

Stručný obsah

Úvod	41
Překlad a sestavení programu: základní dovednosti	43
Překlad a sestavení programu: další možnosti	57
Implementační a hlavičkové soubory	73
Ladění v integrovaném vývojovém prostředí	83
Ladění pomocí programových konstrukcí	97
Program, jeho běh a prostředí	103
Deklarace a proměnné	113
Ukazatele a pole	123
Správa paměti	135
Reference	145
Práce se znaky	149
Znakové řetězce (pole znaků)	159
Znakové řetězce (typ <code>basic_string<></code>)	175
Funkce a ukazatele na ně	187
Preprocesor a makra	205
Celá čísla	215
Reálná čísla	225
Příkazy	237
Výrazy a operátory	249
Výčtové typy	259

Struktury a unie	265
Objektový typ a jeho deklarace	273
Vytvoření instance	289
Kopírování instancí	299
Zánik instance	309
Dědění a polymorfismus	315
Objektový návrh	331
Přetěžování operátorů	341
Zpracování chybových stavů	361
Jmenné prostory	375
Šablony	383
Práce s datovými typy	403
Vstupní a výstupní operace v jazyce C	413
Vstupní a výstupní operace v jazyce C++	435
 Na přiloženém CD naleznete	
Kontejnery ve standardní knihovně	453
Algoritmy ve standardní knihovně	467
Lokální nastavení v C a v C++	487
Komplexní čísla	503
C++0x	511
Několik tipů na závěr	521

Obsah

Úvod	41
Komu je kniha určena	41
Doprovodné CD	42
Překlad a sestavení programu: základní dovednosti	43
1 Překládáme program složený z jediného souboru	43
2 Překládáme program a určujeme jméno výsledného souboru (bcc32, cl)	43
3 Překládáme program a určujeme jméno výsledného souboru (g++)	44
4 Jak přeložit soubor bez sestavování	44
5 Sestavujeme spustitelný program z objektového souboru (bcc32, cl)	44
6 Sestavujeme spustitelný program z objektového souboru (g++)	44
7 Překládáme program složený z několika souborů	45
8 Překládáme program složený z několika souborů a určujeme jméno výsledného souboru (bcc32)	45
9 Překládáme program složený z několika souborů a určujeme jméno výsledného souboru (cl, g++)	46
10 Připojujeme nestandardní knihovnu (bcc32)	46
11 Připojujeme nestandardní knihovnu (C++Builder 2009)	46
12 Připojujeme nestandardní knihovnu (cl)	47
13 Připojujeme nestandardní knihovnu (Visual C++ 2008)	48
14 Určujeme adresář s knihovnamí (g++)	48
15 Připojujeme konkrétní standardní knihovnu (g++)	49
16 Připojujeme konkrétní nestandardní knihovnu (g++)	49
17 Jak ušetřit práci: Program MAKE	49
18 Jak na soubor Makefile	50
19 Pozor na mezeru před pravidlem v souboru makefile	50
20 Když se příkaz v souboru makefile nevejde na jeden řádek	50
21 Odkazujeme na jméno cíle v souboru makefile (mingw32-make)	51
22 Odkaz na předpoklady v souboru makefile (mingw32-make)	51
23 Odkaz na všechny soubory daného typu v souboru makefile (mingw32-make)	51
24 Odkaz na první předpoklad v souboru makefile (mingw32-make)	51
25 Jak na úklid pomocí souboru makefile (mingw32-make)	52
26 Jak na všechny úpravy souboru makefile z předchozích tipů	52
27 Jaký jazyk překládáme? (bcc)	52
28 Jaký jazyk překládáme? (C++Builder)	53
29 Jaký jazyk překládáme? (cl)	54
30 Jaký jazyk překládáme? (Visual Studio 2008)	54

31	Jaký jazyk překládáme? (g++)	55
32	Příklad zdrojového souboru v C99 (g++)	55
33	Jak získat základní nápovědu k překladači (bcc32)	55
34	Jak na jména standardních hlavičkových souborů	55
35	Jak využít hlavičkové soubory z jazyka C v C++	56
	Překlad a sestavení programu: další možnosti	57
36	Jak na optimalizaci	57
37	Optimalizujeme velikost výsledného programu	57
38	Optimalizujeme velikost výsledného programu (C++Builder 2009)	58
39	Optimalizujeme velikost výsledného programu (Visual Studio 2008)	59
40	Optimalizujeme rychlost výsledného programu	60
41	Optimalizujeme rychlost výsledného programu v g++	60
42	Optimalizujeme rychlost výsledného programu (C++Builder 2009)	60
43	Optimalizace rychlosti výsledného programu (Visual Studio 2008)	61
44	Proč definovat podmínková jména	61
45	Definujeme podmínkové jméno	61
46	Jak na definici podmínkového jména (C++Builder 2009)	62
47	Jak na definici podmínkového jména (Visual C++ 2008)	63
48	Jak na další adresář pro hlavičkové soubory	63
49	Jak na další adresář pro hlavičkové soubory (C++Builder 2009)	64
50	Jak na další adresář pro hlavičkové soubory (Visual C++ 2008)	65
51	Vytvoření statické knihovny (bcc32, cl)	66
52	Vytvoření statické knihovny (g++)	66
53	Vytvoření statické knihovny (C++Builder 2009)	67
54	Vytvoření statické knihovny (Visual Studio 2008)	67
55	Co je výstupem preprocesoru	68
56	Co je výstupem preprocesoru (cl)	68
57	Co je výstupem preprocesoru (g++)	68
58	Co je výstupem preprocesoru (Visual Studio 2008)	68
59	Příklad vybraného zdrojového souboru do assembleru (bcc32)	69
60	Příklad vybraného zdrojového souboru do assembleru (cl)	69
61	Příklad vybraného zdrojového souboru do assembleru (g++)	70
62	Příklad vybraného zdrojového souboru do assembleru (C++Builder 2009)	70
63	Příklad vybraného zdrojového souboru do assembleru (Visual Studio 2008)	70
64	Příklad všech souborů projektu do assembleru (C++Builder 2009)	71
65	Příklad všech zdrojových souborů do assembleru (Visual Studio 2008)	72

	Implementační a hlavičkové soubory	73
66	Co je modul a jeho rozhraní	73
67	Použití hlavičkového souboru	73
68	Kde překladač hledá hlavičkový soubor	73
69	Kam lze vložit hlavičkový soubor	74
70	Jak zabránit opakovanému čtení hlavičkového souboru	74
71	Jak zabránit opakovanému čtení hlavičkového souboru ve Visual C++	74
72	Jak exportovat globální proměnnou z modulu	74
73	Jak na globální proměnnou, kterou nechceme exportovat	75
74	Jak na globální proměnnou, kterou nechceme exportovat (jen C++)	75
75	Jak exportovat konstanty (const) z modulů	75
76	Jak exportovat konstanty (const) z modulů – společné řešení pro C a C++	76
77	Jak na konstanty, které z modulu nechceme exportovat	76
78	Jak exportovat funkci z modulu	76
79	Jak na funkci, kterou nechceme exportovat	77
80	Jak na funkci, kterou nechceme exportovat (jen C++)	77
81	Jak na funkci s modifikátorem inline v hlavičkových souborech (C++ a C99)	77
82	Makra v hlavičkových souborech	78
83	Neobjektový datový typ exportovaný z modulu	78
84	Objektový typ exportovaný z modulu	78
85	Vložené (inline) metody	79
86	Datový typ, který nechceme exportovat z modulu	79
87	Šablona funkce exportovaná z modulu – typické řešení	80
88	Šablona funkce exportovaná z modulu – jiná možnost (jen některé nejnovější překladače)	80
89	Šablona třídy exportovaná z modulu – typické řešení	80
90	Šablona třídy exportovaná z modulu – jiná možnost (jen některé nejnovější překladače)	81
	Ladění v integrovaném vývojovém prostředí	83
91	Příklad pro ladění (C++Builder)	83
92	Příklad pro ladění (Visual C++)	83
93	Jak spustit program z IDE (C++Builder)	84
94	Jak spustit program z IDE (Visual Studio 2008)	84
95	Chceme vidět výsledky konzolové aplikace (C++Builder 2009)	84
96	Chceme vidět výsledky konzolové aplikace (Visual Studio 2008)	85
97	Jak krokovat program	85
98	Krokování s překročením podprogramu	85
99	Krokování se vstupem do podprogramu	87
100	Zjištění hodnoty proměnné	87

101	Zjištění hodnoty proměnné (C++Builder)	87
102	Zjištění hodnoty proměnné (Visual Studio 2008)	87
103	Sledované výrazy (C++Builder 2009)	87
104	Sledované výrazy (Visual Studio 2008)	88
105	Měníme hodnotu proměnné (C++Builder 2009)	88
106	Měníme hodnotu proměnné (Visual Studio 2009)	89
107	Krokování od zvoleného místa: doběhni ke kurzoru	89
108	Krokování od zvoleného místa: zarážka	89
109	Podmíněné zarážky (C++Builder 2009)	89
110	Podmíněné zarážky (Visual Studio 2008)	90
111	Jak ukončit krokování	90
112	Jak krokovat v assembleru	91
113	Posloupnost volání funkcí (C++Builder 2009)	92
114	Posloupnost volání funkcí (Visual Studio 2008)	92
115	Jak zjistit obsah registrů (C++Builder 2009)	93
116	Jak zjistit obsah registrů (Visual C++ 2008)	93
117	Jak zjistit obsah paměti (C++Builder 2009)	93
118	Jak zjistit obsah paměti (Visual C++ 2009)	93
119	Zakomentování vybraných řádků zdrojového kódu	94
120	Parametry programu zadávané v příkazovém řádku	94
121	Parametry programu při spouštění v IDE (C++Builder 2009)	94
122	Parametry programu při spouštění v IDE (Visual Studio 2008)	94
	Ladění pomocí programových konstrukcí	97
123	Potřebujeme dočasně odstranit několik řádků z programu	97
124	Potřebujeme odstranit větší počet řádků	97
125	Definice podmínkového jména pro ladění	97
126	Co je ladicí kód	98
127	Jednoduchý ladicí tisk (C)	98
128	Makro pro ladicí tisk	98
129	Podmíněné makro	98
130	Kam umístit direktivu #define, je-li makro v hlavičkovém souboru?	99
131	Ve které funkci jsme? (C99)	99
132	Ve kterém zdrojovém souboru jsme?	99
133	Na kterém řádku zdrojového souboru jsme?	100
134	Při ladění prověřujeme kontrakt	100
135	Test vstupních podmínek při ladění: aserce	100
136	Odstranění asercí z programu	101
137	Aserce, nebo výjimky?	101

138	Test výstupních podmínek	101
139	Je test dostatečný?	102
140	Test invariantů	102
141	Proč nerandomizovat	102
	Program, jeho běh a prostředí	103
142	Funkce main(), wmain() a jiné	103
143	Abychom nemuseli volit: _tmain()	103
144	Formální parametry funkce main() a wmain()	103
145	Formální parametry „funkce“ _tmain()	104
146	Tradiční jména parametrů funkce main()	104
147	Typ funkce main()	104
148	Omezení kladená na funkci main() v C++	104
149	Co je startovací kód	104
150	Parametry příkazové řádky programu	105
151	S kolika parametry v příkazové řádce byl náš program spuštěn?	105
152	Jak se jmenuje náš program?	105
153	Výpis všech parametrů programu	105
154	Jak na proměnné prostředí	105
155	Výpis všech proměnných prostředí (nestandardní řešení)	106
156	Zjištění hodnoty proměnné prostředí	106
157	Zjištění hodnoty proměnné prostředí v Unicode (nestandardní řešení)	107
158	Nastavení hodnoty proměnné prostředí (nestandardní řešení)	107
159	Spouštíme z programu jiný program	107
160	Kód ukončení programu	108
161	Předdefinované konstanty pro kód ukončení	108
162	Když program končí	108
163	Řádné ukončení programu	108
164	Spořádané ukončení programu jinde než ve funkci main()	109
165	Ukončení bez úklidu: _Exit() (C99)	109
166	Abnormální ukončení programu	109
167	Abnormální ukončení programu v C++	109
168	Měníme funkci, kterou volá terminate()	110
169	Vlastní úklid při ukončení programu: funkce registrované voláním atexit()	110
170	Pořadí volání funkcí registrovaných pomocí atexit()	110
171	Podařila se registrace funkce pomocí atexit()?	110
172	Kdy se inicializují globální proměnné	110
173	Jak zajistit včasnou inicializaci objektu?	111

	Deklarace a proměnné	113
174	Jaký je rozdíl mezi deklarací a definicí	113
175	Pravidlo jediné definice pro proměnné	113
176	Informativní deklarace proměnné	113
177	Definiční deklarace globální proměnné	113
178	Deklarace lokální automatické proměnné	114
179	Lokální proměnná, která si pamatuje hodnotu	114
180	Počítáme, kolikrát byla funkce volána	114
181	Jak se inicializuje lokální statická proměnná?	115
182	Co je to registrová proměnná?	115
183	K čemu je registrová proměnná?	115
184	Jak deklarovat ukazatel	115
185	Ukazatel na cokoli	116
186	Jak deklarovat konstantu	116
187	Konstanta v jazyce C	116
188	Konstanta v jazyce C++	116
189	Ukazatel na konstantu	117
190	Konstantní ukazatel	117
191	Konstantní ukazatel na konstantu	117
192	Inicializace proměnné v deklaraci	117
193	Nové jméno pro typ	118
194	Deklarace ukazatele pomocí jména zavedeného deklarací typedef	118
195	Deklarace jednorozměrného pole pojmenovaného typu	118
196	Deklarace vícerozměrného pole	119
197	Deklarace jednorozměrného pole ukazatelů	119
198	Deklarace ukazatele na jednorozměrné pole	119
199	Deklarace proměnné objektového typu s konstruktorem bez parametrů	119
200	Deklarace proměnné objektového typu s konstruktorem s parametry	120
201	Co je implicitní int	120
202	Proměnné, které se mohou měnit „z pozadí“	120
203	Označení typu	121
	Ukazatele a pole	123
204	Deklarace jednorozměrného pole s inicializací	123
205	Jaké prvky má pole?	123
206	Inicializace pole nulami	123
207	Čárka na konci seznamu inicializátorů	123
208	Známe inicializátory, neznáme počet prvků	124
209	Kolik prvků má pole?	124

210	Inicializujeme jen některé prvky (jen C99)	124
211	Pole s nekonstantním počtem prvků	124
212	Pole objektového typu	125
213	Pole objektového typu inicializované skalárními hodnotami	125
214	Pole objektového typu inicializované instancemi	125
215	Explicitní volání konstruktoru v inicializaci pole objektového typu	126
216	Pole s nekonstantním počtem prvků (jen C99)	126
217	Inicializace vícerozměrného pole	126
218	První index v deklaraci vícerozměrného pole můžeme někdy vynechat	126
219	Literál typu pole (jen C99)	127
220	Přístup k prvkům pole	127
221	Indexování je symetrické	127
222	Pole se skoro vždy konvertuje na ukazatel	127
223	Pole lze přiřadit ukazateli	128
224	Používáme proměnnou, na kterou ukazuje ukazatel	128
225	Ukazatel indexujeme jako pole	128
226	Jak vytvořit ukazatel nikam	129
227	Adresová aritmetika (aritmetika ukazatelů)	129
228	Co znamená rovnost nebo nerovnost ukazatelů?	129
229	Který prvek má vyšší index?	129
230	Která složka struktury je definována dříve?	129
231	Chceme s ukazatelem přejít na následující prvek pole	130
232	Chceme s ukazatelem přejít na předcházející prvek pole	130
233	Chceme s ukazatelem přejít o m prvků dál	130
234	O kolik se liší indexy daných prvků?	130
235	Překopírování obsahu jednorozměrného pole	131
236	Překopírování obsahu jednorozměrného pole pomocí adresové aritmetiky	131
237	Kopírování obsahu pole, když se zdrojové a cílové pole nepřekrývá	131
238	Kopírování obsahu pole, když se zdrojové a cílové pole překrývá	132
239	První nulový prvek v jednorozměrném poli pomocí adresové aritmetiky	132
240	První nulový prvek v jednorozměrném poli pomocí indexování	132
241	První záporný prvek ve dvourozměrném poli pomocí indexování	133
242	Výpis všech záporných prvků ve dvourozměrném poli pomocí adresové aritmetiky	133
243	Omezené (restringované) ukazatele v C99	134
	Správa paměti	135
244	Alokace jednoduché proměnné základního typu	135
245	Alokace jednorozměrného pole pomocí funkce malloc()	135
246	Alokace jednorozměrného pole s vynulováním	135

247	Když malloc() nebo calloc() neuspěje	135
248	Uvolnění paměti alokované pomocí malloc() nebo calloc()	136
249	Zvětšení dynamicky alokovaného pole	136
250	Zmenšení dynamicky alokovaného pole	136
251	Když funkci realloc() předáme NULL	137
252	Když realloc() neuspěje	137
253	Alokace jednoduché proměnné základního typu v C++	137
254	Alokace jednoduché proměnné základního typu s inicializací	137
255	Alokace instance objektového typu	137
256	Alokace instance objektového typu inicializovaná konstruktorem s parametry	138
257	Alokace instance objektového typu pomocí malloc()	138
258	Uvolnění paměti jednoduché proměnné základního typu alokované pomocí new	138
259	Uvolnění paměti instance objektového typu	139
260	Alokace hrubé paměti pomocí new	139
261	Uvolnění paměti alokované pomocí new bez volání destruktora	139
262	Když operátor new neuspěje	139
263	Když nechceme, aby operátor new vyvolával výjimku	140
264	Alokace instance objektového typu s přetíženým operátorem new	140
265	Překladač nenašel globální new. Co se děje?	140
266	Alokace instance objektového typu s přetíženým operátorem new pomocí globálního new	140
267	Zrušení instance objektového typu s přetíženým operátorem delete globálním delete	141
268	Alokace jednorozměrného pole základního typu pomocí new	141
269	Alokace jednorozměrného pole objektového typu pomocí new	141
270	Alokace pole objektového typu s přetíženým operátorem new	142
271	Překladač nenašel při alokaci pole globální new. Co se děje?	142
272	Alokace pole objektového typu s přetíženým operátorem new[] globálním new	142
273	Zrušení pole objektového typu s přetíženým operátorem delete[] globálním delete[]	143
274	Vyvolá-li konstruktor alokované instance výjimku	143
275	Nemíchejme malloc() a new	143
276	Alokace dvourozměrného pole v jazyce C	143
277	Alokace dvourozměrného pole neobjektového typu pomocí new	144
278	Uvolnění pole alokovaného pomocí new	144
279	Zapomeneme-li hranaté závorky za delete	144
	Reference	145
280	Co je to reference?	145
281	Deklarace reference	145
282	Jak jsou reference implementovány?	145
283	Hlavní omezení kladená na reference	145

284	Inicializace nekonstantní reference neobjektového typu	145
285	Inicializace reference objektového typu	146
286	Lze změnit odkazovanou proměnnou?	146
287	Měníme hodnotu proměnné prostřednictvím reference	146
288	Získáváme adresu odkazované proměnné	146
289	Zjednodušení zápisu pomocí reference	147
290	Chceme referenci inicializovat výrazem	147
291	Reference na pole	147
292	Reference na funkci	147
293	Voláme odkazovanou funkci	148
294	Reference jako složka objektového typu	148
	Práce se znaky	149
295	Znakové typy: v čem se liší C90, C99 a C++	149
296	Jakého typu je znakový literál	149
297	Literál představující úzký netisknutelný znak	149
298	Přehled řídicích posloupností	149
299	Literál představující široký znak	150
300	Co je to Unicode	151
301	Co jsou vícebajtové znaky	151
302	Převod širokého znaku na úzký	151
303	Převod úzkého znaku na široký	152
304	Lze široký znak reprezentovat jednobajtovým znakem?	152
305	Máme znak. Je to číslice? (C)	152
306	Máme znak. Je to číslice? (C++)	153
307	Máme úzký znak. Je to písmeno? (C)	153
308	Máme široký znak. Je to písmeno? (C)	154
309	Máme znak. Je to písmeno? (C++)	154
310	Jde o malé písmeno? (C)	154
311	Jde o malé písmeno? (C++)	155
312	Klasifikace úzkých znaků – přehled	155
313	Klasifikace znaků při jiném lokálním nastavení (C++)	156
314	Převod znaků na malá písmena	156
315	Převod znaků na malá písmena při jiném lokálním nastavení (C++)	156
316	Převod znaků na velká písmena	157
317	Převod znaků na velká písmena při jiném lokálním nastavení (C++)	157
318	Převod mezi malými a velkými písmeny nemusí být jednoznačný	158
319	Co jsou to trigrafy	158

	Znakové řetězce (pole znaků)	159
320	Hlavičkové soubory pro široké znaky	159
321	Řetězcová konstanta	159
322	Co je to řetězcová konstanta?	159
323	Inicializace znakového pole řetězcem	159
324	Když vynecháme počet prvků	160
325	Spojování řetězcových konstant (všechny verze C a C++)	160
326	Spojování řetězcových konstant (od C90 dále)	160
327	Spojování řetězcových konstant vytvořených makry	160
328	Speciální znaky v řetězcové konstantě	161
329	Jak deklarovat ukazatel na řetězec	161
330	Pozor na neinicializované ukazatele	161
331	Inicializace pole řetězců	162
332	Jak zjistit délku řetězce	162
333	Jak přistupovat ke znakům řetězce	162
334	Kopírování řetězců	162
335	Kopírujeme nejvýše n znaků	163
336	Kopírujeme pole znaků, která se překrývají	163
337	Spojování řetězců	163
338	První výskyt znaku v řetězci	164
339	První výskyt znaku ze zadané množiny	164
340	První výskyt podřetězce	165
341	Rozklad řetězce na slova	165
342	Kolik musíme projít znaků, než narazíme na zadaný znak?	166
343	Lexikografické porovnávání řetězců	166
344	Abeceďní porovnávání řetězců podle lokálních zvyklostí	166
345	Čtení celého čísla z řetězce úzkých znaků	167
346	Čtení celého čísla z řetězce širokých znaků	167
347	Čtení reálného čísla z řetězce	167
348	Převod řetězce na celočíselnou hodnotu	168
349	Převod řetězce na číselnou hodnotu jiného typu než int	168
350	Chceme vědět, kde převod na int skončil	168
351	Chceme vědět, kde skončil převod na jiný celočíselný typ	168
352	Chceme vědět, kde skončil převod na reálný typ	169
353	Převod řetězce širokých znaků na číslo	169
354	Převod z jiných číselných soustav	169
355	Převedení čísla na řetězec úzkých znaků	170
356	Převedení čísla na řetězec širokých znaků	170

357	Převod celého čísla na řetězec – nestandardní řešení	170
358	Další nestandardní funkce pro převod čísel na řetězec	171
359	Zápis celého čísla v zadané soustavě (nestandardní řešení)	171
360	Caesarova šifra (jednoduchá transformace řetězce)	171
361	Převod úzkého řetězce na široký	172
362	Převod širokého řetězce na úzký	172
363	Převod znaků v řetězci na velká písmena	173
364	Převod znaků v řetězci na malá písmena	173
365	Transformace řetězce na místě	174
	Znakové řetězce (typ <code>basic_string<></code>)	175
366	Třídy <code>std::string</code> a <code>std::wstring</code> a dokumentace k nim (C++)	175
367	Třída <code>string</code> je kontejner ze STL	175
368	Zápis řetězce do proudu	175
369	Prázdná instance třídy <code>string</code>	176
370	Inicializace instance typu <code>string</code> polem znaků	176
371	Kolik znaků řetězec pojme?	176
372	Zvětšujeme kapacitu řetězce	176
373	Inicializace instance typu <code>string</code> opakovaným znakem	176
374	Převod instance typu <code>string</code> na pole typu <code>char</code>	177
375	Inicializace pole řetězců typu <code>string</code>	177
376	Délka řetězce typu <code>string</code>	177
377	Je řetězec prázdný?	177
378	Přístup ke znakům řetězce	178
379	Nový obsah instance typu <code>string</code>	178
380	Nový obsah instance typu <code>string</code> : opakovaný znak	178
381	Spojování řetězců typu <code>string</code> nebo <code>wstring</code>	178
382	Spojujeme řetězce, aniž bychom je měnili	179
383	Zkracujeme řetězec	179
384	Prodlužujeme řetězec	179
385	Kopírujeme řetězec do pole	180
386	První výskyt znaku v řetězci	180
387	První výskyt podřetězce	180
388	První výskyt znaku z dané množiny	181
389	První výskyt znaku, který nepatří do dané množiny	181
390	Poslední výskyt znaku v řetězci	181
391	Poslední výskyt znaku, který nepatří do dané množiny	182
392	Bylo hledání úspěšné?	182
393	Vložení nového znaku nebo podřetězce do řetězce	182

394	Připojení znaku na konec řetězce	183
395	Jak získat podřetězec	183
396	Převod typu string na wstring	183
397	Převod typu wstring na string	184
398	Převod řetězce na velká nebo pouze malá písmena	184
399	Kopie řetězce obsahující pouze velká nebo pouze malá písmena	185
400	Lexikografické porovnávání řetězců	185
401	Abecední porovnání řetězců	185
402	Prohození obsahu dvou řetězců	186
	Funkce a ukazatele na ně	187
403	Co je to funkce?	187
404	Parametry funkce	187
405	Co může funkce vrátet	187
406	Co je kontrakt funkce	187
407	Příklad kontraktu	188
408	Princip jediné odpovědnosti	188
409	Příklad příliš široké zodpovědnosti: funkce realloc()	188
410	Definice a deklarace	188
411	Prototyp funkce bez parametrů – C vs. C++	189
412	Statické a externí funkce	189
413	Vložené funkce (inline)	189
414	Vložené funkce nejsou statické (C++)	190
415	Vložené funkce v jazyce C	190
416	Pravidlo jediné definice	190
417	Předávání parametrů funkcím hodnotou	190
418	Chceme změnit parametr: předávání ukazatele	190
419	Chceme změnit parametr: předávání odkazem (jen C++)	191
420	Ukazatelem, nebo odkazem?	191
421	Předávání pole jako parametru funkce	191
422	Funkce svůj parametr nemění: ukazatel na konstantu	192
423	Konstantní reference jako parametr	192
424	Ukazatel na funkci	193
425	Konstanta typu ukazatele na funkci	193
426	Volání funkce zadané ukazatelem	193
427	Funkce jako parametr funkce	194
428	Funkce vracující ukazatel	194
429	Funkce vracující referenci (pouze C++)	194
430	Na co si dát pozor při deklaraci referenční funkce	194

431	Funkce vracející řetězec	195
432	Co by funkce neměla vracet	195
433	Když má funkce příliš mnoho parametrů	195
434	Když překladáč nezná typy parametrů	196
435	Funkce s proměnným počtem parametrů: výpustka	196
436	Jak se předávají parametry na místě výpustky	196
437	Jak pracujeme s parametry předanými výpustkou	197
438	Příklad funkce s výpustkou	197
439	Nepoužitý parametr (jen C++)	197
440	Přetěžování funkcí (jen C++)	198
441	Kdy přetěžovat funkce?	198
442	Implicitní hodnoty parametrů	199
443	Volání funkce s implicitními hodnotami parametrů	199
444	Pravidla pro implicitní hodnoty parametrů	199
445	Funkce podle Kernighana a Ritchieho	200
446	Příklad funkce podle Kernighana a Ritchieho	200
447	Co je implicitní int	200
448	Funkce z jazyka C v C++	200
449	Funkce v C++ se jménem podle pravidel jazyka C	201
450	Zacházení se jmény funkcí v C a v C++: pohled pod pokličku	201
451	Příklad zdobení jmen v C++	201
452	Volací konvence (není součástí standardu)	202
453	Standardní volací konvence jazyka C	202
454	Další volací konvence	202
455	Funkce zná své jméno (jen C99)	203
	Preprocesor a makra	205
456	Podmíněný překlad	205
457	#elif místo #else #if	205
458	Co může být v podmínce v direktivě #if?	205
459	Test, zda je definováno podmínkové jméno	206
460	Zkratky pro #if defined	206
461	Kód překládaný jen v C++	206
462	Zdrojový soubor, který vyžaduje C++	207
463	Varování: direktiva #warn (C99)	207
464	Platnost podmínkového jména	207
465	Zrušení podmínkového jména	207
466	Direktiva pro ozdobu: #	208
467	Manifestační konstanta (makro bez parametrů)	208

468	Proč používat manifestační konstanty	208
469	Proč nepoužívat manifestační konstanty (v C++)	208
470	Jak definovat makro s parametry	208
471	Když se definice makra nevejde na jeden řádek	209
472	Makro jako náhrada funkce	209
473	Proč používat makra	209
474	Proč nepoužívat makra, ale funkce inline	210
475	Makra je třeba pečlivě závorkovat	210
476	Parametr makra by se měl vyhodnocovat jen jednou	210
477	Vytvoření řetězce v makru	211
478	Makro by nemělo končit středníkem	211
479	Spojování symbolů v makru	211
480	Makra s proměnným počtem parametrů (jen C99)	212
481	Jak zjistit verzi Unicode (C99)	212
482	Datum a čas překladu	213
483	Konvence pro pojmenování maker	213
	Celá čísla	215
484	Co všechno je celé číslo	215
485	Celočíselné typy v C90 a v C++	215
486	Alternativní jména celočíselných typů	215
487	Základní celočíselné typy v C99	215
488	Osmibajtová celá čísla v C90 a v C++	216
489	Zobrazení celých čísel bez znaménka v paměti	216
490	Zobrazení celého čísla se znaménkem v paměti	216
491	Malý nebo velký endián	216
492	Rozsahy celočíselných typů	217
493	Minimální rozsah celočíselných typů	217
494	Je jeden celočíselný typ int a ty ostatní jsou od něj odvozeny...	217
495	Různé pohledy na celá čísla (C99)	218
496	Celočíselné typy s přesně danou šířkou (C99)	218
497	Celočíselné typy, které mají alespoň udanou šířku (C99)	218
498	Nejrychlejší celočíselné typy, které mají alespoň udanou šířku (C99)	218
499	Celočíselné typy, do nichž lze uložit ukazatel (C99)	219
500	Celočíselné typy s největší možnou šířkou (C99)	219
501	Jak zapsat celočíselný literál	219
502	Přípony celočíselných literálů	219
503	Jakého typu je desítková celočíselná konstanta?	220

504	Jakého typu je osmičková nebo šestnáctková celočíselná konstanta?	220
505	Konstanta typu short nebo unsigned short	221
506	Meze základních celočíselných typů v C	221
507	Meze dalších celočíselných typů	221
508	Meze celočíselných typů v C++	222
509	Co je to celočíselné přetečení	222
510	Absolutní hodnota celého čísla	222
511	Je číslo liché?	223
512	Je k-tý bit proměnné nastaven?	223
513	Nastavení k-tého bitu proměnné	223
514	Počet nastavených bitů v proměnné	223
515	Jak bezpečně sečíst dvě nezáporná čísla	224
516	Nemíchejme typy se znaménkem a bez něj	224
	Reálná čísla	225
517	Datové typy pro reálná čísla	225
518	Reprezentace a aritmetika reálných čísel	225
519	Vyhovuje daný datový typ standardu IEEE 754?	225
520	Reálný literál	225
521	Typ reálného literálu	226
522	Zjišťujeme rozsah reálných typů	226
523	Zjišťujeme rozsah reálných typů v C++	226
524	Kolik platných desítkových cifer má daný typ?	226
525	Kolik bitů mantisy má daný typ?	227
526	Jaký je rozsah exponentu daného typu?	227
527	Když k číslu a přičtu b, změní se číslo a?	227
528	Jaké nejmenší číslo mohu přičíst k 1, abych dostal číslo jiné než 1?	228
529	Jaké nejmenší číslo mohu přičíst k x, abych dostal číslo jiné než x?	228
530	Když sečtu deset desetin, dostanu 1?	228
531	Komutativní a asociativní zákon nemusí platit	228
532	Porovnávání reálných čísel	229
533	Rozumnější porovnání	229
534	Odložená volba typu reálných čísel	230
535	Známe strany a počítáme úhly v pravouhlém trojúhelníku	230
536	Když známe obě odvěsny, je lepší atan2()	230
537	Jak pracovat s logaritmy	230
538	Logaritmus o libovolném základu	231
539	Mocnina a exponenciální funkce	231

540	Goniometrické a hyperbolické funkce pro typ double	231
541	Zaokrouhlování a absolutní hodnota pro typ double	231
542	Běžné matematické funkce pro ostatní reálné typy	232
543	Co je to strojové nekonečno	232
544	Obsahuje tato implementace strojové nekonečno?	232
545	Jak získat strojové nekonečno	233
546	Když je výsledek matematické funkce nekonečný	233
547	Co jsou to denormální čísla	233
548	Podporuje tento typ denormální čísla? (C++)	234
549	Je daný výsledek denormální, tedy méně přesný?	234
550	Co je to NaN	234
551	NaN mohou vracet i naše vlastní funkce	234
552	Je to NaN?	234
553	Klasifikace reálných čísel (C99)	235
	Příkazy	237
554	Potřebujeme několik příkazů na místě, kde smí být jeden	237
555	Nechceme žádný příkaz na místě, kde má nějaký být	237
556	Deklarace a příkazy v bloku v C90 a starších	237
557	Deklarace a příkazy v bloku v C99 a v C++	237
558	Výraz jako příkaz	238
559	Deklarace je v C++ příkaz	238
560	Chceme provést akci, jen když je splněna podmínka	238
561	Rozhodování mezi dvěma alternativami (příkaz if)	239
562	Příkaz vnořený do if tvoří vždy blok	239
563	Ke kterému if patří to else?	239
564	V podmínce můžeme deklarovat proměnnou (jen C++)	240
565	Větvení programu podle většího počtu hodnot (příkaz switch)	240
566	Rozdělení příkazu switch na alternativy (příkaz break)	240
567	Příkaz switch nebývá přehledný	241
568	Náhrada příkazu switch polem	241
569	Náhrada příkazu switch polymorfizmem	242
570	Opakujeme skupinu příkazů: cyklus	242
571	Tělo cyklu je blok	242
572	Testujeme podmínku před vstupem do těla cyklu	242
573	Zjišťujeme počet nenulových prvků pole (cyklus while)	243
574	V podmínce příkazu while můžeme deklarovat proměnnou (jen C++)	243
575	Čteme znaky z klávesnice až po zadaný znak (cyklus do-while)	243
576	Jak funguje příkaz for	244

577	V inicializaci příkazu for můžeme deklarovat proměnnou (C99 a C++)	244
578	Řetězec pozpátku (deklarace několika proměnných v inicializaci)	245
579	Co můžeme použít jako podmínku	245
580	Nekonečný cyklus	245
581	Cyklus s podmínkou uprostřed (jak předčasně opustit cyklus)	246
582	Nedokončený průchod cyklem (příkaz continue)	246
583	Příkaz skoku (goto)	246
584	Proč nepoužívat příkaz goto	247
585	Tolerované použití příkazu goto	247
586	Návrat z funkce (příkaz return)	247
587	Příkaz asm (jen C++)	247
588	Příkaz asm ve Visual C++ a v C++-Builderu	248
	Výrazy a operátory	249
589	Každý operátor má svou prioritu	249
590	Jaké existují úrovně priority	249
591	Operátor má také asociativitu	249
592	Neváhejme používat závorky	250
593	Priorita a asociativita neurčuje pořadí vyhodnocování operandů	250
594	Kdy je pořadí vyhodnocování pevně dáno	250
595	Když použijeme operátor ++ nebo -- dvakrát na stejnou proměnnou	251
596	Co jsou to sekvenční body	251
597	Přiřazení je výraz	251
598	Složené přiřazovací operátory	252
599	Přiřazení vs. porovnání	252
600	Přístup k zastíněné globální proměnné (C++)	253
601	Zjišťujeme velikost proměnné nebo typu	253
602	Operátor sizeof vyhodnocuje zpravidla překladač	253
603	Volání funkce	254
604	Nejdřív si připravíme data, pak vyhodnotíme podmínku (operátor čárka)	254
605	Unární aritmetické operátory a typ výsledku	255
606	Binární aritmetické operátory a typ výsledku v C++ a v C90	255
607	Binární aritmetické operátory a typ výsledku v C99	256
608	Jak se určí typ aritmetického výrazu	256
609	Výsledky nás mohou překvapit	256
610	Někdy neplatí asociativní ani komutativní zákon	257
611	Jak na přetypování v jazyce C	257
612	Jak na přetypování v C++	257

	Výčtové typy	259
613	Výčtový typ pro dny v týdnu – nejjednodušší deklarace	259
614	Deklarace proměnné výčtového typu	259
615	Proměnnou lze deklarovat v deklaraci typu	259
616	Když nechceme v jazyce C psát enum	259
617	Co jsou to výčtové konstanty	260
618	Když nám implicitně přidělené hodnoty konstant nevyhovují	260
619	Výčtovou konstantu můžeme definovat pomocí předchozí konstanty	260
620	Výčtový typ pro příznaky	260
621	Jména výčtových typů	261
622	Čárka na konci seznamu konstant	261
623	Výčtový typ v programu	261
624	Výčtový typ v C a v C++: Hlavní rozdíl	262
625	Jaké hodnoty může obsahovat proměnná výčtového typu	262
626	Anonymní výčtový typ	262
627	Výčtový typ jako náhrada konstanty	263
	Struktury a unie	265
628	Skupina proměnných jako jeden celek (struktura)	265
629	Proměnné sdílejí místo v paměti	265
630	Deklarace struktury	265
631	Deklarace unie	266
632	Jméno struktury nebo unie v C a v C++	266
633	Když nechceme v jazyce C psát struct nebo union	266
634	Když použijeme typedef, můžeme jmenovku vynechat	267
635	V deklaraci typu můžeme deklarovat proměnnou	267
636	Když dvě struktury odkazují na sebe navzájem	267
637	Přístup ke složkám struktur a unii	268
638	Ukazatel na strukturu nebo unii	268
639	Inicializace struktury	268
640	Inicializace unie	268
641	Inicializace struktury v C99	269
642	Inicializace unie v C99	269
643	Literál typu struktura nebo unie (jen C99)	269
644	Uložení struktury v paměti, velikost struktury	270
645	Jak zjistit velikost unie	270
646	Práce s jednotlivými bity (bitová pole)	270
647	Uložení bitových polí v paměti	271

648	K čemu je bitové pole	271
649	Bitové pole nemusí mít jméno	271
650	Bitové pole nulové šířky	271
651	Co nelze s bitovými poli dělat	271
652	Proměnné, které sdílejí místo v paměti (anonymní unie, jen C++)	272
	Objektový typ a jeho deklarace	273
653	Co je to třída	273
654	Obecné a speciální pojmy a jejich model v programu	273
655	Objektově orientovaný program	273
656	S daty pracujeme zásadně pomocí metod	274
657	Deklarace třídy bez předků	274
658	V deklaraci typu lze deklarovat instance	274
659	Příklad: třída cplx	275
660	Vytvoření instance	275
661	Použití složek	276
662	Jak vyjádřit délku pole, které je složkou třídy	276
663	Právo na přístup ke složkám	277
664	Význam specifikací přístupu	277
665	Výjimka z přístupových práv: přátelé	277
666	Co je to rozhraní třídy	278
667	Přístupové metody a jejich jména	278
668	K čemu jsou dobré přístupové metody	278
669	Datové složky instancí	279
670	Metody instancí	279
671	Jak metoda ví, se kterou instancí pracuje?	279
672	Metody, které lze použít pro konstantní instance	280
673	Datové složky třídy	280
674	Deklarace datové složky třídy jako celku (statické datové složky)	281
675	Použití statické datové složky mimo její třídu	281
676	Statické konstantní složky	282
677	Význačné hodnoty jako statické datové složky	282
678	Metody třídy jako celku (statické metody)	282
679	Statické metody nemají this	283
680	Volání statické metody	283
681	Také konstruktor je metoda třídy jako celku	283
682	Metoda definovaná v těle je inline	284
683	V metodě používáme složky i metody těžce instance bez kvalifikace	284

684	Ve statické metodě nelze používat datové složky instancí bez kvalifikace	284
685	Typ jako složka třídy	284
686	Vnořené a obklopující třídy jsou si cizí	285
687	Třída je obor viditelnosti	285
688	Vnořená třída je v oboru obklopující třídy	285
689	Šablona jako složka třídy	286
690	Některé metody vytvoří překladač sám	286
691	Předběžná deklarace třídy	286
692	Struktura jako objektový typ	286
693	Unie jako objektový typ	287
694	Třída definovaná v těle metody nebo funkce (lokální třída)	287
695	Velikost instance	287
696	Měnitelné složky konstant	288
697	Deklarovaná, ale nedefinovaná metoda	288
	Vytvoření instance	289
698	Deklarace konstruktoru	289
699	Omezení kladená na konstruktor	289
700	Lze instanci inicializovat jako neobjektovou strukturu?	289
701	Inicializační část konstruktoru	290
702	Inicializační část konstruktoru není povinná	290
703	V jakém pořadí inicializace probíhají	290
704	Volání konstruktoru složky	291
705	Konstantní a referenční složky	291
706	Deklarace instance znamená volání konstruktoru	291
707	Konstruktor nelze volat jako ostatní metody	292
708	Překladač si může vytvořit konstruktor sám	292
709	Kdy si překladač konstruktor nevytvoří	292
710	Konstruktor bez parametrů, konstruktor s jedním parametrem	293
711	Automatické konverze pomocí konstruktoru	293
712	Konverzní konstruktor	293
713	Explicitní konverze pomocí konstruktoru	293
714	Když nechceme, aby konstruktor sloužil k implicitním konverzím	294
715	Parametrem konstruktoru je délka pole	294
716	Kopírovací konstruktor	294
717	Explicitní vytvoření instance	295
718	Optimalizace vrácené hodnoty	295
719	Konstruktor nemůže volat jiný konstruktor	295

720	Dva konstruktory sdílí část kódu: jak se neopakovat	296
721	V několika metodách se opakují stejné úvodní a závěrečné operace (hlídka)	296
722	Dynamická instance	297
723	Explicitní volání konstruktora pro dané místo	297
724	Jak zakázat vytváření instancí všem	297
725	Jak zakázat vytváření instancí klientskému programátorovi	298
726	Co je to tovární metoda	298
727	Konstruktor a výjimky	298
728	„Konstruktory“ základních typů (pseudokonstruktory)	298
	Kopírování instancí	299
729	Mělká a hluboká kopie	299
730	Instance třídy obsahující dynamické pole	299
731	Vytvoření nové instance, která je kopií existující instance	299
732	Volání kopírovacího konstruktora	299
733	Inicializace instance výrazem jiného typu	300
734	Když do hry s konstruktory vstoupí přístupová práva	300
735	Překladač může optimalizovat	301
736	Implicitní kopírovací konstruktor	301
737	Kdy překladač nevytvoří kopírovací konstruktor	301
738	Kopírovací přiřazovací operátor	301
739	Typický problém, když instance obsahuje ukazatel na pole	302
740	Implicitní přiřazovací operátor	303
741	Kdy překladač nevytvoří přiřazovací operátor	303
742	Přiřazovací operátor by měl vracet *this	303
743	Třída vektor: příklad deklarace přiřazovacího operátoru	303
744	Pole proměnné délky, chybná implementace	304
745	Rovnost při přiřazování	304
746	Ochrana proti přiřazení sebe sobě	305
747	Jaké problémy mohou nastat, když kopírování hodnot může vyvolat výjimku	305
748	Jak zvládnout možné výjimky při kopírování dat	306
749	Příklad transakčního kopírování: vektor	306
750	Jak zabránit kopírování	306
751	Předávání parametrů objektových typů hodnotou	307
752	Vracení výsledku funkce	307
	Zánik instance	309
753	Co je to destruktork	309

754	Deklarace destruktora	309
755	Implicitní destruktora	309
756	Kdy překladač nedokáže implicitně definovat destruktora	309
757	Implicitní volání destruktora	310
758	Explicitní volání destruktora	310
759	Destruktor může být virtuální	310
760	Omezení kladená na destruktora	310
761	Proč by se z destruktora neměly šířit výjimky	310
762	Jak dlouho žije lokální automatická instance	311
763	Jak dlouho žije globální instance	311
764	Jak dlouho žije lokální statická instance	311
765	Jak dlouho žije dynamická instance	311
766	Jak dlouho žijí pomocné instance	311
767	Jak dlouho žijí pomocné instance, které slouží k inicializaci referencí	312
768	Jak dlouho žije pomocná proměnná vytvořená v inicializační části konstruktora	312
769	V jakém pořadí se volají destruktory složek	313
770	„Destruktory“ vestavěných typů (pseudodestruktory)	313
771	Je destruktora volán v době hledání obsluhy výjimky?	313
	Dědění a polymorfismus	315
772	Dědění neboli specializace	315
773	Potomek může zastoupit předka	315
774	Instance má několik typů	315
775	Vícenásobné dědění	315
776	Deklarace třídy s předky	316
777	Kdo může být předkem	316
778	Které metody se nedědí	316
779	Volání konstruktora předka	317
780	Specifikátor přístupu ve specifikaci předků	317
781	Třída je obor viditelnosti	318
782	Zastínění zděděné metody	318
783	Jak se dovoláme zastíněné metody	318
784	Jak zabráníme zastínění metody předka	319
785	Zveřejnění chráněné metody předka	319
786	Přiřazení potomka předkovi („ořezání dat“)	320
787	Přiřazení ukazatele na potomka ukazateli na předka	320
788	Časná a pozdní vazba	321
789	Jak deklarovat metodu s pozdní vazbou	321

790	Jak překrýt virtuální metodu předka	321
791	Kdy se uplatňuje pozdní vazba	322
792	Volání virtuální metody předka	322
793	Polymorfni třída	322
794	Potřebujeme, aby třída byla polymorfni	323
795	Kreslení grafického objektu: abstraktní metoda, abstraktní třída	323
796	Deklarace abstraktní metody	323
797	Omezení kladená na abstraktní třídy	324
798	Čistě virtuální funkce může mít definici	324
799	Potřebujeme, aby třída byla abstraktní	324
800	Různé druhy metod ve společném předkovi	325
801	Přetypování ukazatele na potomka na ukazatel na předka	325
802	Konflikty jmen v odvozených třídách	326
803	Pozdní vazba: pohled pod pokličku	326
804	Volání virtuálních metod z konstruktorů a destruktorů	327
805	Co se děje při konstrukci a destrukci instance polymorfni třídy	328
806	Virtuální dědění	328
807	Volání konstruktoru virtuálního předka	329
808	Pořadí volání konstruktorů předků při virtuálním dědění	330
809	Přetypování na potomka	330
	Objektový návrh	331
810	Třídy a jejich vztahy	331
811	Co jsou to návrhové vzory	331
812	Proč využívat skládání	331
813	Co je to rozhraní	331
814	Rozhraní v C++	332
815	Úsečka jako potomek bodu?	332
816	Technické dědění	332
817	Je dědění v tomto případě vhodné?	333
818	Třídy vyhovují testu JE – MÁ. Stačí to?	333
819	Substituční princip Barbary Liskovové	333
820	Přetížená třída	334
821	Princip jediné zodpovědnosti	334
822	Příliš mnoho tříd	334
823	Když nechceme zveřejnit konstruktory (tovární metoda)	335
824	Když nechceme, aby od naší třídy bylo možno odvodit potomka	335
825	Chceme jedinou instanci	336

826	Zásobník jako potomek seznamu?	336
827	Jak použít adaptér	336
828	Oddělujeme rozhraní od implementace	337
829	Implementace stromu: schematický příklad mostu	337
830	Nechceme být vázáni na jedinou implementaci	338
831	Kontejner a alokace paměti	338
832	Oddělujeme kontejner od algoritmu	339
	Přetěžování operátorů	341
833	Co je to přetěžování operátorů	341
834	Které operátory nelze přetěžovat	341
835	Operátory, které lze přetěžovat jen jako metody	341
836	Operátory new a delete a ostatní operátory	341
837	Přetěžování operátorů a srozumitelnost programu	341
838	Přetěžování operátorů: základní vodítka	342
839	Jak přetěžujeme binární operátory	342
840	Jak přetěžujeme unární operátory	343
841	Když překladač uvidí operátor	343
842	Použití operátoru	343
843	Třída cplx pro příklady	343
844	Sčítání komplexních čísel	344
845	Operátor jako metoda: Operandy nejsou rovnocenné	344
846	Chceme, aby operandy byly rovnocenné	344
847	Sčítání komplexních čísel bez deklarace friend	345
848	Když je rychlost důležitá, definujeme další verzi operátoru	345
849	Operátory ++ a --	346
850	Co je „prefixovost“ a „postfixovost“ operátorů ++ a --	346
851	Prefixový operátor ++ pro výčtový typ	347
852	Postfixový operátor ++ pro výčtový typ	347
853	Unární minus pro komplexní čísla	347
854	Unární plus pro komplexní čísla	347
855	Chovají se přetížené operátory stejně jako standardní operátory?	348
856	Přiřazovací operátor nemusí jen kopírovat	348
857	Jak je to s operátory @=	349
858	Někdy je lepší definovat nejprve @= a teprve pak operátor @	349
859	Chceme indexovat instanci	350
860	Přístup ke složkám komplexního čísla pomocí indexů	350
861	Přístup ke složkám komplexní konstanty pomocí indexů	350

862	Operátor volání funkce	351
863	Přístup k prvkům matice: volání funkce místo indexování	351
864	Konverzní operátor (konverzní funkce)	352
865	Konverze komplexního čísla na reálné	352
866	Operátor ->	352
867	Chytrý ukazatel	353
868	Jak přetížít operátor new pro alokaci jedné instance	353
869	Jak přetížít operátor new pro alokaci pole	353
870	Globální new lze nahradit	354
871	Jak přetížít operátor delete pro uvolnění jedné instance	354
872	Jak přetížít operátor delete pro uvolnění pole instancí	355
873	Globální delete lze nahradit	355
874	Rozdíl mezi použitím operátoru a voláním operátorové funkce	356
875	Alokace instance	356
876	Alokace pole	356
877	Použití třídního new a delete	357
878	Použití globálního new a delete	357
879	Když opravdu chceme nahradit globální new	357
880	Náhrada globálního delete	358
881	Operátor new s dodatečnými parametry	358
882	Operátor delete s dodatečnými parametry	359
	Zpracování chybových stavů	361
883	Dlouhý skok v jazyce C	361
884	Proměnná pro záznam stavu prostředí v místě návratu	361
885	Místo, kam se chceme vrátit	361
886	Vlastní dlouhý skok	362
887	Co dlouhý skok umí a co ne	362
888	Návratové hodnoty indikující chybu	362
889	Jak používat proměnnou errno	363
890	Samotná kontrola errno nestačí	363
891	Co se vlastně stalo	363
892	Co je to výjimka v C++	363
893	Pouze synchronní výjimky	364
894	Třídy pro přenos informací o výjimce	364
895	Terminologické zmatky	365
896	Logické a běhové chyby	365
897	Další třídy výjimek	365

898	Vyvolání výjimky	366
899	Šíření výjimky	366
900	Úklid zásobníku	366
901	Zachycení a ošetření výjimky	367
902	Rozdělení kódu na normální běh a obsluhu chyb	367
903	Pořadí obsluh	368
904	Chceme zachytit všechny výjimky	368
905	Obsluha se hledá podle typu, ale bez konverzí	368
906	Bližší informace o výjimce	369
907	Částečné ošetření výjimky	369
908	Specifikace výjimek v hlavičce funkce	369
909	Když se z funkce rozšíří neočekávaná výjimka	370
910	Funkce, kterou volá unexpected()	370
911	Když je celé tělo funkce jeden příkaz try	371
912	Inicializační část konstruktoru a výjimky	371
913	Výjimky, konstruktory a destruktory	371
914	Specifikace výjimek a šablony	372
915	Výjimka v konstruktoru složky	372
916	Výjimka při alokaci dynamické instance	372
917	Úniky paměti při výjimkách	373
918	Automatické ukazatele	373
919	Automatický ukazatel a výjimky	374
	Jmenné prostory	375
920	Deklarace jmenného prostoru	375
921	Knihovny umísťujeme do jmenných prostorů	375
922	Deklaraci jmenného prostoru můžeme rozdělit	375
923	Přístup k součástem jmenných prostorů	375
924	Uvnitř jmenného prostoru nemusíme kvalifikaci používat	376
925	Jak se dovoláme proměnné, která neleží v prostoru jmen	376
926	Uvnitř jmenného prostoru stačí deklarace	377
927	Jiné jméno jmenného prostoru	377
928	Když nechceme psát zdlouhavé kvalifikace (deklarace using)	377
929	Zpřístupňujeme celý jmenný prostor	378
930	Direktiva using je tranzitivní	378
931	Také jména zpřístupněná deklarací using se direktivou using „pošlou dál“	379
932	Anonymní jmenné prostory	380
933	Skrýváme jméno a nechceme použít klíčové slovo static	380
934	Jak překladač hledá volanou funkci nebo operátor	380

935	Princip rozhraní	381
	Šablony	383
936	Šablony jako lepší makra	383
937	Definice šablony volné funkce	383
938	Deklarace šablony volné funkce	383
939	Formální parametry šablony	384
940	Šablonové parametry šablony funkce	384
941	Skutečné parametry šablony	384
942	Implicitní vytvoření instance šablony funkce	385
943	Šablona nenahrazuje prototyp	385
944	Jak zavolat šablonovou funkci s parametry jiného typu	386
945	Explicitní vytvoření instance šablony funkce	386
946	Přetěžování šablon funkcí	386
947	Specializace šablony volné funkce	387
948	Ukazatel na instanci šablony	387
949	Šablona objektového typu	387
950	Použití šablonových parametrů šablony objektového typu	388
951	Instance šablony objektového typu	388
952	Implicitní hodnoty parametrů šablony	388
953	Jeden parametr lze použít jako implicitní hodnotu dalšího	389
954	Instance šablony jako parametr jiné šablony	389
955	Šablony metod	390
956	Šablona konstruktoru	390
957	Statická datová složka	391
958	Které součásti se vytvoří, když se vytvoří instance šablony třídy	391
959	Slučování instancí	391
960	Šablona jako složka třídy	392
961	Šablona jako složka šablony třídy	392
962	Co se podle vnořené šablony nevytvoří	393
963	Dvojití čtení	393
964	Jak napovědět, že jde o vnořený typ	393
965	Jak napovědět, že jde o šablonu	393
966	Různé instance šablony objektového typu jsou různé třídy	394
967	Sprátenelé funkce a šablony	394
968	Šablony jako přátelé: příklad s lomenými závorkami	395
969	Šablony jako přátelé: příklad (kvalifikovaná jména)	395
970	Zvláštní případy šablony objektového typu	396
971	Příklad primární šablony: obecná dvojice	396

972	Příklad částečné specializace: dvojice ukazatelů	397
973	Příklad explicitní specializace: dvojice ukazatelů typu void*	398
974	Různé počáteční hodnoty statických složek pro různé parametry	398
975	Nepodařené dosazení není chyba	399
976	Šablona jako aserce	399
977	Šablona jako nástroj pro zobrazení čísla na typ	400
978	Šablona jako nástroj pro zobrazení typu na typ	400
979	Překladač vypočte faktoriál	400
	Práce s datovými typy	403
980	K čemu je dynamická identifikace typů	403
981	Jak použít operátor typeid	403
982	Pro nepolymorfní typy určí operátor typeid statický typ	403
983	Chceme-li dynamický typ, musí být třídy polymorfní	404
984	Použití operátoru typeid k určení typu	404
985	Kdy (ne)používat operátor typeid	405
986	Nové přetypované operátory	405
987	Jak se nové přetypované operátory používají	405
988	Běžná přetypování	406
989	Modifikátory const a volatile	406
990	Podivná přetypování	406
991	Jak funguje operátor dynamic_cast	407
992	Přetypování ukazatele na předka na ukazatel potomka	407
993	Přetypování reference na předka na referenci na potomka	408
994	Patří skutečný typ instance mezi potomky jisté třídy?	408
995	Adresa celého objektu	409
996	Přetypování na „sousední třídu“	409
997	Co když předáme ukazatel s hodnotou 0	410
998	Přetypování na virtuálního potomka	410
999	Co je to přístupný předek	411
1000	Co je to jednoznačný předek	411
	Vstupní a výstupní operace v jazyce C	413
1001	Co je to datový proud	413
1002	Struktura FILE	413
1003	Standardní datové proudy v jazyce C	414
1004	Otevření datového proudu	414
1005	Binární a textový soubor, binární a textový režim	415
1006	Čtení a zápis v textovém režimu	415

1007	Když se operace nepodaří (EOF, WEOF a wint_t)	415
1008	Uzavření proudu	416
1009	Orientace datového proudu na široké a úzké znaky	416
1010	Změna orientace proudu	416
1011	Neformátovaný zápis do souboru	416
1012	Neformátované čtení ze souboru	417
1013	Aktuální pozice v proudu	418
1014	Aktualizace souboru	418
1015	Přečtení jednoho bajtu (znaku)	419
1016	Zápis jednoho bajtu (znaku)	419
1017	Zápis znakového řetězce	419
1018	Čtení jednoho znaku ze standardního vstupu	419
1019	Zápis jednoho znaku do standardního výstupu	420
1020	Vrácení jednoho znaku do proudu	420
1021	Čtení přímo z klávesnice (nestandardní řešení)	421
1022	Čtení funkčních kláves	421
1023	Formátovaný zápis	421
1024	Formátovací řetězec funkce fprintf()	422
1025	Specifikace konverze pro funkci fprintf()	422
1026	Určení typu ve specifikaci konverze pro funkci fprintf()	423
1027	Specifikace konverze pro další celočíselné typy z C99	423
1028	Specifikace velikosti pro funkci fprintf()	424
1029	Příznak pro funkci fprintf()	425
1030	Alternativní tvar výstupu (příznak #)	425
1031	Šířka vystupující hodnoty	426
1032	Přesnost ve specifikaci konverze pro funkci fprintf()	426
1033	Tisk tabulky funkce	427
1034	Formátovaný výstup na konzolu	427
1035	Zápis do znakového řetězce	427
1036	Výpis parametrů předaných na místě výpustky	428
1037	Když pracujeme se širokými znaky	428
1038	Formátované čtení	428
1039	Formátovací řetězec	429
1040	Specifikace konverze pro funkci fscanf()	429
1041	Specifikace typu pro funkci fscanf()	430
1042	Specifikace konverze pro další celočíselné typy v C99	430
1043	Čteme celá čísla	431
1044	Čteme jeden znak pomocí fscanf()	431

1045	Čteme řetězec pomocí fscanf()	431
1046	Čteme jen vybrané znaky	432
1047	Čteme reálná čísla	432
1048	Specifikace velikosti	432
1049	Nechceme uložit načtenou hodnotu (potlačené přiřazení)	433
1050	Další funkce pro formátované čtení úzkých znaků	433
1051	Funkce pro formátované čtení širokých znaků	433
1052	Jak zjistit stav proudu	433
1053	Jsmo na konci souboru?	434
1054	Spláchnutí vyrovnávací paměti	434
	Vstupní a výstupní operace v jazyce C++	435
1055	Třídy objektových datových proudů	435
1056	K čemu jsou tyto třídy dobré	435
1057	Připravené instance proudových tříd	436
1058	Formátované čtení a zápis: první přiblížení	436
1059	Formátované čtení a zápis podrobněji	437
1060	Hlavičkové soubory	437
1061	Proud jako parametr nebo výsledek funkce	437
1062	Otevření datového proudu pro čtení ze souboru	438
1063	Jaké existují režimy otevření	438
1064	Podařilo se otevření? Podařila se poslední operace?	439
1065	Čteme z textového souboru, dokud je co číst	439
1066	Čteme, dokud načtené hodnoty splňují danou podmínku	439
1067	Příznaky chyb v datových proudech	440
1068	Co se stane při zjištění chyby	440
1069	Odstranění příznaků chyby	440
1070	Výjimka jako reakce na chybu	441
1071	Chceme načíst následující znak	441
1072	Chceme načíst skupinu znaků	441
1073	Chceme při čtení přeskočit bílé znaky	442
1074	Chceme načíst celý řádek a známe délku	442
1075	Chceme načíst soubor po řádcích a neznáme jejich délku	442
1076	Otevření souborového proudu pro výstup	443
1077	Formátování výstupu: manipulátory	443
1078	Tiskneme tabulku funkce	445
1079	Vstup a výstup vlastních datových typů	446
1080	Operátor pro výstup komplexních čísel	446

1081	Operátor <<, který respektuje zadanou šířku	446
1082	Vlastní výstupní manipulátor bez parametrů: vložení 5 mezer	447
1083	Operátor pro vstup komplexních čísel	447
1084	Operátor >> kontrolující formát vstupních dat	448
1085	Uzavření proudu	448
1086	Výstup v šestnáctkové soustavě	448
1087	Čtení v šestnáctkové soustavě	449
1088	Spláchnutí proudu	449
1089	Svázané proudy	449
1090	Neformátovaný výpis jednoho nebo několika znaků	450
1091	Jaký znak je na řadě?	450
1092	Přeskočíme několik znaků	450
1093	Zjištění a změna aktuální pozice v proudu	450
1094	Vyrovňovací paměť proudu	451
1095	Přesměrování objektového datového proudu	451

Na přiloženém CD naleznete

	Kontejnery ve standardní knihovně	453
1096	Co jsou kontejnery a jak se dělí	453
1097	Jaké posloupnosti máme v STL k dispozici	453
1098	Jaké asociativní kontejnery máme k dispozici	454
1099	Další kontejnery	454
1100	Současná verze STL není úplná	454
1101	Jak používat kontejnery z STL: parametry šablony	455
1102	Jak používat kontejnery z STL: konstruktory	455
1103	Jak používat kontejnery ze STL: metody	455
1104	Jak používat kontejnery z STL: zveřejňované datové typy	456
1105	Typy zveřejňované frontou a zásobníkem	456
1106	Co jsou to iterátory	457
1107	Proč máme různé kategorie iterátorů	457
1108	Rozdělení iterátorů v STL	457
1109	Dereferencování iterátorů	458
1110	Vlastnosti vstupních iterátorů	458
1111	Vlastnosti výstupních iterátorů	458
1112	Vlastnosti dopředných iterátorů	458
1113	Vlastnosti obousměrných iterátorů	459

1114	Vlastnosti iterátorů pro náhodný přístup	459
1115	Platnost iterátorů	459
1116	Co lze ukládat do kontejnerů z STL	460
1117	Jak se ukládají data do kontejneru	460
1118	Vkládáme data do vektoru	460
1119	Počáteční kapacita vektoru	460
1120	Výpis všech prvků vektoru	461
1121	Přístup k prvkům vektoru	461
1122	Odstranění prvku z vektoru	462
1123	Chceme jen tolik paměti, kolik potřebujeme	462
1124	Tabulka proměnných: mapa	462
1125	Vytvoření nové proměnné: přidání prvku do mapy	463
1126	Přístup k prvkům mapy	463
1127	Výpis obsahu mapy	463
1128	Hodnota na vrcholu zásobníku	464
1129	Prvek v čele fronty	464
1130	Proč dvě metody?	465
	Algoritmy ve standardní knihovně	467
1131	Predikáty a funktoary	467
1132	Třídění neboli řazení v jazyce C	467
1133	Třídění neboli řazení v C++	468
1134	Řazení vektoru	468
1135	Řazení typů, pro které není definována relace <	468
1136	Stabilní řazení	469
1137	Částečné řazení	469
1138	Částečné řazení kopie posloupnosti	470
1139	Řazení seznamu	470
1140	k-tý prvek podle velikosti	471
1141	Otočení obsahu posloupnosti	471
1142	Náhodné promíchání prvků kontejneru	472
1143	První prvek s danou hodnotou	472
1144	První prvek splňující danou podmínku	472
1145	První prvek z dané množiny	473
1146	Počet prvků se zadanou hodnotou	473
1147	Počet prvků vyhovujících dané podmínce	474
1148	Kopírování do jiného kontejneru	474
1149	Odstranění prvků s danou hodnotou	475

1150	Kopírování prvků, které splňují danou podmínku	475
1151	Transformace všech prvků v kontejneru po jednom	475
1152	Fibonacciova čísla: transformace prvků po dvou	476
1153	Procházíme postupně permutace prvků	476
1154	Binární vyhledávání	477
1155	Kam vložit nový prvek?	477
1156	Vyplňujeme kontejner zadanou hodnotou	478
1157	Vyplňujeme kontejner generovanými hodnotami	478
1158	Sloučení dvou setříděných úseků	479
1159	Sloučení dvou setříděných úseků na místě	479
1160	Množiny a operace s nimi	480
1161	Je A podmnožinou B?	480
1162	Průnik množin A a B	481
1163	Větší nebo menší ze dvou hodnot	481
1164	Nejmenší nebo největší prvek v daném úseku kontejneru	481
1165	Prohození obsahu dvou proměnných	482
1166	Prohození úseků dvou kontejnerů	482
1167	Prohození obsahu kontejnerů	483
1168	Standardní predikáty	483
1169	Funkční objekt vracející větší ze dvou hodnot	484
1170	Další iterátory	484
1171	Kopírování přímo do proudu	485
	Lokální nastavení v C a v C++	487
1172	Čeho všeho se lokální nastavení týká	487
1173	Lokální nastavení v jazyce C	487
1174	Lokální nastavení v C++	488
1175	Jméno lokálního nastavení	488
1176	Kódové stránky	489
1177	Výpis textu do souboru v daném kódování (jazyk C)	489
1178	Čtení textu ze souboru v daném kódování (jazyk C)	489
1179	Výpis textu do souboru v daném kódování pomocí objektových proudů (C++)	490
1180	Čtení textu ze souboru v daném kódování pomocí objektových datových proudů (C++)	490
1181	Výpis českého textu na konzolu a čtení z ní	490
1182	Abecední řazení: obecné problémy	491
1183	Abecední řazení v češtině	491
1184	Porovnání dvou řetězců podle pravidel abecedního řazení v C	492
1185	Abecední řazení pole řetězců v jazyce C	492

1186	Abecední porovnání dvou instancí typu string v C++	493
1187	Abecední řazení pole řetězců v C++	493
1188	Formátování čísel v jazyce C	494
1189	Údaje o nastavených hodnotách pro formátování čísel	494
1190	Formátování čísel při výstupu v C++	494
1191	Čtení formátovaných čísel v C++	495
1192	Převody mezi úzkými a širokými znaky	495
1193	Fazety lokálního nastavení	496
1194	Jak použít fazetu lokálního nastavení	496
1195	Zjištění systémového času a data	496
1196	Formátování data a času: funkce strftime nebo wcsftime()	497
1197	Výpis aktuálního data a času	497
1198	Formátovací řetězec pro strftime() [37-26]	498
1199	Význam specifikací konverzí pro funkci strftime()	498
1200	Modifikátory E a O	500
1201	Nestandardní modifikátor # (Visual C++)	500
1202	Počítání týdnů v roce podle ISO 8601	500
1203	Datum ve formátu ISO 8601	501
1204	Identifikátory v programech	501
	Komplexní čísla	503
1205	Základní pojmy	503
1206	Datové typy pro komplexní čísla v C99 a odpovídající reálné typy	503
1207	Imaginární čísla (C99)	503
1208	Základní matematické operace s komplexními čísly	503
1209	Imaginární jednotka (C99)	504
1210	Alternativní modifikátory (C99)	504
1211	Deklarace proměnné s inicializací (C99)	504
1212	Reálná a imaginární část (C99)	504
1213	Vstup a výstup komplexních čísel (C99)	505
1214	Komplexně sdružené číslo (C99)	505
1215	Konverze mezi typy komplexních čísel (C99)	505
1216	Konverze mezi komplexními, reálnými a ryze imaginárními čísly (C99)	505
1217	Matematické funkce (C99)	506
1218	Poznámky k inverzním trigonometrickým funkcím pro komplexní čísla (C99)	506
1219	Poznámky k inverzním hyperbolometrickým funkcím pro komplexní čísla (C99)	506
1220	Poznámky k dalším funkcím pro komplexní čísla (C99)	507
1221	Goniometrický tvar komplexního čísla	507

1222	Komplexní čísla v C++	507
1223	Deklarace komplexní proměnné	508
1224	Konverze komplexních čísel (C++)	508
1225	Přiřazování komplexních čísel	508
1226	Aritmetické operace pro komplexní čísla (C++)	508
1227	Relace pro komplexní čísla (C++)	509
1228	Reálná a imaginární část komplexního čísla (C++)	509
1229	Vstup a výstup komplexních čísel pomocí objektových proudů (C++)	509
1230	Matematické funkce pro komplexní čísla (C++)	509
1231	Komplexně sdružené číslo	510
1232	Goniometrická reprezentace komplexního čísla (C++)	510
1233	Vytvoření komplexního čísla na základě goniometrické reprezentace (C++)	510
	C++0x	511
1234	Co je C++0x	511
1235	Podpora C++0x v překladači g++	511
1236	Nové znakové typy a řetězcové literály	511
1237	Ukazatel nikam	511
1238	Nová podoba příkazu for (foreach)	512
1239	Zapomeňte na klíčové slovo auto pro paměťovou třídu	512
1240	Jsme líní specifikovat typ proměnné	512
1241	Reference na r-hodnotu	513
1242	Aserce v době překladu	513
1243	Inicializace kontejnerů	513
1244	Volání jiného konstruktora téže třídy	514
1245	Inicializace složek v deklaraci	514
1246	Když nevíme přesně typ	514
1247	Alternativní zápis deklarace funkce	514
1248	Funkce, jejíž typ výsledku závisí na několika parametrech šablony	515
1249	Co je to lambda-výraz	515
1250	Transformace vektoru (nepojmenovaná funkce v C++0x)	516
1251	Lambda-výraz	516
1252	Okolní proměnné	517
1253	Podrobnější specifikace záchyty	517
1254	Jakého typu je lambda-výraz	518
1255	Deklarace šablony s proměnným počtem parametrů	518
1256	Definice šablony s proměnným počtem parametrů	518
1257	Speciální funkce ve standardní knihovně	519

	Několik tipů na závěr	521
1258	Několik slov o optimalizaci	521
1259	Standard popisuje pozorovatelné chování	521
1260	Ukazatele na metody	522
1261	Jak získáme ukazatel na metodu	522
1262	Třídní ukazatel na datové složky	522
1263	Jak získáme hodnotu třídního ukazatele na datovou složku	523
1264	Dereferencování třídního ukazatele na metodu	523
1265	Dereferencování třídního ukazatele na datovou složku	523
1266	A co statické metody a složky?	524
1267	Třídní ukazatele a pozdní vazba	524
1268	Třídní ukazatele: pohled pod pokličku	524
1269	Jak zjistit aktuální čas	524
1270	Jak zjistit čas procesu	525
1271	Přesnější měření času (nestandardní řešení)	525
1272	Typově generická makra V C99	525
1273	Generování náhodných čísel	525
1274	Randomizace náhodné posloupnosti	526
1275	Generátory náhodných čísel v C++: hudba budoucnosti	526
1276	Náhrada operátorů	526
1277	Co jsou to signály	527
1278	Obsluha signálu	527
1279	Vyvolání signálu	527
1280	Čtení z paměťového proudu	528
1281	Zápis do paměťového proudu	528
1282	Chceme přejmenovat soubor	528
1283	Chceme odstranit soubor	528
1284	Potřebujeme dočasný soubor	529
1285	Jak vytvořit platné jméno dočasného souboru	529
1286	Co zavádí tato deklarace?	529

Úvod

Začnu velmi osobně: Když jsem poznal jazyk C, začalo se mi programování líbit, ale jeho skutečnou krásu jsem pochopil, až když jsem porozuměl jazyku C++.

Něco z tohoto pocitu jsem se pokusil zprostředkovat vám v této knize. Najdete v ní více než 1000 tipů a triků, které můžete využít při programování v těchto dvou jazycích. Najdete tu jak základní obraty, které ocení začátečníci, tak i věci, které se budou hodit pokročilým programátorům při řešení speciálních úloh.

Nesnažil jsem se napsat učebnici C nebo C++. Mým cílem bylo ukázat programátorské obraty a známé či méně známé knihovní nástroje, bez nichž se programátor v C++ při seriózním programování neobejde. V případech, kdy to má smysl, ukazuji jak řešení v C, tak i řešení v C++.

Při psaní jsem převážně vycházel z platných standardů obou jazyků [ISOC] a [ISOCPP]. Uvádím ovšem i některá běžná nestandardní rozšíření; skutečnost, že nejde o součást standardu a tedy že řešení, které je na tom založeno, nemusí být přenositelné, vždy výslovně zdůrazňuji. V některých případech se také zmiňuji o očekávaných rozšířeních v příští verzi standardu.

Při psaní jsem vycházel především ze zkušeností s těmito komerčními vývojovými prostředími a překladači:

- CodeGear C++Builder2009 a překladač bcc32,
- Microsoft Visual C++ 2008 a překladač cl.

Vedle toho jsem měl k dispozici nekomerční překladač g++, který je součástí MinGW32 verze 3.4.5 a je šířen pod licencí Lesser GPL. V době korektur jsem měl také již ostrou verzi překladače Microsoft Visual C++ 2010.

V celé knize používám označení C90 pro implementaci jazyka C podle standardu ISO 9899 z roku 1990, která je stále široce používána, a označení C99 pro implementaci podle současného standardu ISO 9899 z roku 1999.

Na závěr bych chtěl poděkovat všem, kteří svými radami a připomínkami přispěli ke zdárnému dokončení tohoto díla.

Komu je kniha určena

Každý tip a trik je v knize označen pomocí jedné ze tří úrovní pokročilosti, které po čtenářích buď vyžadují, nebo nevyžadují určité znalosti C++. Kniha je tak vhodná pro všechny skupiny programátorů.



začátečník

Vyžaduje základní orientaci v C++



pokročilý

Předpokládá pokročilé znalosti jazyka C++, které rozšiřuje.



znalec

Předpokládá velmi dobré znalosti C++, popisuje pokročilé a rafinované postupy.

Doprovodné CD

Doprovodný disk obsahuje kromě zdrojových kódů také řadu odkazů na užitečné stránky a také několik užitečných nástrojů, jež vám programování v jazyce C++ výrazně usnadní nebo alespoň zpříjemní. Najdete na něm také instalační programy vývojových prostředí, která jsou v knize zmíněna.

CD stačí vložit do počítače a rozhraní se spustí automaticky. Pokud nemáte automatické spouštění disků povoleno, vyhledejte na CD kořenový adresář a otevřete soubor `spustit_CD.html`.

Jestliže rozhraní CD otevřete v prohlížeči Internet Explorer, Opera nebo Google Chrome, budete z CD moci instalovat doprovodný software okamžitě. V případě jiných prohlížečů se zobrazí výzva k uložení instalačního souboru na pevný disk. V tomto případě doporučujeme spustit instalaci přímo z CD. Obsah CD najdete ve složce obsah.