edice
začínáme s ...

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

soubory ke stažení na www.grada.cz

MIROSLAV VIRIUS

GRADA®

edice
začínáme s ...

Začínáme programovat v jazyku C++

MIROSLAV VIRIUS

GRADA Publishing

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.

ISBN 978-80-247-5935-7 (ePub)
ISBN 978-80-247-5902-9 (pdf)
ISBN 978-80-271-5157-8 (print)

Obsah

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 zkoušet 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 zavadíme 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 ladicí 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 háčků a čárek, neboť prostředí OnlineGDB Beta jejich použití nepodporuje. Vím, 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. Vím, 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 cppreference.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 Virius
miroslav.virius@fjfi.cvut.cz
http://people.fjfi.cvut.cz/virius

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 potřebujeme textový editor. Pak zdrojový program přeložíme.

Přitom se může stát, že program nenapíšeme 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. Takovémuto 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 slýchám 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 veliké 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 zapsat 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".

Součástí projektů jsou zpravidla i jiné než zdrojové soubory – některé slouží jako pomůcka pro integrované vývojové prostředí (v nich si IDE uchovává informace o tom, které soubory k projektu patří, různá nastavení, kterými se bude překladač řídit atd.), jiné mohou obsahovat tzv. *prostředky* (anglicky *resource*, obrázky a texty, které bude program používat apod.).

Překladač přeloží jednotlivé zdrojové soubory, které projekt tvoří, a pak z nich *sestaví* výsledný spustitelný program. Integrovaná vývojová prostředí umějí zpravidla pracovat i se skupinami projektů (v některých vývojových nástrojích se označují jako řešení). Takováto skupina projektů obvykle představuje skupinu programů, které nějakým způsobem spolupracují – ale to už přesahuje možnosti úvodního povídání.

1.8 Proč bych si měl vybrat jazyk C++

- Pochopil jsem, že programovacích jazyků je více. Proč bych si měl vybrat právě C++?

Důvodů může být mnoho. Jazyk C++ vytvořil a publikoval v polovině 80. let minulého století Bjarne Stroustrup a tento jazyk velice rychle získal popularitu. V 90. letech byl programovacím jazykem číslo 1 a od té doby – i když vznikla řada nových jazyků – je stále mezi pěticí nejpoužívanějších. To je jistě dobrý důvod, proč ho používat.

Dalším důvodem může být, že jazyk C++ není vázán na žádnou konkrétní platformu (tedy zhruba řečeno na konkrétní druh počítačů a jejich vybavení); to znamená, že jsou k dispozici překladače pro převážnou většinu platforem – a samozřejmě pro všechny běžné.

1.9 Jak vypadá program v C++

- Vraťme se k programování. Jak takový program vypadá?

Nejjednodušší program v C++ si ukážeme hned v následující kapitole. Pojďme už na to; další podrobnosti si povíme později.

1.10 Vývojové prostředí a překladače

- Než se pustíme do programování, ještě jednu otázku: Jaká vývojová prostředí – a jaké překladače – si mohu vybrat? Když na to přijde, je překladač vždy součástí nějakého vývojového prostředí?

Některé překladače jsou k dispozici samostatně, jen s doprovodnými knihovnami a některými dalšími doplňkovými nástroji, jiné jsou k dostání pouze jako součást IDE; některé překladače a některá vývojová prostředí jsou k dispozici bezplatně, některá je třeba zakoupit.

Asi nejznámějším volně dostupným překladačem je g++, jeden ze sady překladačů různých programovacích jazyků projektu GNU (GNU Compiler Collection, viz např. [7]). Tento překladač používá i prostředí OnlineGDB Beta. Z nekomerčních vývojových prostředí, která si musíme instalovat, se zmíníme o nástroji Code::Blocs, jehož instalační program si lze stáhnout na adrese [8], a o pokročilém nástroji Qt Creator [9], který nabízí mimo jiné i knihovnu pro tvorbu grafického uživatelského rozhraní, jež sice není standardní součástí jazyka C++, ale je velmi oblíbená.

Mezi komerčními překladači je dnes asi nejznámější překladač firmy Microsoft, který je součástí vývojového prostředí MS Visual Studio. Toto prostředí je sice placené, ale pro nekomerční použití je k dispozici jeho verze označovaná Community Edition, která je zdarma (za registraci). Vedle toho nabízí Microsoft zdarma také nástroj MS Visual Studio Code [16]. Dalším komerčním nástrojem je C++Builder od firmy Embarcadero, u něhož je také k dispozici bezplatná Community Edition [11]; podmínkou je opět registrace.

2 První program v C++

Nejlepší cestou, jak se seznámit s programováním v jakémkoli programovacím jazyce, je zkusit si napsat jednoduchý program, přeložit ho a spustit ho, abychom viděli, že dělá právě to, co jsme zamýšleli, nic víc, nic míň. My jsme se rozhodli pro C++ a už víme, že k tomu potřebujeme vhodné vývojové prostředí (IDE). Řekli jsme si také, že některá IDE jsou k dispozici online, takže si je nemusíme instalovat. Za toto pohodlí ovšem zaplatíme – nebudeme mít k dispozici přeložený program, náš program budeme moci spouštět pouze v tomto prostředí. Při prvním seznamování s programováním v C++ to však není důležité.

V této kapitole poznáme – alespoň povrchně – vývojové prostředí OnlineGDB Beta, které je k dispozici na webu, napíšeme první jednoduchý program, přeložíme ho a spustíme. Ukážeme si také, co se stane, když překladač jazyka C++ v programu najde chybu, takže mu nebude rozumět.

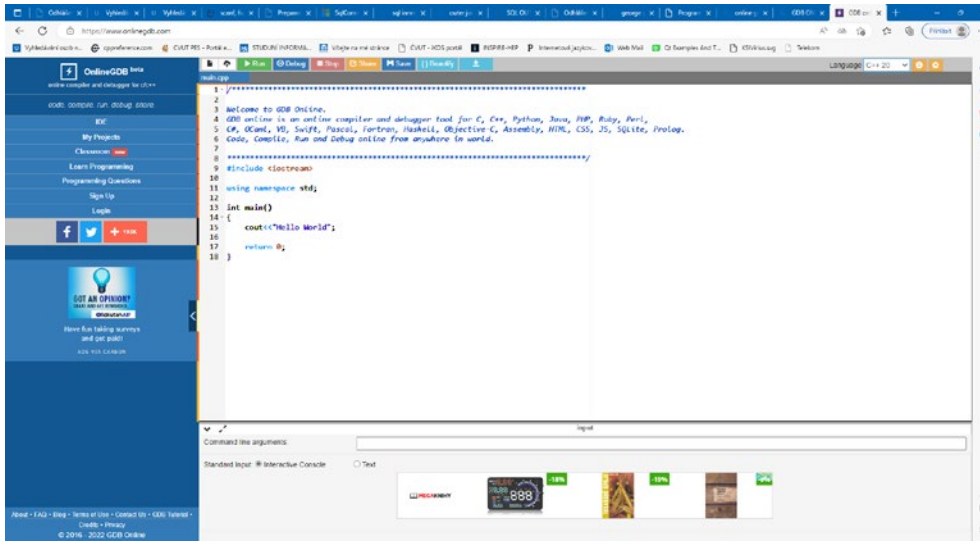
2.1 Prostředí OnlineGDB Beta

Už jsme si řekli, že v této knize budeme používat prostředí OnlineGDB Beta, které najdete na webové adrese [1]. Toto prostředí umožňuje programovat v řadě programovacích jazyků, mezi jinými i v C++. Po otevření této webové stránky uvidíte nejspíš něco podobného jako na obrázku 2.1. Prostředí pro nás vytvořilo projekt obsahující jediný zdrojový soubor nazvaný `main.cpp`, který obsahuje jednoduchý program.

Než se budeme zabývat programem, který nám OnlineGDB Beta nabídlo, alespoň povrchně se s tímto prostředím seznámíme.

Na levé straně je panel s odkazy, z nichž většina vyžaduje vytvoření účtu a přihlášení, a proto je ponecháme stranou. Tlačítkem se znakem <, které najdeme uprostřed levého okraje, můžeme tento panel zavřít. Přitom se na levém okraji stránky objeví tlačítko s bleskem (viz obr. 2.2), kterým tento panel v případě potřeby zase otevřeme.

Střed stránky tvoří panel editoru, ve kterém budeme zapisovat své programy. Po otevření tam bude již zobrazen jednoduchý program v některém z programovacích jazyků, které toto prostředí podporuje.



Obr. 2.1 Vývojové prostředí OnlineGDB Beta po otevření

Nad panelem editoru je lišta, jež obsahuje tlačítka pro spuštění programu, pro jeho ladění apod. a také pro volbu programovacího jazyka, který chceme používat; mělo by tam být uvedeno C++ následované rokem vydání standardu. Pokud není, klepnutím myší na toto pole rozvineme seznam jazyků a vybereme např. C++ 20.[3] Prostředí zobrazí jednoduchý program v C++.

Poznamenejme, že čísla řádek, která vidíte na levé straně panelu editoru, nejsou součástí programu, doplnilo je prostředí. Oceníme je, až si budeme povídat o tom, co která řádka programu znamená, a také při luštění zpráv o chybách.

V pravé části této webové stránky je – nebo může být – zobrazen panel s nástroji, které slouží k ladění programů – tedy k hledání a nápravě chyb, které způsobují, že program sice běží, ale dělá něco jiného, než chceme, nebo že se rozběhne a skončí chybou. Později s ním budeme pracovat.

Ve spodní části stránky je panel, ve kterém nás po otevření upoutají reklamy. Podíváme-li se pozorněji, uvidíme zde vstupní pole nadepsané *Command line arguments* (parametry příkazové řádky) a přepínače, které umožňují volit, zda budeme našemu programu zadávat data z konzole, nebo zda je zde předem napíšeme (také k tomu se vrátíme později).

V tomto prostoru se bude zobrazovat výstup našeho programu, tedy text, který napíše náš program po spuštění.

2.2 První program

Stalo se tradicí, že první program napíše pozdrav a skončí. To dělá i program, který se ve vývojovém prostředí OnlineGDB Beta zobrazil po otevření. My si ho ale trochu upravíme: Smažeme úvodní řádky č. 1–8, které obsahují *komentář* – text, který je určen programáto-

[3] Číslo za jménem jazyka říká rok, kdy byl vydán jeho mezinárodní standard. Zatím poslední standard jazyka C++ byl vydán v roce 2020. Pro účely prvního seznámení ale můžeme ve skutečnosti vzít kteroukoli z verzí tohoto jazyka.

rovi, ale který neříká počítači, co má dělat. (Když komentář ponecháme nebo i nějak změníme, nic se nestane, program bude fungovat úplně stejně jako bez něj. Navíc zde komentář informuje o účelu prostředí OnlineGDB Beta, takže je pro nás zcela zbytečný.)

Dále odstraníme řádku 16, která je prázdná (jestliže jste odstranili komentář, je to nyní řádka 8). Nakonec změníme text pozdravu, který má program vypsat a který je zapsán v uvozovkách na řádce 7, z

```
"Hello, World"
```

na

```
"Nazdárek, lidičky".
```

Výsledný program bude mít tvar

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Nazdárek, lidičky";
    return 0;
}
```

To je vše.

2.3 Spouštíme první program

Nyní náš program přeložíme a spustíme. K tomu použijeme tlačítko *Run*, které je třetí na liště nad panelem editoru. Jestliže jste program upravili přesně tak, jak jsme si ukázali – musí se shodovat i velikost písmen, nesmějí chybět středníky apod. –, vývojové prostředí náš program úspěšně přeloží a spustí. V panelu ve spodní části vývojového prostředí zmizí nejen reklama, ale i vstupní pole a přepínače a objeví se tam nápis

```
Nazdárek, lidičky
```

který napsal náš program – viz obr. 2.2. Pod ním bude ještě sdělení

```
...Program finished with exit code 0
Press ENTER to exit console.
```

které připojilo vývojové prostředí a které říká, že program skončil s kódem 0 (tedy bez problémů) a že po stisknutí klávesy Enter se toto okno (tzv. *konzole*) uzavře. Když skutečně stiskneme klávesu Enter, tento výstup – tedy text napsaný naším programem a doprovodný komentář – zmizí a vrátí se tam původní obsah.