IAL Algoritmy
Obsah 3. přednášky

- Datové struktury, statické, dynamické, homogenní, heterogenní; konstruktor, selektor, iterátor, destruktor.
- Dynamické struktury , seznamy (jednosměrný, dvojsměrný, kruhový, hlavička seznamu).
- ADT - syntaktická a sémantická specifikace, diagramy signatury.

Obsah 3. přednášky

- ADT seznam - jednosměrný, dvousměrný, kruhový, dvojsměrný seznam v poli s jedním ukazatelem, seznamy s hlavičkou, bez hlavičky, operace, principy implementace.

- Základní operace, některé další operace (ekvivalence, relace uspořádání, počet prvků, uchování a obnova aktivity, kopie, destrukce, inserce podseznamu, rušení podseznamu, konkatenace, dekatenace aj.)

Datové struktury statické, dynamické, operace nad DS

- Datová struktura je homogenní, když všechny její prvky jsou téhož typu. Nejčastěji pak prvkům říkáme položky struktury.
- Datová struktura, jejíž prvky nejsou téhož typu ,je heterogenní. Jejím prvkům nejčastěji říkáme složky. Představitelem heterogenní (nehomogenní) datové struktury v Pascalu je záznam. Ostatní pascalovské datové struktury jsou homogenní.
- Mezi základní operace nad datovou strukturou je operace přiřazení.

Konstruktor a selektor

- Konstruktor je operace, jejímiž vstupními parametry je výčet (všech) komponent struktury a výsledkem operace je datová struktura obsahující tyto komponenty. Konstruktor “vytvoří” datovou strukturu z jejich komponent. Příkladem konstruktoru v jazyce Pascal je textový řetězec v příkazu str=‘Textovy retezec’ nebo množina [1,3,5,7,9]
- Selektor je operace, která umožní přístup k jednotlivé komponentě datové struktury na základě uvedení jména struktury a zápisu přístupu (“reference”). Příklad selektoru nad textovým řetězcem z předcházejícího odstavce je zápis str[3], kde prvek řetězce má hodnotu ‘x’. Přístup k prvku datové struktury se používá za účelem změny hodnoty prvku nebo k získání jeho hodnoty.

Destruktor a iterátor

- Destruktor je operace nad dynamickými strukturami. Tato operace zruší dynamickou strukturu a vrátí prostor jí zaujímaný systému DPP.
- Iterátor je operace, která provede zadanou činnost nad všemi prvky homogenní datové struktury. Příklad: většina složitých grafických útvarů je realizována seznamem grafických elementů, z nichž je útvar sestaven. Operace, která jediným příkazem typu iterátor provede operaci "vykreslit" nad všemi elementy, zobrazí útvar jako celek.

Abstraktní datový typ

Abstraktní datový typ (ADT) je definován množinou hodnot, kterých smí nabýt každý prvek tohoto typu a množinou operací nad tímto typem.

Smyslem ADT je zvýšit datovou abstrakci a snížit algoritmickou složitost programu (algoritmu).

Kardinalita datového typu vyjadřuje množství různých hodnot ADT. Je výrazem paměťové náročnosti ADT.

ADT zdůrazňuje „co dělá“ a potlačuje „jak to dělá“. ADT připomíná „černou skříňku“

Příklady elementárních ADT : typ real.

Syntax a sémantika

Syntaxe vyjadřuje pravidla korektního zápisu jazykové konstrukce.

Sémantika vyjadřuje (popisuje) účinek (význam) zápisu jazykové konstrukce.

Algebraická signatura typu Kladný integer - Posint

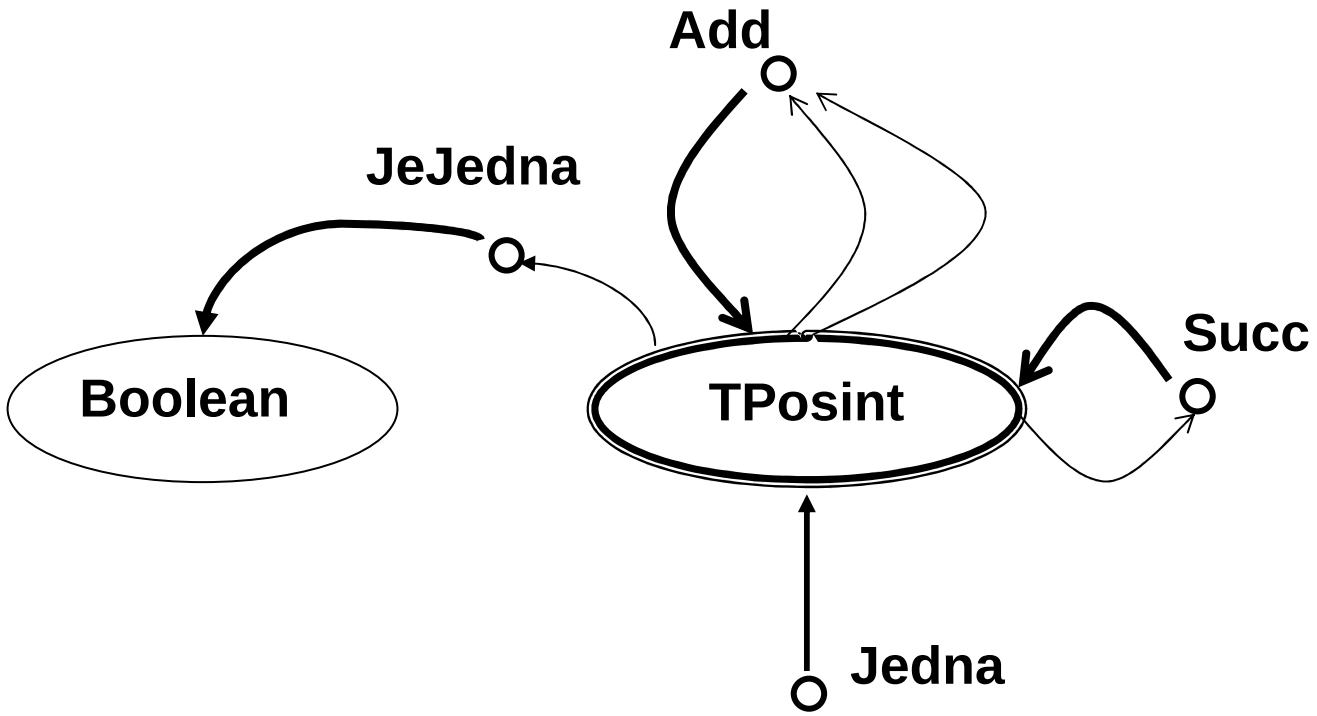
One : → Posint (Této operaci se říká generátor nebo "inicializace". Musí se použít před první operací nad typem

ADD : Posint x Posint → Posint

SUCC : Posint → Posint

IsOne : Posint → Boolean

Diagram signatury ADT Posint



Sémantika operace

Slovní sémantika:

- Operace One ustaví hodnotu typu Posint rovnu jedné. Tato operace je inicializace typu (generátor).
- Operace ADD vytvoří aritmetický součet dvou prvků typu Posint.
- Operace Succ vytvoří hodnotu následující danou hodnotu (hodnotu o jednu větší).
- Operace (predikát) IsOne nabude hodnoty true, pokud je argument hodnota rovna jedné. V jiných případech má operace hodnotu false.

Axiomatická specifikace sémantiky

1. ADD(X,Y) = ADD(Y,X)
2. ADD(One,X)=SUCC(X)
3. ADD(SUCC(X),Y)=SUCC(ADD(X,Y))
4. IsOne(One)=true
5. IsOne(SUCC(X))=false.

Jiné způsoby specifikace sémantiky

- Slovní popis
- Operační (funkční popis) – popis prostřednictvím procedur/funkcí
  - Nevýhoda -> Podkládá způsob implementace
- Specifikace úplným výčtem účinků

ADT seznam (angl. List)

Seznam je lineární, homogenní, dynamická datová struktura. Patří mezi nejobecnější datové struktury.

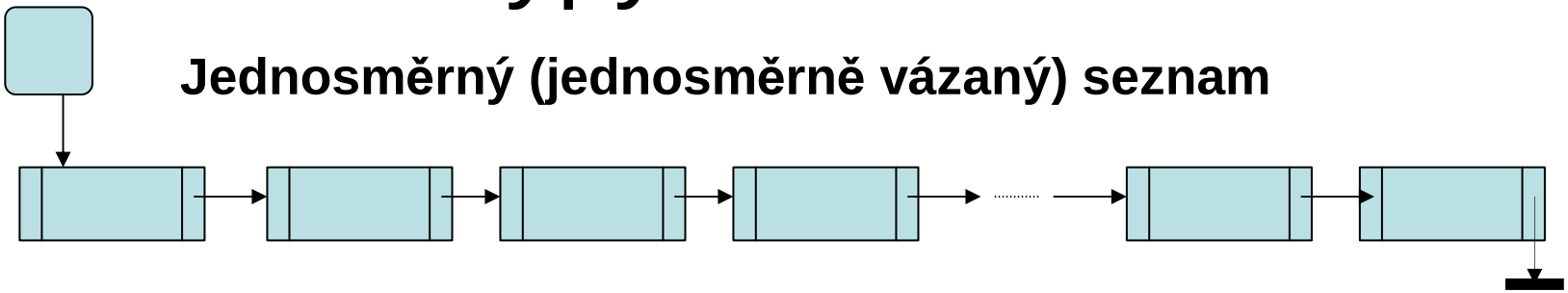
Lineárnost znamená, že každý prvek struktury má právě jednoho předchůdce (*predecessor*) a jednoho následníka (*successor*). Výjimku tvoří první a poslední prvek.

Prvkem seznamu může být libovolný jiný datový typ – také strukturovaný. Např. seznam seznamů.

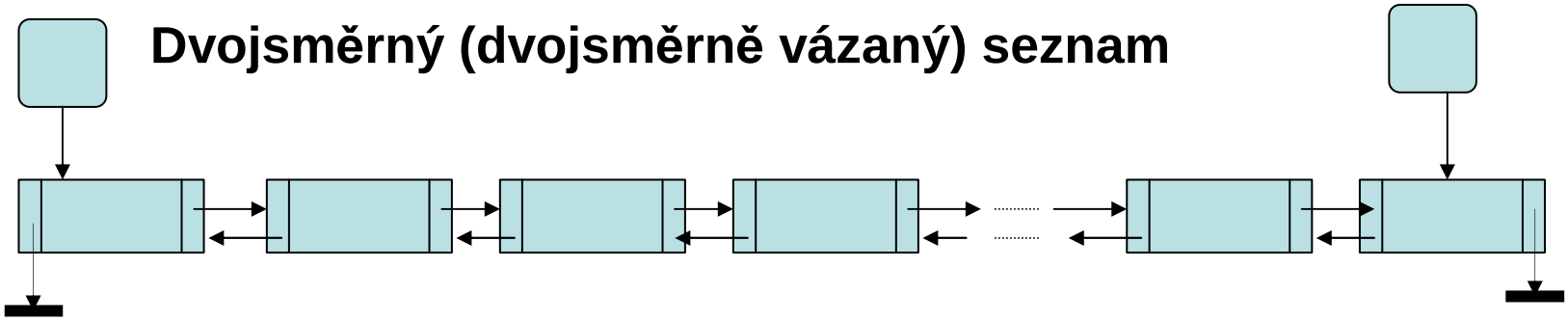
Seznam může být prázdný.

Typy seznamů

Jednosměrný (jednosměrně vázaný) seznam



Dvojsměrný (dvojsměrně vázaný) seznam



Kruhový jednosměrný seznam

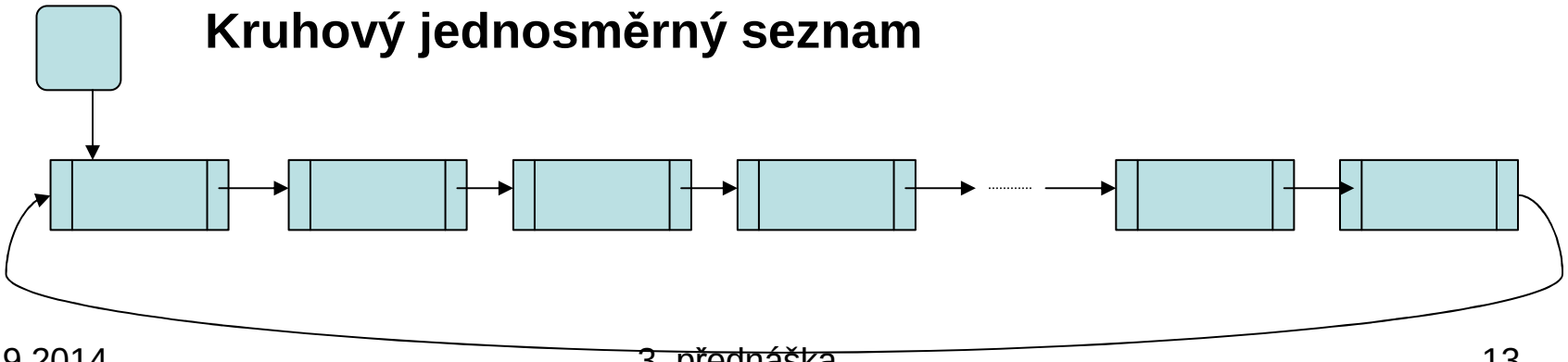
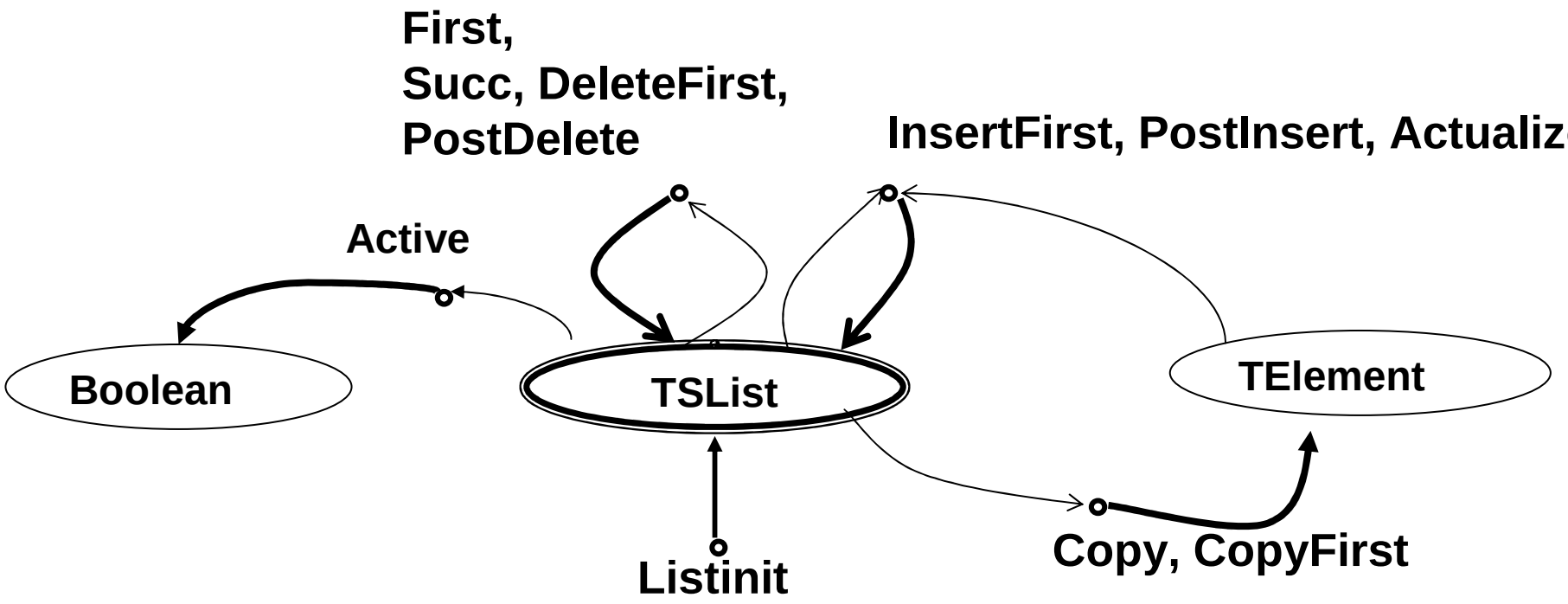


Diagram signatury ADT TList



Symetricky doplňkové operace pro dvousměrný seznam:
Last, Pred, DeleteFirst, PostDelete, InsertLast, PreInsert, CopyLast

Sémantika operací nad jednosměrným seznamem

- **ListInit(L)** – vytvoří prázdný seznam prvků daného typu
- **InsertFirst(L,El)** – vloží element El do seznamu L jako první prvek (jediná operace, která může vložit prvek do prázdného seznamu)
- **CopyFirst(L,V)** – v proměnné V vrátí hodnotu prvního prvku. V případě prázdného seznamu CHYBA! (stav Error)
- **DeleteFirst(L)** – zruší první prvek seznamu

- **First(L)** – první prvek se stane aktivním (pro prázdný seznam bez účinku)
- **Succ(L)** – aktivita se přenese na následující prvek (pro prázdný seznam bez účinku, ja-li aktivním poslední, aktivita se ztratí - seznam přestane být aktivním)
- **Copy(L,V)** - v proměnní V vrátí hodnotu aktivního prvku (v případě neaktuvního seznamu CHYBA! – stav ERROR).
- **Actualize(L,El)** – hodnota aktivního prvku je přepsána hodnotou El. (V případě neaktivního seznamu bez účinku).

- **PostInsert(L, El)** - vloží prvek El jako nový za akticní prvek (v případě neaktivního seznamu bez účinku)
- **PostDelete(L)** – zruší prvek za aktivním prvkem (v případě neaktivního seznamu nebo neexistence prvku za aktivním – bez účinku)
- **Active(L)** – predikát (Booleovská funkce) – vrací hodnotu true v případě, že seznam je aktivní; v jiném případě vrací hodnotu false

Symetrické operace pro dvojsměrný seznam

- **Insertlast(L, EL)** - vloží prvek El jako poslední (do prázdného seznamu současně i první – spolu s InsertFirst je to jediná operace, která může vložit prvek do prázdného seznamu)
- **CopyLast(L, V)** – V proměnné V vrátí hodnotu posledního prvku. V případě prázdného seznamu Chyba!!
- **Pred(L)** - posune aktivitu zpět

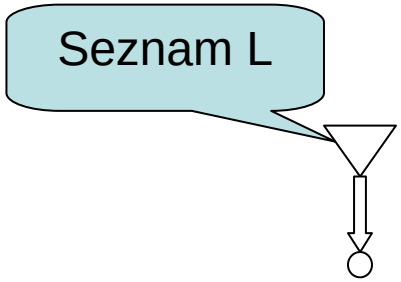
- **Last(L)** – nastaví aktivitu na poslední prvek. V případě prázdného seznamu bez účinku-

- **DeleteLast(L)** – zruší poslední prvek (V případě, že poslední prvek je současně jediný a první, vznikne prázdný seznam. Je s operací DeleteFirst jediná operace, která může odstranit jediný - poslední prvek. Pro prázdný seznam bez účinku).

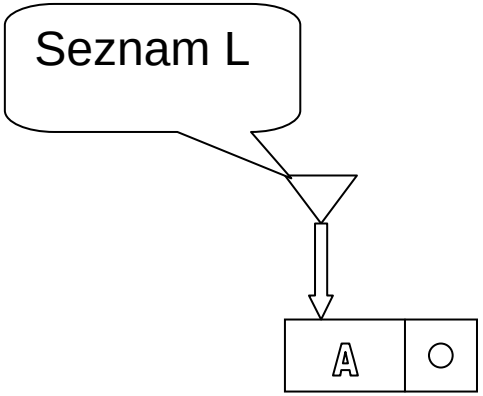
- PreInsert(L, El) – vloží hodnotu prvku El před aktivní prvek. Je-li seznam neaktivní je operace bez účinku.

- PreDelete(L) – operace zruší prvek před aktivním prvkem. Je-li seznam neaktivní nebo před aktivním prvkem (nalevo od něj) není žádný prvek, je operace bez účinku.

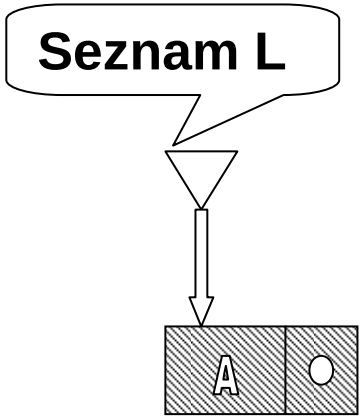
Presentace účinků operace s jednosměrným seznamem



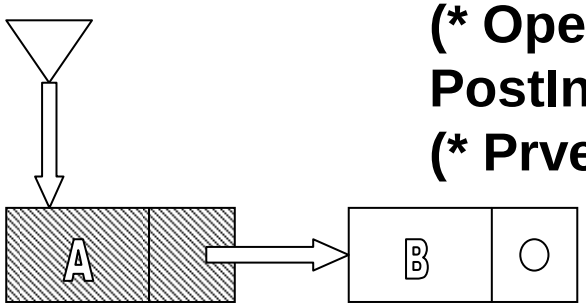
ListInit(L);
Operace vytvoří prázdný seznam.



Operace InsertFirst(L, A);

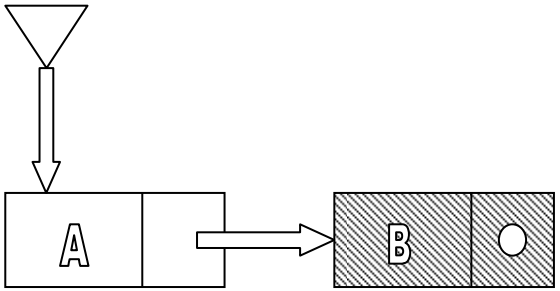


(* Operace *)
First (L);
(* První je aktivní *)

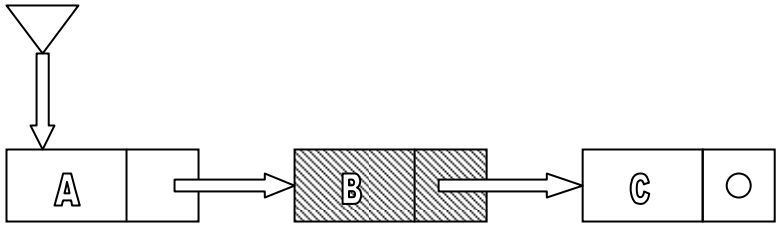


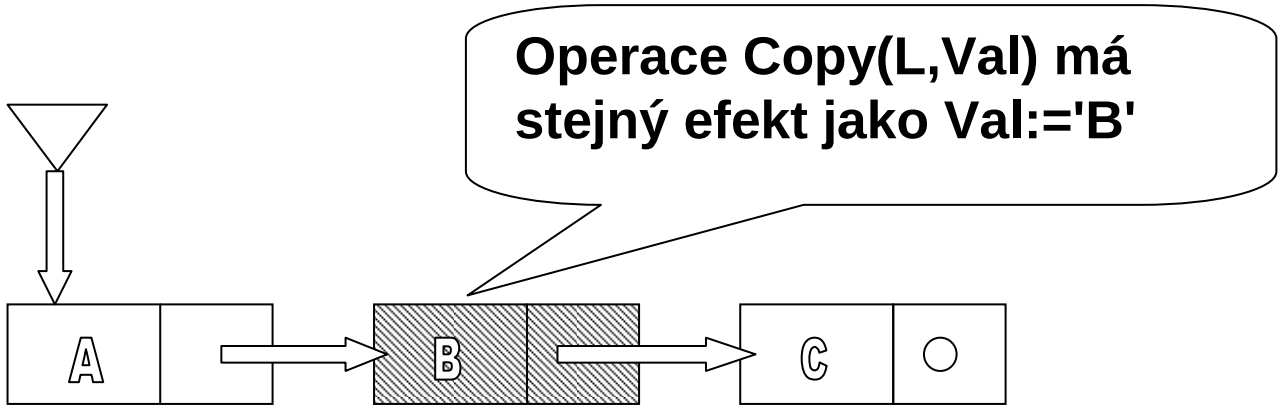
(* Operace *)
PostInsert(L,B);
(* Prvek B se vloží za aktivní *)

(* Operace *)
Succ(L);
(* posune aktivitu na další prvek*)

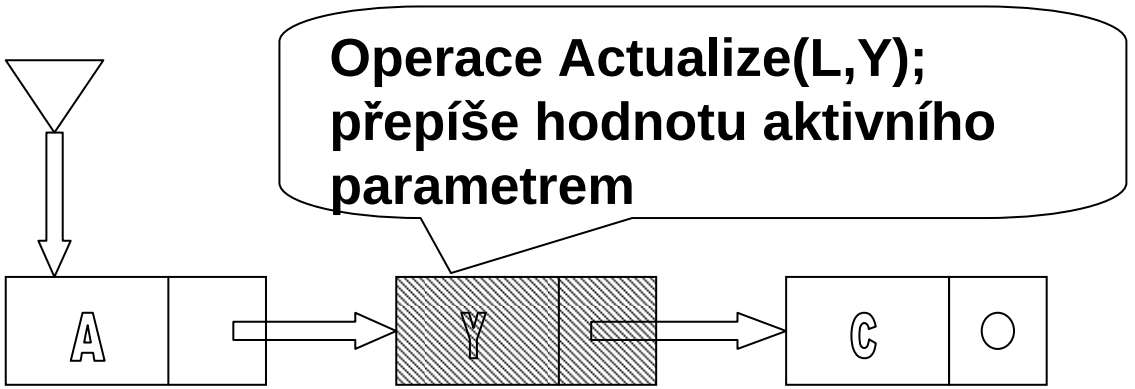


(*Operace*)
PostInsert(L,C)
(* vloží nový prvek C za aktivní *)



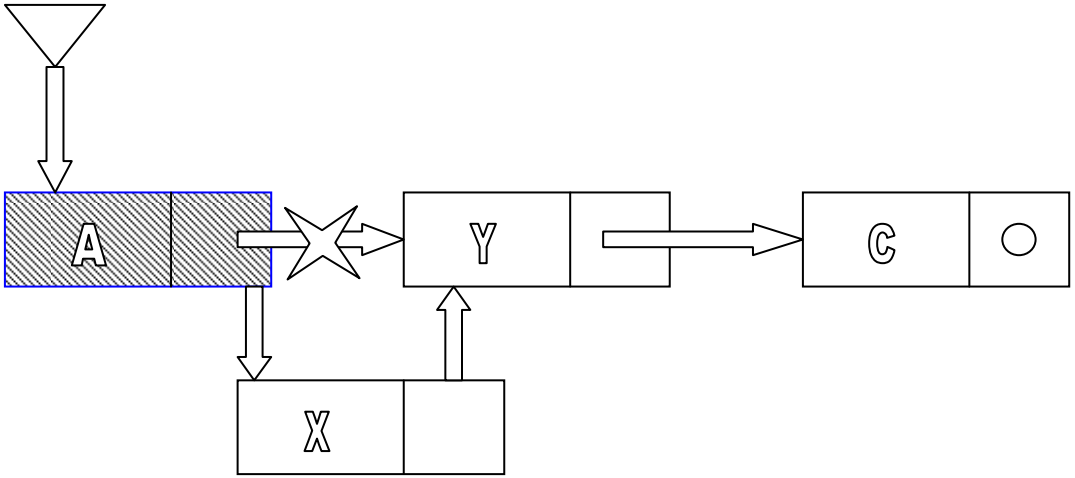


V případě neaktivního seznamu dojde k chybě !!!



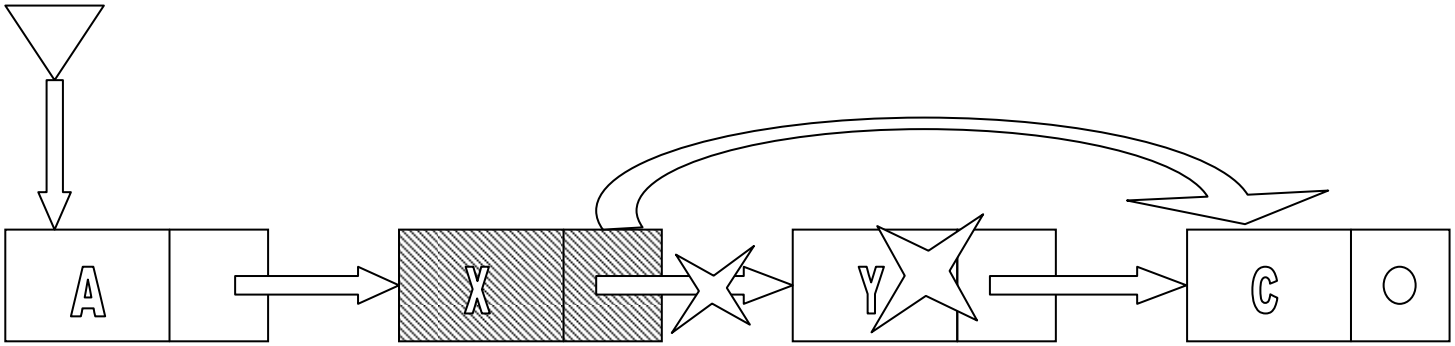
(* Dvojice operací *)

First(L); **(* Učiní první aktivním*)**
PostInsert(L,X) **(* Vloží X za aktivního*)**



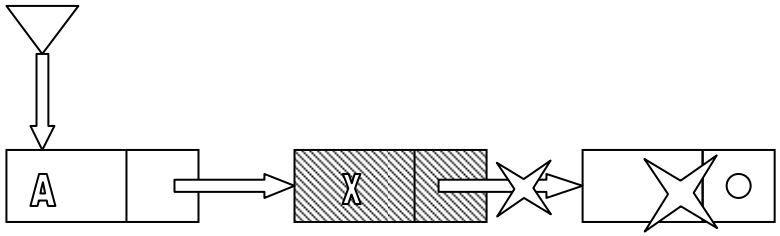
(* Dvojice operací*)

Succ(L); (* Posune aktivitu na další prvek *)
PostDelete(L); (* zruší prvek za aktivním *)

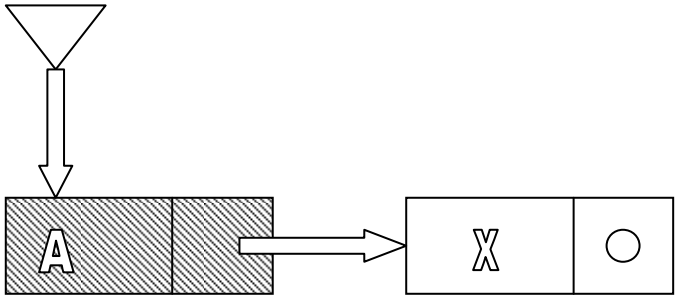


(* Operace*)

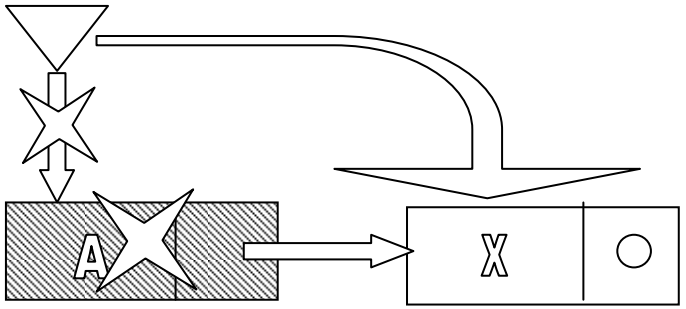
Postdelete(L); (* Zruší prvek za aktivním *)



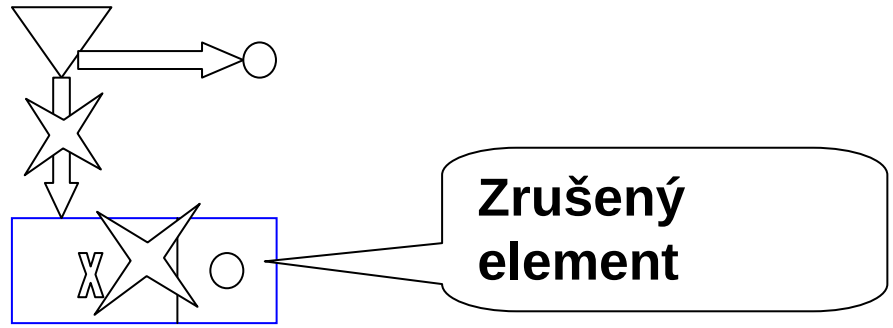
(* Operace *)
First(L); (* První je aktivní *)

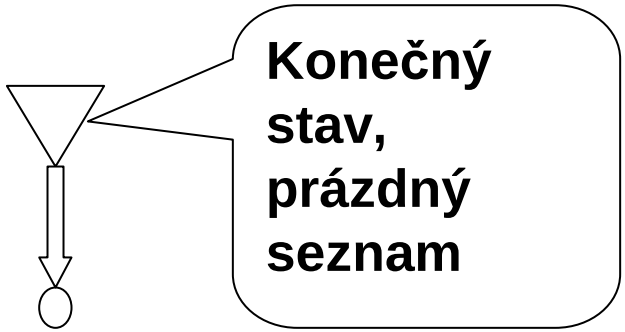


(* Operace *)
DeleteFirst(L) (* Zruší první prvek *)



(* Operace *)
DeleteFirst(L); (* Zruší první a jediný prvek.
Vytvoří prázdný seznam *)





Typické algoritmy nad ADT seznam

Typické operace nad seznamem:

1. Délka seznamu
2. Kopie (duplikát) seznamu
3. Zrušení seznamu
4. Ekvivalence dvou seznamů
5. Relace (lexikografická) dvou seznamů
6. Vkládání nových prvků do seznamu (na začátek, na pozici danou ukazatelem, pořadím, aktivitou, na konec)

7. Vkládání a rušení podseznamu
8. Vyhledávání prvku v seznamu
9. Rušení prvku seznamu (prvního, prvku na a za pozicí dané ukazatelem, pořadím, aktivitou, posledního)
10. Seřazení prvků seznamu podle velikosti (klíče)
11. Konkatenace (zřetězení) dvou a více seznamů (podseznamů) do jednoho seznamu (např. podseznamů obsahující textové „slovo“ do „věty“).
12. Dekatenace (rozčlenění) jednoho seznamu na podseznamy (např. rozčlenění textové „věty“ na „slova“)

Rekurzívní definice ekvivalence dvou seznamů

Dva seznamy jsou ekvivalentní, když jsou oba prázdné nebo když se rovnají jejich první prvky a také jejich zbytky.

Algoritmy nad ADT List s použitím abstraktních operací (pouze!!!)

Délka seznamu:

```
function Length(L:TL):integer;
var Count:integer;
begin
   Count:=0;
   First(L);
   while Active(L)do begin
     Count:=Count+1;
     succ(L)
   end; (* while *)
   Length:= Count
end
```

Domácí úloha:

• Do seznamu celých čísel seřazeného podle velikosti vložte nový prvek tak, aby seřazení seznamu zůstalo zachováno. V případě, že seznam obsahuje prvek rovný vkládanému, vložte vkládaný za shodný (poslední ze shodných). Problém řešte procedurou.

• Napište proceduru, která ze zadaného seznamu vyřadí prvek zadaný parametrem. (příklad: vyřazení hodnoty 5 ze seznamu "1,3,6,8" nemá žádný účinek. Zrušení hodnoty 5 ze seznamu "1,3,5,7" má za výsledek seznam "1,3,7"). V případě, že seznam obsahuje více shodných prvků, rovných zadanému, zrušte poslední z nich.

• V seznamu celých čísel nalezněte nejdelší neklesající posloupnost. Procedura vrátí počáteční index a délku nalezené posloupnosti daného seznamu (příklad: Výsledky pro seznam: "4,3,2,1" je začátek=1, délka=1. Výsledek pro seznam: 4,1,3,5,2,4,1,8,9" je začátek=2, délka=3;)
Nápověda: Algoritmus hledá maximální délku. Algoritmus uchovává index a délku každé kandidátské posloupnosti (posloupnosti delší než nejdelší z předcházejících).

• V jednom průchodu seznamem najděte průměrnou hodnotu a rozptyl (dispersi) délek všech neklesajících posloupností daného seznamu celých čísel.
Nápověda: Rozptyl D daného souboru hodnot je průměrná hodnota kvadratických odchylek všech hodnot souboru od průměrné hodnoty.

$$D = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2$$

• Jsou dány dva seřazené seznamy celých čísel. Sloučením z nich vytvořte jeden nový seznam seřazených celých čísel. (příklad: je dán seznam L1="1,3',5,7,9" a L2="2,3",4,6,8" . Výsledkem bude L3="1,2,3',3",4,5,6,7,8,9").
Pozn. Při práci s ukazateli jsou k vytvoření nového seznamu použity prvky zdrojových seznamů, které tím zaniknou. Jejich původní ukazatele vynilujte. V případě práce s abstraktními operacemi nad ADT List zachovejte seznamy L1 a L2 a vytvořte nový seznam L3.

Kontrolní otázky

- Co je to abstrakce?
- Je rozdíl v operaci + jsou-li použity v následujících kontextech:
- integer + real
- integer + integer
- Má jazyk Pascal silnou nebo slabou typovou kontrolu?
- Vysvětlete pojmy zapouzdření a kardinalita.
- Vysvětlete pojmy syntaxe a sémantika.
- Jaké jsou základní vlastnosti ADT seznam?
- Co se stane při použití DeleteFirst na prázdném seznamu?
- Co se stane při použití CopyFirst na prázdném seznamu?
- Definujte ekvivalenci seznamu.
- K čemu slouží generátor (inicializace) u ADT.

Doplňte na místa označená XXXXX a YYYYY správným zápisem

```
function Length(L:TL):integer;
var Count:integer;
begin
   Count:=0;
   First(L);
   while Active(L)do begin
     XXXXX;
     YYYYY
   end; (* while *)
   Length:= Count
end
```