Anotace

- Ukazatele a jejich aplikace:
- Spojové seznamy,
- fronta, zásobník,
- uspořádaný spojový seznam,
- samoopravující seznamy,
- hashování,
- stromové datové struktury.

Pointery alias ukazatele

- Paměť je organizována lineárně v podobě jednotlivých adres.
- Na těchto adresách jsou ukládány hodnoty (kód, data).
- Paměť zpravidla adresujeme přirozenými čísly, která zapisujeme v šestnáctkové soustavě.
- Na data v paměti si tedy můžeme ukazovat.
- Pod těmito ukazateli můžeme mít údaje libovolných typů, z čehož vyplývá jisté nebezpečí.
- Z toho vzplývá jisté nebezpečí a nutnost být obezřetný.

Pointery – syntax a sémantika

- S pointery pracujeme zpravidla tak, že definujeme datový typ *ukazatel na něco.*
- *Ukazatel na* řekneme pomocí operátoru stříšky:
- `type pint=^integer; {ukazatel na integer}`
- Následně definujeme proměnnou příslušného typu:
  `var ukaz:pint;`
- Pod pointer "koukneme" opět pomocí operátoru stříšky:
  `writeln(ukaz^);`
- Jenže ono to zdaleka není tak snadné!

Organizace paměti s ohledem na program

- Program sestává z kódu, tzv. statických dat, prostoru zásobníku a oblasti haldy.
- Kam ukazuje pointer, který si lehkomyslně vyrobíme?
- Pokud s ním budeme zacházet rozumně, bude ukazovat někam do oblasti haldy, k tomu ale máme ještě daleko.
- Kam tedy pointer ukazuje?
- V lepším případě na adresu 0, v horším na náhodně vybraný kus paměti.
- Pořádek v paměti hlídá allokátor, který ví, které části paměti jsou použité a které ne a může nám přidělit prostor.

Allokátor a jeho použití

- Abychom mohli pod pointer kouknout, musíme ho někam nasměrovat. A to buďto na už existující pointer (`var a,b:pint;... naalokuj b;... a:=b;`),
- anebo "urafnutím" nové paměti: `new(a);`
- Funkci `new` předáváme jako parametr proměnnou typu ukazatel.
- Funkce `new` najde v paměti vhodné místo a nasměruje na něj příslušný pointer.
- Cvičení zimního učiva: Bere `new` parametr hodnotou nebo referencí?
- Od chvíle, kdy máme naalokováno, můžeme pod pointer koukat i zapisovat:
- `new(a); a^:=5; writeln(a^);` Ale pozor na přesměrování ukazatele!

Pointery a práce s nimi

- `var a,b:pint;`
- `new(a);` – naallokuje místo pro proměnnou typu integer
- `a^:=5;` – zapíše pod pointer hodnotu 5.
- `b^:=a^;` – okopíruje pod pointer `b` hodnotu, na kterou ukazuje pointer `a`.
- `b:=a;` – okopíruje pointer (tedy `a` a `b` ukazují na stejné místo).
- Co v kontextu uvedeného kódu udělá `b^:=10; writeln(a^);`?
- Pozor, více pointery si můžeme ukazovat na to samé místo!

Deallokace

- Nepotřebujeme-li už příslušné místo, musíme to říci allokátoru voláním funkce `dispose`:
- `dispose(a);`
- Tím prohlásíme, že pod dotyčný pointer už nepotřebujeme.
- Pozor pokud si na totéž místo ukazujeme víckrát, nesmíme provést vícenásobné odallokování!
- Pokud pointer přesměrujeme bez odallokování (ničím si na dotyčné místo neukazujeme), vzniká tzv. garbage, o které má allokátor za to, že je používaná, my však už nemáme, jak se k dotyčnému kusu paměti dostat (tak ho nemůžeme ani odallokovat).
- Některé jazyky (Java, C#) takto odallokovávají (přestaneme si na kus paměti ukazovat). Říká se tomu garbage collector, je to nevýchovné a v Pascalu není implementován.

Typičtější použití pointerů

- V zimě jsme se učili o strukturách (records).
- Struktury nám umožňovaly udržovat údaje různých typů spolu souvisejících. Například dvě souřadnice bodu v rovině, údaje o lidech, údaje o knihách...
  Do recordu se přistupuje operátorem tečky.
- Typicky se dělají pointery na struktury:
- Příklad:

```
type tbod=record
x,y:integer;
end;
pbod=^tbod;
```

- Stále ještě neřešíme původní problém, co když máme bodů víc a nevíme předem kolik?