

# Obsah

---

Tipy podle zaměření .....	11
Úvod .....	17
Jednejte s opatrností .....	20
<i>Seb Rose</i>	
Aplikujte principy funkcionálního programování .....	22
<i>Edward Garson</i>	
Položte si otázku: „Co by udělal uživatel?“ (vy jím nejste) .....	24
<i>Giles Colborne</i>	
Automatizujte své kódovací standardy .....	26
<i>Filip van Laenen</i>	
V jednoduchosti je krása .....	28
<i>Jørn Ølmheim</i>	
Před refaktorováním .....	30
<i>Rajith Attapattu</i>	
Pozor na sdílený kód .....	32
<i>Udi Dahan</i>	
Skautské pravidlo .....	34
<i>Robert C. Martin (Uncle Bob)</i>	
Než začnete obviňovat ostatní, zkontrolujte vlastní kód .....	36
<i>Allan Kelly</i>	

Nástroje volte s rozvahou .....	38
<i>Giovanni Asproni</i>	
Programujte v jazyce domény .....	40
<i>Dan North</i>	
Kód představuje návrh .....	42
<i>Ryan Brush</i>	
Na odsazení kódu záleží .....	44
<i>Steve Freeman</i>	
Revize kódu .....	46
<i>Mattias Karlsson</i>	
Odůvodnění správnosti kódu .....	48
<i>Yechiel Kimchi</i>	
Pár slov ke komentářům .....	50
<i>Cal Evans</i>	
Komentujte pouze to, co kód sám nedokáže říct .....	52
<i>Kevlin Henney</i>	
Nepřetržité učení .....	54
<i>Clint Shank</i>	
Pohodlnost nepatří mezi ctnosti .....	56
<i>Gregor Hohpe</i>	
Nasazujte brzy a často .....	58
<i>Steve Berczuk</i>	
Rozlište technické a business výjimky .....	60
<i>Dan Bergh Johnson</i>	
Procvičujte se .....	62
<i>Jon Jagger</i>	
Jazyky specifické pro doménu .....	64
<i>Michael Hunger</i>	
Nebojte se, že něco rozbijete .....	66
<i>Mike Lewis</i>	
Nesnažte se okouzlit testovacími daty .....	68
<i>Rod Begbie</i>	

Neignorujte chyby .....	70
<i>Pete Goodliffe</i>	
Nestačí pouze naučit se jazyk, ale porozumět jeho kultuře.....	72
<i>Anders Norås</i>	
Nesnažte se svůj program přibít do vzpřímené polohy .....	74
<i>Verity Stob</i>	
Nevěřte na kouzla.....	76
<i>Alan Griffiths</i>	
Neopakujte se .....	78
<i>Steve Smith</i>	
Nedotýkej se toho kódu.....	80
<i>Cal Evans</i>	
Zapouzdřete chování, ne pouze stav.....	82
<i>Einar Landre</i>	
Čísla s plovoucí řádovou čárkou nejsou reálná .....	84
<i>Chuck Allison</i>	
Naplňte své ambice pomocí open-source.....	86
<i>Richard Monson-Haefel</i>	
Zlaté pravidlo návrhu API .....	88
<i>Michael Feathers</i>	
Mýtus guru .....	90
<i>Ryan Brush</i>	
Dřina se nevyplácí .....	92
<i>Olve Maudal</i>	
Použití systému pro sledování chyb.....	94
<i>Matt Doar</i>	
Vylepšete kód tím, že ho odstraníte.....	96
<i>Pete Goodliffe</i>	
Nainstaluj si mě.....	98
<i>Marcus Baker</i>	
Komunikace mezi procesy ovlivňuje reakční dobu aplikace.....	100
<i>Randy Stafford</i>	

Udržujte build čistý .....	102
<i>Johannes Brodwall</i>	
Naučte se používat nástroje příkazového řádku .....	104
<i>Carroll Robinson</i>	
Ovládejte více než dva programovací jazyky .....	106
<i>Russel Winder</i>	
Poznejte své IDE .....	108
<i>Heinz Kabutz</i>	
Poznejte své limity .....	110
<i>Greg Colvin</i>	
Poznejte, kdy dokončíte práci .....	112
<i>Dan Bergh Johnsson</i>	
Rozsáhlá vzájemně propojená data patří do databáze .....	114
<i>Diomidis Spinellis</i>	
Naučte se cizí jazyky .....	116
<i>Klaus Marquardt</i>	
Naučte se odhadovat .....	118
<i>Giovanni Asproni</i>	
Naučte se říct: „Ahoj, světe“ .....	120
<i>Thomas Guest</i>	
Nechte projekt mluvit sám za sebe .....	122
<i>Daniel Lindner</i>	
Linker není žádným magickým programem .....	124
<i>Walter Bright</i>	
Dlouhověkost provizorních řešení .....	126
<i>Klaus Marquardt</i>	
Rozhraní by se měla snadno používat správným způsobem a těžko špatným .....	128
<i>Scott Meyers</i>	
Ať se neviditelné stane více viditelným .....	130
<i>Jon Jagger</i>	

Předávání zpráv vede v paralelních systémech k lepší rozšiřitelnosti .....	132
<i>Russel Winder</i>	
Odkaz budoucnosti .....	134
<i>Linda Rising</i>	
Nevyužité příležitosti k polymorfizmu .....	136
<i>Kirk Pepperdine</i>	
Novinka: Testeři jsou vaši kamarádi .....	138
<i>Burk Hufnagel</i>	
Pouze jeden binární soubor .....	140
<i>Steve Freeman</i>	
Pouze kód říká pravdu .....	142
<i>Peter Sommerlad</i>	
Nezapomínejte na sestavení. ....	144
<i>Steve Berczuk</i>	
Párové programování .....	146
<i>Gudny Hauknes, Kari Røssland a Ann Katrin Gagnat</i>	
Preferujte typy specifické pro doménu před primitivními typy ....	148
<i>Einar Landre</i>	
Předcházejte chybám .....	150
<i>Giles Colborne</i>	
Profesionální programátor .....	152
<i>Robert C. Martin (Uncle Bob)</i>	
Všechno verzujte. ....	154
<i>Diomidis Spinellis</i>	
Odložte myš a odstupte od klávesnice .....	156
<i>Burk Hufnagel</i>	
Čtete kód .....	158
<i>Karianne Berg</i>	
Zajímejte se o humanitní vědy. ....	160
<i>Keith Braithwaite</i>	

Často znovu vynalézejte kolo .....	162
<i>Jason P. Sage</i>	
Odolejte pokušení návrhového vzoru jedináček .....	164
<i>Sam Saariste</i>	
Honbu za výkonem komplikuje špinavý kód .....	166
<i>Kirk Pepperdine</i>	
Jednoduchost jde ruku v ruce s redukcí .....	168
<i>Paul W. Homer</i>	
Princip jedné odpovědnosti .....	170
<i>Robert C. Martin (Uncle Bob)</i>	
Začněte u Ano .....	172
<i>Alex Miller</i>	
Odstupte a automatizujte, automatizujte, automatizujte .....	174
<i>Cay Horstmann</i>	
Využijte nástrojů pro analýzu kódu .....	176
<i>Sarah Mount</i>	
Testujte požadované, nikoli nahodilé chování .....	178
<i>Kevlin Henney</i>	
Testujte přesně a konkrétně .....	180
<i>Kevlin Henney</i>	
Testujte ve spánku (a přes víkendy) .....	182
<i>Rajith Attapattu</i>	
Testování je přísností softwarového vývoje .....	184
<i>Neal Ford</i>	
Myšlení ve stavech .....	186
<i>Niclas Nilsson</i>	
Dvě hlavy jsou často lepší než jedna .....	188
<i>Adrian Wible</i>	
Dvakrát špatně může znamenat dobře (a těžko se opravuje) .....	190
<i>Allan Kelly</i>	
Ubuntu pro vaše přátele .....	192
<i>Aslam Khan</i>	

Unixové nástroje jsou vaši kamarádi .....	194
<i>Diomidis Spinellis</i>	
Použijte správný algoritmus a datovou strukturu .....	196
<i>Jan Christiaan „JC“ van Winkel</i>	
Přehnané protokolování vám na klidném spánku nepřidá .....	198
<i>Johannes Brodwall</i>	
WET pomáhá překonat výkonnostní překážky.....	200
<i>Kirk Pepperdine</i>	
Když programátoři a testeři spolupracují.....	202
<i>Janet Gregory</i>	
Pišťe kód tak, jako byste museli zajišťovat jeho podporu po zbytek svého života .....	204
<i>Yuriy Zubarev</i>	
Vytvářejte malé funkce na základě příkladů .....	206
<i>Keith Braithwaite</i>	
Vytvářejte testy pro lidi.....	208
<i>Gerard Meszaros</i>	
O kód se musíte starat.....	210
<i>Pete Goodliffe</i>	
Vaši zákazníci neuvažují nad tím, co říkají .....	212
<i>Nate Jackson</i>	
Příspěvatelé .....	214
Rejstřík .....	235





# Tipy podle zaměření

---

## **Chyby a opravy**

Než začnete obviňovat ostatní, zkontrolujte vlastní kód .....	36
Nedotýkej se toho kódu .....	80
Použití systému pro sledování chyb .....	94
Dvakrát špatně může znamenat dobře (a těžko se opravuje) .....	190

## **Vývoj**

Nasazujte brzy a často .....	58
Nedotýkej se toho kódu .....	80
Nainstaluj si mě .....	98
Udržujte build čistý .....	102
Nechte projekt mluvit sám za sebe .....	122
Pouze jeden binární soubor .....	140
Nezapomínejte na sestavení .....	144

## **Vzhled kódu**

Automatizujte své kódovací standardy .....	26
Na odsazení kódu záleží .....	44
Revize kódu .....	46
Pár slov ke komentářům .....	50
Komentujte pouze to, co kód sám nedokáže říct .....	52
Využijte nástrojů pro analýzu kódu .....	176

## Techniky návrhu

Aplikujte principy funkcionálního programování . . . . .	22
Položte si otázku: „Co by udělal uživatel?“ (vy jím nejste) . . . . .	24
V jednoduchosti je krása . . . . .	28
Nástroje volte s rozvahou . . . . .	38
Programujte v jazyce domény . . . . .	40
Kód představuje návrh . . . . .	42
Odůvodnění správnosti kódu . . . . .	48
Pohodlnost nepatří mezi ctnosti . . . . .	56
Rozlište technické a business výjimky . . . . .	60
Neopakujte se . . . . .	78
Zapouzdřete chování, ne pouze stav . . . . .	82
Zlaté pravidlo návrhu API . . . . .	88
Komunikace mezi procesy ovlivňuje reakční dobu aplikace . . . . .	100
Rozhraní by se měla snadno používat správným způsobem a těžko špatným . . . . .	128
Předávání zpráv vede v paralelních systémech k lepší rozšiřitelnosti . . . . .	132
Nevyužité příležitosti k polymorfismu . . . . .	136
Pouze kód říká pravdu . . . . .	142
Preferujte typy specifické pro doménu před primitivními typy . . . . .	148
Předcházejte chybám . . . . .	150
Odolejte pokušení návrhového vzoru jedináček . . . . .	164
Princip jedné odpovědnosti . . . . .	170
Myšlení ve stavech . . . . .	186
WET pomáhá překonat výkonnostní překážky . . . . .	200

## Myšlení v souvislostech

Programujte v jazyce domény . . . . .	40
Jazyky specifické pro doménu . . . . .	64
Naučte se cizí jazyky . . . . .	116
Preferujte typy specifické pro doménu před primitivními typy . . . . .	148
Zajímejte se o humanitní vědy . . . . .	160
Myšlení ve stavech . . . . .	186
Vytvářejte malé funkce na základě příkladů . . . . .	206

## **Chyby, výjimky a ladění**

Rozlište technické a business výjimky .....	60
Neignorujte chyby .....	70
Nesnažte se svůj program přibít do vzpřímené polohy .....	74
Předcházejte chybám .....	150
Přehnané protokolování vám na klidném spánku nepřidá .....	198

## **Výuka a dovednosti**

Nepřetržitě učení .....	54
Procvičujte se .....	62
Nestačí pouze naučit se jazyk, ale porozumět jeho kultuře .....	72
Naplňte své ambice pomocí open-source .....	86
Mýtus guru .....	90
Dřina se nevyplácí .....	92
Čtěte kód .....	158
Zajímejte se o humanitní vědy .....	160
Často znovu vynalézejte kolo .....	162

## **Tajemno a magično**

Nevěřte na kouzla .....	76
Nedotýkej se toho kódu .....	80
Mýtus guru .....	90
Naučte se používat nástroje příkazového řádku .....	104
Linker není žádným magickým programem .....	124
Testujte ve spánku (a přes víkendy) .....	182
Přehnané protokolování vám na klidném spánku nepřidá .....	198
Pišťe kód tak, jako byste museli zajišťovat jeho podporu po zbytek svého života .....	204

## **Výkon a optimalizace**

Aplikujte principy funkcionálního programování .....	22
Čísla s plovoucí řádovou čárkou nejsou reálná .....	84
Vylepšete kód tím, že ho odstraníte .....	96
Komunikace mezi procesy ovlivňuje reakční dobu aplikace .....	100
Poznejte své limity .....	110

Rozsáhlá vzájemně propojená data patří do databáze .....	114
Předávání zpráv vede v paralelních systémech k lepší rozšiřitelnosti .....	132
Honbu za výkonem komplikuje špinavý kód .....	166
Použijte správný algoritmus a datovou strukturu .....	196
WET pomáhá překonat výkonostní překážky .....	200
Profesionální přístup .....	
Nepřetržité učení .....	54
Procvičujte se .....	62
Dřina se nevyplácí .....	92
Dlouhověkost provizorních řešení .....	126
Profesionální programátor .....	152
Odložte myš a odstupejte od klávesnice .....	156
Testování je přísnoostí softwarového vývoje .....	184
Piště kód tak, jako byste museli zajišťovat jeho podporu po zbytek svého života .....	204
O kód se musíte starat .....	210

## **Programovací jazyky a paradigmatata**

Aplikujte principy funkcionálního programování .....	22
Jazyky specifické pro doménu .....	64
Nestačí pouze naučit se jazyk, ale porozumět jeho kultuře .....	72
Ovládejte více než dva programovací jazyky .....	106
Naučte se cizí jazyky .....	116

## **Refaktorování a údržba kódu**

Jednejte s opatrností .....	20
Před refaktorováním .....	30
Skautské pravidlo .....	34
Komentujte pouze to, co kód sám nedokáže říct .....	52
Nebojte se, že něco rozbijete .....	66
Vylepšete kód tím, že ho odstraníte .....	96
Udržujte build čistý .....	102
Poznejte, kdy dokončíte práci .....	112
Dlouhověkost provizorních řešení .....	126
Odkaz budoucnosti .....	134

Pouze kód říká pravdu .....	142
Nezapomínejte na sestavení. ....	144
Profesionální programátor .....	152
Honbu za výkonem komplikuje špinavý kód .....	166
Jednoduchost jde ruku v ruce s redukcí .....	168
Ubuntu pro vaše přátele. ....	192
O kód se musíte starat. ....	210

## **Znovupoužití verzus opakování**

Pozor na sdílený kód .....	32
Pohodlnost nepatří mezi ctnosti .....	56
Procvičujte se .....	62
Neopakujte se .....	78
Často znovu vynalézejte kolo .....	162
Použijte správný algoritmus a datovou strukturu .....	196
WET pomáhá překonat výkonnostní překážky .....	200

## **Plány, termíny a odhady**

Jednejte s opatrností .....	20
Kód představuje návrh .....	42
Poznejte, kdy dokončíte práci .....	112
Naučte se odhadovat .....	118
Ať se neviditelné stane více viditelným. ....	130

## **Jednoduchost**

V jednoduchosti je krása .....	28
Naučte se říct: „Ahoj světe“ .....	120
Odkaz budoucnosti .....	134
Jednoduchost jde ruku v ruce s redukcí .....	168

## **Práce v týmu**

Revize kódu. ....	46
Naučte se cizí jazyky .....	116
Párové programování .....	146
Začněte u Ano .....	172

Dvě hlavy jsou často lepší než jedna .....	188
Ubuntu pro vaše přátele.....	192
Když programátoři a testeři spolupracují.....	202

## **Testy, testování a testeři**

Aplikujte principy funkcionálního programování.....	22
Kód představuje návrh .....	42
Nesnažte se okouzlit testovacími daty .....	68
Zlaté pravidlo návrhu API .....	88
Rozhraní by se měla snadno používat správným způsobem a těžko špatným .....	128
Ať se neviditelné stane více viditelným.....	130
Novinka: Testeři jsou vaši kamarádi .....	138
Testujte požadované, nikoli nahodilé chování.....	178
Testujte přesně a konkrétně .....	180
Testujte ve spánku (a přes víkendy) .....	182
Testování je přísností softwarového vývoje.....	184
Když programátoři a testeři spolupracují.....	202
Vytvářejte malé funkce na základě příkladů .....	206
Vytvářejte testy pro lidi.....	208

# Úvod

---

*I ten nejnovější počítač pouze znásobuje pradávný problém vztahu mezi lidskými bytostmi a na konci bude zprostředkovatel muset čelit problému toho, co říci a jak.*

*Edward R. Murrow*

V hlavách programátorů se honí mnoho myšlenek. Programovací jazyky, programovací techniky, vývojová prostředí, způsoby kódování, nástroje, proces vývoje, termíny, schůzky, softwarová architektura, návrhové vzory, dynamika týmu, kód, požadavky, chyby, kvalita kódu a mnoho a mnoho dalšího.

Programování je umění, dovednost a věda, zdaleka přesahující rámec samotného programu. Programování spojuje diskrétní svět počítačů s fluidním světem lidských záležitostí. Programátoři jsou prostředníky mezi vyjednanými a nejistými obchodními fakty a neústupnou oblastí bitů a bajtů a vyšších jazykových konstruktů.

Vzhledem k množství potřebných znalostí a bezpočtu možných postupů nemůže nikdo označovat právě svůj postup za ten jediný správný. Kniha *97 klíčových znalostí programátora* staví na kolektivních vědomostech a zkušenostech a spíše než ucelený větší obrázek poskytuje představu odborné veřejnosti o tom, co by měl každý programátor znát. Svým rozsahem tak pokrývá vše od rad týkajících se kódu po vzdělávání, od použití algoritmů po agilní uvažování, od implementačního know-how po profesionalismus, od způsobu k podstatě.

Příspěvky na sebe vzájemně nenavazují a není to ani záměr – spíše opak je pravdou. Hodnota příspěvků pramení právě z jejich osobitosti. Hodnota kolekce spočívá v tom, jak se vzájemně doplňují, potvrzují, nebo jsou dokonce v rozporu. Nenajdete zde žádné oslí můstky – je na vás reagovat, uvážit a spojit si dohromady, co čtete, vzít v potaz váš vlastní kontext a zkušenosti.

## Oprávnění

Autorskou ochranu každého článku lze přirovnat k open-source řešením. Každý tip je dostupný online a chráněn na základě licence Creative Commons, což znamená, že můžete každý z článků samostatně využívat ve své praxi, pokud uvedete jeho autora.

## Poděkování

Mnoho lidí přispělo svým časem či názory ke vzniku knihy *97 klíčových znalostí programátora*, a to přímo i nepřímo. Ti všichni si zaslouží uznání.

Richard Monson-Haefel je původce myšlenky edice *97 klíčových znalostí* a také sestavil první knihu této série – *97 klíčových znalostí softwarového architekta*, do které jsem přispěl. Chtěl bych Richardovi poděkovat za tuto sérii, její koncept otevřeného přispívání a hlavně za podporu mého nápadu realizovat tuto knihu.

Chci poděkovat všem, kteří věnovali čas a úsilí přispět do tohoto projektu. Příspěvatelům, kteří jsou se svými radami uvedeni v této knize, i těm, kteří vybrání nebyli. Velký objem a kvalita příspěvků učinily výběr hodně složitým, pevně daný počet bohužel vyřadil i několik těch, které by se normálně do knihy podařilo vmáčknout. Velký dík za zpětnou vazbu, komentáře a další nápady patří Giovanni Aspronimu, Paulu Colin Glosterovi a Michaelu Hungerovi.

Děkuji O'Reilly za podporu, kterou věnovalo tomuto projektu, velkou pozornost a lidem v O'Reilly jmenovitě pak Miku Loukidesovi, Laurelu Ackermanovi, Ediemu Freedmanovi, Edu Stephensonovi a Rachel Monaghanové.

Chtěl bych také poděkovat svojí ženě Carolyn za to, že vnesla pořádek do mého chaosu, a také mým dvěma synům, Stefanovi a Yannickovi za znovuvyvoření části chaosu. Doufám, že kniha vám poskytne informace, vhled a inspiraci.

## Poznámka redakce českého vydání

Nakladatelství Computer Press, které pro vás tuto knihu přeložilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press  
redakce PC literatury  
Holandská 8 639 00 Brno

nebo

*knihy@cpress.cz*

Další informace a případné opravy českého vydání knihy najdete na internetové adrese <http://knihy.cpress.cz/k1834>. Prostřednictvím uvedené adresy můžete též naší redakci zaslat komentář nebo dotaz týkající se knihy. Na vaše reakce se srdečně těšíme.