Sedm tahounů

V tomto článku se podíváme na sedm překladačů jazyka C++, s nimiž se můžeme dnes setkat, a pokusíme se porovnat je z různých hledisek.

PŘEKLADAČE

Porovnávané překladače uvádíme v abecedním pořadí.

Borland C++ Builder 6 (BC6). Byl uveden na trh na počátku roku 2002 jako součást stejnojmenného vývojového nástroje firmy Borland. K porovnání jsme použili provedení Enterprise.

Borland C++BuilderX (BCX). Objevil se ve druhé polovině roku 2003 a je také součástí stejnojmenného vývojového nástroje firmy Borland. I tentokrát jsme k porovnání použili provedení Enterprise.

Borland C++BuilderX - Technology Preview (BCXP). Je součástí nástroje Borland C++BuilderX a lze jej instalovat samostatně. Jde v podstatě o beta verzi překladače, který bude součástí příští verze tohoto nástroje.

GNU mingw32 (GNU). Jde o freewarový překladač g++, verzi 3.2, tj. o překladač GNU C++ pro dvaatřicetibitová Windows, konfigurovaný jako mingw. Lze jej získat z webových stránek GNU nebo například spolu s nástrojem Borland C++BuilderX.

Intel C++ 7.1 (INT71). Tento překladač je součástí sady vývojových nástrojů firmy Intel z roku 2003. Nelze ho však používat samostatně, vyžaduje instalaci Microsoft Platform SDK. Využívá hlavičkové soubory a linker překladačů firmy Microsoft. Pokud je k dispozici MS Visual C++ .NET 2002 nebo 2003, integruje se do jeho IDE jako implicitní překladač. Překladač INT71 existuje ve dvaatřicetibitovém a čtyřiašedesátibitovém provedení; zde jsme se zabývali pouze verzí dvaatřicetibitovou.

Microsoft Visual C++ .NET v. 2002 (VC02). Tento překladač je součástí Microsoft Visual Studia .NET 2002 a umožňuje překlad pro Win32 a pro platformu .NET. Protože ostatní překladače neumožňují překládat programy pro .NET, budeme se tímto překladačem zabývat pouze z hlediska překladu programů pro platformu Win32.

Microsoft Visual C++ .NET v. 2003 (VC03). Nová verze Visual C++ byla uvedena na trh v polovině roku 2003. I o ní budeme hovořit pouze z hlediska překladu programů pro platformu Win32.

O CO NÁM JDE

Překladače jazyka C++ lze srovnávat z řady různých hledisek. V tomto článku se zaměříme na následující kritéria:

shoda implementovaného jazyka C se standardem ISO/IEC 9899:1999,
shoda implementovaného jazyka C++ se standardem ISO/IEC 14882:1998,
rychlost přeložených programů (tedy kvalita optimalizace).

I když na první pohled nejzajímavější je asi porovnání překladačů z hlediska efektivity přeloženého kódu, začneme od shody překladačů se standardy, neboť ta ovlivňuje přenositelnost programů mezi překladači a platformami, a to také není zanedbatelné.

JAZYK C

Nedílnou součástí všech porovnávaných překladačů jazyka C++ je i překladač jazyka C. Pro všechny porovnávané produkty platí, že jako program v jazyce C se implicitně překládá zdrojový soubor, který má příponu .c.

První mezinárodní standard jazyka C, tedy ISO/IEC 9899:1990, byl přijat v r. 1990, takže na konci devadesátých let jej implementovaly již všechny běžně dostupné překladače. Všechny porovnávané překladače kromě GNU také implementují tzv. strukturované výjimky (structured exception handling) a některá další microsoftská rozšíření jazyka C pro Win32.

V roce 1999 však byla přijata nová verze standardu, která přinesla řadu úprav a rozšíření. Ze změn, které nová verze normy jazyka C přinesla, jsme pro účely našeho porovnání vybrali tyto rysy jazyka C:

restringované ukazatele (modifikátor restrict),
inicializaci jen vybraných složek polí a struktur,
vestavěná komplexní čísla, podporu typů long long a unsigned long long,

nekonstantní horní mez v deklaraci lokálního pole, deklaraci zapisovanou mezi příkazy,
literály typu pole a struktura,
funkci s modifikátorem inline, hlavičkový soubor <stdint.h>,
přinášející alternativní přístup k implementovaným celočíselným typům.

Podrobnější porovnání, jež zahrnuje větší počet nově požadovaných rysů, najdete spolu se stručnými poznámkami o významu uvedených konstrukcí v rozšířené verzi tohoto článku na Chip CD, který je přílohou tohoto čísla.

VÝSLEDKY

Podívejme se nyní, jak jednotlivé překladače z tohoto hlediska dopadly. Začneme od věcí, které většina překladačů pomíjí:

Modifikátor restrict v deklaraci ukazatele a inicializaci pouze vybraných složek polí nebo struktur podporuje pouze překladač BCXP a INT71.

Komplexní čísla podle nového standardu (datové typy float _Complex atd.) implementuje pouze překladač BCXP. V omezeném rozsahu ho implementují také GNU a INT71. Ty sice dovolují datové typy pro komplexní čísla použít a provádět s nimi běžné aritmetické operace, neobsahují však hlavičkový soubor <complex.h> s deklaracemi pomocných maker, jako je I, představující imaginární jednotku, a další.

Pokud jde o celočíselné typy long long a unsigned long long, je situace poněkud zmatená. Všechny překladače dovolují deklarovat proměnné těchto typů a zapisovat v programu literály těchto typů. Ve všech případech jde o ekvivalent typů __int64 a unsigned __int64 z prostředí Win32. Žádný z překladačů však nepodporuje standardem požadovanou specifikaci "%lld" (resp. "%llu") pro výstup těchto čísel realizovaný pomocí funkcí z rodiny printf(). Ve všech překladačích je třeba použít specifikaci "%I64d", resp. "%I64u".

Možnost specifikovat horní mez pole pomocí nekonstantního výrazu, možnost zapisovat deklarace kdekoli mezi výrazy a možnost používat literály, jež představují pole nebo struktury, poskytují pouze překladače BCXP, GNU a INT71.

Již po několik let se v překladačích jazyka C objevuje jako rozšíření možnost v deklaraci funkce použít modifikátor __inline, který může vést k rychleji běžícímu přeloženému programu. Současný standard ovšem požaduje klíčové slovo inline (bez úvodních podtržítek). Tomuto požadavku vyhovují pouze překladače BCXP, GNU a INT71, v ostatních musíme použít __inline.

Hlavičkový soubor <stdint.h> můžeme používat v překladačích BC6, BCX, BCXP a GNU; překladače MSVC02, MSVC03 a INT71 ho neposkytují. (Připomeňme si však, že překladač Intel využívá hlavičkové soubory z MS Visual C++.)

Přehledně tyto výsledky shrnuje tabulka 1.

V překladačích BCXP a INT71 je třeba povolit podporu nového standardu jazyka C zvláštním přepínačem. Shrnuto jazykem politiků, je zřejmé, že implementace nových rysů jazyka C nepředstavuje pro tvůrce překladačů C++ prioritu.

JAZYK C++

Zde je situace přece jen veselejší. Nástroje, které využívají běžní programátoři, jsou v převážné většině již stabilní součástí dodávaných překladačů a odchylky od standardu se týkají hlavně rysů tohoto jazyka, které ocení především autoři knihoven. (Existují však i výjimky z tohoto pravidla.) To se týká především šablon: Jejich základní vlastnosti jsou implementovány všemi porovnávanými překladači. Na odchylky od standardu lze však narazit v některých pro běžného programátora spíše okrajových oblastech, jako je např. parciální specializace šablon.

Pro porovnání uvedených překladačů jsme vybrali následující rysy jazyka C++:

Koenigovo vyhledávání (vyhledávání přetížených funkcí v prostorech jmen parametrů),
specifikace typů výjimek, které se mohou z funkce rozšířit,
exportní šablony,
vnořené šablony,
parciální specializace šablon objektových typů,
parciální řazení šablon obyčejných funkcí.

VÝSLEDKY

Začneme opět od vlastností, které překladače nejvíce opomíjejí:

O implementaci klíčového slova export, tedy tzv. exportních šablon, se pokusil pouze jediný z uvedených překladačů, a to BCXP. K tomu je však třeba poznamenat, že řada členů standardizační komise se netají názorem, že tento rys by měl být z jazyka odstraněn.

Možnost specifikovat typy výjimek, které se mohou z dané funkce nebo metody rozšířit, pomocí klíčového slova throw za hlavičkou funkce byla součástí jazyka C++ už v dobách dávno před přijetím standardu. Přesto se kupodivu najdou překladače, které tuto možnost dodnes neimplementují, nebo ji implementují jen omezeně. Překladače INT71, MSVC02 a MSVC03 sice dovolují tuto specifikaci k funkci připojit, ale ignorují ji. Ostatní porovnávané překladače ji implementují v plném rozsahu.

I když vyhledávání jmen přetížených funkcí v prostorech jmen určených operandy (Koenigovo vyhledávání) usnadňuje především používání přetížených operátorů, platí toto pravidlo i pro "obyčejné" funkce. Ač jde na první pohled o neproblematickou záležitost, překladače BCXP a MSVC02 provádějí implementaci pouze pro přetížené operátory.

Překladač MSVC02 sice dovoluje definovat vnořenou šablonu, požaduje však, abychom do těla obklopující třídy zapsali celou její definici, nikoli pouze deklaraci. (Jinými slovy, vnořená šablona musí být v MSVC02 deklarována in-line.) Ostatní porovnávané překladače dovolují deklarovat vnořené šablony uvnitř i vně obklopující funkce.

Překladač MSVC02 nepodporuje parciální specializaci šablon objektových typů ani parciální řazení šablon obyčejných funkcí. Ostatní porovnávané překladače tyto možnosti implementují.

Přehled výsledků porovnávání překladačů z hlediska implementace jazyka C++ najdete v tabulce 2.

EFEKTIVITA PŘELOŽENÉHO KÓDU

Z tohoto porovnávání jsme vypustili překladač BCXP, neboť jde v podstatě o beta verzi Technology Preview sice zní vznešeněji, ale neznamená nic jiného - a nenabízí možnost optimalizace přeloženého kódu.

Pro porovnání jsme zvolili několik testovacích programů, které ukazují různé stránky optimalizace - práci s reálnými čísly, práci s pamětí, efektivitu vstupních a výstupních operací apod. Použili jsme následující programy:

Řešení problému 13 dam. V této úloze jde o to, rozmístit 13 dam na šachovnici o rozměru 13 x 13 polí tak, aby se žádné dvě navzájem neohrožovaly. Program má určit celkový počet možných řešení, nevypisuje je. (Poznamenejme, že tato úloha má přes 70 000 řešení.)

Přečtení 30 000 reálných čísel ze souboru a jejich výpis na konzolu. Ač by se mohlo zdát, že vstupní a výstupní operace, prováděné ve všech případech stejným operačním systémem, zaberou převážnou většinu času, není tomu tak a výsledky se pro jednotlivé překladače liší. Liší se pochopitelně i výsledky pro týž překladač, a to podle toho, zda použijeme vstupní a výstupní funkce z jazyka C (z hlavičkového souboru <stdio.h>), nebo objektové datové proudy z jazyka C++ (z hlavičkového souboru <iostream>).

Zaplňování a vyprazdňování spojového seznamu. Tento program testuje rychlost práce s pamětí tak, že opakovaně (10 000krát) uloží do spojového seznamu 1000 celých čísel a pak tento seznam vyprázdní.

Pro test výpočtů s reálnými čísly jsem použil program, který řeší jistou zvláštní soustavu 40 lineárních algebraických rovnic o 40 neznámých.

Efektivitu implementace ošetřování výjimek jsem testoval na opakovaném volání funkce, která vždy vyvolá výjimku. Testovací program ji volá 100 000krát. (Tomuto kritériu však není třeba přikládat příliš velkou váhu - koneckonců, výjimky by měly být v programech výjimečné. Na druhé straně ho však nelze zcela pominout.)

Dobu potřebnou pro jednotlivé testovací programy jsem měřil pomocí knihovní funkce ftime(), jež sice není součástí standardu, je však natolik běžným rozšířením, že ji implementují všechny testované překladače. Testovací programy jsem spouštěl na PC vybaveném 1GHz procesorem Athlon a 512 MB RAM pod operačním systémem Windows 2000. Každý program jsem spustil desetkrát jako jedinou aplikaci v dosovém okně a tabulka 3 ukazuje průměry dosažených časů.

I když jsou dosažené časy poměrně vyrovnané, lze z nich získat představu o kvalitě porovnávaných překladačů.

CO JE TEDY NEJLEPŠÍ?

Je asi jasné, že na tuto otázku neexistuje jednoduchá odpověď. Přesto z tohoto porovnání můžeme udělat následující závěry:

Z hlediska implementace nových rysů jazyka C je standardu nejbližší BCXP a INT71, ovšem překladač BCXP zatím není prakticky použitelný. Na druhém místě je GNU. Ostatní porovnávané překladače jsou z tohoto hlediska v podstatě nevyhovující.

Z hlediska kvality implementace jazyka C++ jsou nejlepší překladače BC6, BCX a GNU.

Z hlediska efektivity přeloženého programu vítězí možná trochu překvapivě MSVC02, tedy starší verze microsoftského překladače. (Možná jde o důsledek skutečnosti, že nová verze překladače

implementuje mnohé z rysů C++, které předchozí verze opomíjela, a proto se mohla více soustředit na optimalizaci.) Na opačném konci spektra najdeme překladač GNU.

Podrobnější článek, zahrnující nejen větší počet kritérií, ale i ukázky zdrojového kódu, stručný výklad o použitých testech apod., najdete na Chip CD v rubrice Chip Plus.

Miroslav Virius

Tab. 1: Shoda implementovaného jazyka C se standardem ISO/IEC 9899:1999							
	BC6	**BCX**	**BCXP**	**GNU**	**INT71**	**VC02**	**VC03**
Nekonstantní meze polí	ne	ne	ano	ano	ano	ne	ne
Modifikátor restrict	ne	ne	ano	ne	ano	ne	ne
Deklarace mezi příkazy	ne	ne	ano	ano	ano	ne	ne
Pojmenované inicializátory	ne	ne	ano	ne	ano	ne	ne
Strukturové literály	ne	ne	ano	ano	ano	ne	ne
Typy _Complex	ne	ne	ano	ano (?)	ano (?)	ne	ne
Typy long long	ano (?)	ano (?)	ano (?)	ano (?)	ano (?)	ano (?)	ano (?)
Hlavička <stdint.h>	ano	ano	ano	ano	ano	ano	ano
Funkce inline	__inline	__inline	ano	ano	ano	__inline	__inline

Tab. 2: Shoda implementovaného jazyka C++ se standardem ISO/IEC 14882:1998							
	BC6	**BCX**	**BCXP**	**GNU**	**INT71**	**VC02**	**VC03**
Koenigovo vyhledávání	ano	ano	jen op.	ano	ano	jen oper.	ano
Exportní šablony	ne	ne	ano	ne	ne	ne	ne
Vnořené šablony	ano	ano	ano	ano	ano	jen inline	ano
Parciální specializace	ano	ano	ano	ano	ano	ne	ano
Parciální řazení	ano	ano	ano	ano	ano	ne	ano
Specifikace throw	ano	ano	ano	ano	ne	ne	ne

Tab. 3: Rychlost přeložených programů (kvalita optimalizace)						
	CB6	**CBX**	**GNU**	**INT71**	**VC02**	**VC03**
13 dam	7,5	7,5	13,8	5,5	6,1	6,0
stdio	6,6	3,4	3,2	3,0	2,6	3,0
iostream	7,2	8,2	11,2	8,5	2,6	8,5
paměť	3,0	2,9	7,2	5,3	5,3	5,0
soustava	2,4	1,6	2,8	1,6	1,3	1,9
výjimky	0,8	7,4	1,7	0,6	0,5	0,6