Aplikovaná informatika

Podklady předmětu
Aplikovaná informatika
pro akademický rok 2013/2014
Radim Farana

7

Obsah

- Dynamické datové struktury.
- Lineární seznam.
- Realizace zásobníku a fronty.
- Vkládání do setříděného seznamu.
- Třídění lineárního seznamu.
- Kruhový zásobník.

Dynamické datové struktury

- Struktury, které během zpracování mění svoji strukturu (velikost zabírané paměti).
- K realizaci je nutný datový typ ukazatel (pointer) – adresa dat v paměti,
  - lze pracovat přímo s obsahem proměnné – měnit adresu na kterou ukazuje: p = q.
  - lze pracovat s daty na která ukazuje: p^.data = q.
- Data tvoří obvykle složená struktura záznam:
  - klíč (index) – je využit k třídění dat
  - data – zastupují zbytek uložených dat
  - ukazatel nebo ukazatelé na další prvky struktury.

Přidělování paměti

- Dynamické přidělování paměti
  - přísná typová kontrola: p je ukazatel na konkrétní datovou strukturu:
    - new (p) – přidělí dynamické proměnné paměť a její adresu uloží do ukazatele,
    - dispose (p) – uvolní paměť dynamické proměnné, hodnota ukazatele se obvykle nemění.
  - volná typová kontrola: p je obecný ukazatel:
    - GetMem (p, size) – přidělí paměť určené velikosti,
    - FreeMem (p, size) – uvolní paměť určené velikosti.
- Určení prázdného ukazatele – nil.

Lineární seznam

- Nejjednodušší dynamická struktura (také vázaný seznam). Každý prvek má nejvýše jednoho následníka.
- Na strukturu (první prvek) ukazuje proměnná (ukazatel) kořen.
- Základní operace:
  - vložení prvku,
  - zpracování seznamu.
  - zrušení struktury.

prvek lineárního seznamu

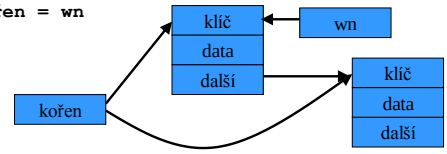
Vložení prvku do lineárního seznamu

- Pozor na prázdný seznam (kořen = nil).
- Vložení prvku:
  - na začátek – struktura zásobník (LIFO – last in, first out),
  - na konec – struktura fronta (FIFO – first in, first out),
  - podle hodnoty klíče – setříděný seznam.
- Zpracování prvků vždy probíhá od začátku seznamu do konce.

Zásobník – LIFO

- Nový prvek vkládáme na začátek seznamu

```
new (wn)
wn↑.klíč = hodnota klíče nového prvku
wn↑.data = data nového prvku
wn↑.další = kořen
kořen = wn
```



Fronta – FIFO

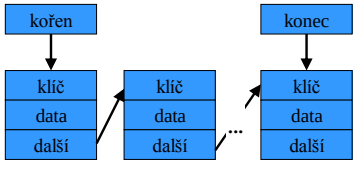
- Nový prvek vkládáme na konec seznamu.

```
new (wn)                                REM nový prvek
wn↑.klíč = hodnota klíče nového prvku
wn↑.data = data nového prvku
wn↑.další = nil
if kořen = nil                  REM prázdný seznam
      kořen = wn
else
      w = kořen                 REM nalezení posledního
      while not w↑.další = nil
            w = w↑.další
      end while
      w↑.další = wn
end if
```


Přístup ke konci seznamu

- Udržování ukazatele na poslední prvek seznamu – ukazatel konec.

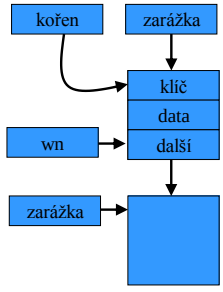
```
new (wn)
wn↑.klíč = klíč
wn↑.data = data
wn↑.další = nil
if kořen = nil
      kořen = wn
      konec = wn
else
      konec↑.další = wn
      konec = wn
end if
```



Přístup ke konci seznamu

- Vytvoření prvku zarážka.

```
wn = zarážka
wn↑.klíč = klíč
wn↑.data = data
new(zarážka)
wn↑.další = zarážka
```



Vkládání do nesetříděného seznamu

- Hledáme výskyt prvku
  - po nalezení: h = False, w = odkaz na prvek,
  - nenalezen: h = True, wk = odkaz na poslední prvek,
- Podle výsledku
  - manipulace s nalezeným prvkem, nebo
  - přidání prvku na konec seznamu.

Vkládání do nesetříděného seznamu

nalezení prvku

```
w = kořen
h = true
while (w<>nil) and (h)
      if w↑.klíč = klíč then
            h = false
      else
            wk = w
            w = w↑.další
      end if
end while
```

vložení nebo zpracování prvku

```
if h then
      new (wn)
      wn↑.klíč = klíč
      wn↑.data = data
      wn↑.další = nil
      if kořen = nil then
            kořen = wn
      else
            wk↑.další= wn
      end if
else
      práce s daty w↑.data
end if
```

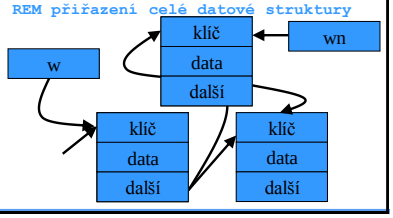
Vkládání do setříděného seznamu

- Hledáme výskyt prvku (se zarážkou)
  - po nalezení: k = False, w = odkaz na prvek,
  - nenalezen: k = True,
    - h = False, w = odkaz na zarážku,
    - h = True, w = odkaz na následující (větší) prvek.
- Podle výsledku
  - manipulace s nalezeným prvkem, nebo
  - přidání prvku do seznamu, varianty:
    - na začátek seznamu pro případ prázdného seznamu,
    - na konec seznamu, za poslední prvek,
    - před nalezený prvek s větší hodnotou klíče.

Vložení prvku před aktuální prvek

- Chybí odkaz na předchozí prvek.
- Řešením je přesun obsahu prvku do nového.

```
new (wn)
wn↑ = w↑
w↑.klíč = klíč
w↑.data = data
w↑.další = wn
```



Vkládání do setříděného seznamu

```
w = kořen
h = true
k = true
while h and w<>zarážka then
  if w↑.klíč = klíč then
    h = false
    k = false
  else
    if w↑.klíč<klíč then
      w = w↑. další
    else
      h = false
    end if
  end if
end while
```

```
if k then
  new (wn)
  wn↑ = w↑
  w↑.klíč = klíč
  w↑.data = data
  w↑.další = wn
  if w = zarážka then
    zarážka = w↑.další
  end if
else
  práce s daty w↑.data
end if
```

Zrychlení vyhledávání

- Cíl – zkrácení doby vyhledávání prvků (vyhledávání je vždy lineární).
- Využití četnosti výskytu prvků a jejich tendence ke shlukování.
- Řešení – nalezený prvek posuneme na začátek seznamu.
- Nový prvek tedy vkládáme na začátek seznamu.

Zrychlení vyhledávání

```
w = kořen
h = true
while w<>nil and h
  if w↑.klíč = klíč then
    h = false
  else
    wk = w
    w = w↑.další
  end if
end while
```

```
if h then
  new (wn)
  wn↑.klíč = klíč
  wn↑.data = data
  wn↑.další = kořen
  kořen = wn
else
  zpracování dat w↑.data
  if w<>kořen then
    wk↑.další = w↑.další
    w↑.další = kořen
    kořen = w
  end if
end if
```


Zpracování lineárního seznamu

- Vždy od začátku do konce:

```
w = kořen
while w<>nil
      zpracování dat w↑.data
      w = w↑.další
end while
```

- Opačný postup pomocí rekurze:

```
průchod(w)
  if w<>nil  then
     průchod (w↑.další)
     zpracování dat w↑.data
  end if
end průchod
```

nebezpečí přeplnění zásobníku volání

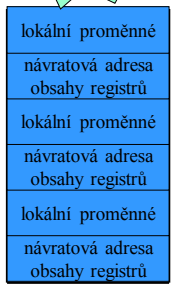
Zásobník volání

Stack overflow error

volání procedury
- uložení návratových hodnot
- deklarace lokálních proměnných
- volání procedury
  - uložení návratových hodnot
  - deklarace lokálních proměnných
- ukončení procedury

ukončení procedury

Nebezpečí rekurzivního volání



zásobník

Setřídění lineárního seznamu

- Použití dvou seznamů
  - třídění přímým vkládáním (Insertion Sort),
  - třídění přímým výběrem (Selection Sort).
- Použití jednoho seznamu
  - třídění přímou výměnou (Bubble Sort).
- Využití algoritmů externího třídění.

Třídění přímou výměnou

h – indikuje zda proběhla nějaká výměna

```
new(wp)
h = true
while h
  h = false
  w1 = kořen
  if w1 <> nil then
    w2 = w1
    w1 = w1↑.další
  end if
  while w1<> nil
    if w2↑.klíč>w1↑.klíč then
      wp↑.klíč = w2↑.klíč
      wp↑.data = w2↑.data
      w2↑.klíč = w1↑.klíč
      w2↑.data = w1↑.data
      w1↑.klíč = wp↑.klíč
      w1↑.data = wp↑.data
      h = true
    end if
    w2 = w1
    w1 = w1↑.další
  end while
end while
dispose(wp)
```

Obousměrně vázaný seznam

- Prvek ukazuje na svého následníka i předchůdce (ukazatelé další a minulý).
- Po vybudování jednosměrně vázaného seznamu se aktualizují odkazy na předchůdce.

```
wp = nil
w = kořen
while w<>nil
      w↑.minulý = wp
      wp = w
      w = w↑.další
end while
```

- Vhodný například pro třídění přetřásáním (Shake Sort).

Vynětí prvku ze seznamu

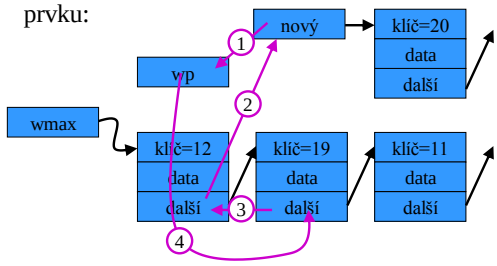
- Nejjednodušší je vynětí prvního prvku.
- Rušení celého seznamu:

```
while kořen <> nil
      wp = kořen
      kořen = kořen↑.další
      dispose(wp)
end while
```

- Jediné přípustné použití ukazatele kořen.

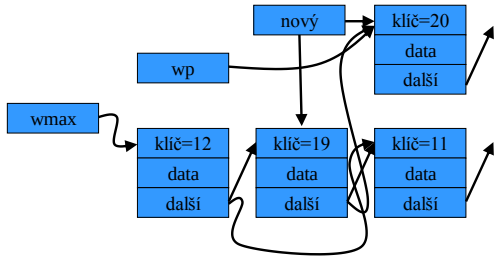
Vynětí prvku ze seznamu

- Nejjednodušší vynětí následníka aktuálního prvku:



Vynětí prvku ze seznamu

- Realizace



Třídění přímým výběrem

```
nový = nil
while kořen <> nil do
  w1 = kořen
  w2 = nil
  wmax = w2
  kmax = w1↑.klíč
  while w1<>nil
    if w1↑.klíč >= kmax
      wmax = w2
      kmax = w1↑.klíč
    end if
    w2 = w1
    w1 = w1↑.další
  end while
  if wmax<>nil then
    wp = nový
    nový = wmax↑.další
    wmax↑.další = nový↑.další
    nový↑.další = wp
  else
    wp = nový
    nový = kořen
    kořen = nový↑.další
    nový↑.další = wp
  end if
end while
kořen = nový
```


Kruhový zásobník

- Kruhový zásobník má pevný počet prvků.
- Poslední prvek ukazuje na první prvek.
- Zpracovaný prvek je nahrazen novým prvkem.
- Při vytváření zásobníku se prvky pouze vkládají.
- Při rušení se prvky pouze odstraňují.
- Využití pro speciální potřeby jako realizace dopravního zpoždění.

Kruhový zásobník

Vytvoření kruhového zásobníku

```
new (zásobník)
w1 = zásobník
inicializace dat w1↑.data
i = 1
while i<n
        new(w1↑.další)
        w1 = w1↑.další
        inicializace dat w1↑.data
        i = i+1
end while
w1↑.další = zásobník
```


Kruhový zásobník

Průchod kruhovým zásobníkem

```
přečtení dat zásobník↑.data
uložení nových dat zásobník↑.data
zásobník = zásobník↑.další
```

Zrušení kruhového zásobníku

```
w1 = zásobník
repeat
        wp = w1
        w1 = w1↑.další
        dispose(wp)
until w1 = zásobník
```