

David Matoušek

C++

BEZ
PŘEDCHOZÍCH
ZNALOSTÍ



Datové typy, operátory, řídicí struktury
Funkce, ukazatele, odvozené datové typy
Pole, řetězce, vstup a výstup programu
Práce se soubory, obsluha výjimek

computer
press®

David Matoušek

C++ bez předchozích znalostí

**Computer Press
Brno
2016**

C++ bez předchozích znalostí

David Matoušek

Obálka: Martin Sodomka

Odpovědný redaktor: Martin Herodek

Technický redaktor: Jiří Matoušek

Objednávky knih:

<http://knihy.cpress.cz>

www.albatrosmedia.cz

eshop@albatrosmedia.cz

bezplatná linka 800 555 513

ISBN 978-80-251-4640-8

Vydalo nakladatelství Computer Press v Brně roku 2016 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 23 292.

© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

1. vydání

ALBATROS  **MEDIA** a.s.

Obsah

Předmluva	13
Zpětná vazba od čtenářů	14
Zdrojové kódy ke knize	15
Errata	15
KAPITOLA 1	
Úvod do programování v jazyce C++	17
Základní pojmy	17
Proměnné a konstanty	18
Typy příkazů	18
IDE – integrované vývojové prostředí	19
IDE Dev-C++	19
Stažení a instalace Dev-C++	19
První program	25
Klíčové položky nabídky	27
Překlad programu	28
Stručné vysvětlení zápisu programu	30
Pár zajímavostí	31
Komentáře neboli poznámky	31
Pomocné nástroje na Internetu	31
Rozdělení základních datových typů	31
KAPITOLA 2	
Celočíselné datové typy	33
Celá čísla se znaménkem a bez znaménka	33
Celá čísla bez znaménka	34
Celá čísla se znaménkem	35
Charakteristiky celočíselných datových typů	35
Základní vstupně/výstupní operace	36
Základní výstupní operace	36
Základní vstupní operace	40

Pokročilejší operace s proměnnými a proudy	40
Deklarace proměnné	40
Výstupní manipulátory dec, hex, oct	41
Aritmetické operace s celými čísly	42
Základní aritmetické operátory	42
Unární aritmetické operátory	43
Priorita a asociativita	44
Zadávání číselných literálů v různých soustavách	45

KAPITOLA 3

Datové typy pro reálná čísla	47
Vlastnosti datových typů pro reálná čísla	47
Vstupně/výstupní operace z pohledu reálných čísel	49
Aritmetické operace s reálnými čísly	51
Přípony pro rozlišení literálů reálných čísel	52
Implicitní a explicitní typové konverze	52
Implicitní typové konverze	52
Možné problémy implicitních převodů	54
Když implicitní převod nestačí	54
Explicitní typová konverze	55
Priorita a asociativita dosud probraných operátorů	56

KAPITOLA 4

Větvení programu	59
Konstrukce logických výrazů	59
Typ bool	59
Relační operátory (operátory pro porovnání)	59
Logické operátory	60
Priorita a asociativita	61
Vývojové diagramy	62
Příklad	62
Podmíněný příkaz if	63
Základní varianta (bez větve při nesplnění podmínky)	63
Varianta s příkazy v obou větvích	64
Varianta s další podmínkou v záporné větvi	64
Používání bloků	65

Složitější větvení	66
Podmíněný příkaz switch	68
KAPITOLA 5	
Cykly	71
Cyklus while – cyklus s podmínkou na začátku	71
PROG_01 – výpis řady čísel	72
Cyklus do..while – cyklus s podmínkou na konci	73
PROG_02 – výpis řady čísel pomocí cyklu do..while	73
Ošetření chybného zadání z klávesnice	74
Cyklus for – cyklus s určeným počtem opakování	77
PROG_04 – výpis řady čísel pomocí cyklu for	78
Break – předčasné ukončení cyklu	79
Continue – vynechání jednoho kroku cyklu	80
KAPITOLA 6	
Pole	83
Deklarace pole	83
Vlastnosti polí v jazyce C++	84
Inicializace prvků pole	84
Základní operace s poli	85
Konstanty	86
Příklady	87
Základní operace s jednorozměrným polem	87
Základní operace s „dvourozměrným“ polem	89
Míříme k funkcím!	90
KAPITOLA 7	
Funkce	91
Základy používání funkcí	91
Výhody používání funkcí:	91
Obecný zápis funkce	91
Předávání parametrů hodnotou	92
Návratová hodnota	93
Typ void	93

Příklady	94
Funkce min	94
Funkce pro práci s poli	95
Dopředná deklarace funkce	97
Základní knihovní funkce jazyka	99
Matematické funkce	99
Funkce pro práci se znaky	100
Další užitečné funkce	100
Globální a lokální data	100
KAPITOLA 8	
Datový typ ukazatel	103
Deklarace proměnné typu ukazatel	103
Reference proměnné	103
Dereference ukazatele	104
Další informace k ukazatelům	106
Ukazatel void*	106
Hodnota NULL	106
Velikost ukazatele	106
Ukazatel na ukazatel	106
Nové operátory a jejich priorita a asociativita	107
Dynamická alokace paměti	107
Operátor new	108
Operátor delete	108
Příklad	108
KAPITOLA 9	
Používání ukazatelů	111
Předávání parametrů funkce přes ukazatel – výstupní parametry	111
Předávání parametrů funkce odkazem – výstupní parametry podruhé	112
Deklarace proměnné typu odkaz (reference)	112
Ukazatelová aritmetika	114
Přetypování ukazatele na logickou hodnotu	115
Přetypování ukazatele na celé číslo	115
Souvislost ukazatele a pole	116
Problémy s používáním polí ve funkcích	117

KAPITOLA 10

Znaky	121
Datový typ char	121
Funkce pro práci se znaky	123
Vstup a výstup znaků	126
Použití funkce system	126
Vstup znaků přes vstupní proud cin pomocí extraktoru	127
Vstup znaků přes vstupní proud cin pomocí metody get	129
Vstup znaků pomocí funkcí z knihovny conio.h	130

KAPITOLA 11

Řetězce	133
Datový typ char*	133
Řetězcové literály	134
Deklarace spojená s inicializací	134
Operace	134
Funkce pro práci s řetězci	138
Vstup a výstup řetězců	142
Vstup řetězců	142
Výstup řetězců	143
Objektová podpora řetězců	144

KAPITOLA 12

Odvozené datové typy	145
Definice nového datového typu	145
Přehled datových typů	145
Datový typ enum – výčet	146
Další vlastnosti výčtu:	147
Datový typ struct – sktruktura	147
Další vlastnosti struktury:	148
Datový typ union – sjednocení (unie)	151
Datový typ bitové pole	154
Datový typ class – třída	156

KAPITOLA 13

Operátory	157
Rozdělení operátorů	157
Rozdělení operátorů podle počtu operandů	157
Rozdělení operátorů podle typu operace	158
Ternární operátor	158
Bitové operátory	158
Bitová negace ~	158
Bitový součet	159
Bitový součin &	159
Výlučný bitový součet ^	159
Posuv vlevo <<	160
Posuv vpravo >>	160
Příklad použití	160
Operátory přiřazení	161
Operátor čárka (operátor zapomenutí)	162
Souhrnná tabulka priority a asociativity operátorů	163
Přetěžování operátorů	163

KAPITOLA 14

Direktivy, paměťové třídy, modulární programování	165
Direktivy (příkazy preprocesoru)	165
#include (česky zahrnout)	165
#define (česky definovat)	166
#if, #else, #elif, #ifdef, #ifndef, #endif (řízení překladu)	167
#pragma pack (zarovnání)	167
Paměťové třídy	168
Auto (automatická proměnná)	168
Register (registrová proměnná)	169
Static (statická proměnná)	170
Příklad	170
Extern (externí ≡ vnější proměnná)	171
Modulární programování	171
Používané pojmy:	172
Příklad	172

KAPITOLA 15

Přetížení funkcí a implicitní parametry funkcí	177
Přetížení funkcí	177
Přetížení funkce pomocí typů parametrů	177
Přetížení funkce pomocí počtu parametrů	179
Implicitní parametry funkcí	180
Dopředná deklarace a implicitní parametry funkce	182

KAPITOLA 16

Základy objektově orientovaného programování	183
Definice třídy	184
Třída TClovek – 1. varianta (základní)	185
Problematika zapouzdření a inline metody	188
Třída TClovek – 2. varianta (zapouzdření a inline metody)	188
Konstruktory	190
Třída TClovek – 3. varianta (s parametrickým konstruktorem)	191
Jak funguje standardní kopirovací konstruktor	192
Destruktor	193
Třída TClovek – závěrečná varianta	194
Dědičnost – základní informace	198
Krátký příklad na vysvětlení základů dědičnosti	199
Změna přístupových úrovní při dědění	202

KAPITOLA 17

Přetěžování operátorů, výjimky	203
Přetěžování operátorů	203
Přetížení operátoru přiřazení	203
Přetížení insertoru	205
Výjimky	207
Výjimka je třída aneb hierarchie standardních výjimek	208
Syntaxe	208
Příklad – vylepšení třídy TClovek	209

KAPITOLA 18

Proudová knihovna a práce se soubory	213
Hierarchie proudů	213

Standardně deklarované proudy	213
Metody proudů ios, istream a ostream	214
Souborové proudy	216
Otevření souboru	217
Zavření souboru	217
Test úspěšnosti operace	218
Příklady	218
PROG_01 – Zápis čísel do souboru	218
PROG_02 – Čtení čísel ze souboru	220
PROG_03 – Práce s binárním souborem	221
KAPITOLA 19	
Třída string	227
Stručný popis	227
Konstruktory	227
Operátory	228
Vybrané metody	229
Příklad	230
KAPITOLA 20	
Parametry a návratová hodnota programu	233
Parametry argc a argv	233
Návratová hodnota	235
Program na kopírování souborů	235
PŘÍLOHA A	
Číselné soustavy a reprezentace čísel	239
Jednotky informací	239
Číselné soustavy	240
Hornerovo schéma	240
Dvojková soustava	240
Šestnáctková soustava	240
Reprezentace celých čísel v paměti počítače	241
Celá čísla bez znaménka	241
Celé čísla se znaménkem	241
Uložení vícebajtových hodnot do paměti	243

Reprezentace čísel v plovoucí řádové čárce v paměti počítače	244
Standard IEEE 754	244
Logické operace	245
NOT – logická negace (inverze)	245
AND – logický součin	245
OR – logický součet	246
XOR – výlučný logický součet	246
Souvislost s jazykem C++	246
PŘÍLOHA B	
Popis vývojového prostředí Dev-C++	247
Položky nabídky	247
Soubor	247
Editace	248
Hledat	249
Zobrazit	249
Projekt	251
Spustit	256
Nástroje	258
AStyle	260
Okna	260
Nápověda – klasické položky nápovědy (bez komentáře)	261
Ukázka ladění programu	261
Slovo závěrem	263
Seznam doporučené literatury pro další studium	264
Rejstřík	265

Předmluva

Knih, kterou právě držíte v ruce, je určena zájemcům, kteří se chtějí naučit programovat v jazyce C++. Provádí čtenáře od začátků programování až k pokročilým záležitostem, jako je objektově orientované programování a práce se soubory. Text je doplněn četnými příklady.

Prakticky se pracuje s vývojovým prostředím **Dev-C++**, které bylo zvoleno především s ohledem na nízké nároky při instalaci, dobrou lokalizaci do češtiny a relativně snadné ovládání. Prostředí je volně k dispozici, je poskytováno na základě GNU licence.

V úvodní kapitole jsou vysvětleny základní pojmy programování, projdeme instalaci vývojového prostředí (IDE) Dev-C++ a sestavíme první program. Nakonec se seznámíme se základním dělením datových typů.

Kapitoly 2 a 3 jsou věnovány datovým typům pro celá a reálná čísla. Jsou také vysvětleny základní vstupně/výstupní operace (načítání čísel z klávesnice a jejich výpis na obrazovku) a operátory. Pozornost je věnována i problematice tzv. konverzí (převodů hodnot různých datových typů).

Kapitoly 4 a 5 jsou věnovány větvení programu pomocí příkazů `if` a `switch` a konstrukci cyklů typu `while`, `do...while` a `for`. Použita jsou i související klíčová slova `break` a `continue`.

Kapitola 6 vysvětluje pojem pole, základní operace s poli a směřuje výklad k používání funkcí.

Kapitola 7 ukazuje používání funkcí (tedy podprogramů) za účelem přehlednější tvorby programů v jazyce C++. Je vysvětleno předávání parametrů hodnotou a návratová hodnota. Poté je vytvořena skupina funkcí pro práci s poli a jsou uvedeny základní knihovní funkce (tedy již hotové funkce). Závěr je věnován otázce globálních a lokálních proměnných.

Kapitoly 8 a 9 jsou věnovány ukazatelům. Jsou vysvětleny pojmy reference a dereference, dynamická alokace paměti (operátory `new` a `delete`), předávání parametrů přes ukazatel a předávání parametrů odkazem. Jsou také zařazeny pokročilejší záležitosti, jako jsou ukazatelová aritmetika a souvislost ukazatele s polem.

Kapitoly 10 a 11 se zaměřují na znaky a řetězce. Je vysvětlen způsob reprezentace znaků a řetězců, jsou předvedeny základní knihovní funkce pro práci s těmito typy dat společně se vstupně/výstupními operacemi.

Kapitola 12 popisuje odvozené datové typy. Kromě úvodního rozdělení datových typů jsou postupně probírány nové datové typy: výčet, struktura, sjednocení a bitové pole. Datový typ třída je popsán samostatně až v kapitole 16.

Kapitola 13 je zařazena jako shrnutí operátorů. Doplněny jsou informace k dalším operátorům, jako jsou ternární operátor, bitové operátory, operátory přiřazení, operátor čárka a na závěr je uvedena přehledová tabulka priority a asociativity všech operátorů.

Kapitola 14 představuje modulární programování, tedy tvorbu programů založenou na více zdrojových souborech (používá se pro tvorbu rozsáhlejších programů). Souběžně s tím jsou vysvětleny direktivy překladu a paměťové třídy proměnných.

Kapitola 15 vysvětluje pojmy přetížení funkcí a implicitní parametry funkcí.

Kapitola 16 je úvodem do objektově orientovaného programování (OOP). Vysvětluje jeho výhody a základní rysy na postupně budovaném příkladu třídy `TC1ovek`. Jsou tak postupně vysvětleny pojmy: atribut, metoda, zapouzdření, přístupové úrovně, konstruktor, destruktork. Rovněž se seznámíme s pokročilejšími termíny, jako je mělká a hluboká kopie nebo statické a konstantní členy třídy. Na závěr je pak vysvětlen pojem dědičnost.

Kapitola 17 ukazuje možnosti přetěžování operátorů (možnost definovat vlastní význam operace pro danou třídu) a použití výjimek (řeší havarijní stavy programu).

Kapitola 18 popisuje další možnosti proudové knihovny a ukazuje použití proudů pro ovládání souborů. Kromě textových souborů je ukázána i práce s binárními soubory.

Kapitola 19 je věnována třídě `string`, která zjednodušuje práci s řetězci.

Kapitola 20 se zaměřuje na „napojení“ programu na zbytek operačního systému. Řeší parametry příkazové řádky programu a návratovou hodnotu programu. Vše je předvedeno na příkladu programu pro kopírování souborů. Příklad používá pokročilejší práci se soubory včetně datového bufferu.

Dále jsou zařazeny dvě přílohy. Příloha A vysvětluje číselné soustavy a reprezentaci čísel v počítači. Jedná se o vysvětlení základních a násobných jednotek informace, převody mezi číselnými soustavami, reprezentaci celých čísel bez znaménka a se znaménkem a reprezentaci čísel v plovoucí řádové čárce. Dále jsou vysvětleny základní logické operace. Příloha B je věnována stručnému popisu vývojového prostředí Dev-C++ (popis položek nabídky) a také ukazuje ladění programu na krátkém příkladu.

Zpětná vazba od čtenářů

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

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Zdrojové kódy ke knize

Z adresy <http://knihy.cpress.cz/K2237> si po klepnutí na odkaz Soubory ke stažení můžete přímo stáhnout archiv s ukázkovými kódy.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatelé tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese <http://knihy.cpress.cz/K2237> po klepnutí na odkaz Soubory ke stažení.

Úvod do programování v jazyce C++

V této kapitole:

- Základní pojmy
- IDE – integrované vývojové prostředí
- První program
- Pár zajímavostí
- Rozdělení základních datových typů

V této úvodní kapitole se seznámíme s úplnými základy programování, které jsou společně nejen pro programovací jazyk C++, ale pro všechny programovací jazyky. Budou vysvětleny základní pojmy (termíny), které budeme v dalším textu používat. Vzhledem k obsáhlosti celé problematiky budeme výklad pojmů „dávkovat“ a s dalšími (složitějšími) termíny se budeme seznamovat postupně v následujících kapitolách.

Základní pojmy

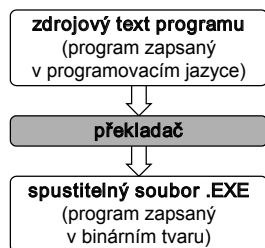
Než začneme psát jakýkoli program, musíme si provést rozbor řešeného problému. Pokud tuto fázi vynecháme („To je přeci otrava – přemýšlet, lepší je začít psát program!“), může se nám lehce stát, že tvorbou programu strávíme mnohem více času, než jsme původně předpokládali.

Následně pak dojdeme k tomu, jaké činnosti má počítač provádět a v jakém pořadí. Přesný návod, jak vyřešit daný typ úlohy, se nazývá **algoritmus**.

Pro zápis algoritmu do počítače používáme speciální **programovací jazyk**. Programovací jazyk musí být sestaven tak, aby bylo možno snadno vyjadřovat algoritmy a aby byl blízký lidskému uvažování.

Počítačový **program** vznikne tak, že algoritmus vyjádřený v programovacím jazyce převedeme do tvaru spustitelného na počítači. Přejít od algoritmu zapsaného v programovacím jazyce k programu spustitelnému na počítači se nazývá **překlad** a provádí jej speciální program označovaný jako **překladač** (kompilátor). Překladem se zdrojový text (je zapsaný v programovacím jazyce) převede do tzv. binární formy, což je vlastně sled instrukcí, které bude provádět procesor počítače.

Činnost, kterou počítač provádí při vykonávání programu, označujeme jako **proces**. Označení proces je na místě, přetváří vstupní údaje na výstupní.



Obrázek 1.1. Průběh překladač (zjednodušeno)

Pro správný zápis programu pomocí programovacího jazyka musíme dodržet syntaxi (věcnou správnost) a sémantiku (význam konstrukcí):

- **Syntaxe** popisuje, z čeho se může skládat zápis programu. Určuje například tzv. *klíčová slova*. Nemůžeme tedy použít slova, která daný programovací jazyk nezná. Syntaktická kontrola pak znamená, že se kontroluje dodržení korektnosti zdrojového textu (tedy „dodržení pravopisu“).
- **Sémantika** přiřazuje jednotlivým konstrukcím jazyka význam. Je jistě možné napsat syntakticky správný program, který však nemá správný smysl.

Syntaxe + sémantika = správně fungující program

Proměnné a konstanty

Předměty, se kterými program pracuje (čte jejich hodnoty nebo je mění), nazýváme **data**. Může se jednat o čísla, znaky apod.

Většina dat obvykle během provádění programu mění svůj obsah, proto je označujeme jako **proměnné**. Data, jejichž hodnota se v průběhu provádění programu nemění, označujeme jako **konstanty**.

Typy příkazů

Zápis programu v programovacím jazyce se skládá z popisu použitých dat (tzv. deklarace) a z jednotlivých příkazů. Nejčastějšími variantami příkazů jsou:

- **deklarace proměnné** – nahlášení datového typu a názvu proměnné, kterou budeme používat (překladač pak může sledovat, jestli s proměnnými provádíme správné operace, a kontroluje, zda má dost paměti),
- **příkaz vstupu** – příkaz, který zajistí načtení dat například z klávesnice nebo jiného vstupního zařízení (například ze souboru),

- **příkaz výstupu** – příkaz, který vypíše data například na obrazovce nebo je uloží do jiného výstupního zařízení (například do souboru),
- **přiřazovací příkaz** – příkaz, který přiřadí do proměnné novou hodnotu,
- **větvení** – příkaz, který podle určité podmínky rozdělí další postup do více cest,
- **volání podprogramu** – příkaz, který provede dříve vytvořenou část programu; takto můžeme například provést výpočet funkce $\sin(x)$, pokud je tento podprogram v daném jazyce k dispozici; rovněž můžeme vytvářet vlastní podprogramy (funkce).

IDE – integrované vývojové prostředí

V souvislosti s programováním se zkratka IDE objevuje s anglickým souslovím Integrated Development Environment. Tedy doslovně přeloženo jako „integrované vývojové prostředí“.

IDE znamená pro programátora komfort, kdy v rámci jediného programu (přesněji aplikace) zapíše zdrojový text, nastaví (pokud je to nutné) parametry překladu, provede překlad, testuje a ladí činnost hotového programu.

Existuje řada překladačů jazyka C++, které však vyžadují další znalosti uživatele, což je pro začátečníky nevhodné. Totiž, pokud nemáme k dispozici IDE, musíme napsat zdrojový text například v poznámkovém bloku a poté spustit překlad pomocí příkazové řádky obvykle s použitím speciálního dávkového souboru. Možnosti ladění jsou v takovém případě velmi omezené.

IDE Dev-C++

Vývojových prostředí existuje celá řada. K nejznámějším patří **Microsoft Visual Studio** od Microsoftu nebo **C++ Builder** od firmy Embarcadero (původně Borland).

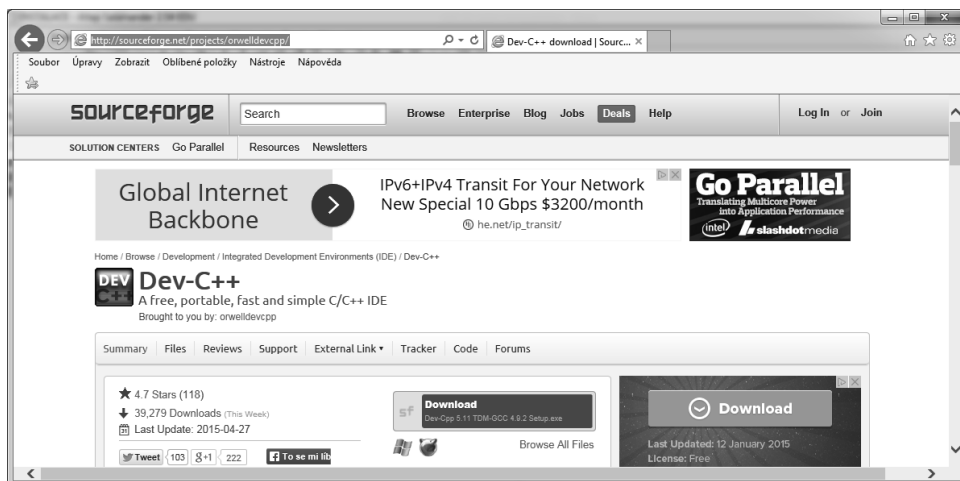
My jsme pro tuto knihu vybrali vývojové prostředí **Dev-C++**, které je vyvíjeno na základě licence GNU pod hlavičkou Bloodshed Software. Důvody byly: dobrá znalost tohoto prostředí, časově neomezená licence pro nekomerční použití, nízké nároky při instalaci a částečná lokalizace prostředí do češtiny (většina příkazů hlavní nabídky je v češtině, ale například chyby překladu jsou hlášeny v angličtině).

Vývojové prostředí Dev-C++ lze stáhnout ze stránek <http://www.bloodshed.net/dev/devcpp.html>.

Instalační soubor verze 5.9.2 z února 2015 měl velikost okolo 45 MB (tedy mnohem méně než ostatní výše uváděné produkty).

Stažení a instalace Dev-C++

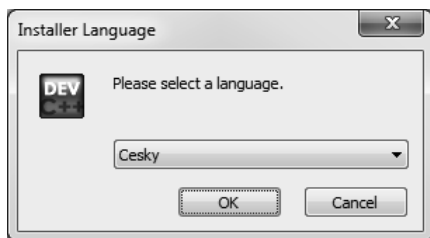
Odkaz pro stažení instalace Dev-C++ je možné najít na výše uvedených stránkách (<http://sourceforge.net/projects/orwelldevcpp/>) nebo lze použít přímý odkaz (jak také ukazuje obrázek 1.2).



Obrázek 1.2. Stažení instalačního programu ze stránek SourceForge

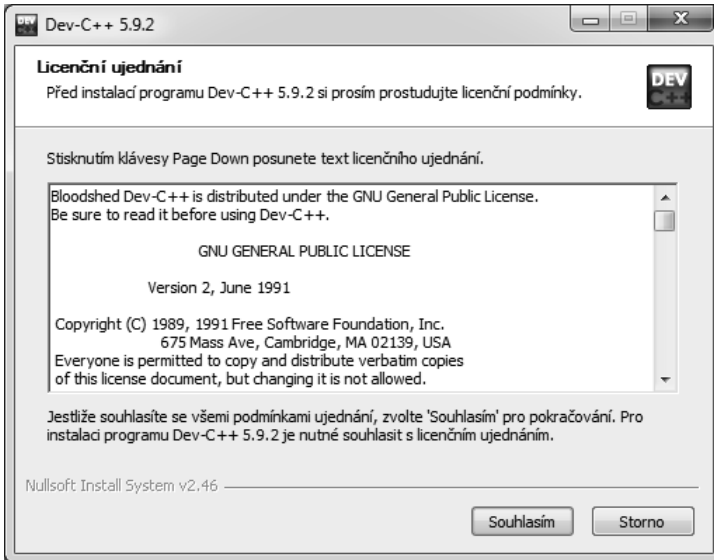
Instalační soubor stáhneme pomocí tlačítka **Download**.

Spustíme instalaci pomocí souboru s názvem (název může být upraven při stažení novější verze): **Dev-Cpp 5.9.2 TDM-GCC 4.8.1 Setup.exe**. Nejdříve je provedena dekomprese a následně se zobrazí dialog pro volbu jazyka instalace dle obrázku 1.3. Vybereme **Cesky** a potvrdíme stiskem tlačítka **OK**.

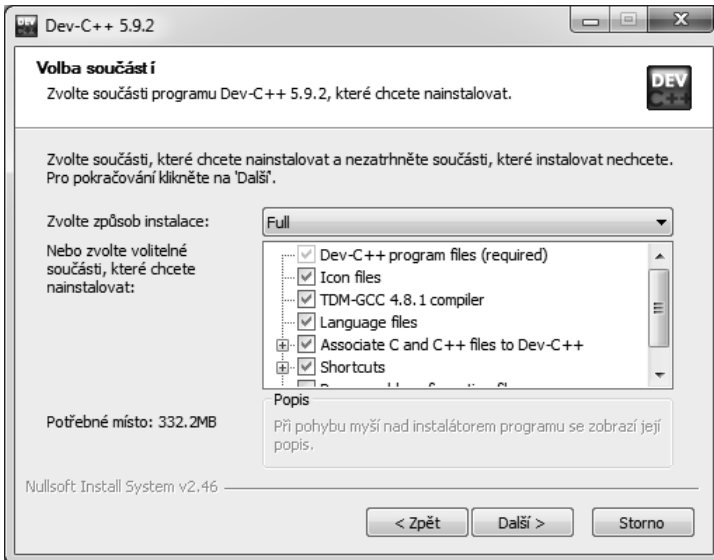


Obrázek 1.3. Volba jazyka instalace

Následuje dialog s licenčním ujednáním dle obrázku 1.4, který je nutné potvrdit stiskem tlačítka **Souhlasím**.



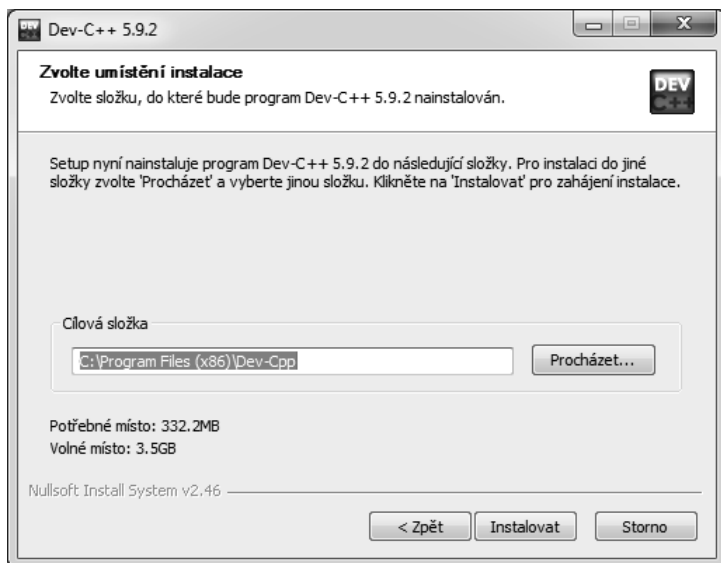
Obrázek 1.4. Dialog s licenčním ujednáním



Obrázek 1.5. Volba součástí pro instalaci

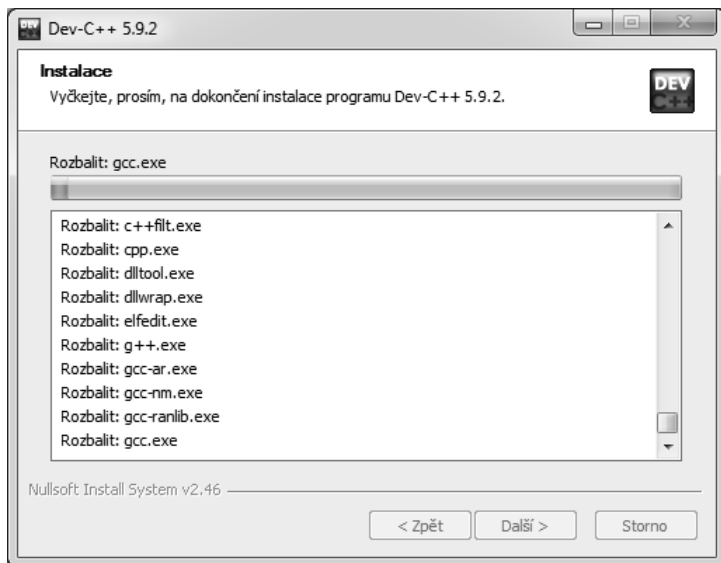
Následuje dotaz na výběr součástí pro instalaci dle obrázku 1.5. Zde kromě jiného dochází k asociaci s koncovkami souborů, které se používají v jazyce C++. Doporučujeme ponechat výchozí nastavení (vše vybrané) a poté pokračovat stiskem tlačítka **Další**. V této verzi je pro instalaci třeba asi 330 MB diskového prostoru.

Nakonec je třeba vybrat adresář, do kterého bude instalace provedena, viz obrázek 1.6. Doporučujeme ponechat výchozí volbu a pokračovat stiskem tlačítka **Instalovat**.



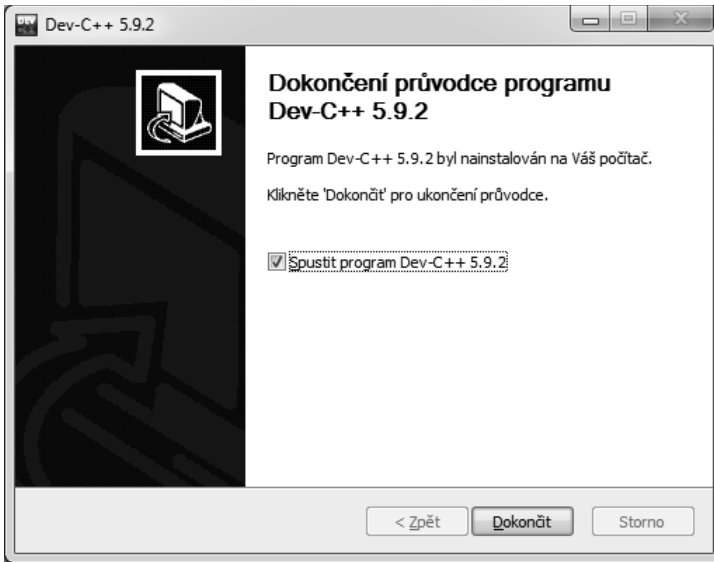
Obrázek 1.6. Volba adresáře pro provedení instalace

Poté se již rozbíhá instalace, jak dokumentuje obrázek 1.7.



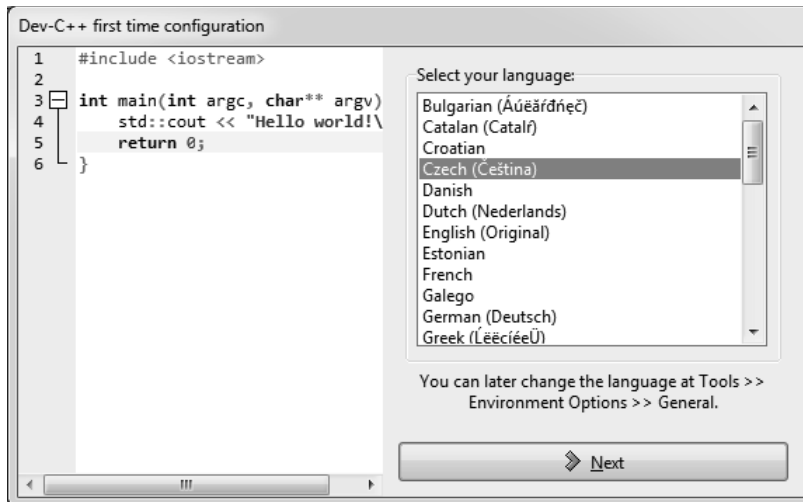
Obrázek 1.7. Průběh vlastní instalace

Na závěr se zobrazí dialog dle obrázku 1.8, který umožňuje přímé spuštění vývojového prostředí. Volbu **Spustit program Dev-C++** necháme aktivní a stiskneme tlačítko **Dokončit**.

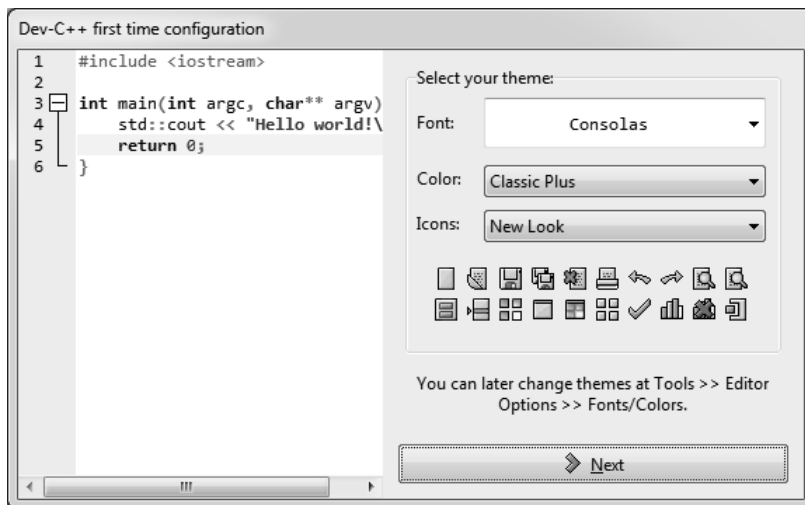


Obrázek 1.8. Dokončení instalace

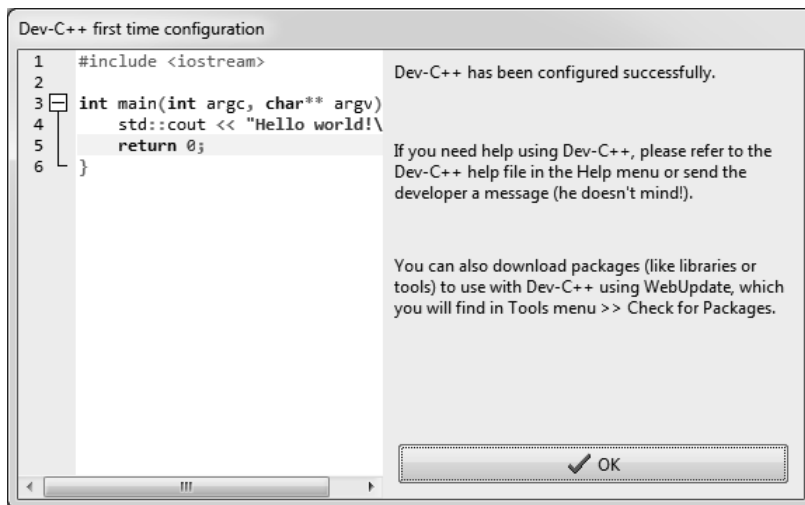
Následně se objeví tři velmi podobné dialogy dle obrázků 1.9 až 1.11, které slouží pro konfiguraci prostředí. Jedná se o možnost změny jazyka instalace, schématu a nástrojů. Poslední dialog potvrzuje dokončení konfigurace.



Obrázek 1.9. Volba jazyka



Obrázek 1.10. Volba schématu



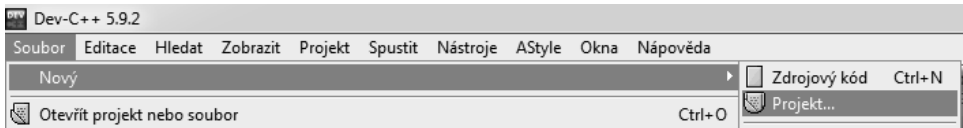
Obrázek 1.11. Dokončení konfigurace

Následně již nabíhá prostředí.

První program

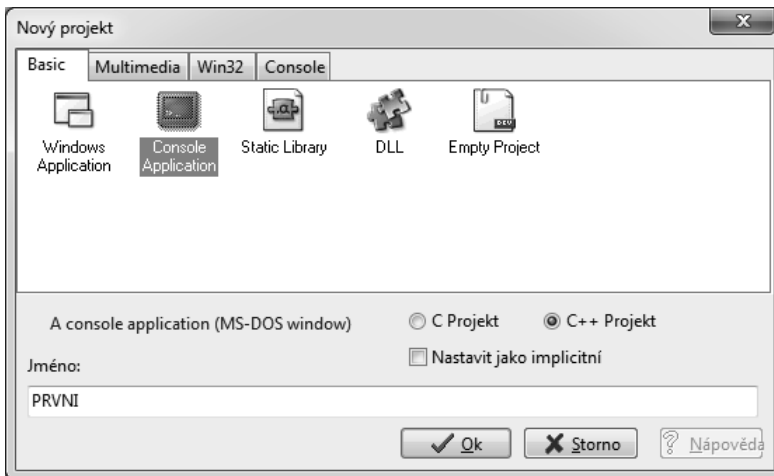
Nyní tedy můžeme vytvořit svůj první program v jazyce C++.

Spustíme vývojové prostředí Dev-C++ (pokud již není spuštěno) a pomocí příkazu nabídky **Soubor** → **Nový** → **Projekt** (viz obrázek 1.12) vyvoláme dialog volby typu projektu dle obrázku 1.13.



Obrázek 1.12. Založení nového projektu

V dialogu volby typu projektu dle obrázku 1.13 se přepneme na kartu **Basic** a jako typ projektu zvolíme **Console Application**. Konzolová aplikace je typ aplikace pro operační systém Windows, která pracuje v textovém režimu (jako MS-DOS okno). Dále vybereme projekt typu **C++ Projekt**. V textovém poli **Jméno** zadáme název projektu **PRVNI**. Pokračujeme stiskem tlačítka **Ok**.

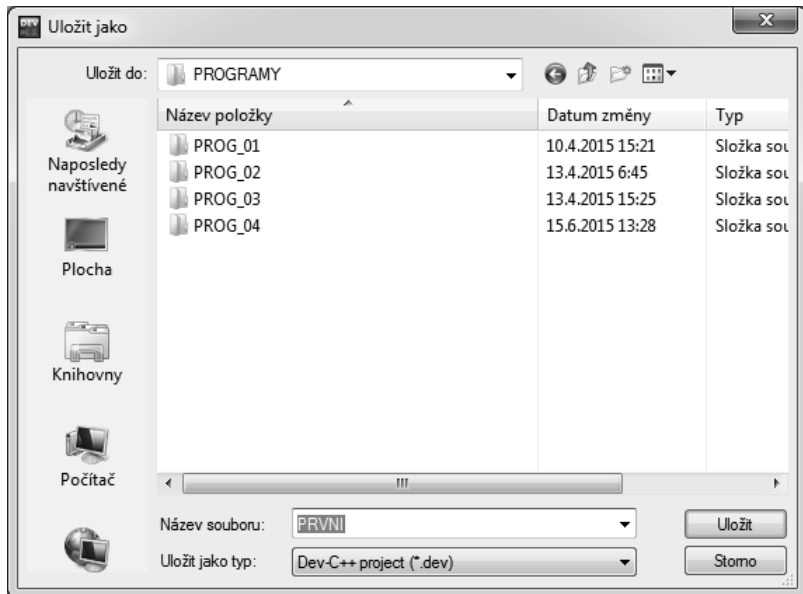


Obrázek 1.13. Volba typu a názvu projektu

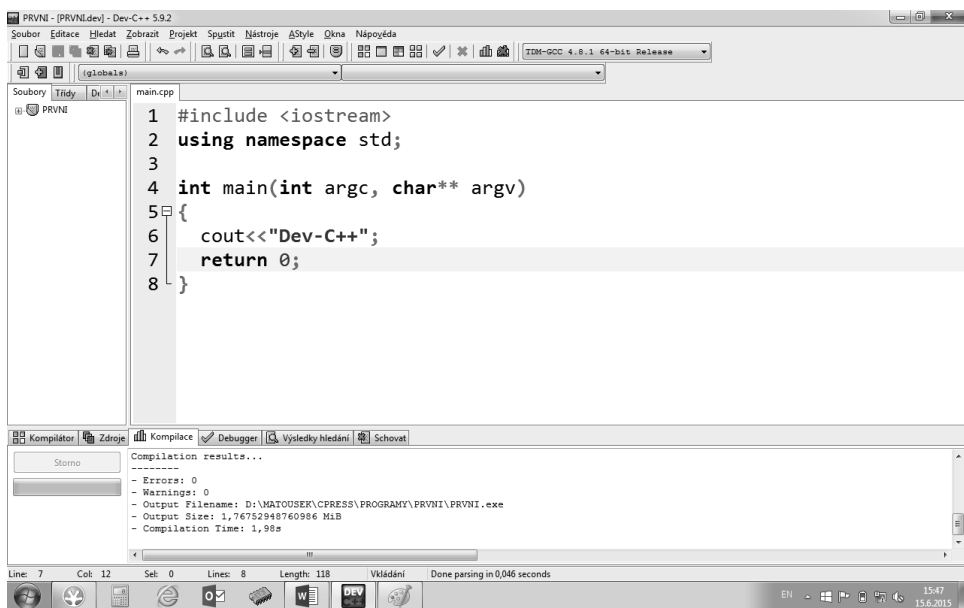
Následně se zobrazí dialog dle obrázku 1.14, což je klasický dialog volby adresáře pro uložení projektu. Zvolíme příslušnou složku a pokračujeme tlačítkem **Uložit**.

Zápis našeho prvního programu dokumentuje obrázek 1.15. Soubor byl standardně pojmenován jako **MAIN.CPP** (main = hlavní program, CPP značí zkratku pro „cé plus plus“). Všimněte si, že textový editor rozlišuje syntaxi pomocí barev a stylů písma. Příkazy preprocesoru (povíme si o nich podrobněji v průběhu našeho seznamování s jazykem C++) se značí zeleně,

tučně jsou označena klíčová slova jazyka C++, červeně speciální znaky, modře textové řetězce a fialově zápisy čísel.



Obrázek 1.14. Volba adresáře pro uložení souborů projektu



Obrázek 1.15. Zápis programu v textovém editoru

Výpis programu je ještě uveden v textové formě níže.

```

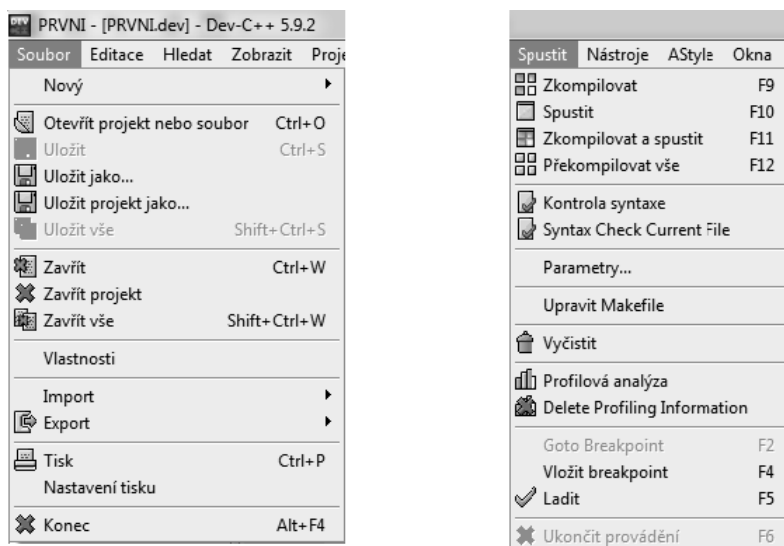
MAIN.CPP
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    cout<<"Dev-C++";
    return 0;
}

```

Klíčové položky nabídky

Na začátku naší práce s vývojovým prostředím Dev-C++ vystačíme pouze s několika málo příkazy z nabídek **Soubor** a **Spustit**, pro informaci jsou tyto dvě nabídky uvedeny formou obrázku 1.16.

Nabídka **Soubor** obsahuje známé příkazy. Nejčastěji budeme používat příkaz **Nový** buď pro založení nového projektu, nebo pro vložení nového zdrojového souboru do stávajícího projektu. Příkaz **Zavřít projekt** zavře aktuální projekt, ale neukončí vývojové prostředí. Příkaz **Zavřít** zavře aktuální zdrojový soubor otevřený v editoru. Více nyní vědět nemusíme. Příkazem **Otevřít projekt nebo soubor** otevřeme dříve vytvořený projekt nebo zdrojový soubor (který jsme třeba nedopatřením zavřeli).



Obrázek 1.16. Nabídky Soubor a Spustit

Nabídka **Spustit** obsahuje čtyři klíčové příkazy, které jsou svými názvy velmi podobné, proto vysvětlíme rozdíly (viz obrázek 1.16):

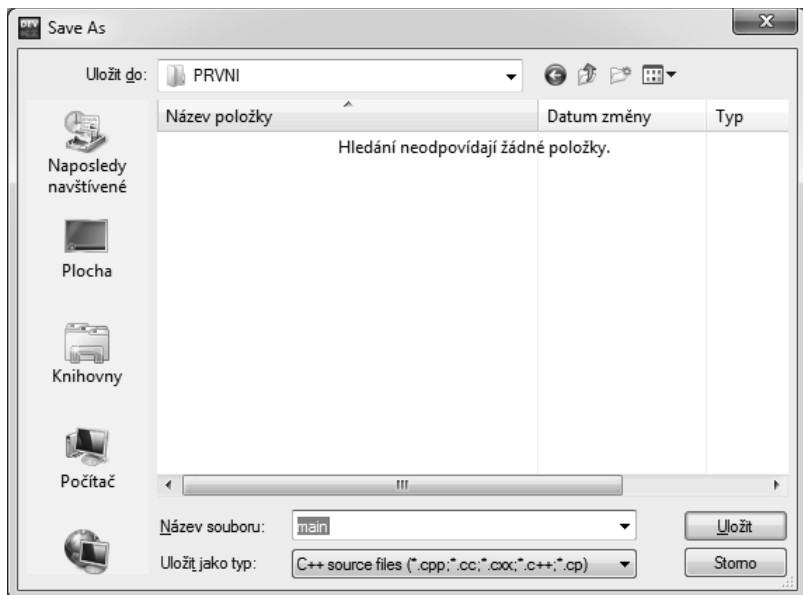
- **Zkompilovat F9** – přeloží aktuální projekt, vznikne cílový spustitelný soubor s koncovkou EXE,
- **Spustit F10** – spustí dříve vytvořený cílový soubor (nový překlad se neprovádí),
- **Zkompilovat a spustit F11** – pokud došlo ve zdrojovém souboru ke změnám, provede nový překlad; pokud je překlad v pořádku, spustí následně program,
- **Překompilovat vše F12** – provede opětovný překlad všech zdrojových souborů v projektu.

Překlad programu

Nyní již víme, že nejsnazší způsob pro překlad a následné spuštění programu je použít zkratkovou klávesu **F11** (vyvolá se příkaz **Zkompilovat a spustit**).

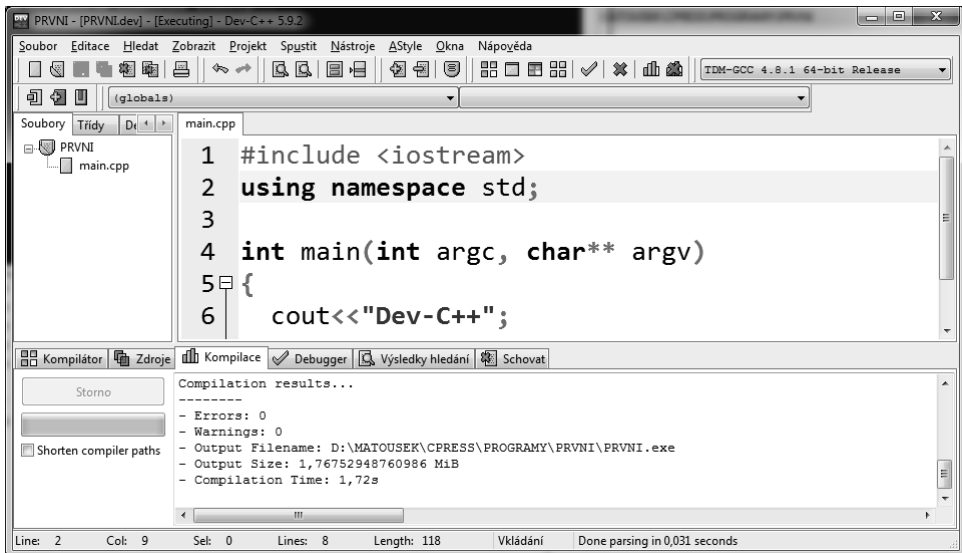
Před prvním překladem budeme dotázáni na název zdrojového souboru proto, aby vývojové prostředí mohlo pod tímto názvem soubor uložit. Výchozí název **MAIN.CPP** lze změnit, obvykle však tento název ponecháme. Viz obrázek 1.17.

Při opětovných překladech již vývojové prostředí ukládá zdrojové soubory automaticky. Situaci po úspěšném překladu zachycuje obrázek 1.18, následně je přeložený program spuštěn v konzoli, jak dokladuje obrázek 1.19.



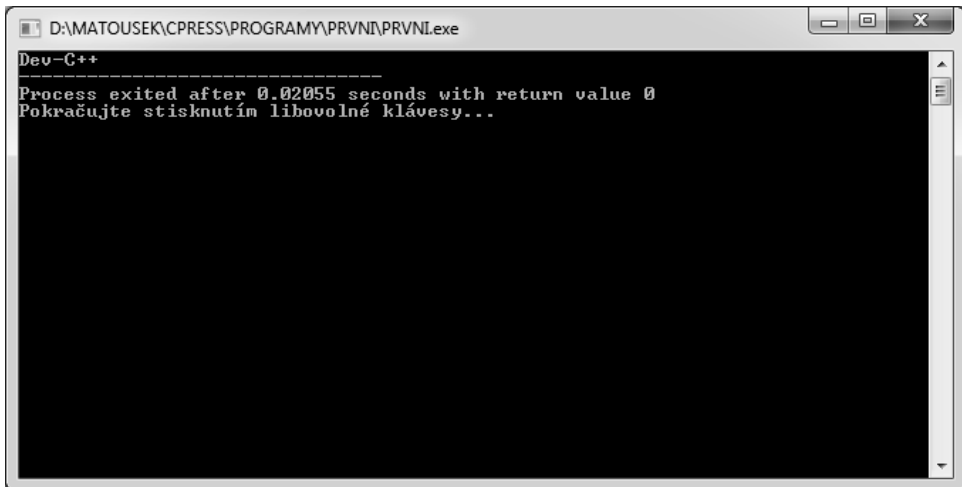
Obrázek 1.17. Dialog pro volbu názvu zdrojového souboru pro jeho uložení

Je možné, že se vám nepodaří zdrojový text opsat správně. Hlášení chyb je sice v angličtině, ale chybu lze rozoznat pomocí čísla řádku. Dále víme, že například klíčová slova jsou psána tučně (uděláme-li chybu při zápisu klíčového slova, nebude příslušné slovo vyobrazeno tučně).



Obrázek 1.18. Situace po překladu, v dolní části jsou informace o výsledcích překladu

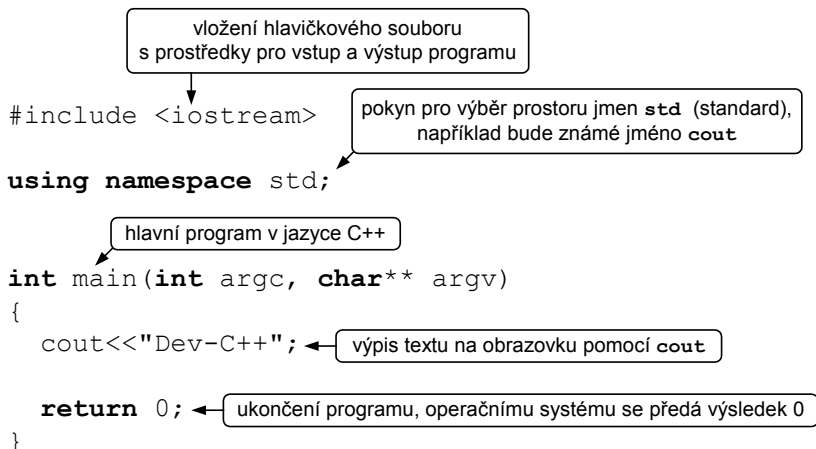
Výsledkem běhu programu je výpis textu Dev-C++. Prostředí samo dále vypíše hlášení ohledně času běhu programu (zhruba 20 ms) a návratové hodnotě 0. Rovněž vloží čekání na stisk klávesy, aby bylo možné si vypsané výsledky prohlédnout. Viz obrázek 1.19.



Obrázek 1.19. Běh programu v konzoli

Stručné vysvětlení zápisu programu

V této chvíli není možné zcela vysvětlit konstrukce, které jsou v zápisu tohoto jednoduchého programu použity. Pokusme se o to alespoň stručně pomocí obrázku 1.20.



Obrázek 1.20. Rozbor programu

Na začátku zdrojového textu se obvykle uvádí tzv. direktivy neboli příkazy preprocesoru. Tyto příkazy rozeznáme podle toho, že začínají symbolem # (mřížka). Příkaz `#include <iostream>` slouží pro vložení zdrojového souboru s názvem `iostream` (input/output stream, tedy vstupně/výstupní proudy; proudy jsou v jazyce C++ prostředky pro vstup a výstup programu; například načtení dat z klávesnice nebo výpis na obrazovku, již v kapitole 2 se dozvíme, jak se proudy používají).

Příkaz `using namespace std;` vybere k použití prostředky poskytované tzv. jmenným prostorem `std` (standard). Do tohoto prostoru patří zejména prostředky `cin` (vstupní konzolový proud, používá se pro načtení dat z klávesnice) a `cout` (výstupní konzolový proud, používá se pro výpis dat na obrazovku).

Jazyk C++ používá (podobně jako většina jiných programovacích jazyků) blokovou strukturu. Program lze rozdělit do určitých celků, tzv. podprogramů, které se v případě jazyka C++ nazývají funkce. Hlavní program je pak zapsán jako funkce `main` (česky hlavní). Každá funkce může mít návratovou hodnotu a parametry (o tom později v kapitole 7).

V hlavním programu je zapsán příkaz výstupu, který do proudu `cout` odešle uvedený textový řetězec. V programu to způsobí vypsaní příslušného řetězce na obrazovce.

Program je zakončen příkazem `return 0;` (návrat), který ukončí jeho běh a operačnímu systému předá návratový kód. Zde zvolená hodnota 0 se většinou používá jako znamení, že program došel normálně (bez hlášení chyby). Různými hodnotami návratových kódů můžeme operačnímu systému sdělovat různé chyby, které při své činnosti program zjistil.

Pár zajímavostí

Na tomto místě připojíme pár zajímavostí, které se mohou hodit při běžném psaní programů.

Komentáře neboli poznámky

Pokud potřebujeme určitou část zdrojového textu označit poznámkou (neboli napsat k ní komentář), lze použít dvě varianty:

1. Jedná-li se o krátký komentář, který se vejde na jeden řádek, použijeme dvě lomítka a za ně zapíšeme komentář. Například:

```
cout<<"Dev-C++"; //vypíše Dev-C++
```

2. Jedná-li se o delší komentář na více řádků, začneme kombinací lomítka – hvězdička, poté zapíšeme příslušný text a komentář ukončíme opačnou kombinací, tedy hvězdička – lomítko. Například:

```
/*toto je komentář,  
který zabírá více řádků*/
```

Komentáře slouží pouze pro orientaci ve zdrojovém textu. Všechny komentáře překladač ignoruje.

Proto také komentáře často používáme k tomu, abychom určitou část zdrojového textu „vypustili“. Říkáme, že tuto část textu „zakomentujeme“. Překladač pak tuto část nepřeloží (tedy „vynechá“).

Pomocné nástroje na Internetu

Na internetu lze nalézt mnoho užitečných informací k programování. Doporučujeme zejména stránky:

- **www.cplusplus.com** – zde je k dispozici řada informací k jazyku C++, například podrobnější výklad klíčových slov a knihoven, nápověda a tutoriály. Text je v angličtině.
- **cpp.sh** – zajímavé stránky obsahující tzv. C++ Shell. Jedná se o online překladač jazyka C++. Můžete ho používat zejména pro testování krátkých programů, a přitom nepotřebujete mít v počítači nainstalovaný překladač jazyka C++.

Rozdělení základních datových typů

V následujících kapitolách se postupně seznámíme s datovými typy pro uložení různých hodnot. To nám dovlí vytvářet programy, které budou například schopny provádět jednoduché matematické operace.

Datový typ určuje obor hodnot (v jakém rozsahu proměnná uchovává hodnoty), interpretaci obsahu (číslo, znak apod.) a množinu přípustných operací (například čísla lze sčítat nebo porovnávat). Mezi základní datové typy patří:

- *celočíselné (v pevné řádové čárce)*: například **int** (viz kapitolu 2),
- *reálné (v pohyblivé řádové čárce)*: například **float** (viz kapitolu 3),
- *logické*: **bool** (viz kapitolu 4),
- *znakové (pro uchování jednoho znaku)*: **char** (viz kapitolu 10),
- *odvozené datové typy (jsou odvozeny z číselných nebo znakových typů)*: například pole (viz kapitolu 6), ukazatel (viz kapitoly 8 a 9), struktura (viz kapitolu 12), třída (viz kapitolu 16) atd.

Celočíselné datové typy

Paměť počítače ani jeho výpočetní výkon nejsou neomezené, proto používáme různé datové typy přiměřené pro konkrétní použití. Základní charakteristiky datových typů:

- **Obor hodnot** určuje omezení rozsahu hodnot uchovávaných v proměnné příslušného datového typu. Má-li být rozsah hodnot větší, musí proměnná zabírat větší prostor v paměti a také provádění operací zabírá obvykle více času.
- **Interpretace obsahu** určuje, jakým způsobem se nahlíží na obsah proměnné. Na hardwarové úrovni je obsah proměnné reprezentován „shlukem jedniček a nul“. Obsah proměnné můžeme ale chápat jako číslo nebo znak či mnoha jinými způsoby.
- **Množina operací** určuje, které operace lze vykonávat nad proměnnými daného datového typu, případně jak tyto operace probíhají. Například dělení dvou celých čísel je prováděno celočíselně. Kdežto dělení dvou reálných čísel je prováděno v reálné aritmetice. Sčítat můžeme proměnné číselných typů, kdežto pole sčítat nelze. Atd.

Nejjednodušší skupinu datových typů představují celá čísla, proto touto skupinou začneme. Níže se dozvíme, že celočíselné proměnné mohou být bez znaménka (kladné hodnoty včetně nuly) nebo se znaménkem. Dále se seznámíme se základními vstupně/výstupními operacemi (naučíme se načíst hodnotu čísla z klávesnice do proměnné a vypsat obsah proměnné na obrazovku). Poté se seznámíme s aritmetickými operacemi poskytovanými jazykem C++.

Celá čísla se znaménkem a bez znaménka

Základní rozdělení celých čísel je na *čísla bez znaménka* (kladné hodnoty včetně nuly) a *čísla se znaménkem* (kladné i záporné hodnoty včetně nuly). Pro rozlišení těchto dvou skupin používáme klíčová slova `unsigned` (označuje proměnnou bez znaménka) a `signed` (označuje proměnnou se znaménkem).

V této kapitole:

- Celá čísla se znaménkem a bez znaménka
- Základní vstupně/výstupní operace
- Pokročilejší operace s proměnnými a proudy
- Aritmetické operace s celými čísly
- Zadávání číselných literálů v různých soustavách

Pro deklaraci proměnné typu celé číslo používáme datový typ `int`. Je to zkratka anglického označení *integer*, která odpovídá českému pojmu celé číslo. Proměnná tohoto typu zabírá v paměti počet bajtů, který je obvykle určen typem překladače. *V současnosti je obvyklé, že překladače berou základní velikost celého čísla v rozsahu 32 bitů, tedy jako 4bajtové číslo.*

Je-li třeba, lze původní rozsah typu `int` upravit buď zkrácením na polovinu rozsahu v bajtech, takový typ označujeme jako `short` (česky „krátký“). Případně je možné původní rozsah v bajtech rozšířit na dvojnásobek, takový typ označujeme jako `long` (česky „dlouhý“). Dokonce existuje ještě „širší“ typ označovaný jako `long long`.

Celá čísla bez znaménka

Začněme nejdříve typem `unsigned int`, který odpovídá celému číslu bez znaménka v rozsahu 4 bajtů. Do proměnné tohoto typu lze uložit hodnoty 0 až $2^{32} - 1$. Počet odlišných čísel kódovaných ve dvojkové soustavě ve 32 bitech odpovídá 2^{32} . Vzhledem k tomu, že začínáme od nuly, je maximální hodnota vždy o jedničku nižší než 2^{32} , tedy maximální hodnota je $2^{32} - 1$.

Datový typ `unsigned short int` má rozsah v bajtech poloviční proti předchozímu případu (tedy 2 bajty). Do proměnné tohoto typu lze uložit hodnoty 0 až $2^{16} - 1$.

Datový typ `unsigned long int` má obvykle rozsah shodný jako typ `unsigned int`.

Datový typu `unsigned long long int` má proti typu `unsigned int` rozsah v bajtech dvojnásobný (tedy 8 bajtů). Do proměnné tohoto typu lze uložit hodnoty 0 až $2^{64} - 1$.

Zvláštností je pak datový typ `unsigned char`. Později se dozvíme (viz kapitolu 10), že se používá pro znakové proměnné. Nicméně ho lze použít i pro celá čísla a rozsah odpovídá jednomu bajtu, číselně 0 až $2^8 - 1$.

Rychlý přehled o celočíselných datových typech bez znaménka získáme z tabulky 2.1. Najdeme zde příslušný datový typ, jeho zkrácený název (nemusíme tak například zapisovat `unsigned int`, stačí zapsat pouze `unsigned`), velikost proměnné v bajtech (tedy informaci o místě, které zabere v paměti), číselný rozsah a příponu. Přípona se používá v případech, kdy je nutné zapisovat číselné hodnoty ve vyšším rozsahu, než odpovídá (znaménkovému) datovému typu `int`. Příponu uvádíme také v případech, že je nutné zdůraznit, jaký má být typ číselné konstanty.

Tabulka 2.1. Datové typy pro celá čísla bez znaménka

Datový typ	Zkrácený název	Velikost	Číselný rozsah	Přípona
<code>unsigned int</code>	<code>unsigned</code>	4 bajty	0 až $2^{32}-1$	UL
<code>unsigned long int</code>	<code>unsigned long</code>		(0 až 4 294 967 295)	
<code>unsigned short int</code>	<code>unsigned short</code>	2 bajty	0 až $2^{16}-1$ (0 až 65 535)	není

Datový typ	Zkrácený název	Velikost	Číselný rozsah	Přípona
unsigned long long int	unsigned long long	8 bajtů	0 až $2^{64}-1$ (0 až 18 446 744 073 709 551 615)	ULL
unsigned char	není	1 bajt	0 až 2^8-1 (0 až 255)	není

Celá čísla se znaménkem

Celá čísla se znaménkem používají jeden bit pro uložení znaménka, takže rozsah čísel je pak o jeden bit užší. Záporných hodnot je stejný počet jako kladných hodnot včetně nuly.

Pro 4bajtové typy `signed int` a `signed long int` je rozsah -2^{31} až $2^{31}-1$. Pro 2bajtový typ `signed short int` je rozsah -2^{15} až $2^{15}-1$. Pro 8bajtový typ `signed long long int` je rozsah -2^{63} až $2^{63}-1$. Pro 1bajtový typ `signed char` je rozsah -2^7 až 2^7-1 .

Tabulka 2.2. Datové typy pro celá čísla se znaménkem

Datový typ	Zkrácený název	Velikost	Číselný rozsah	Přípona
signed int	int	4 bajty	-2^{31} až $2^{31}-1$	není
signed long int	long		(-2 147 483 648 až +2 147 483 647)	L
signed short int	short	2 bajty	-2^{15} až $2^{15}-1$ (-32 768 až +32 767)	není
signed long long int	long long	8 bajtů	-2^{63} až $2^{63}-1$ (-9 223 372 036 854 775 808 až +9 223 372 036 854 775 807)	LL
signed char	char	1 bajt	-2^7 až 2^7-1 (-128 až +127)	není

Charakteristiky celočíselných datových typů

Chceme-li snadno zjistit rozsahy jednotlivých číselných typů v dané implementaci překladače, stačí použít soubor **LIMITS.H**. Tento soubor nalezneme v adresáři instalace vývojového prostředí.

V tabulce 2.3 jsou uvedeny nejpoužívanější symboly a jejich význam (v závorce je pak uvedena hodnota, která platí pro případ překladačů, které jako základní typ uvažují 32bitové celé číslo). Podobně jsou definovány symboly `LONG_MIN`, `LONG_MAX`, `ULONG_MAX`, `LLONG_MIN`, `LLONG_MAX`, `ULLONG_MAX`.

Tabulka 2.3. Vybrané symboly ze souboru LIMITS.H

Symbol	Význam
CHAR_BIT	počet bitů pro reprezentaci proměnné typu char (8)
SCHAR_MIN	minimální hodnota proměnné typu signed char (-128)
SCHAR_MAX	maximální hodnota proměnné typu signed char (+127)
UCHAR_MAX	maximální hodnota proměnné typu unsigned char (255)
SHRT_MIN	minimální hodnota proměnné typu signed short (-32 768)
SHRT_MAX	maximální hodnota proměnné typu signed short (+32 767)
USHRT_MAX	maximální hodnota proměnné typu unsigned short (65 536)
INT_MIN	minimální hodnota proměnné typu signed int (-2 147 483 648)
INT_MAX	maximální hodnota proměnné typu signed int (+2 147 483 647)
UINT_MAX	maximální hodnota proměnné typu unsigned int (4 294 967 295)

Základní vstupně/výstupní operace

Vstupně/výstupní operace, tedy načítání hodnot proměnných a jejich výpis, se v jazyce C++ provádí pomocí tzv. proudů (anglicky streams). Proudů poskytují velmi přehledný a výkonný prostředek pro vstupně/výstupní operace.

Standardní (tzv. konzolové) proudy se nazývají `cin` (vstupní) a `cout` (výstupní). Pro jejich použití je třeba zapsat pokyn (použití symbolů z tzv. prostoru jmen s označením `std`, tím se nám zjednoduší zápisy: místo plné formy `std::cin` a `std::cout` budeme zapisovat pouze `cin` a `cout`):

```
using namespace std;
```

Základní výstupní operace

Výstupní proud se nazývá `cout`, což je zkratka od anglického označení *console output* (výstupní konzole). Jako výstupní konzole pracuje většinou obrazovka počítače. Hodnoty odeslané do výstupního proudu `cout` uvidíme tedy vypsáné na obrazovce.

Pro odeslání hodnoty do výstupního proudu používáme operátor `<<`, který se odborně označuje jako *insertor* (insert – vložit, vkládá hodnoty do proudu). Symbol `<<` představuje myšlenou šipku, která říká, že hodnota je nasměrována do proudu. Operátor `<<` se používá v této formě zápisu:

```
výstupní_proud << výraz;
```

Je-li třeba odeslat do proudu (provést výpis) více výrazů, lze operátor `<<` použít v daném příkazu opakovaně, například:

```
výstupní_proud << výraz1 << výraz2 << výraz3;
```

Chceme-li vypsát text, musíme jej uvést v rámci uvozovek:

```
výstupní_proud << "text";
```

Pro ukončení řádku při výpisu používáme *manipulátor end1* (zkratka od anglického označení *end line*):

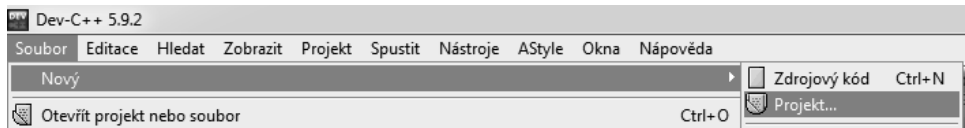
```
výstupní_proud << end1;
```

Po tomto krátkém výkladu můžeme již předvést výstupní operace pomocí proudu `cout` na krátkém příkladu. Vypíšeme na obrazovku základní vlastnosti výše popsanych datových typů `int` a `unsigned`. Pro získání informace o velikosti proměnné nebo datového typu používáme operátor `sizeof` (vychází z anglického označení „*size of*“, píše se však bez mezery). Tento operátor se používá ve formě zápisu:

```
sizeof(datový_typ|proměnná)
```

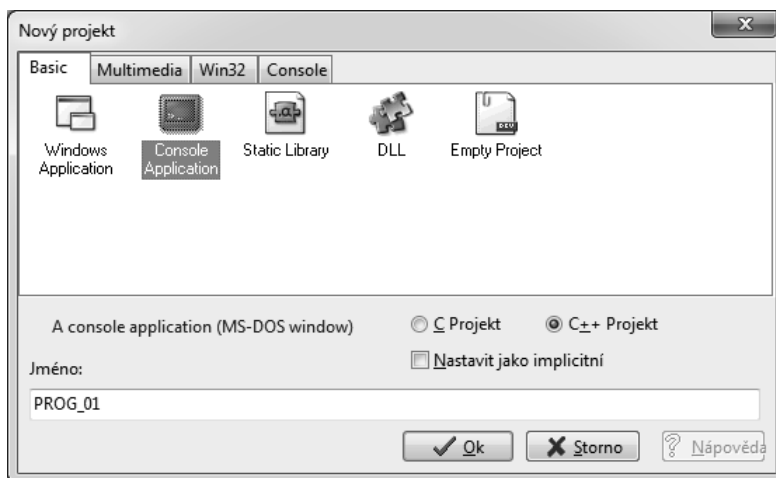
Výsledkem použití operátoru `sizeof` je celé číslo, které určuje počet bajtů potřebných pro uložení proměnné příslušného datového typu do paměti.

Nyní spustíme vývojové prostředí Dev-C++. Pomocí položky nabídky **Soubor** → **Nový** → **Projekt**, viz obrázek 2.1, vyvoláme dialog volby typu projektu dle obrázku 2.2.



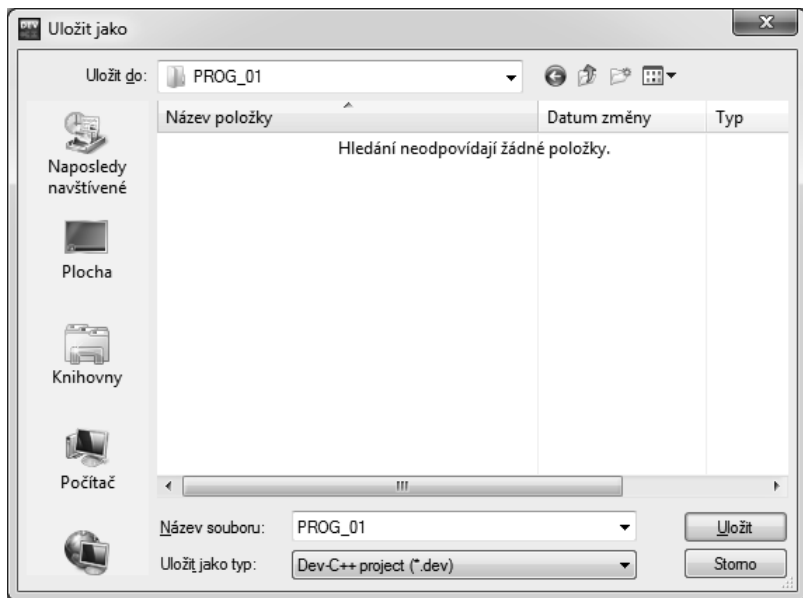
Obrázek 2.1. Založení nového projektu

V dialogu volby typu projektu dle obrázku 2.2 se přepneme na kartu **Basic** a jako typ projektu zvolíme **Console Application**. Konzolová aplikace je typ aplikace pro operační systém Windows, která pracuje v textovém režimu (jako MS-DOS okno). Dále vybereme projekt typu **C++ Projekt**. V textovém poli **Jméno** zadáme název projektu **PROG_01**. Pokračujeme stiskem tlačítka **Ok**.

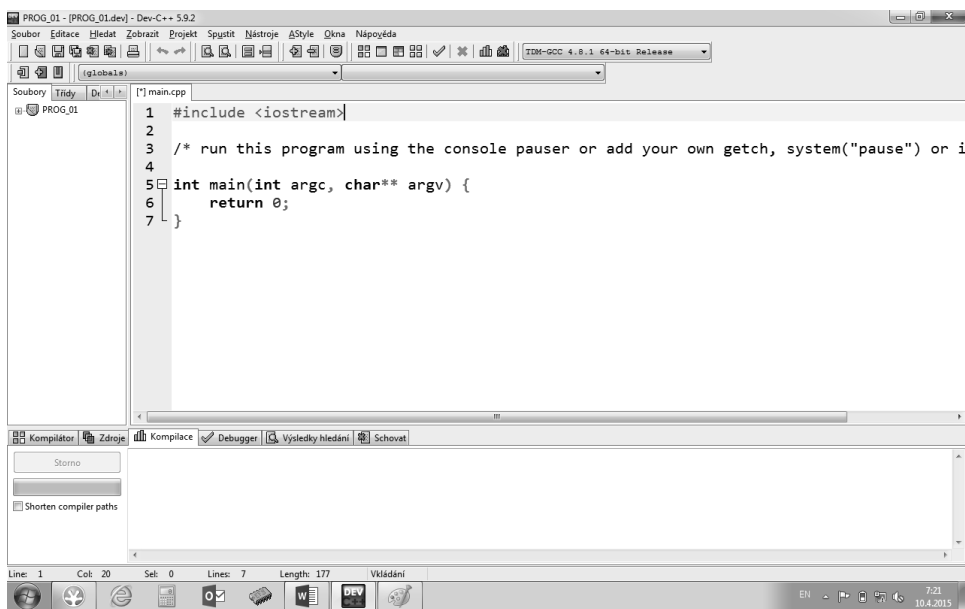


Obrázek 2.2. Volba typu projektu

Následně se zobrazí dialog dle obrázku 2.3, což je klasický dialog volby adresáře pro uložení projektu. Zvolíme příslušnou složku a pokračujeme tlačítkem **Uložit**.



Obrázek 2.3. Volba adresáře pro uložení souborů projektu



Obrázek 2.4. Nyní můžeme začít psát program

Po těchto krocích je již vývojové prostředí připraveno na vlastní programování. Situace je zřejmá z obrázku 2.4.

Vývojové prostředí samo vloží hlavičkový soubor **iostream**, který je potřebný pro použití proudů. Pro vložení externího zdrojového souboru se používá direktiva `#include` (*include* – zahrnout) v této formě:

```
#include <název_souboru>
```

Následně požadujeme použití jmenného prostoru `std` a sami požadujeme vložení hlavičkového souboru **limits.h**, který obsahuje informace o celočíselných datových typech.

Hlavní program zapisujeme do funkce `main`.

Program pomocí proudu `cout` vypíše informace o velikosti datového typu `int` v bajtech a minimální a maximální hodnotu proměnné typu `int`.

Následně jsou podobně vypsané informace o datovém typu `unsigned`.

```

PROG_01:
#include <iostream>
using namespace std;
#include <limits.h>
int main(int argc, char** argv)
{
    cout<<"velikost typu int: "<<sizeof(int)<<endl;
    cout<<"minimalni hodnota: "<<INT_MIN<<endl;
    cout<<"maximalni hodnota: "<<INT_MAX<<endl;
    cout<<endl;
    cout<<"velikost typu unsigned: "<<sizeof(unsigned)<<endl;
    cout<<"minimalni hodnota: "<<0<<endl;
    cout<<"maximalni hodnota: "<<UINT_MAX<<endl;
    return 0;
}

```

Výpis programu v konzoli:

```

velikost typu int: 4
minimalni hodnota: -2147483648
maximalni hodnota: 2147483647
velikost typu unsigned: 4
minimalni hodnota: 0
maximalni hodnota: 4294967295

```