

Anotace

- Soutěž Checkpoint už v pátek!
- Definice vlastních datových typů (struktury)
- Základní třídící algoritmy
- Jednotky (units)

Datový typ record

- Typicky míváme objekt popsany několika hodnotami (bod v rovině: dvojice hodnot).
- Je nešikovné mít každou hodnotu někde úplně jinde (je vhodnější mít data co nejvíce pohromadě).
- Definujeme tedy strukturu, kde budeme mít všechny údaje pohromadě.
- Definujeme pomocí klíčového slova `record`.

Typ record – příklad

```
type bod=record
    x,y:integer;
end;
var a,b:bod;
    kn:record
        autor: string[100];
        nazev: string[100];
        rok: integer;
    end;
```

Přístup do struktury

Do struktur přistupujeme pomocí operátoru tečky:

`a.x` – prvek `x` struktury `a`

Jednotlivé prvky se chovají jako běžné proměnné, můžeme tedy číst jejich hodnoty, zapisovat do nich...

Typ record – příklad použití

```
var a,b:bod; kn:kniha;  
begin  
    a.x:=1;  
    a.y:=2;  
    b.x:=10;  
    b.y:=10;  
    kn.autor:='Topfer, P.';  
    kn.nazev:='Algoritmy a programovací techniky';  
    kn.rok:=1995;  
end.
```

Pole struktur

```
var knihovna:array [1..100] of record
    autor:string[25];
    nazev:string[45];
    rok:integer;
end;
begin
    for i:=1 to 100 do begin
        readln(knihovna[i].autor);
        readln(knihovna[i].nazev);
        readln(knihovna[i].rok);
    end; ...
```

Klíčové slovo `with`

Chceme-li pracovat se strukturami, je otravné stále říkat, ve které struktuře se pohybujeme

(`struktura_s_dlouhym_nazvem_kterou_jsme_vymysleli_v_opilosti`)

Proto můžeme říct:

`with` `struktura` do příkaz nebo blok;

a v sekci `with` můžeme přistupovat k jednotlivým prvkům struktury rovnou.

Konstrukce with – příklad:

```
procedure vypis(kniha_s_dlouhym_nazvem:kniha);  
begin  
    with kniha_s_dlouhym_nazvem do  
        writeln(autor:25,nazev:46,rok:5);  
end;  
...  
    for i:=1 to 100 do  
        vypis(knihovna[i]);  
...
```


Problém třídění – motivace

- Máme načtena data (například čísla),
- chceme je zpracovat například v rostoucím pořadí.
- Jak to udělat? Setřídíme, zpracujeme.
- Předpokládejme, že data jsou v poli.

Problém třídění – jednoduché třídící algoritmy

- Bublínkové třídění (BubbleSort),
- zatřídování alias třídění přímým vkládáním (InsertSort),
- třídění výběrem (SelectSort),
- QuickSort.

Bublínkové třídění

- Geometrická interpretace:
Bublínky v kapalině jdou zpravidla vzhůru
- Myšlenka: Porovnáváme po sobě jdoucí čísla (ve smyslu sousední v zadaném poli) od prvního k poslednímu, jsou-li v nesprávném pořadí, prohodíme je.
- Prvky "probublávají" "správným" směrem.
- Opakujeme bublání, dokud se prohazuje.

Bubblesort v pseudokódu

- `prohazovalose:=true;`
- `while prohazovalose:=true do`
 - `begin`
 - `for i:=1 to pocet - 1 do`
 - `begin`
 - `prohazovalose:=false;`
 - `if pole[i]>pole[i+1] then`
 - `begin prohod(pole[i],pole[i+1]);`
 - `prohazovalose:=true;`
 - `end;`
 - `end;`
 - `end;`

Složitost bubble-sortu

- Kolikrát se provede vnější `while`-cyklus?

Složitost bubble-sortu

- Kolikrát se provede vnější `while`-cyklus?
- V i -tém kroku dojde i -tý největší na své místo!

Složitost bubble-sortu

- Kolikrát se provede vnější `while`-cyklus?
- V i -tém kroku dojde i -tý největší na své místo!
- Stačí tedy n -krát probublát, jedno probublání porovná po sobě jdoucí dvojice, tedy má složitost lineární.

Složitost bubble-sortu

- Kolikrát se provede vnější `while`-cyklus?
- V i -tém kroku dojde i -tý největší na své místo!
- Stačí tedy n -krát probublát, jedno probublání porovná po sobě jdoucí dvojice, tedy má složitost lineární.
- Složitost BubbleSortu je tedy $O(n^2)$.

Složitost bubble-sortu

- Kolikrát se provede vnější `while`-cyklus?
- V i -tém kroku dojde i -tý největší na své místo!
- Stačí tedy n -krát probublát, jedno probublání porovná po sobě jdoucí dvojice, tedy má složitost lineární.
- Složitost BubbleSortu je tedy $O(n^2)$.
- Implementaci, kdy se střídavě bublá z jedné strany na druhou a z druhé na první se říká ShakeSort a funguje pro něj stejný odhad složitosti.

Třídění přímým výběrem a zatřídováním

Přímý výběr:

- Opakuj, dokud není tříděné pole prázdné:
- Najdi v poli minimum a přesuň ho na konec setříděného pole.

Zatřídování:

- Opakuj, dokud není tříděné pole prázdné:
- Vyjmi z něj první prvek a zatříd do cílového pole, tedy: najdi pozici, kam prvek patří, přidej ho tam a zbytek setříděného pole posuň (o jedna dál).

Analýza složitosti: n krát opakujeme proces, který trvá nejvýše n kroků, tedy také $O(n^2)$.

Quicksort – třídění za pomoci rekurze – idea:

- Pokud třídíme jedno číslo, nic nedělej (posloupnost je setříděna),
tedy vrať posloupnost tak, jak jsme ji dostali.
- V poli POLE vyber jeden prvek (dále pivot).
- Rozděl POLE na pole A obsahující prvky menší než pivot
- a na pole B obsahující prvky větší nebo rovné pivotu.
- Pomocí sebe sama setříd' pole A ,
- pomocí sebe sama setříd' pole B ,
- Vypiš: pole A , pivot, pole B .

Jednotky – oddělený překlad

- Občas máme obecně využitelné funkce, které chceme používat v různých programech současně.
- Buďto je můžeme okopírovat mezi zdrojovými soubory (špatný nápad)
- nebo je dáme do zvláštního souboru, který budeme kompilovat zvlášť
- Tomu (druhému) řešení říkáme jednotky.

Jednotky – výhody a nevýhody

- Kód se rozlézá ve více souborech,
- jeden kód nemusíme psát vícekrát, chceme-li jej sdílet ve více programech.

Jednotky – syntax a sémantika

- Místo slovem `program` zahájíme soubor slovem `unit`,
- opět následuje jméno jednotky a tentokrát už se kontroluje!
- Jednotka má `interface` (popis, co je vidět zvenku)
- a část implementační (zahájenou slovem `implementation`).

Jednotky – interface

- Interface popisuje, co z jednotky je "vidět".
- V části interface jsou:
 - definice proměnných (viditelných zvenku),
 - prototypy funkcí a procedur (rovněž viditelných zvenku),
 - prototypem rozumíme hlavičku funkce ("první řádek").

Jednotky – implementace

- To, co nemá být vidět zvenku, tedy:
- Samotné definice funkcí,
- definice proměnných, které nemají být vidět zvenku,
- definice pomocných funkcí (které nemají být vidět zvenku).
- Jednotku ukončíme `end`. (!)

Jednotky – příklad

```
unit trideni;  
interface  
    type po=array[0..9] of integer;  
    procedure bubblej(var pole:array of integer);  
    procedure vytrid(var a:po);  
    procedure zatridovani(var a:po);  
    procedure quicksort(var pole:array of  
integer;kolik:integer);  
    procedure vypis(a:array of integer);
```

Jednotky – příklad (pokr.)

```
...
implementation
    var zatrideno:integer;
    procedure bublej(var pole:array of integer);
        ...
    function najdi_min(var a:po):integer;
        {Pozor, funkci najdi_min neni zvenku videt!}
        ...
    procedure vytrid(var a:po):integer;
        ...
        ...
end.
```

Použití jednotky

- Chceme-li jednotku použít, oznámíme to pomocí klíčového slova `uses`, před názvem dotyčné jednotky:
- Příklad: `uses trideni;`

Použití jednotky – příklad

```
program trid;  
uses trideni;  
var p:array [0..9] of integer;  
i:integer;  
begin  
for i:=0 to 9 do  
read(p[i]);  
quicksort(p,10);  
vypis(p);  
end.
```

Standardní jednotky

Turbo Pascal je vybaven několika standardními jednotkami:

- crt,
- dos,
- graph,
- printer,
- ...

Jednotky se mohou lišit pro různé překladače!

Jednotka crt

- Jednotka pro práci s obrazovkou a klávesnicí (barvy, zvuky)
- Proměnné: `LastMode` (údaj o posledním textovém módu před přechodem do grafiky),
- `TextAttr` (aktuální atribut zobrazení ovládaný pomocí `TextBackground` a `TextColor`),
- Procedura `TextBackground` nastaví barvu pozadí, proc. `TextColor` nastaví barvu popředí,
- funkce `keypressed` (vrací boolean a říká, zda byla stisknuta klávesa), `clrscr` (smaže obrazovku).

Jednotky dos, graph a printer

- Jednotka dos slouží především k práci se soubory, adresáři, disky...
- Jednotka graph slouží k práci s grafikou (InitGraph, CloseGraph, GraphResult, SetColor, GetColor...).
- Jednotka Printer slouží k tisku.
- Všechny tyto jednotky obsahují mnoho funkcí, které si nastudujete v případě potřeby.

Podivný příklad:

Ukazoval jsem několikrát program obsahující:

```
program nic;  
uses crt;  
...  
begin  
... repeat until keypressed;  
end.  
Co to bylo?
```


Podivný příklad:

Ukazoval jsem několikrát program obsahující:

```
program nic;  
uses crt;  
...  
begin  
... repeat until keypressed;  
end.
```

Co to bylo?

Použití jednotky crt, resp. její funkce keypressed.

Direktiva forward

- Občas nám mezi dvěma funkcemi vznikne vzájemná závislost:
- Jedna funkce volá druhou a druhá funkce volá funkci první.
- Překladač Pascalu se začne bouřit při prvním volání druhé funkce, protože ta ještě není definována!
- Proto před definicí první funkce musíme oznámit, že bude existovat druhá funkce.
- Napíšeme její prototyp a místo definice těla dáme klíčové slovo `forward`:
- `procedure dva(a:integer);forward;`

Forward příklad:

```
program qq;
procedure dva(a:integer);forward;
procedure jedna(a:integer);
begin
    dva(a);
end;
procedure dva(a:integer);
begin
    jedna(a);
end;
begin
    jedna(1);
    {Ponechme stranou, že program nic nedela!}
end.
```

Medián v lineárním čase

- Minule byl algoritmus Quicksort.
- Problémem bylo, jak volit pivot.
- Volíme-li špatně, děláme mnoho iterací rekurze.
- Hledáme-li pivot dlouho, nepříjemně to trvá.
- Jak hledat medián v lineárním čase?
- Ve skutečnosti nebudeme hledat medián, ale k -tý nejmenší prvek.

Medián v lineárním čase

- Rozděl vstup na pětice,
- v každé pětici najdi medián,
- najdi medián mediánů (tedy medián mezi mediány pětic),
- rozděl vstup na menší a větší,
- zjisti, zda se k -tý nejmenší nachází mezi většími nebo menšími
- pokračuj s hledáním příslušné hromádky.

Medián lineárně detaily

- Jak najít mediány pětic?

Medián lineárně detaily

- Jak najít mediány pětic?
- Hrubou silou (v konstantním čase, opakujeme lineárně-krát).

Medián lineárně detaily

- Jak najít mediány pětic?
- Hrubou silou (v konstantním čase, opakujeme lineárně-krát).
- Jak najít medián mediánů?

Medián lineárně detaily

- Jak najít mediány pětic?
- Hrubou silou (v konstantním čase, opakujeme lineárně-krát).
- Jak najít medián mediánů?
- Rekurzívně (zavolej se na pole mediánů pětic).

Medián lineárně detaily

- Jak najít mediány pětic?
- Hrubou silou (v konstantním čase, opakujeme lineárně-krát).
- Jak najít medián mediánů?
- Rekurzivně (zavolej se na pole mediánů pětic).
- Jak "pokračovat na příslušné hromádce?"

Medián lineárně detaily

- Jak najít mediány pětic?
- Hrubou silou (v konstantním čase, opakujeme lineárně-krát).
- Jak najít medián mediánů?
- Rekurzívně (zavolej se na pole mediánů pětic).
- Jak "pokračovat na příslušné hromádce?"
- Rekurzívně (zavolej se buďto na menší hromádku a hledej k -tý nejmenší, nebo máme-li hledat v hromádce větších hodnot budiž l počet prvků na menší hromádce a hledej v hromádce větších $k - l$ -tý nejmenší.

Proč je algoritmus lineární?

Protože:

- mediánů pětic je přibližně pětina délky vstupu,
- hromádka menších čísel stejně jako hromádka větších čísel bude mít velikost aspoň $3/10$,
- tudíž každá z hromádek bude mít též velikost nejvýš $7/10$.
- Zbytek je jen indukce.

Indukce:

Složitost algoritmu vede k rekurenci:

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + kn.$$

Ukážeme, že existuje l takové, že $T(n) \leq ln$:

$$T(n) \leq \frac{ln}{5} + \frac{7ln}{10} + kn = kn + \frac{9}{10}ln$$

a tedy stačí volit $l \geq 10k$.