

knihovna programátora

- Přehledná a praktická učebnice C++ pro začátečníky i uživatele ostatních programovacích jazyků
- Základní programovací konstrukce, šablony, vestavěné datové typy, příkazy jazyka C++, pole a ukazatele
- Uživatelem definované neobjektové typy, výrazy a deklarace, funkce, práce s preprocesorem jazyka C++, objektové typy, výjimky, vstupní a výstupní operace
- Více než 170 příkladů použití v jednoduchých programech
- Podle mezinárodního standardu jazyka C++ z roku 2017



Programování v C++ od základů k profesionálnímu použití

MIROSLAV VIRIUS



soubory
ke stažení na

WWW.GRADA.CZ



knihovna programátora

C Programování v **++**

**od základů
k profesionálnímu
použití**

Miroslav Virius

Upozornění pro čtenáře a uživatele této knihy

Všechna práva vyhrazena. Žádná část této tištěné či elektronické knihy nesmí být reprodukována a šířena v papírové, elektronické či jiné podobě bez předchozího písemného souhlasu nakladatele. Neoprávněné užití této knihy bude **trestně stíháno**.

Miroslav Vírúš

Programování v C++

od základů k profesionálnímu použití

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401, fax: +420 234 264 400
jako svou 6791. publikaci

Odpovědný redaktor Petr Somogyi
Sazba Petr Somogyi
Počet stran 416
První vydání, Praha 2018
Vytiskly Tiskárny Havlíčkův Brod, a. s.

© Grada Publishing, a.s., 2018
Cover Design © Grada Publishing, a. s., 2018

Názvy produktů, firem apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

ISBN 978-80-271-0956-2 (pdf)
ISBN 978-80-271-0502-1 (print)

Obsah

	Předmluva.....	15
1	Než začneme.....	17
1.1	Počítač.....	17
1.1.1	Operační paměť.....	18
1.1.2	Soustava SI, bity a bajty.....	18
1.2	Datové typy a proměnné.....	18
1.3	Programy a programovací jazyky.....	19
1.3.1	Jazyk C++.....	20
1.3.2	Asembler.....	20
1.4	Operační systém.....	21
1.5	Program a algoritmus.....	21
1.5.1	Metoda shora dolů.....	22
1.5.2	Metoda zdola nahoru.....	23
1.6	Objekty a třídy.....	23
1.6.1	Zapouzdření.....	23
1.6.2	Dědění.....	26
1.6.3	Polymorfismus.....	28
1.6.4	Objektové a neobjektové programování.....	29
1.6.5	Poznámka k terminologii.....	29
1.7	Céčko se dvěma plusy.....	29
1.7.1	Vývoj C++.....	30
1.7.2	Dvě úrovně znalostí C++.....	30
1.7.3	Vývojové nástroje.....	31
1.7.4	Kde získat potřebné nástroje.....	31
1.7.5	Co budeme používat.....	32
2	První program.....	33
2.1	Bez použití vývojového prostředí.....	33
2.1.1	Zdrojový kód.....	33
2.1.2	Překlad a spuštění.....	34
2.1.3	Když se něco nezdaří.....	35
2.2	Používáme vývojové prostředí.....	36
2.2.1	Vytváříme projekt.....	36
2.2.2	Píšeme program.....	39
2.2.3	Překládáme program.....	39
2.2.4	Soubory projektu.....	39
2.2.5	Spouštíme program z IDE CB.....	40
2.3	Co jsme naprogramovali.....	40
2.3.1	Komentář.....	40
2.3.2	Hlavičkový soubor.....	41
2.3.3	Funkce main().....	41
2.3.4	Hlavička funkce main().....	42
2.3.5	Tělo funkce main().....	42
2.3.6	Výstup textu.....	42

2.3.7	Výsledek funkce main().....	43
2.3.8	Středník.....	43
2.3.9	Jméno a příjmení (jmenné prostory).....	43
2.3.10	Jak zapisujeme zdrojový program.....	44

3	Složitější programy.....	46
3.1	Dvojnásobek zadaného čísla.....	46
3.1.1	Nástroje.....	46
3.1.2	Zdrojový kód programu.....	47
3.2	Větší ze dvou čísel.....	48
3.2.1	Co bude program dělat.....	48
3.2.2	Přiřazení.....	48
3.2.3	Rozhodování: které číslo je větší?.....	49
3.2.4	Operátor podmíněného výrazu.....	50
3.3	Počítáme faktoriál.....	50
3.3.1	Co bude program dělat.....	50
3.3.2	Zdrojový kód.....	51
3.3.3	Zkrácený zápis některých výpočtů.....	52
3.4	Samostatná funkce.....	53
3.4.1	Neopakujte se!.....	53
3.4.2	Funkce faktorial().....	54
3.4.3	Celý program.....	55
3.5	Deklarace a použití.....	55
3.5.1	Definice a volání funkce.....	56
3.5.2	Lokální a globální proměnné.....	56
3.6	Několik zdrojových souborů.....	58
3.6.1	Struktura programu.....	58
3.6.2	Obsah hlavičkového souboru.....	59
3.6.3	Překlad z příkazové řádky.....	60
3.6.4	Projekt v prostředí CB.....	60
3.7	Testování programu.....	61
3.7.1	Nesmyslný vstup.....	61
3.7.2	Velká čísla.....	62
3.8	Rozmezí hodnot.....	64
3.8.1	Konjunkce dvou podmínek.....	64
3.8.2	Magická čísla.....	65
3.8.3	Pojmenované konstanty.....	65
3.8.4	Upravený program.....	65
3.9	Zpřehledňujeme program.....	66
3.9.1	Logické hodnoty (typ bool).....	67
3.9.2	Vstup hodnoty od uživatele.....	67
3.9.3	Test rozmezí hodnot.....	67
3.9.4	Tisk zprávy o chybě.....	68
3.9.5	Struktura programu.....	68
3.9.6	Funkce main().....	68
3.10	Proč počítat spočítané.....	69
3.10.1	Rekurzivní výpočet faktoriálu.....	69
3.10.2	Jiná možnost.....	70

3.11	Jmenný prostor pro naše funkce	71
3.11.1	Deklarace jmenného prostoru	72
3.11.2	Úprava programu	72
3.12	Program neumí pod Windows česky	73
3.12.1	Podivné znaky	73
3.12.2	Co s tím?	73
3.12.3	Nastavujeme kódovou stránku ručně	74
3.12.4	Nastavujeme kódovou stránku programově	74
4	Pokročilejší konstrukce a techniky	77
4.1	První objektový program	77
4.1.1	Třída Napis	77
4.1.2	Vytvoření a použití instance	80
4.1.3	Přístup ke složkám instance	82
4.1.4	Sledujeme počet nápisů (statické složky)	83
4.2	Mnoho verzí téže funkce	85
4.2.1	Větší ze dvou čísel	85
4.2.2	Přetěžování funkcí	85
4.2.3	Šablona funkce	86
4.2.4	Program se šablonami	87
4.3	Mnoho verzí téže třídy	87
4.3.1	Deklarace šablony objektového typu	88
4.3.2	Šablony metod	88
4.3.3	Program se šablonami	89
4.4	Potřebujeme počítat	89
4.4.1	Třída vektor	89
4.4.2	Výchozí hodnoty parametrů	90
4.4.3	Přetěžování operátorů	91
4.5	Ladění programu	93
4.5.1	Zdrojový kód	93
4.5.2	Příprava programu pro ladění v CB	94
4.5.3	Krokujeme program v CB	95
4.5.4	Skok na dané místo programu	95
4.5.5	Zarážka	96
4.5.6	Zjišťujeme hodnotu proměnné	96
4.5.7	Testujeme opravený program	97
4.5.8	Další možnosti	97
5	Začínáme naostro	98
5.1	Jak budeme jazyk C++ popisovat	98
5.1.1	Pravidla popisu syntaxe	98
5.2	Základní pojmy	99
5.2.1	Komentář	99
5.2.2	Identifikátor	100
5.2.3	Klíčová slova	100
5.2.4	Zápis programu	101
5.2.5	L-hodnota a r-hodnota	102

5.3	Organizace programu	102
5.3.1	Implementační a hlavičkové soubory	102
5.3.2	Standardní hlavičkové soubory	103

6 Základní datové typy

6.1	Celočíselné datové typy	104
6.1.1	Typy se znaménkem a bez znaménka	105
6.1.2	Celočíselné literály	106
6.1.3	Mezní hodnoty celočíselných typů	107
6.1.4	Operace s celočíselnými typy	107
6.1.5	Přiřazování celočíselných hodnot	109
6.2	Znakové typy	110
6.2.1	Znakové typy v C++	111
6.2.2	Znakové literály	112
6.2.3	Operace se znakovými typy	114
6.2.4	Znakové řetězce	114
6.3	Logické hodnoty	115
6.3.1	Automatické konverze	115
6.3.2	Operace s logickými hodnotami	115
6.4	Reálná čísla	117
6.4.1	Reálné literály	118
6.4.2	Informace o reálných typech	118
6.4.3	Operace s reálnými čísly	119
6.4.4	Matematické funkce	120
6.5	Neúplný typ void	122

7 Příkazy

7.1	Jednoduché příkazy	123
7.1.1	Výrazový příkaz	123
7.1.2	Prázdný příkaz	124
7.1.3	Deklarace	125
7.2	Blok (složený příkaz)	125
7.2.1	Co je to blok	125
7.2.2	Blok, platnost a viditelnost deklarací	126
7.3	Rozhodování (větvení algoritmu)	126
7.3.1	Příkaz if	126
7.3.2	Příkaz switch	127
7.4	Cykly	130
7.4.1	Příkaz while	130
7.4.2	Příkaz do-while	131
7.4.3	Příkaz for (klasický)	132
7.4.4	Příkaz for pro kontejnery	133
7.5	Příkazy pro přenos řízení	135
7.5.1	Příkaz return	135
7.5.2	Příkaz break	135
7.5.3	Příkaz continue	136
7.5.4	Příkaz goto a návěští	137

7.6	Další příkazy	138
7.6.1	Deklarace asm.....	138
7.6.2	Příkaz static_assert.....	139
7.6.3	Příkazy pro práci s výjimkami.....	139

8 Pole 140

8.1	Jednorozměrné pole	140
8.1.1	Rozsah pole.....	140
8.1.2	Inicializace jednorozměrného pole	141
8.1.3	Operace s poli.....	142
8.1.4	Pole jako parametr funkce.....	143
8.1.5	Znakové řetězce ukončené nulou.....	147
8.1.6	Surové řetězcové literály.....	148
8.2	Vícerozměrná pole.....	150
8.2.1	Deklarace vícerozměrného pole	151
8.2.2	Uložení prvků v paměti	151
8.2.3	Inicializace vícerozměrného pole.....	151
8.2.4	Vícerozměrné pole jako parametr funkce.....	152

9 Práce s pamětí..... 153

9.1	Ukazatele.....	153
9.1.1	Doménový typ ukazatele.....	153
9.1.2	Deklarace proměnné typu ukazatel.....	153
9.1.3	Užití ukazatele k nepřímému přístupu.....	154
9.1.4	Ukazatel bez doménového typu.....	156
9.1.5	Ukazatel nikam	156
9.1.6	Přiřazování ukazatelů	157
9.1.7	Relace mezi ukazateli.....	157
9.1.8	Funkce, které vracejí ukazatel	157
9.2	Vztah polí a ukazatelů.....	158
9.2.1	Konverze pole na ukazatel.....	158
9.2.2	Funkce, které vracejí pole.....	159
9.2.3	Adresová aritmetika	160
9.2.4	Ukazatele a znakové řetězce	161
9.3	Dynamické proměnné.....	164
9.3.1	Paměť přidělená programu	164
9.3.2	Vytvoření dynamické proměnné.....	165
9.3.3	Když se alokace nepodaří	167
9.3.4	Zrušení dynamické proměnné	167
9.3.5	Běžné chyby.....	168
9.4	Reference.....	168
9.4.1	Reference na l-hodnotu.....	168
9.4.2	Reference na r-hodnotu.....	170
9.5	Program, který hádá zvířata.....	171
9.5.1	Jak bude program fungovat.....	171
9.5.2	Jak bude program hádat	172
9.5.3	Strom v programu	172

10	Výčtové typy, struktury a unie	179
10.1	Výčtové typy.....	179
10.1.1	Deklarace slabého výčtového typu.....	179
10.1.2	Operace se slabými výčtovými typy.....	181
10.1.3	Deklarace silného výčtového typu.....	182
10.1.4	Použití silného výčtového typu.....	182
10.2	Struktury.....	183
10.2.1	Deklarace struktury (datového typu).....	183
10.2.2	Operace se strukturami.....	185
10.3	Unie.....	186
10.3.1	Deklarace unie.....	186
10.3.2	Operace s uniemi.....	187
10.3.3	Neúplná deklarace.....	190
11	Výrazy	191
11.1	Pořadí operací.....	191
11.1.1	Priorita operátorů.....	191
11.1.2	Asociativita operátorů.....	192
11.1.3	Příprava operandů před provedením operace.....	192
11.2	Přehled operátorů.....	193
11.2.1	Operátory, u nichž je určeno pořadí přípravy operandů.....	194
11.2.2	Některé další operátory.....	197
11.3	Typ výrazu.....	198
11.3.1	Konverze číselných typů.....	198
11.3.2	Automatické konverze ukazatelů.....	200
11.3.3	Typ výrazu.....	201
11.4	Konstantní výrazy.....	204
12	Deklarace a proměnné	205
12.1	Deklarace a definice.....	205
12.2	Paměťové třídy proměnných.....	205
12.2.1	Automatické proměnné.....	205
12.2.2	Registrové proměnné.....	206
12.2.3	Externí proměnné.....	206
12.2.4	Statické proměnné.....	207
12.3	Konstantní a nestálé proměnné.....	208
12.3.1	Konstanty (const).....	208
12.3.2	Konstanty (constexpr).....	209
12.3.3	Nestálé proměnné.....	209
12.4	Deklarace proměnné.....	209
12.4.1	Začínáme.....	209
12.4.2	Deklarace je odvozena od výrazu.....	210
12.4.3	Jak číst deklaraci.....	210
12.4.4	Deklarace proměnné bez uvedení typu.....	211
12.4.5	Inicializace.....	212

12.5	Nové jméno existujícího typu.....	213
12.5.1	Deklarace typedef	213
12.5.2	Deklarace using.....	213
12.5.3	Označení typu.....	214
12.6	Platnost a viditelnost deklarace	215
12.6.1	Obor platnosti	215
12.6.2	Obor viditelnosti	216
12.7	Atributy.....	217
13	Funkce.....	218
13.1	Deklarace funkce.....	218
13.1.1	Hlavička funkce.....	218
13.1.2	Tělo funkce.....	220
13.1.3	Různé modifikátory.....	222
13.2	Parametry funkce.....	223
13.2.1	Předávání parametrů	224
13.2.2	Proměnný počet parametrů.....	225
13.2.3	Nepojmenované parametry.....	228
13.3	Přetěžování funkcí.....	229
13.3.1	Kdy je vhodné přetěžovat funkce.....	229
13.3.2	Která funkce se zavolá?.....	230
13.3.3	Přetěžování a výchozí hodnoty parametrů.....	232
13.4	Ukazatele na funkce.....	232
13.4.1	Deklarace ukazatele na funkci.....	232
13.4.2	Operace s ukazateli na funkce.....	233
13.5	Lambda-výrazy.....	237
13.5.1	Zápis lambda-výrazu.....	237
13.5.2	Použití lambda-výrazu s prázdným záchytem.....	238
13.5.3	Záchyt.....	239
14	Program a jeho běh.....	240
14.1	Vlastnosti funkce main().....	240
14.1.1	Omezení kladená na funkci main().....	240
14.1.2	Co vrací funkce main().....	240
14.1.3	Parametry funkce main().....	241
14.2	Systémové proměnné (proměnné prostředí).....	242
14.2.1	Standardní přístup.....	242
14.2.2	Nestandardní přístup.....	243
14.3	Běh programu.....	244
14.3.1	Startovací a ukončovací kód.....	244
14.3.2	Operace po ukončení funkce main().....	244
14.3.3	Předčasné ukončení programu	245
15	Preprocesor.....	247
15.1	Co dělá preprocesor.....	247
15.2	Direktivy preprocesoru.....	247

15.2.1	Syntaktická pravidla pro direktivy preprocesoru.....	247
15.2.2	Prázdná direktiva #.....	248
15.2.3	Vložení souboru	248
15.2.4	Makra.....	248
15.2.5	Podmíněný překlad.....	253
15.2.6	Vyvolání chyby	255
15.2.7	Další direktivy	256
15.3	Samostatný preprocesor	256

16	Objektové typy.....	257
16.1	Deklarace objektového typu.....	257
16.1.1	Třída, struktura, unie	257
16.1.2	Tělo třídy.....	258
16.1.3	Přístup ke složkám	258
16.1.4	Metody.....	260
16.1.5	Datové typy deklarované ve třídě (vnořené typy)	261
16.2	Vytvoření a zánik instance.....	262
16.2.1	Konstruktor.....	262
16.2.2	Konstruktory, shrnutí.....	268
16.2.3	Volání jiného konstruktora téže třídy.....	271
16.2.4	Destruktor	271
16.3	Kopírování instancí.....	272
16.3.1	Vlastní kopírovací konstruktor	272
16.3.2	Vlastní přiřazovací operátor.....	274
16.4	Datové složky a metody třídy jako celku.....	277
16.4.1	Statické datové složky	277
16.4.2	Statické metody	278
16.5	Konstantní instance.....	279

17	Dědění.....	281
17.1	Odvozená třída.....	281
17.1.1	Deklarace odvozené třídy	281
17.1.2	Jak vypadá odvozená třída.....	283
17.2	Polymorfismus.....	286
17.2.1	Přiřazování instancí různých typů	286
17.2.2	Virtuální metody	287
17.2.3	Pohled pod pokličku	289
17.2.4	Konstruktory, destruktory a virtuální metody.....	290
17.3	Abstraktní třídy a čistě virtuální metody.....	293
17.3.1	Deklarace čistě virtuální metody	293
17.3.2	Použití abstraktní třídy.....	293
17.3.3	Finální třídy a finální metody.....	294
17.4	Dědění, nebo skládání?	295

18	Chyby za běhu programu	306
18.1	Aserce	306
18.1.1	Makro assert	306
18.1.2	Odstranění aserce	307
18.2	Výjimky	308
18.2.1	Jak používáme výjimky	308
18.2.2	Vyvolání výjimky	309
18.2.3	Zachycení a obsluha výjimky	312
18.2.4	Co se děje v programu s výjimkami	314
18.2.5	Výjimky, konstruktory a destruktory	316
18.2.6	Specifikace výjimek v deklaraci funkce	318
18.2.7	Třídy pro práci s výjimkami	319
19	Vstupy a výstupy	320
19.1	Soubor a proud	320
19.1.1	Soubor v C++	320
19.1.2	Datový proud	320
19.1.3	Textové a binární soubory	320
19.2	Datové proudy v C++	321
19.2.1	Třídy datových proudů	321
19.2.2	Vstupní a výstupní proudy	322
19.2.3	Formátování výstupu	326
19.3	Nástroje zděděné od jazyka C	336
19.3.1	Datový proud a základní operace s ním	336
19.3.2	Další operace s datovými proudy	342
19.3.3	Formátovaný zápis	344
19.3.4	Formátované čtení	349
19.3.5	Některé další funkce	353
20	Příklad	355
20.1	Filtr sort, základní verze	355
20.1.1	Zadání	355
20.1.2	Kontejnery v C++	355
20.1.3	Řazení	358
20.1.4	Čtení a zápis souboru	358
20.1.5	Implementace	359
20.1.6	Testujeme první verzi	360
20.1.7	Rozdělení zodpovědnosti	361
20.1.8	Neobjektová implementace	362
20.2	Filtr sort s volbami	363
20.2.1	Zadání	363
20.2.2	Volby	363
20.2.3	Řazení	366
20.2.4	Testujeme druhou verzi	367
20.3	Čeština pod Windows	367
20.3.1	Knihovna pro nastavení konzoly	367
20.3.2	Lokální nastavení	368
20.3.3	Překlad řetězce do požadované kódové stránky	369

20.4	Příprava na další volby	371
20.4.1	Odstraňujeme typ zvoleno	372
20.4.2	Určení komparátoru	373
20.4.3	Implementace	375
20.5	Abecední řazení	376
20.5.1	První implementace	377
20.6	Vlastní implementace abecedního řazení	377
20.7	Opačné řazení	382
20.7.1	První úvahy	383
20.7.2	Šablonové řešení	383
20.7.3	Testujeme opačné řazení	384
20.8	Jak snadno změnit pravidla abecedního řazení	384
20.8.1	Více složených písmen	384
20.8.2	Abecední řazení v různých jazycích	385
20.8.3	Obecný postup	385
20.8.4	Inicializační soubor	386
20.8.5	Reprezentace inicializačního souboru	386
20.8.6	Testujeme tuto verzi	393
20.9	Objektová implementace komparátorů	393
20.9.1	Volatelná instance objektového typu	393
20.9.2	Třídy komparátorů	393
20.9.3	Inicializační soubor	395
20.9.4	Třída volba	396
20.9.5	Řazení	397
20.9.6	Opačné řazení	398
20.10	Komparátor na přání	401
20.10.1	Nejjednodušší implementace tovární funkce	401
20.10.2	Implementace tabulkou	402
20.10.3	Třída volba	403
20.10.4	Třída tridic	404
20.10.5	Uvolnění komparátoru	404
20.11	Několik slov na závěr	405

21	O čem jsme nehovořili	406
21.1	Jazyk C++	406
21.2	Standardní knihovna	407

Literatura	409
------------------	-----

Rejstřík	411
----------------	-----

Předmluva

Kniha, kterou držíte v ruce, vás seznámí se základy programování v jazyce C++. Je určena jak čtenářům, kteří ještě vůbec neprogramovali, tak i těm, kteří již v nějakém jazyce programovali a chtějí se seznámit s jazykem C++. Předpokládám, že čtenář je sice začátečníkem v programování, ale umí s počítačem zacházet uživatelsky, tedy dokáže spustit program, zkopírovat nebo smazat soubor atd. V průběhu dvaceti kapitol této knihy se jej pokusím dovést na úroveň lehce pokročilého programátora.

Co v této knize najdete

Jediným způsobem, jak se naučit nějaký programovací jazyk, je psát v něm programy, a proto už ve druhé kapitole začneme programovat. Cílem druhé, třetí a čtvrté kapitoly je uvést na scénu vybrané konstrukce jazyka C++ a umožnit čtenářům zkusit si vše, o čem si v následujících kapitolách povíme, na vlastních programech – jednoduchých a možná nešikovných, ale fungujících.

Než se ovšem pustíme do programování, musíme si ujasnit některé pojmy a sjednotit terminologii. Proto se v první kapitole seznámíme s pojmy, které budeme používat. Povíme si, co je třeba vědět o počítači, o programovacích jazycích, algoritmech, objektivě orientovaném programování a některých dalších věcech, bez nichž se při programování neobejdeme.

Ve druhé kapitole napíšeme svůj první program, ukážeme si, jak ho přeložit (neboť jazyk, v němž ho píšeme, je jiný než jazyk, v němž „myslí“ počítač) a jak ho spustit. Seznámíme se s integrovaným vývojovým prostředím Code::Blocks a s překladačem g++, které budeme v této knize používat.

Ve třetí kapitole napíšeme několik verzí jednoduchých programů, které přečtou číslo, vy počtou z něj jistou hodnotu a výsledek vypíší. To nám umožní seznámit se alespoň povrchně s některými základními programovacími konstrukcemi, s rozdělením programu do několika souborů, s pojmem projektu apod. Se znalostmi z této kapitoly byste měli být schopni zkusit si vše, co se v dalších kapitolách naučíte, na vlastních programech.

Ve čtvrté kapitole si ukážeme jednoduchý objektový program, seznámíme se velmi povrchně se šablonami a předvedeme si, jak ladit program – tedy jak zjistit, proč dělá něco jiného, než co jsme chtěli naprogramovat.

Od páté kapitoly začíná výklad „naostro“. Po seznámení se základními pojmy postupně poznáme vestavěné datové typy, příkazy jazyka C++, naučíme se používat tzv. pole a ukazatele. V deváté kapitole napíšeme první větší program – jednoduchou hru, v níž počítač klade otázky a podle uživatelových odpovědí hádá zvíře.

V následujících kapitolách se seznámíme s uživatelem definovanými neobjektovými typy, s výrazy a deklaracemi, s funkcemi, s preprocesorem jazyka C++, s objektovými typy a s tzv. výjimkami (nástroj pro ošetřování chyb za běhu). V předposlední kapitole probereme nástroje pro vstupní a výstupní operace (tedy pro čtení údajů z klávesnice, ze souboru apod.) a pro zápis údajů na obrazovku, do souboru apod.

V poslední kapitole postupně vyvineme několik verzí programu pro seřazení obsahu textového souboru. Jde o analogii filtru *sort*, který je standardní součástí běžných operačních systémů; naše verze se ovšem bude od standardního programu v mnoha ohledech lišit. To nám umožní seznámit se s řadou nástrojů ze standardní knihovny jazyka C++ a s některými užitečnými programátorskými technikami.

V poslední kapitole si krátce povíme o věcech, o nichž jsme v této knize nehovořili nebo jsme si řekli jen nejnnutnější informace, ale které by měl pokročilý programátor v C++ znát.

Nástroje

Výklad v této knize je založen na mezinárodním standardu jazyka C++ [3] z roku 2017. Pro psaní, překlad, sestavení a ladění programů používáme vývojové prostředí Code::Blocks, které lze zdarma získat na stránkách [15] spolu s překladačem g++. Vedle toho zde najdeme poznámky pro čtenáře, kteří se rozhodnou pro Visual C++, jehož základní verzi lze také získat zdarma. Poznamenejme, že verze 4.9.2 překladače g++, která je implicitně dodávána s prostředím Code::Blocks, ve skutečnosti implementuje jazyk C++ na úrovni standardu z roku 2011. Není však problém instalovat si a používat nejnovější verzi tohoto překladače (v době dokončování této knihy to byla verze 6.4), která plně implementuje standard z roku 2014 a většinu novinek standardu z roku 2017. Místo překladače g++ si také můžete instalovat jakýkoli jiný překladač C++. Je však třeba poznamenat, že rozšíření, která přinesly pozdější verze standardu, v této knize nepoužívám, neboť výklad o nich přesahuje možnosti knihy určené začátečníkům.

Terminologie

V celé knize používám důsledně českou terminologii. Víím, že má mnoho odpůrců, jsem však přesvědčen, že použití vhodných českých názvů výrazně usnadní pochopení, oč jde. Anglické termíny samozřejmě uvádím alespoň při prvním výskytu také.

Příklady

Výklad v této knize doprovází velké množství příkladů. Jejich zdrojové texty si můžete stáhnout z webových stránek nakladatelství Grada Publishing (www.grada.cz, vyhledejte příslušnou sekci této knihy), nebo z mých osobních stránek, jejichž adresu najdete dále. Tyto příklady jsem odladil pod překladačem g++ verze 4.9.2 a pod Visual C++ 2015.

Poděkování

Na závěr bych chtěl poděkovat všem, kteří svými radami a připomínkami napomohli zdárnému dokončení tohoto díla, především pak Ing. Rudolfu Pecinovskému, CSc., z VŠE v Praze a Ing. Vladimíru Jarému, Ph.D., z KSI FJFI ČVUT, kteří četli vybrané části této knihy a měli k nim řadu připomínek. I přes veškerou péči, kterou jsem této knize věnoval, se do ní ovšem mohly vloudit chyby; za ty nesu pochopitelně veškerou zodpovědnost já. Jestliže v této knize nějakou chybu najdete, pošlete mi prosím zprávu na níže uvedenou adresu. Bude-li to možné, uveřejním na svých webových stránkách opravu.

Ženeva, CERN, 21. srpna 2017

Miroslav Virius
miroslav.virius@fjfi.cvut.cz
<http://people.fjfi.cvut.cz/virius>

1 Než začneme

Než si začneme povídat o programovacím jazyce C++, musíme se alespoň povrchně seznámit s některými pojmy, které budeme používat. Začneme od počítače jako takového, pak si něco povíme o programovacích jazycích, algoritmech a objektově orientovaném programování a skončíme u C++ a nástrojů, které budeme potřebovat. Některé z uvedených pojmů nepochybně znáte; přesto vás prosím, abyste si toto povídání alespoň zběžně přečetli, abychom se mohli shodnout na terminologii.

1.1 Počítač

S osobními počítači se v současné době setkáváme denně – najdeme je nejen v každé kanceláři, ale i v téměř každé domácnosti. Dovolím si tedy předpokládat, že s počítačem umíte zacházet jako uživatelé – umíte ho zapnout, vytvořit na něm textový dokument apod. Ovšem programátor (a to i začínající) o něm musí přece jen vědět víc než běžný uživatel. I když bude povídání v této kapitole silně zjednodušené, pro naše účely bude stačit.

Obvykle se říká, že počítač obsahuje čtyři základní části:

- **Procesor** je součást, která opravdu „počítá“, lépe řečeno která zpracovává informace. Zároveň také řídí činnost všech ostatních částí počítače.
- **Operační paměť** slouží k ukládání dat (informací), která počítač zpracovává, a programů, tedy příkazů, které určují, co má dělat. Vše, co je v této paměti, se při vypnutí počítače ztratí, „zapomene“. Používá se pro ni označení *RAM*, což je zkratka anglických slov *Random Access Memory*, tedy **paměť s náhodným přístupem**.
- **Vstupní a výstupní zařízení** slouží k výměně informací s okolím. (Počítač by nám nebyl nic platný, kdyby nám nemohl předat výsledky své práce.) Typickými příklady vstupních zařízení jsou klávesnice, myš, skener atd. Typickým příkladem výstupních zařízení může být obrazovka monitoru nebo tiskárna. Pro vstupní a výstupní zařízení se používá zkratka *VV* nebo *I/O* (z anglického *input/output*).
- **Trvalá paměť** slouží k trvalému ukládání dat a programů.¹ Data v ní se při vypnutí počítače neztrácejí, má zpravidla mnohonásobně větší kapacitu než operační paměť, ale práce s ní je mnohonásobně pomalejší než práce s operační pamětí. Jako trvalá paměť se používají převážně magnetické disky (pevný disk, disk SSD – jeho název pochází z anglických slov *solid state drive* a je to paměťové zařízení podobné jako flash disk neboli „fleška“, ovšem s větší kapacitou) a samozřejmě také disky CD nebo DVD.

1 Setkáme se také s označením **vnější paměť**, neboť v dobách počítačové prehistorie byly magnetické disky opravdu uloženy mimo vlastní počítač. I dnes lze ovšem připojit k počítači vnější (externí) paměťové médium.

1.1.1 Operační paměť

Základem operační paměti jsou elektronické obvody, které mohou mít dva stavy – například vypnuto nebo zapnuto. Jeden z těchto stavů obvykle odpovídá číslici 0, druhý číslici 1. Údaje v paměti jsou proto vyjádřeny jen pomocí nul a jedniček, v tzv. dvojkové soustavě. Místo, na které můžeme uložit číslici 0 nebo 1, označujeme jako **bit**. Operační paměť je tedy dlouhá řada bitů.

Ovšem práce s jednotlivými bity je nepohodlná, a proto se bity sdružují do větších celků. V dnešních počítačích se téměř bez výjimky používají skupiny velikosti 8 bitů, které se nazývají **bajty** (anglicky *byte*, tedy slabika). Poznamenejme, že pro bity se používá značka *b* a pro bajty značka *B*.

Jednotlivé bajty, tvořící operační paměť, jsou očíslovány. Počáteční bajt má číslo 0, následující má číslo 1 atd. Toto pořadové číslo se nazývá **adresa** bajtu. (Ve skutečnosti může být záležitost s adresami trochu složitější, ale to nás v souvislosti s C++ vůbec nebude zajímat.)

1.1.2 Soustava SI, bity a bajty

Vnitřní paměť běžného počítače má dnes velikost několik miliard bajtů, kapacita disků a podobných zařízení se vyjadřuje v bilionech bajtů. Podobně jako v běžném životě nevyjadřujeme například vzdálenosti měst v metrech, ale v kilometrech, i v informatice se používají k vyjádření kapacity paměti (nebo obecně množství informace) větší jednotky. Běžně se hovoří o kilobajtech (kB), megabajtech (MB), gigabajtech (GB) atd.

Vzhledem k tomu, že počítače jsou založeny na dvojkové soustavě, je však přirozenější používat jednotky založené na mocninách dvou, nikoli na mocninách deseti, jak je tomu v soustavě SI. Proto by se měly používat jednotky kibibajt ($1 \text{ KiB} = 2^{10} \text{ B} = 1024 \text{ B}$), mebibajt ($1 \text{ MiB} = 1024 \text{ KiB} = 2^{20} \text{ B} = 1048576 \text{ B}$), gibibajt ($1 \text{ GiB} = 2^{30} \text{ B} = 1073741824 \text{ B}$) atd. Předpony těchto jednotek jsou složeny z první slabiky odpovídající předpony soustavy SI a slabiky **bi** ze slova *binary*.²

1.2 Datové typy a proměnné

Snadno zjistíme, že nejmenší číslo, které může jeden bajt obsahovat, se skládá z osmi nul a představuje i v desítkové soustavě nulu. Největší takové číslo se bude skládat z osmi jedniček a v desítkové soustavě představuje 255. To je samozřejmě málo – s počítačem, který by znal jen čísla od 0 do 255, bychom si nedokázali ani přepočítat výplatu. Proto se pro ukládání dat obvykle používají různé velké skupiny za sebou následujících bajtů.

Ani to ovšem nestačí. Snadno se přesvědčíme, že kdybychom vzali například skupinu dvou za sebou následujících bajtů, mohli bychom do ní uložit celá čísla v rozmezí od 0 do 65 535. Ale co když budeme potřebovat záporná čísla? Co když budeme potřebovat reálná čísla? Co když budeme chtít vyjádřit znaky nebo logickou hodnotu (nějaké tvrzení platí nebo neplatí)?

Musíme tedy zajistit způsob, který nám umožní reprezentovat data různých „druhů“ v paměti počítače. Jinými slovy, musíme najít způsob, jakým určité skupině bitů přiřadíme hodnotu, kterou tato skupina představuje – jak ji v počítači **zakódovat**.

2 Tyto jednotky byly zavedeny Mezinárodní elektrotechnickou komisí r. 1998, přijaty hlavními standardizačními organizacemi (jsou například součástí standardu [5]) a tvoří součást mezinárodního systému měrných jednotek.

Můžeme se například dohodnout, že bajt s hodnotou 01000001 bude představovat znak 'A'. Táž skupina bitů může za jiných okolností ovšem také představovat celé číslo, které má hodnotu 65. Stejný bajt ale může být i součástí většího celku s úplně jiným významem.

Předchozí příklady ukazují, že pracuje-li počítač s nějakým kouskem paměti, s nějakou skupinou za sebou následujících bajtů na určité adrese, musí vědět, jak je tato skupina velká a jak má její obsah interpretovat. Jinými slovy, musí znát **datový typ** hodnoty, která je tam uložena, musí vědět, zda jde o celé číslo, znak, logickou hodnotu atd.

Podle datového typu se také budou lišit operace, které lze s danou hodnotou provádět. Celá čísla lze například sčítat a odečítat, znaky lze spojovat do řetězců, tedy do souvislého textu.

Nyní se na celý problém podívejme trochu jinak. Už víme, že když budeme chtít pracovat s nějakou hodnotou, musíme si ji uložit do paměti. To ale znamená, že si tam pro ni musíme vyhradit místo a říci, jakého typu budou údaje, které bude obsahovat. Takové místo pro ukládání hodnoty budeme nazývat **proměnná**.

Abychom s proměnnou mohli zacházet, musíme ji pojmenovat, musíme jí přidělit **identifikátor**. Tomu se v programování říká **deklarace** proměnné.³

1.3 Programy a programovací jazyky

Aby mohl počítač nějak zpracovávat data, která mu předložíme, musíme mu také říci, co má vlastně dělat – musíme mu dát **program**.

Procesor umí s daty řadu operací, ovšem velice jednoduchých. Lze mu například říci „vezmi celé číslo, které je na adrese 6548, a přičti k němu celé číslo z adresy 7895“. Protože do jeho paměti nelze uložit nic jiného než čísla, musí být tyto příkazy vyjádřeny – zakódovány – také čísly. Toto číselné vyjádření instrukcí (příkazů) pro procesor se nazývá **strojový kód** a je to jediná věc, které procesor rozumí. Jedním z problémů je, že různé druhy počítačů používají různé strojové kódy, takže programy ve strojovém kódu nejsou přenositelné mezi počítači s různými procesory.

Jiným – a možná horším – problémem je, že programování ve strojovém kódu je velice namáhavé a nepřehledné. Také to téměř nikdo nedělá; místo toho se používají tzv. **vyšší programovací jazyky** – PHP, Java, C, Basic, a také C++.

Zápis programu ve vyšším programovacím jazyce se zpravidla skládá z vybraných anglických slov a z výrazů zapsaných podobně jako v matematice. Programování ve vyšším jazyce je pochopitelně daleko jednodušší než programování ve strojovém kódu. Je tu ovšem jeden háček: takovýto program nelze přímo spustit, neboť počítač mu nerozumí. Program ve vyšším programovacím jazyce se proto musí buď **přeložit** do strojového kódu, nebo **interpretovat**. V obou případech k tomu potřebujeme další program (nebo skupinu programů), které to za nás udělají.

Zápis programu ve vyšším programovacím jazyce se zpravidla označuje jako **zdrojový kód** nebo **zdrojový program**, v „programátorštině“ **zdroják**. Textový soubor, který tento zápis obsahuje, označujeme jako **zdrojový soubor**. O jeho **překlad** do strojového kódu (**kompilaci**) se stará program zvaný **překladač** neboli **kompilátor** (anglicky nazývaný *compiler*). V mnoha případech s ním spolupracuje ještě **sestavovací program** neboli *linker*, který může spojit několik nezávisle přeložených částí programu do jednoho celku. Sestavovací program také připojí knihovny – části programu, které už někdo naprogramoval předem a které můžeme už jen používat.

³ Později uvidíme, že identifikátor – tedy jméno – mohou mít i jiné části programu, nejen proměnné.

Překladem a sestavením programu vznikne soubor obsahující strojový kód, který lze na cílovém počítači rovnou spustit. Mezi často používané překládané programovací jazyky patří například C nebo C++.

Můžeme však také použít speciální program, jenž bude číst zdrojový text a provádět příkazy, které v něm najde – interpretovat ho. Typickým interpretovaným jazykem je klasický Basic.⁴

Interpretované programy zpravidla běží výrazně pomaleji než překládané programy. Navíc musíme na cílový počítač spolu s naším programem dodat také interpretační program.

Doplňme, že překladač libovolného programovacího jazyka (i C++) zároveň kontroluje syntaktickou správnost programu – tedy zda program je napsán podle jistých formálních pravidel, které zaručují, že mu počítač porozumí. (Syntaktická správnost programu bohužel nezaručuje věcnou správnost programu, tedy nezaručuje, že program bude dělat to, co si přejeme. Zaručuje pouze, že ho překladač dokáže přeložit.)

1.3.1 Jazyk C++

Už víme, že C++ patří mezi překládané programovací jazyky. Jeho zdrojový kód se může skládat z několika souborů. Překlad takového programu probíhá ve třech základních fázích:

1. Nejprve preprocesor upraví zdrojové texty – odstraní z nich komentáře (v následující kapitole se dozvíme, co to znamená) a provede některé další úpravy. Výsledkem jsou zdrojové soubory připravené k vlastnímu překladu.
2. Pak překladač přeloží samostatně jednotlivé zdrojové soubory. Překladem z nich vzniknou tzv. relativní soubory (anglicky nazývané *object file*). Relativní soubor se typicky jmenuje stejně jako zdrojový soubor, má však jinou příponu – pod Windows to je zpravidla `.obj`, v prostředí Linuxu to je `.o`.
3. Nakonec sestavovací program (linker) spojí relativní soubory v jeden celek, připojí k nim potřebné knihovny (už hotové součásti, které má překladač k dispozici) a vytvoří z nich spustitelný soubor; ten má pod Windows příponu `.exe`. V jiných operačních systémech nemusí mít žádnou zvláštní příponu.

Preprocesor, překladač a linker mohou být – a dnes zpravidla jsou – spojeny do jednoho programu, který provede postupně všechny potřebné kroky. Přepínači zadanými při spuštění tohoto programu lze ovšem zajistit, že proběhne například jen zpracování preprocesorem, jen zpracování preprocesorem a překlad nebo jen sestavení relativních souborů do spustitelného programu. Podrobněji se s tím seznámíme v příští kapitole.

1.3.2 Asembler

Mezistupněm mezi strojovým kódem a vyššími programovacími jazyky je **assembler** (*assembler, assembly language*). Je to jazyk složený především z mnemotechnických zkratk jednotlivých instrukcí strojového kódu. Například instrukce:

```
MOV EAX, 0
```

znamená v jednom z assemblerů procesorů Intel používaných na PC „ulož 0 do registru EAX“. (Registry jsou součástí procesoru, v nichž se provádějí operace s daty.) Asembler také zavádí proměnné jako pojmenovaná místa v paměti, a proto se pro ně dříve používalo označení **ja-**

4 Nehovoříme o Visual Basicu. Poznamenejme, že ve skutečnosti může být kterýkoli jazyk překládaný nebo interpretovaný (existuje například interpret jazyka C++), takže by bylo přesnější hovořit o jazycích *zpravidla* překládaných a *zpravidla* interpretovaných.