen jedním směrem
- hodí se pro implementaci zásobníku, fronty, i v mnoha jiných případech

Obousměrný spojový seznam

- vyžaduje více místa a složitější režii změn
- efektivně průchozí oběma směry

Spojový seznam s hlavou

- součástí seznamu speciální uzel (hlava) sloužící jako zarážka (nemusíme testovat existenci dalšího uzlu, stačí si dát hledanou hodnotu do hlavy a ušetříme jedno testování podmínky v každém uzlu)
- vyžaduje o trošku více paměti než jednoduchý seznam, ale snáze se prochází
- seznam zapojený do kruhu

Operace nad jednoduchými dynamickými datovými strukturami

- ověříme, zda struktura není prázdná a ošetříme
- zjistíme, zda už nejsme v požadovaném uzlu
- vybereme vhodného následníka (je-li více možností)
- pokračujeme dalším uzlem

Operace nad strukturami s hlavou

- využíváme hlavu jako zarážku – hledaný prvek dáme do hlavy
- procházíme seznam, dokud nenajdeme hledaný prvek
- je-li tím uzlem, kde jsme prvek našli, hlava, prvek v seznamu není (a třeba jej můžeme vložit nebo nahlásit chybu)
- pokud ne, máme skutečný výskyt prvku a uzel, v němž je (a tedy provedeme požadovanou operaci)

Práce se seznamy – prostředí

- Abychom si mohli se seznamy mohli hrát, potřebujeme vhodné podmínky
- Testovací příkazy a data budeme načítat ze souboru

Práce se seznamy – prostředí (2)

- Příkazy budou na každém řádku jeden (případně i s parametry)
- Pro jednoduchost si můžeme příkazy zkrátit na jedno písmeno
- Určitě se bude hodit kontrolní tisk – mělo by být poznatelné, co je prázdný seznam a co prázdný řádek

Práce se seznamy – analýza příkazu

```
TYPE
  prvek = integer;
  ukaz = ^uzel;
  uzel = RECORD
    next : ukaz;
    data : prvek;
    END;
  prikazy = (pNothing, pInsert, pPrint, pPop, pError);
```

Práce se seznamy – RozdělŘádek

```
PROCEDURE RozdelRadek(radek:string;
          VAR prikaz:prikazy; VAR param:prvek);
  VAR
    x, Code : Integer;
    s : String;
  BEGIN
    IF radek='' THEN prikaz:=pNothing
    ELSE CASE radek[1] OF
      'P': prikaz:=pPrint;
```

```
    'I': BEGIN
           prikaz:=pInsert;
           s:=Copy(radek,3,length(radek)-2);
           Val(s,x,Code);
           param:=x;
         END;
    'O': prikaz:=pPop;
    ELSE prikaz:=pError;
    END;
END; { of RozdělŘádek }
```

Práce se seznamy – Akce

```
PROCEDURE Akce(radek:string; VAR seznam:ukaz);
  { zpracuje radek (prikaz) }
  VAR
    prikaz : prikazy;
    param : prvek;
  BEGIN
    RozdelRadek(radek,prikaz,param);
    CASE prikaz OF
      pPrint: HezkyTisk(seznam);
```

```
    pInsert: Push(param,seznam);
    pPop: IF seznam=NIL
            THEN WriteLn('Error: pop on empty stack.')
            ELSE
              BEGIN Pop(param,seznam); WriteLn(param) END;
    pNothing: ;
    ELSE WriteLn('Processing an unsupported command.');
    END;
END; { of Akce }
```

Práce se seznamy – Program

```
{$I-}
BEGIN
  Assign(f,inName);
  Reset(f);
  IF IOResult<>0 THEN
    BEGIN
      WriteLn('Soubor ',inName,' se nepovedlo otevrit.');
      Halt(10)
    END;
```

```
  WHILE NOT SeekEof(f) DO
    BEGIN
      ReadLn(f,radek);
      WriteLn(radek);
      Akce(radek,seznam);
    END;
  Close(f);
  WriteLn('Bye!');
END.
```