Jak psát přehledný kód

- pěkně napsaný kód je přehledný, srozumitelný, snadno upravitelný (i po čase) a rozšiřitelný
- nižší riziko chyb věcných i čistě syntaktických
- vyšší efektivita práce

Odsazování

- správné odsazování (ideální jsou 4 mezery; ne tabulátor – má proměnnou délku)
- mezi logickými částmi kódu (např.: inicializace dat, výpočet, tisk výsledků) vynecháváme řádek
- mezi funkcemi vynecháváme rozumné množství řádků (u menších funkcí většinou 2)
- řádky nejsou příliš dlouhé, případně je lze vhodně rozdělit
- neužíváme přemíru závorek a interpunkce (středníky apod.)

Názvy

- názvy proměnných i funkcí odpovídají tomu, co obsahují či dělají
- názvy nejsou přehnaně dlouhé
- většinou nepoužíváme jednopísmenné názvy
  kromě proměnné cyklu for (i, j, k, n) či argumentů matematických funkcí (x, n)
- v textu nenecháváme záhadné číselné hodnoty (raději definujeme konstanty s výstižným názvem)

Komentáře

- volba výstižných názvů eliminuje mnohé přebytečné komentáře
  omezí se tak možnost chyby (při změně kódu se občas zapomene upravit příslušný komentář)
- program začíná stručným popisem toho, co dělá

Funkce

- funkce by měla být relativně malá a průhledná – dělat jednu věc a pořádně
- neměla by dělat „další změny na pozadí“ (činnosti a změny v datech, které k vykonání jejího úkolu nejsou nutné)
- funkce nemá mít příliš mnoho argumentů (zpravidla 0–3)
- problém by měl být na funkce rozložen přirozeně – logicky

Knihovny

- pokud je soubor příliš dlouhý, můžeme z funkcí vytvářet knihovny
- přehledný soubor zpravidla nemá více než 100 – 200 řádků
- funkce použitelné i v jiném programu sdružujeme do nových knihoven, neprogramujeme je stále znovu
- neměla by vznikat potřeba kopírovat nějakou část kódu na více míst (kandidát na zapouzdření do funkce)

Postup

- přehledný kód často (zejména u větších projektů) nevzniká hned
- *ihned a automaticky: správně odsazujeme a volíme vhodné názvy*
- většinou už při návrhu: úlohu rozdělíme na menší části; některé funkce však vzniknou až při psaní kódu
- přehlednosti kódu věnujeme přiměřenou pozornost (na začátku se nejvíce soustředíme zejména na základní funkčnost, poté můžeme kód refaktorovat (zpřehledňovat), někdy dojde i k podstatnému zjednodušení)
- nepotřebný kód uvážlivě mažeme (máme-li už lepší) či odkládáme do jiného souboru (může-li se hodit k něčemu jinému)

Literatura: M. Fowler: Refaktoring; http://knihy.cpress.cz/cisty-kod.html (R. C. Martin: Čistý kód)
http://knihy.cpress.cz/dokonaly-kod.html (S. McConnell: Dokonalý kód)