C++ programok fordítása

Pataki Norbert

2012. február 24.

Információk

- Pataki Norbert, `patakino@elte.hu`
- `http://patakino.web.elte.hu/Levelezo`
- Jegy: gyakorlat, Szűgyi Zalán

Bevezetés

- A processzor csak a gépikódot érti, pl. C/C++ kódot nem tud egyből végrehajtani.
- Futtatás előtt a C/C++ programokat le kell fordítani.
- Fordítóprogramot használunk (pl. g++).
- Fordítóprogramok a programok bizonyos tulajdonságait ellenőrzik, ez növeli a program helyességének esélyét, de nem garantálja azt.
- A fordítóprogramok optimalizálhatják a kódot.

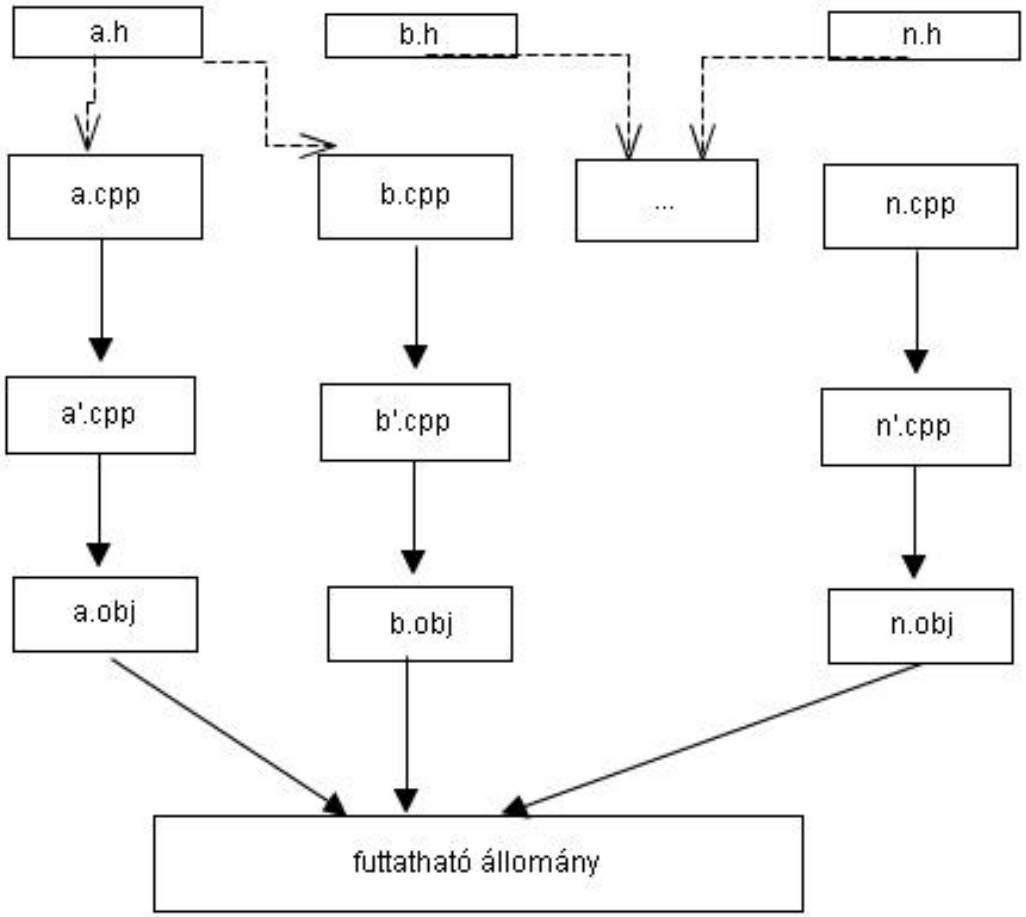
Modularizálás

- Egy programot általában több (sok) file-ban írunk meg, *modularizáljuk*
- Csapatmunka növelése
- Karbantarthatóság: könnyebb bizonyos kódrészleteket megtalálni
- Csak a szükséges file-okat fordítsuk újra
- .cpp/.c/.h

A fordítás lépései

1. A *preprocesszor* feldolgozza a preprocesszor direktívákat.
2. A ,,nyelvi" fordítóprogram az átalakított magasszintű kódból alacsony szintű *tárgykódot* hoz létre.
3. A *linker* összeszerkeszti a tárgykódokat egy futtatható állománnyá.

A fordítás folyamata



Fogalmak

- A lefordítandó program a *forrásprogram*.
- A fordítás eredményeként kapott program a *tárgyprogram*.
- Az az idő, amikor az utasítások fordítása folyik, a *fordítási idő*.
- Az az idő, amikor az utasítások végrehajtása folyik, a *végrehajtási idő*.

Preprocesszor

- Szövegátalakító rendszer
- Nem ismeri a C++ nyelvet, független eszköz
- search & replace
- szöveg bemásolása adott pozícióra
- szöveg kivágása, benntartása feltételtől függően
- Használatát minimalizáljuk (helyette C++ *nyelvi* konstrukciók)

Preprocesszor direktívák

- ```
  #include "filenev"
  #include <iostream>
  ```
- ```
  #define N 20
  #define MAX(a,b) (((a)<(b))?(b):(a))
  ```
- ```
  #undef N
  ```
- ```
  #ifndef N
  #ifdef N
  #if
  #else
  #elif
  #endif
  ```
- ```
  #warning
  #error
  ```
  ```

Include guard

date.h:

```
#ifndef DATE__H
#define DATE__H

class Date { ... };


#endif
```

a.h:

```
#include "date.h"
...
```

b.h:

```
#include "date.h"
#include "a.h"
```

Preprocesszor: tömb méret paraméterezése

```
#define N 20
#define M 30
...
int x[ N ];
int y[ N ];
int z[ M ];

for( int i = 0; i < N; ++i )
{
  y[ i ] = x[ i ];
}
```

Preprocesszor: Kommentek

```
#ifdef KOMMENT

... /* magyarazat */
... /* magyarazat */
... /* magyarazat */
...


#endif
```

Preprocesszor: C/C++

```
#ifdef __cplusplus
extern "C" {
#endif

// ...

#ifdef __cplusplus
}
#endif
```

A preprocesszor veszélye

```
#define MAX(a,b) a > b ? a : b

MAX( x, y )*2  -->   x > y ? x : y*2

#define MAX(a,b)    ((a) > (b) ? (a) : (b))

MAX( ++x, y ) -->    ++x > y ? ++x : y
```

Preprocesszor hibák

```
pperror.cpp:1:1: error: unterminated #ifndef

pperror.cpp:1:17: error: aaa.h: No such file or directory

pperror.cpp:1:2: error: invalid preprocessing
  directive #inddd
```

Header file-ok

Mi ne kerüljön header file-ba?

- Függvénydefiníció (függvénytörzs)
- Globális adatdefiníció
- Névtelen névtér

Mi kerülhet header file-ba?

- Osztálydefiníció include guard-dal ellátva (esetleg helyben kifejtett metódus törzsekkel)
- Sablon osztályok teljes implementációja
- Konstans definíció

C++ nyelvi fordító

- Feldolgozza a preprocesszor által átalakított forráskódot.
- A külön fordítási egységek nem feltétlenül egy időben fordulnak le.
- Ellenőrzi, hogy a nyelv szabályainak megfelel-e a kód.
- Esetlegesen warning-okat és fordítási hibákat ad vissza.
- Ha nincs fordítási hiba, akkor az alacsony szintű tárgykód létrejön.

Compiler hibák

```
comperror.cpp: In function 'int main()':
comperror.cpp:7: error: invalid conversion from 'int'
  to 'const char*'
comperror.cpp:7: error:   initializing argument 1
  of 'void f(const char*)'

comperror.cpp: In function 'int main()':
comperror.cpp:8: error: increment of read-only
  variable 'i'
```

Linker

- A linker feladata a külső szerkesztésű hivatkozások feloldása (pl. függvények, globális változók).
- A linker lefordított tárgykódokkal (object-ekkel) dolgozik.
- A linker ,,nyelvfüggetlen".
- Különböző nyelven írodott kódokból készült object-ek összeszerkeszthetők (pl. Assembly, Ada, Pascal, C, C++, stb.).
- Több object-ből könyvtár állítható össze statikus vagy dinamikus linkeléshez (.a, .lib, .dll, .so).

Compiler/Linker

a.cpp → a.obj:

```
int fac( int );

int main()
{
  std:: cout << "4! == " << fac( 4 ) << std::endl;
}
```

utilities.cpp → utilities.obj:

```
int fac( int i )
{
  // ...
}
```

Linker hibák

```
/tmp/ccChVOFY.o: In function ‘main’:
linkererror.cpp:(.text+0x5): undefined reference to ‘f()’
collect2: ld returned 1 exit status

/tmp/ccyT9yAH.o: In function ‘f()’:
a2.cpp:(.text+0x0): multiple definition of ‘f()’
/tmp/ccG87vIa.o:a1.cpp:(.text+0x0): first defined here
collect2: ld returned 1 exit status
```

Linkelés

- Statikus
- Dinamikus (dll, so)
  - Betöltődéskor
  - Igény szerint

Tulajdonságok

- Statikus
  - csak egyszer kell megcsinálni, nem minden futtatáskor
- Dinamikus
  - Nagy könyvtárak megoszthatóak különböző alkalmazások között, kisebb méretű futtatott programok
  - Verziókhoz könnyebben alkalmazkodó programok
  - ,,DLL-hell”

Warning

```
int i = -2;
unsigned int j = 5;

if ( i < j )
{
  std::cout << "OK";
}
else
{
  std::cout << "Nem OK";
}
```

```
warning.cpp: In function 'int main()':
warning.cpp:8: warning: comparison between signed
  and unsigned integer expressions
```

Warning

```
void f( char* p )
{
   //...
}

f( "Hello" );
```

```
warning.cpp: In function 'int main()':
warning.cpp:9: warning: deprecated conversion from
  string constant to 'char*'
```

Warning

```
int i, j;
std::cin >> i >> j;

if ( i = j )
{
  // ...
}
```

```
warning.cpp: In function 'int main()':
warning.cpp:8: warning: suggest parentheses around
  assignment used as truth value
```

Warning

```
// kivétel: main
int f( int i )
{
   // ...
   // nincs return a függvényben
}

warning.cpp: In function 'int f(int)':
warning.cpp:6: warning: control reaches end of
  non-void function
```

Warning

```
void f( int i )
{
  // i-t nem használjuk a függvény törzsében
  // ...
}
```

```
warning.cpp:3: warning: unused parameter 'i'
```

Warning

```
void f( int )
{
  // ...
}

// Nincs warning...
```

Más nyelvek

- Fordítóprogramok
- Interpreterek
- Virtuális gépek

Interpreter

- Az interpreter program beolvassa a forrásprogramot, és azonnal végrehajtja az utasításokat.
- Nem fordul gépi kód
- Dinamikus típusrendszer
- Flexibilitás
- Kevésbé hatékony

Más nyelvek

Néhány interpretált nyelv:

- Perl
- PHP
- Javascript
- shell

Virtuális gép

- pl. Java, .NET
- A forráskódot lefordítjuk egy univerzális tárgykódra (Java: bytecode)
- Ellenőrzések, optimalizációk végrehajtása
- Ez a tárgykód egy virtuális számítógép által végrehajtható forma
- A virtuális gép lényegében egy interpreter, de nem ,,forráskódot" értelmez, hanem ezt a ,,tárgykódot".
- JIT fordítás: futás közben a bytecode-ból gépi kódot fordít (pl. VM sokszor meghív egy függvényt; futásidejű statisztikák alapján)