Programování

Psaní čistého kódu

There are
10 types
of people
in the world:

Those who
understand binary
and those
who don't.

Martin Urza

Motivace

- Pro kompilátor je jedno, jestli je kód oddělen mezerami, odřádkováními či tabulátory.
- Přeložitelný kód může vypadat třeba i jako ten úchvatně přehledný výtvor v rámečku napravo.
- Proč tak nepsat, když to jde? Z téhož důvodu, proč Vám nepřednáším s roubíkem v puse. Váš kód může někdo chtít číst (i Vy sami, nezajímají-li Vás ostatní).
- Ač by většinu z Vás nejspíše nenapadlo napsat tu krásu, která je uvedena v modrém rámečku, napadá Vás mnoho jiných nečitelných půvabů.

```
program modifyChars;var
numIn:string;numOut:string;func
tion
max(a,b:integer):integer;begin
if (a>b) then max:=a else
max:=b;end;begin
readln(numIn);while
(0<length(numIn)) do begin
numOut:=copy(numIn,length(numIn
)-
2,3)+numOut;delete(numIn,max(1,
length(numIn)-2),3);if
(0<length(numIn)) then
numOut:=concat(',',numOut);end;
writeln(numOut);end.
```

Práce v týmu

- Píšete-li programy dlouhé desítky až stovky řádek, jste typicky jedinými autory. Reálné projekty mívají desítky až stovky tisíc řádek a jsou tvořeny prakticky výhradně v týmech složených z více programátorů.
- Pracuje-li více programátorů na jednom projektu, je nutné, aby po sobě tito byli schopni číst kód pokud možno co nejsnáze.
  - **V jistých ohledech je jednoznačné, co je dobře a co špatně. Někdy je hypoteticky správných možností více; pak je důležité, aby se programátoři shodli.**
- I když nebudete pracovat v týmu, pak vězte, že svůj kód po půl roce budete číst podobně jako cizí.

Odsazení kódu

- Ač mohou být různé názory na to, jakým způsobem odsazovat, **rozhodně je špatné neodsazovat vůbec.**
- V kódu se mohou vyskytovat celistvé bloky příkazů, které k sobě patří (typicky se jedná o takové, které mají být provedeny spolu, například podmíněně). Ty je třeba určitě odsazovat.
  - Tímto odsazením znázorníte, o které příkazy se jedná.
  - Doplnit případné chybějící begin/end bude snadnější.
- Dále je vhodné jednotně odsazovat při psaní cyklů a podmínek (příkaz a blok příkazů se může lišit).
  - Tady je dobré si pohlídat i to, aby kód neodjížděl příliš doprava (měl by se stále vejít na obrazovku).

Příklady odsazení

```
program programName;

var
  var1 : type1;
  var2 : type2;

begin

  order1;
  order2;

  while (condition1)
    do order3;

  if (condition2)
    then
    begin
      order4;
      order5;
    end
    else order6;

  while (condition3)
    do
    begin
      order7;
      order8;
    end;

end.
```

```
program programName;

var
  var1 : type1;

begin
  order1;

  while (condition1) do
    order2;

  if (condition2) then
  begin
    order3;
    order4;
  end
  else
    order5;

  while (condition3) do
  begin
    order6;
    order7;
  end;

  if (condition4) then
    order8
  else
    order9;
end.
```

```
program programName;

var
var1 : type1;

begin
order1;

while (condition1) do
  order2;

if (condition2) then
  begin
  order3;
  order4;
  end
else
  order5;

while (condition3) do
  begin
  order6;
  order7;
  end;

if (condition4) then
  order8
else
  order9;
end.
```

```
program programName;
var var1 : type1;
begin
order1;
if (condition2) then
begin
order2;
order3;
end
else order4;
end.
```

```
program programName;
  var var1 : type1;
begin
  order1;
  while (condition1) do order2;
  if (condition2) then begin
                         order3;
                         order4;
                       end
                  else order5;
  while (condition3) do begin
                         order6;
                         order7;
                       end;
  if (condition4) then order8
                  else order9;
end.
```

Špatná odsazení

- Téměř nejhorší je neodsazovat vůbec (viz. minulý slajd vpravo nahoře).
- Horší, než neodsazovat vůbec, je odsazovat náhodně a bez ohledu na logiku věci. Tento nápad vypadá tak hloupě, že by se mohlo zdát zbytečné jej zmiňovat, avšak se s tím lze kupodivu u některých lidí setkat.
- Další ne moc praktické -ač systematické- odsazování lze najít na minulém slajdu vpravo dole.
  - Při opakovaném vnoření (například podmínky do jiné podmínky) jde kód moc doprava.
  - Změna délky řádky (např. řádky s if) může vyžadovat změny v odsazení více jiných řádek.

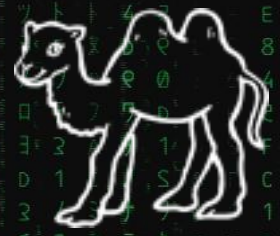
Jména proměnných

- Pro překladač je zcela irelevantní, jak jsou proměnné pojmenovány, beztak jejich jména vyoptimalizuje a nepoužívá.
- Programátorům mohou jména proměnných výrazně pomoci při orientaci v kódu; proměnné pojmenované špatně v tomto směru poskytují medvědí službu.
- Nejčastější špatný způsob pojmenování proměnných je naprosto náhodný (jména nesouvisí s obsahem).
- Další oblíbený způsob je zavedení „systému“ jmen ve stylu a, b, c, d, …., což také „usnadňuje“ orientaci.
- Problémy nastávají v každém kódu, kde je třeba více proměnných (než udržíte v hlavě, což je tak 2 až 8).

Jak správně pojmenovávat proměnné

- Ze jména proměnné by mělo být jasné, co proměnná označuje a co v ní je uloženo.
  - Že to děláte špatně, poznáte podle toho, že při použití nějaké hodnoty musíte výše v kódu hledat, kde je ona hodnota uložena.
  - Ještě zjevnějším příznakem toho, že je něco špatně, je pak situace, kdy koukáte na proměnnou a netušíte, co v ní vlastně je uloženo (což už je trošku extrém).
- Speciálním oříškem jsou proměnné typu boolean. Ty slouží typicky k uložení nějakého stavu. Jméno volte tak, aby bylo jasné, co znamená true a co false.
  - Například název proměnné prime (prvočíslo) není moc vypovídající; lepší je použít isPrime.

Různé notace pojmenování proměnných



- Je-li jméno proměnné jednoslovné, pak není žádný problém.
- Obsahuje-li jméno proměnné více slov, pak je více možností, jak od sebe tato slova oddělit.
- Oddělovat slova v proměnných mezerami či tečkami není možné (neprojde to přes kompilátor).
- Neoddělovat slova v proměnných vůbec nijak a psát je do jedné „nudle“ není dobrý nápad (nepřehledné).
- Oddělovat slova v proměnných podtržítky je dobré.
  - Příklad: is_prime
- Oddělovat slova velkými písmeny je také přehledné.
  - Příklad: isPrime

Funkce a procedury

- Každý program, který je možno zapsat pomocí funkcí a procedur, lze napsat i bez nich. Proč je používáme?
  - Hlavní důvod používání funkcí a procedur byl zmíněn v přednášce o funkcích a procedurách: používáme je proto, aby mohl být stejný kód volán (spuštěn) víckrát.
  - Existuje i jiný důvod. Funkce a procedury mohou být pojmenováním kódu, což zvyšuje přehlednost.
- Použití funkcí i procedur může být rozumné i tehdy, když se v celém kódu vyskytuje jen jedno volání.
  - Uvažte kód, který má přečíst vstup, potom jej upravit, následně z něj něco spočítat a pak vypsat na výstup. Rozdělíte-li kód do readInput(), editInput(), calculate() a writeOutput(), zpřehledníte jej a snížíte riziko chyby.

Jména funkcí a procedur

- Logika pojmenovávání funkcí a procedur je přibližně stejná jako logika pojmenovávání proměnných.
  - Pointou je pojmenovávat funkce podle toho, co vlastně dělají. Ta jména by měla být jasná, výstižná a přesná.
- Pro funkce typu boolean platí stejné doporučení jako pro booleovské proměnné. Jméno isPrime je jasnější než prime. Jméno isSet je lepší než set. Důvody jsou stejné jako u proměnných.
- Pozor na moc krátká jména. Problém je, že se mohou překrývat se jmény jiných funkcí. Například write() a read() jsou vestavěné funkce, jméno open() je celkem nicneříkající (nikdo neví, co se má otevřít).

Konstanty

- Hodnoty (všech typů, tedy čísla, řetězce a tak dále) je možné pojmenovávat.
- Klíčové slovo pro deklaraci konstant je const a používá se podobně jako var.
- Syntaxe deklarace konstant:
identifikátor = hodnota;
- Každá hodnota, která znamená něco jiného než sebe samu, by měla být pojmenována kvůli přehlednosti.
  - 12 neznamená počet měsíců v roce, 4 neznamená délku slova ahoj, 8 neznamená počet opakování
  - „s = ½ * a * b“ je vzorec pro výpočet obsahu pravoúhlého trojúhelníku; ½ v něm znamená ½

```
program showConst;

var
  a : integer;

const
  b = 2;
  c = 'Jadzia Dax';

begin
  writeln(b);
  a := 2 * b;
  writeln(a);
  writeln(c);
end.
```

Komentáře

- Komentáře jsou kusy textu, které je možné vložit do kódu a které budou překladačem zcela ignorovány.
- Jsou dvě možnosti, jak zapisovat komentáře:
  - Do složených závorek {}. Cokoliv je psáno ve složených závorkách, je překladačem ignorováno, příklad použití:

    ```
    { následující příkaz vypíše ahoj na výstup } write('ahoj');
    ```
  - Za dvě lomítka //. Cokoliv je psáno za dvěma lomítky, je až do konce řádky ignorováno, příklad použití:

    ```
    write('ahoj'); //na výstup vypsáno ahoj, toto je ignorováno
    ```
- Do komentářů si může programátor zapisovat různé poznámky a postřehy ke kódu. Ve větších projektech jsou komentáře naprosto nezbytné.

Další možnost použití komentářů

- Představte si situaci, že ladíte kód a chcete nějakou jeho část smazat, ale nejste si jisti, zda je tento krok správný (a třeba jej nebude třeba znovu obnovit).
- V takovém případě není třeba kód odstraňovat, stačí jej umístit do složených závorek, čímž z něj uděláte komentář. Takový kód pak bude ignorován (protože se z pohledu kompilátoru vlastně už nejedná o kód, nýbrž o komentář).
  - Budete-li dotyčný kód někdy znovu potřebovat, stačí jej zas z komentáře vyjmout.
- Pozor na kód, jehož valná většina je zakomentována, protože je pak trochu problém se v něm vyznat.

Jazyk

- Vždy je lepší používat angličtinu místo češtiny, neboť se jedná o jazyk mezinárodní, takže tomu bude více lidí rozumět.
- Je ale také možné, že Vaše vlastní angličtina není na takové úrovni, jak byste si možná přáli.
  - V takovém případě lze přistoupit na kompromis, který spočívá v tom, že budete psát anglický kód a k němu české komentáře (v kódu není nutné třeba sestavovat celé věty, stačí slovní zásoba na jména identifikátorů).
  - Nejste-li schopni ani takového kompromisu, pište celý kód česky (proměnné, funkce, komentáře, ….).
- Nekombinujte český kód s anglickým!!

Rekapitulace

- Měli byste chápat, proč je důležité nepsat kód jako prase ;o)
- Nejdůležitější znaky „hezkého“ kódu jsou zejména:
  - správné (systematické) odsazování
  - rozumné pojmenovávání proměnných
  - používání funkcí a procedur
  - správné pojmenovávání funkcí a procedur
- Měli byste vědět, proč a které konstanty pojmenovat, což je velmi těžké k pochopení. Samotné čtení slajdů k tomu stačit rozhodně nebude.
- Měli byste umět okomentovat kód, případně využít komentářů k „mazání“ částí kódu.