

Začínáme programovat v jazyku

C++

- » Poznáte programování v jazyce C++ od základních pojmů
- » Kniha ukazuje na řadě příkladů různé stránky tohoto programovacího jazyka
- » Seznámíte se s nástroji pro ladění programů
- » Nahlédnete i do pokročilých možností, jako jsou šablony nebo objektově orientované programování
- » Výklad je založen na vývojovém prostředí OnlineGDB Beta, které je dostupné na webu





edice
začínáme s ...

Začínáme programovat v jazyku

C++

MIROSLAV VIRIUS



GRADA Publishing

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 Virius

Začínáme programovat v jazyku C++

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401

jako svou 8642. publikaci

Odpovědná redaktorka Věra Slavíková

Sazba Jaroslav Kolman

Počet stran 200

První vydání, Praha 2023

Vytiskly Tiskárny Havlíčkův Brod a.s.

© Grada Publishing, a.s., 2023

Cover Design © Grada Publishing, a.s., 2023

Cover Photo © Shutterstock.com/anna_pustynnikova/

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

ISBN 978-80-247-5935-7 (ePub)

ISBN 978-80-247-5902-9 (pdf)

ISBN 978-80-271-5157-8 (print)

Obsah

Předmluva	9	
1	Několik slov, než začneme	12
1.1	Co je C++	12
1.2	Program: sada pokynů pro počítač	13
1.3	Co potřebuji vědět o počítači	13
1.4	.cpp	14
1.5	Neopakujme se	15
1.6	Knihovna	15
1.7	Projekt	15
1.8	Proč bych si měl vybrat jazyk C++	16
1.9	Jak vypadá program v C++	16
1.10	Vývojové prostředí a překladače	16
2	První program v C++	18
2.1	Prostředí OnlineGDB Beta	18
2.2	První program	19
2.3	Spouštíme první program	20
2.4	Když uděláme chybu	21
2.5	Co vlastně náš program dělá	22
2.6	Zdrojový text programu	23
2.7	Upravený program	24
2.8	Komentáře	25
3	Konverzace s programem	27
3.1	Kam si uložíme přečtené číslo	27
3.2	Jak přečteme číslo z konzole	28
3.3	Jak zapisujeme aritmetické operace	29
3.4	Jak vypíšeme celé číslo na konzoli	30
3.5	Ještě několik slov o celočíselných typech	30
3.6	Náš druhý program	31
3.7	Vylepšujeme náš druhý program	32

3.8	Něco je jinak	33
3.9	Uložíme si výsledek pro proměnné	34
3.10	Počítáme délku kružnice	35
3.11	Počítáme s reálnými čísly	36
3.12	Přiřazování	38
3.13	Grafická úprava programu	38
3.14	Jak tvoříme jména	40
3.15	Jméno a příjmení	40
3.16	Jména v C++, čeština a my	41

4	Složitější počítání	42
4.1	O co jde	42
4.2	Jak vypočítat součin předem neznámého počtu hodnot	42
4.3	Cyklus v C++	43
4.4	Program pro výpočet faktoriálu	44
4.5	Porovnávání hodnot	45
4.6	Stručnější zápis	46
4.7	Rozhodování: Co když uživatel zadá záporné číslo?	47
4.8	Upravený program (použijeme úplné if)	48
4.9	Jiná možnost (použijeme neúplné if)	49
4.10	Příliš velké číslo (složená podmínka)	49
4.11	Operace s logickými hodnotami	51
4.12	Magická čísla	52
4.13	Cyklus trochu jinak: příkaz for	55
4.14	Když uživatel zadá nesmysl	56
4.15	O čem jsme nehovořili	57

5	Dílčí algoritmy (funkce v C++)	59
5.1	Proč samostatná funkce	59
5.2	Jak programujeme funkce v C++	59
5.3	Funkce pro výpočet faktoriálu	60
5.4	Změníme pořadí funkcí	62
5.5	Funkce v samostatném zdrojovém souboru	64
5.6	Menší a větší ze dvou čísel	65
5.7	Hlavičkový soubor	66
5.8	Čtení vstupu a výstup výsledku jako samostatné funkce	67
5.9	Jak snadno změnit typ	68
5.10	Ochrana proti opakovanému čtení hlavičkového souboru	69
5.11	Maximum pro jiný číselný typ: přetěžování funkcí	73
5.12	Jak se vyhnout opakování: šablony funkcí	74
5.13	Měníme ve funkci skutečný parametr	77
5.14	Funkce volá sama sebe	79

6	Ladění programu	81
6.1	Příklad	81
6.2	Ladicí režim	82
6.3	Ladíme náš program: první kroky	83
6.4	Krokujeme funkci factorial	85
6.5	Program, který nekončí	86
6.6	Ladicí výpisy	88
6.7	Aserce	89
6.8	Když se aserce nehodí	92
7	Pole	93
7.1	Co je pole	93
7.2	Globální proměnná	94
7.3	Kolikátý den v roce je?	95
7.4	Program požádá o zadání data	97
7.5	Typ string	98
7.6	Vstup pomocí cyklu	99
7.7	Výčtový typ	99
7.8	Proměnná sdílená mezi zdrojovými soubory	101
7.9	Zpracování pole v cyklu	102
7.10	Znaky	104
7.11	Nulou ukončené řetězce	106
7.12	Zjišťujeme délku řetězce	107
7.13	Převádíme malá písmena na velká	108
7.14	Převádíme malá písmena na velká včetně češtiny	109
8	Vylepšujeme datum (od struktury k objektu)	112
8.1	Co je struktura	112
8.2	Datum a operace s ním	115
8.3	Třídy a objekty	120
8.4	Datum jako třída	121
8.5	Co jsme získali	128
9	Objektově orientované programování v C++	129
9.1	Objektový typ jako model	129
9.2	Statické složky třídy datum	132
9.3	Kopírování instancí	135
9.4	Kolik instancí třídy datum existuje	137
9.5	Ještě jednou počet instancí	138
9.6	Třída datum implementovaná pomocí pole	140
9.7	Příprava na kalendářní výpočty	143

10	Další zajímavé nástroje jazyka C++	152
10.1	Připravujeme se na kalendářní výpočty (přetěžujeme operátory) ...	152
10.2	Výjimky	158
10.3	Šablona třídy Datum	162
10.4	Omezení šablonových parametrů	165
11	Program na hádání zvířat	169
11.1	Ukazatel a adresa	169
11.2	Dynamické proměnné	172
11.3	Hádáme zvířata	173
11.4	Ukládáme strom do souboru	183
11.5	Čteme strom ze souboru	186
12	O čem jsme nehovořili	189
12.1	Jazyk C++	189
12.2	Knihovny	191
12.3	Co standardní knihovna C++ zatím nenabízí	195
	Odkazy	196
	Rejstřík	197

Předmluva

V této knize nabízím čtenářům seznámení s programovacím jazykem C++. Jak brzy zjistíte, je to krásný a mocný nástroj.

Co budete potřebovat

Odpověď je jednoduchá – kromě chuti seznámit se s programováním v C++ a trochy trpělivosti vlastně nic jiného než počítač připojený k internetu, neboť výklad je založen na volně dostupném prostředí OnlineGDB Beta, což je webová stránka dostupná na adrese [1]. Tato webová stránka funguje jako vývojový nástroj, ve kterém si lze zkusit programování v řadě programovacích jazyků, samozřejmě včetně C++.

Co v této knize najdete

Na rozdíl od běžných knih pro začátečníky se snažím ukázat C++ v celé jeho šíři. Samozřejmě začneme nejjednodušším programem a v duchu nejlepších tradic to bude program vypisující pozdrav, pak se seznámíme se základními konstrukcemi, jako jsou proměnné, vybrané příkazy a funkce. To nám zabere prvních pět kapitol. Potom se ale spolu podíváme i na pokročilé konstrukce, jako jsou šablony funkcí, objektové typy a jejich šablony, a zavedíme i o jednu z dlouho očekávaných novinek standardu jazyka z roku 2020, o omezení šablonových parametrů (koncepty). Za to ovšem zaplatíme, neboť výklad o náročnějších konstrukcích nebude podrobný, poskytne pouze první přiblížení.

Podívejme se stručně, jak budeme postupovat.

V první kapitole najdete rozhovor se čtenářem, který se zajímá o programování. Najdete v něm vysvětlení základních pojmů, které budeme v knize potřebovat.

Ve druhé kapitole se seznámíme s prostředím OnlineGDB Beta, napíšeme první program, spustíme ho a vysvětlíme si, co vlastně dělá. Dále si ukážeme, co se stane, když uděláme chybu. Ve třetí kapitole napíšeme program, který si od uživatele vyžádá zadání celého čísla a vypíše jeho dvojnásobek. Pak ho vylepšíme, takže si vyžádá zadání poloměru kružnice a vypíše její délku. To nám umožní seznámit se s celými a reálnými čísly v C++, s proměnnými a s některými dalšími konstrukcemi.

Ve čtvrté kapitole upravíme náš program, aby si vyžádal zadání celého nezáporného čísla a vypsal jeho faktoriál. To nám umožní seznámit se s cykly, s rozhodováním a s deklarací a používáním konstant. Poznáme také logické hodnoty a jejich použití v programu. V páté kapitole si ukážeme, jak naprogramovat relativně samostatnou část programu, kterou pak můžeme používat na více místech, tedy – jak říkáme v C++ – funkci. Ukážeme si také, jak rozdělit program na několik samostatně překládaných částí.

Šestá kapitola je věnována ladění, tedy hledání, proč náš program dělá něco jiného, než by měl, co jsme naprogramovali špatně. Seznámíme se nejen s nástroji pro krokování programu v prostředí OnlineGDB Beta, ale i s nástroji, které nám poskytuje jazyk C++, jako jsou aserce a ladící výpisy.

V sedmé kapitole si stanovíme úkol na základě zadaného data spočítat, kolikátý den v roce je. To nám umožní seznámit se s konstrukcí zvanou pole, s globálními proměnnými a s tzv. výčtovými typy. Poznáme také typy pro práci se znaky, typ string pro práci se znakovými řetězci a tzv. nulou ukončené řetězce. Nakonec si zkusíme převádět v programu malá písmena na velká a naopak.

V osmé kapitole se vrátíme k vyjádření kalendářního data v programu. Pokusíme se odstranit nedostatky, které způsobovalo použití pole, a to nás přivede nejprve k tzv. strukturám a od nich pak k objektovým typům.

V deváté kapitole se nejprve seznámíme se základními pojmy objektově orientovaného programování a pak upravíme a rozšíříme nástroj pro práci s datem, který jsme naprogramovali v předchozí kapitole. Naším cílem bude udělat z něj nástroj pro kalendářní výpočty, tedy například ke zjištění, kolik dnů uplynulo od přistání Kryštofa Kolumba u břehů Ameriky. Tyto úpravy dokončíme v desáté kapitole a přitom se seznámíme s přetěžováním operátorů, s nástroji pro obsluhu chyb (tzv. výjimkami), se šablonami objektových typů a s omezeními parametrů šablon.

V jedenácté kapitole si zadáme jiný úkol: Napsat program, který bude hádat zvířata. Přesněji program bude na počátku znát pouze dvě zvířata a znak, kterým se liší. Zeptá se uživatele, zda má hádané zvíře tento znak, a podle jeho odpovědi mu napíše, které zvíře určil. Pak se zeptá, zda je jeho odpověď správná, a když není, zeptá se ho, co to tedy je a čím se liší, a nové zvíře si zapamatuje. To nám umožní seznámit se s tzv. ukazateli a s dynamickými proměnnými. Vedle toho se naučíme zapsat data do souboru a při příštím spuštění je přečíst, takže program bude zvíře, které se nyní naučil, při příštím běhu znát.

V poslední kapitole najdete stručný přehled dalších možností jazyka C++, tedy nástrojů, na které se v této knize nedostalo.

Příklady

Výklad doprovází řada příkladů. Jejich zdrojové texty si můžete stáhnout na webových stránkách nakladatelství Grada Publishing (www.grada.cz, vyhledejte příslušnou sekci této knihy), nebo na mých webových stránkách; odkaz na ně najdete dále. V příkladech důsledně používám české identifikátory – na rozdíl od mých ostatních knih bez háčeků a čárek, neboť prostředí OnlineGDB Beta jejich použití nepodporuje. Víím, že profesionální programátoři se tomu vyhýbají a že v mezinárodních týmech jsou anglické identifikátory samozřejmostí. Mám ale dlouholetou zkušenost, že při výkladu určeném začátečníkům mohou české identifikátory výrazně usnadnit orientaci v ukázkách zdrojového kódu – a to je můj hlavní cíl. Má to ale ještě jednu přednost: V příkladech to umožňuje snadno rozlišit, co jsou třídy z knihovny a co naše vlastní, popřípadě které metody jsou zděděné po předchůdcích z knihovny a které jsou naše vlastní.

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 – a to je v knize určené naprostým začátečníkům velice důležité. Anglické termíny samozřejmě uvádím alespoň při prvním výskytu také.

Náročnější pasáže

Části textu, které jsou poněkud náročnější, jsou označeny čarou po straně; při prvním čtení je můžete přeskočit. Doporučuji však se k nim později vrátit, neboť mohou obsahovat velmi užitečné informace.

Obrázky

Většina obrázků v této knize představuje snímky okna editoru nebo konzolového okna vývojového prostředí OnlineGDB Beta. Pozadí těchto oken je černé, text v něm barevný, a prostředí OnlineGDB Beta to – alespoň v současné verzi – neumožňuje změnit. Protože však barevný text na černém pozadí při černobílém tisku bývá málo výrazný, dohodl jsem se s nakladatelstvím Grada Publishing, že pozadí změní na bílé. V textu knihy ovšem na několika místech zmiňuji černé pozadí. Věřím, že se tím nenecháte zmást.

Kde získat další informace

Při čtení této knihy mějte prosím na paměti, že představuje pouze seznámení s jazykem C++, takže výklad o většině témat je neúplný. Jestliže se po přečtení této knihy rozhodnete pokračovat a naučit se jazyk C++ do hloubky, můžete zkusit např. moji knihu *Programování v C++ od základů k profesionálnímu použití*, kterou vydalo nakladatelství Grada v roce 2018 [20]. Podrobné a přesné informace o základních konstrukcích jazyka, o jeho standardní knihovně a o mnoha dalších věcech najdete na stránkách cpreference.com [21].

Na závěr

I přes veškerou péči, kterou jsem této knize věnoval, se do ní mohly vloudit chyby. Jestliže zde nějakou najdete, dejte mi prosím zprávu na níže uvedenou adresu; bude-li to možné, uveřejním na svých webových stránkách opravu.

Praha 26. 6. 2023
Miroslav Vírúš
miroslav.virus@fjfi.cvut.cz
<http://people.fjfi.cvut.cz/virus>

1 Několik slov, než začneme

1.1 Co je C++

- Co se skrývá za názvem C++?
C++ je programovací jazyk, a to jeden z nejpoužívanějších.
- A co to je, programovací jazyk?
Programovací jazyk je jazyk, v němž zapisujeme *programy* – dnes s oblibou říkáme také *aplikace*, i když tohle slovo může znamenat skupinu programů, které fungují jako jeden celek. Program říká počítači, co má dělat, co od něj požadujeme.
- A jak program vypadá?
Program v C++ se obvykle podobá směsi anglických slov, občas obsahuje i zápisy podobné matematickým vzorcům. Tato slova a vzorce vyjadřují pokyny – příkazy –, které říkají počítači, co má udělat.
- To znamená, že počítač zná ona vybraná anglická slova a rozumí jim?
To je skoro pravda; skutečnost je ale o něco složitější. Každý počítač rozumí pouze jakémusi svému vlastnímu jazyku (který může být u různých typů počítačů různý) a my potřebujeme nástroj, který náš program nejprve z jazyka C++ (nebo z nějakého jiného programovacího jazyka) přeloží, aby mu daný počítač rozuměl. Tomuto nástroji se říká *překladač* neboli *kompilátor*.
- Takže když se chci seznámit s C++, musím si instalovat překladač C++?
To není třeba – alespoň pro první seznámení ne. Na webu je k dispozici několik portálů, na nichž můžeme napsat jednoduché programy, můžeme je online překládat a spouštět. Ovšem pro skutečné programování budeme potřebovat nejen překladač, ale i vhodné *vývojové prostředí*.
- Co je zas tohle?
To je složitější povídání. Už jsme si řekli, že program musíme nejdříve napsat ve zvoleném programovacím jazyce; napsaný program uložíme do souboru – tomu říkáme *zdrojový program* (v programátorštině, tedy v programátorském slangu, mu říkáme *zdroják*). K napsání programu potřebojeme textový editor. Pak zdrojový program přeložíme.

Přitom se může stát, že program nenapišeme správně, budou v něm chyby – to, co napíšeme, nebude správné C++ a překladač tomu nebude rozumět a nedokáže to přeložit. Může se ale také stát, že překladač náš program sice přeloží, ale ten bude dělat něco jiného, než chceme (uděláme chybu v popisu postupu, který chceme počítač naučit). V obou případech musíme chyby odstranit, musíme program *odladit*.

Vývojové prostředí (přesněji *integrované vývojové prostředí*; často se používá zkratka IDE z anglického *Integrated Development Environment*) je nástroj, který v sobě spojuje textový editor, překladač, nástroje pro spouštění a ladění programů a některé další pomůcky.

1.2 Program: sada pokynů pro počítač

- Program je tedy nějaká sada pokynů pro počítač?

Ano, to je správně.

- Ale to znamená, že když chci napsat program, musím úlohu, kterou bude tento program řešit, umět vyřešit sám?

Přesně tak. A nejen to: Musíme ji umět přepsat do řeči počítače.

- Přeložit návod do nějakého jazyka, to snad není tak těžké.

Je a není. Ve skutečnosti je to podobné, jako když to, co budu chtít naučit počítač (tedy co budu chtít naprogramovat), chci naučit třeba kamaráda. Musím to rozložit na kroky, tedy dílčí operace, kterým kamarád – nebo počítač – rozumí. V případě počítače ty kroky v podstatě odpovídají jednotlivým příkazům programovacího jazyka.

- To znamená, že si požadovaný úkol musím přepsat po jednotlivých operacích, které počítač umí?

Ano. Musíme si sestavit – na papíře, nebo spíše v nějakém editoru – posloupnost kroků, které bude počítač dělat, aby zadaný úkol vyřešil. Takovému návodu říkáme také *algoritmus*. Tento návod pak přepíšeme do C++ a tak vznikne program.

- A je nějaký doporučený postup, jak ten algoritmus vytvořit?

Nejobvyklejší je postup *shora dolů*. Když známe řešení dané úlohy, napíšeme je jako posloupnost kroků, kterým rozumíme. Pak se podíváme, zda jim bude rozumět i počítač – nebo lépe zda je budeme umět naprogramovat. Když ne, rozložíme tyto kroky na menší a tento postup budeme opakovat, dokud nedospějeme na úroveň kroků, které naprogramovat dokážeme.

- Takže na program se můžeme dívat jako na zápis algoritmu v C++?

I tak to lze říci.

1.3 Co potřebuji vědět o počítači

- Když chci zkusit programovat, co potřebuji vědět o počítači? Občas slychám něco o velikosti paměti, o nějakých megabajtech, gigabajtech a tak.

Dnešní počítač je typicky složen ze čtyř základních součástí. Jednou z nich je *procesor* – to je část, která opravdu počítá. Druhou z nich je *operační paměť*. Tam si počítač ukládá program, data, s nimiž bude počítat, mezivýsledky, výsledky atd. Ovšem obsah operační paměti se při vypnutí počítače ztratí, a proto mají počítače také *trvalou* – *permanentní* – pa-

měť. Donedávna to zpravidla byly magnetické disky, dnes je zvolna nahrazují paměti SSD, které jsou konstrukčně podobné jako např. populární „flešky“. Poslední důležitou součástí počítače jsou vstupní a výstupní zařízení – obrazovka, klávesnice, USB porty apod., jejichž prostřednictvím počítač komunikuje s okolím.

■ Prima. A ty gigabajty?

Ty vyjadřují velikost paměti. Paměť dnešních počítačů se skládá z tzv. *bitů* – to jsou místa, na kterých může být uložena buď nula, nebo jednička, nic jiného. To vypadá jako velké omezení, ale není – pomocí nul a jedniček lze zapsat jakékoli číslo – a nejen data, ale i příkazy pro počítač jsou vyjádřeny jako čísla. (Počítač ve skutečnosti nerozumí ničemu jinému než číslům.)

Bity jsou na dnešních počítačích sdruženy do skupin po osmi. Tyto skupiny označujeme *bajty* (anglicky *byte* = slabika). Snadno se přesvědčíte, že osm nul nebo jedniček může vytvořit 256 různých kombinací čili že do jednoho bajtu lze zapsat 256 různých hodnot, které mohou představovat např. čísla od 0 do 255 nebo čísla od -128 do +127. To je ovšem velice málo, a proto se často pracuje se skupinami složenými ze 2, 4 nebo 8 bajtů. Do skupiny 8 bajtů už můžeme uložit 18 triliónů různých hodnot.

Paměť je tedy vlastně dlouhá řada bajtů. Ty jsou očíslovány, každý bajt má své pořadové číslo, kterému říkáme *adresa*.

Názvy jako kilobajt, megabajt atd. se používají pro vyjádření velikosti úseku paměti (nebo celé paměti) a mají podobnou logiku jako názvy kilometr, megahertz apod., i když ne úplně stejnou. Jeden kilobajt, 1 kB, je 1 024 bajtů (tedy 2^{10} B), jeden megabajt je 1 048 576 bajtů (tedy 2^{20} B) atd.¹

1.4 .cpp

■ Vraťme se k programování v C++. Je obvyklé, že typy souborů lze rozlišit podle přípony – soubory s obrázky mívají např. příponu `.jpg` nebo `.png`, soubory se zvukovým záznamem zase např. `.mp3`. Jak poznám zdrojový soubor v C++? Mají takovéto soubory také nějakou typickou příponu?

Pod Windows a v některých dalších prostředích mají zdrojové soubory v jazyce C++ zpravidla příponu `.cpp`. V některých jiných prostředích, například v některých verzích Unixu, mohou mít příponu `.C` (velké C). Přípona `.c` (malé c) je zpravidla vyhrazena pro příbuzný jazyk C.

Jak ale uvidíme, některé speciální zdrojové soubory mohou mít příponu `.h` nebo mohou být bez přípony.

¹ Tak to běžně – ale nesprávně – používáme. Ve skutečnosti 1 kilobajt – 1 kB – je 1 000 B a 1 024 B je kibibajt, 1 KiB. Podobně 2^{20} B je 1 mebibajt, 1 MiB atd. Viz též [6]. V běžné řeči však lidé od počítačů používají označení kilobajt apod. ve významu, který jsme si řekli v hlavním textu.

1.5 Neopakujme se

- Ještě jednou bych rád zopakoval: Program je tedy soubor s příponou `.cpp` nebo podobnou, který obsahuje posloupnost příkazů pro počítač?

V prvním přiblížení to tak je. Ve skutečnosti však program zpravidla rozložíme na několik částí, které představují dílčí postupy, jako je příprava uživatelského rozhraní programu, zjištění požadavků uživatele, jejich zpracování, uložení nebo vypsání výsledků atd.

- Představme si takovouto situaci. V programu budu potřebovat na sedmi místech zapísat data do souboru a na dvou místech data ze souboru přečíst. Budu muset programovat zápis dat sedmkrát a čtení dvakrát?

Ne. Jazyk C++ nám umožňuje naprogramované dílčí postupy pojmenovat a pak se na ně odvolávat. Tyto pojmenované dílčí postupy se v programech v C++ nazývají *funkce*.

- To znamená, že počítač vlastně naučíme nové kroky, které předtím neuměl a které po něm nyní můžeme požadovat? Funkce tedy představuje takový dílčí krok?

Přesně tak. To je ve skutečnosti základní programátorské pravidlo: *NEOPAKUJTE SE*. Když už jednou něco naprogramujeme, nebudeme to programovat znovu, budeme se na to už jen odvolávat.²

1.6 Knihovna

- Dobře, ale mohu použít něco, co jsem naprogramoval v jednom programu, v jiném? Nebo platí zásada neopakování jen v jednom programu?

Zásada neopakování platí samozřejmě i mezi různými programy. Když už něco naprogramujeme, můžeme si uschovat zdrojový text a příště ho použít. Můžeme si také uschovat přeloženou část programu a pak ji připojovat podle potřeby k dalším programům – můžeme si vytvořit vlastní programovou *knihovnu*.

Ve skutečnosti je řada běžných dílčích postupů, jako je už zmíněný zápis dat do souboru, čtení dat ze souboru, výpočet hodnot často používaných matematických funkcí a mnoho a mnoho dalších, součástí tzv. *standardní knihovny jazyka C++*, takže je nemusíme programovat, můžeme se na ně odvolávat jako na kroky, které počítač už zná.

1.7 Projekt

- Zdrojový kód programu zapíšu v textovém editoru, který je součástí integrovaného vývojového prostředí, a uložím ho do souboru. Co když ale bude opravdu program rozsáhlý? Někde jsem četl, že kód jistého komerčního programu – tedy asi jeho zdrojový kód – měl několik miliónů řádek. Jak se v tom pak má člověk orientovat? Je možné rozdělit jeden program do několika zdrojových souborů?

Jistě, a v této knize si to ukážeme. Skupině souborů, které tvoří jeden program, se obvykle říká *projekt* a integrovaná vývojová prostředí ho umějí spravovat – vytvořit ho, přidávat do něj nové soubory, přeložit jednotlivé soubory i celý projekt atd.

² Anglicky *Don't Repeat Yourself*, ve zkratce DRY; někdy to bývá žertem překládáno jako „suchý princip“.