Aplikasi Translator Kode Dari Bahasa C ke Pascal

Ipam Fuaddina Adam

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Jl. Ganesha 10, Bandung, Indonesia

adam.first84@gmail.com

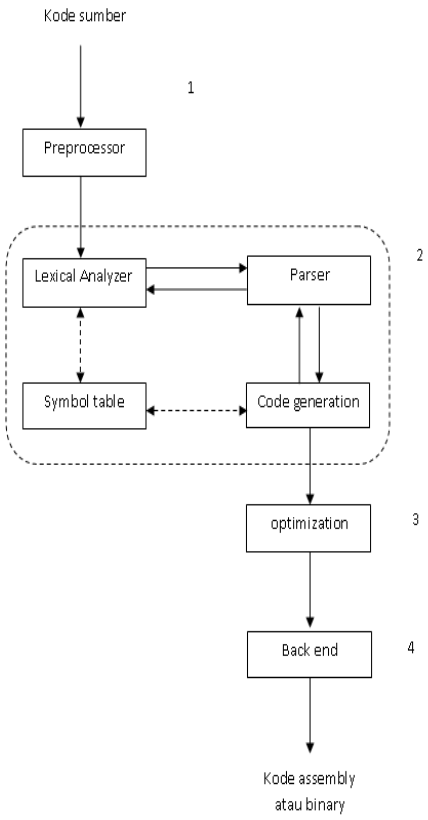
Abstraksi

Translator merupakan solusi otomatis untuk proses penerjemahan kode sumber program dari satu bahasa ke bahasa lainnya. Translator C ke Pascal bekerja dengan melakukan parsing kode sumber, lalu menerapkan aturan penerjemahan, dan terakhir menulis hasil terjemahan ke keluaran. Komponen scanner dan parser dapat dibuat dengan kakas. Untuk mendapatkan aturan penerjemahan yang dimaksud, perlu dilakukan perbandingan grammar dari C dan Pascal. Elemen grammar yang harus dieksplorasi adalah struktur program, deklarasi (variabel, tipe, konstanta, dll), ekspresi, statement, aturan penamaan obyek program, dll. Kemiripan antara C dan Pascal memungkinkan keduanya untuk saling diterjemahkan secara fully translation.

1. Pendahuluan

Proses penerjemahan kode perangkat lunak dari satu bahasa ke bahasa lain seringkali menjadi kebutuhan dalam dunia industri. Ada berbagai alasan yang melatarbelakangi hal ini, misal perubahan proses bisnis, integrasi, regulasi perusahaan, adaptasi dengan teknologi baru, *hardware*, dll. Proses konversi kode tidaklah mudah, apalagi jika kode sumber aplikasi yang diterjemahkan kompleks dan memiliki ukuran yang besar. Proses konversi akan memakan banyak waktu dan tenaga, serta rawan terhadap *error*. Maka diperlukan suatu solusi untuk melakukan proses konversi secara otomatis dengan bantuan program. Program yang menerjemahkan kode sumber ke bahasa lain yang dikehendaki disebut translator.

Translator adalah program yang menerjemahkan suatu kalimat dari bahasa asal ke bahasa target. Mirip dengan kompilator, translator memiliki komponen berupa : *scanner, parser,* dan *code generator*[HOL92]. *Scanner* berfungsi memecah kode sumber menjadi rangkaian *token*, *parser* berfungsi mencocokkan rangkaian *token* dengan *grammar* bahasa, sedangkan *code generator* berfungsi menerapkan aturan penerjemahan dan menulis hasil keluaran. *Scanner* dapat dibuat secara otomatis dengan program Lex / Flex. Demikian juga *parser* dapat dibuat dengan program Yacc/Bison. Berikut adalah skema alur kerja dan komponen dari kompilator



Skema kompilator [HOL92]

Desain kompilator, sebagaimana digambarkan diatas, secara langsung mempengaruhi desain translator. Hanya saja, dalam prakteknya, kompleksitas implementasi translator jauh lebih sederhana daripada kompilator. Semakin banyak kemiripan antara bahasa sumber dan bahasa target, implementasi translator juga semakin mudah. Pembuangan fitur-fitur kompleks dari bahasa sumber

yang tidak ada di bahasa target, juga sangat berpengaruh terhadap tingkat kesulitan implementasi suatu translator.

C dan Pascal memiliki banyak kesamaan, diantaranya sama-sama berparadigma prosedural, penerus seri bahasa ALGOL, waktu pertama kali muncul yang hampir bersamaan, dll. C bersifat *robust*, fleskibel, efisien, ekspresif dan permisif [STR87]. C banyak dipakai di lingkungan industri dan sering digunakan sebagai antarmuka dengan *software* sistem. Pascal memiliki sintaks yang jelas dan mudah dimengerti, sehingga banyak dipakai di lingkungan akademik.

Untuk mendapatkan aturan penerjemahan dari C ke Pascal, kita perlu memadankan *grammar* C dan Pascal. Aspek yang perlu ditinjau misalnya : deklarasi variabel, tipe, konstanta, struktur program, ekspresi, *statement*, dll. C bersifat *case sensitive* sedangkan Pascal tidak, hal ini memungkinkan penerjemahan nama suatu obyek C secara *straight forward* ke Pascal dapat menimbulkan *error*. Pascal juga tidak mengijinkan nama struktur menjadi nama variabel, sedangkan C membolehkan. Deklarasi variabel atau tipe pada C diperbolehkan pada tiap blok program, sedangkan pada Pascal, deklarasi harus diawal program atau awal suatu subrutin.

Perbedaan paling mencolok antara C dan Pascal adalah dalam memperlakukan *pointer*. C menganggap *pointer* dan senarai sebagai dua hal yang sama, sedangkan Pascal menganggapnya berbeda. Hal ini menimbulkan banyak masalah. Perbedaan berikutnya adalah C cenderung sangat luwes dalam memperlakukan tipe, sedangkan Pascal cenderung ketat dan kaku. Hal ini menyebabkan penerjemahan beberapa aspek bahasa C ke Pascal tidak bisa dilakukan secara *straight-forward*, namun memerlukan proses tambahan untuk penyesuaian dengan lingkungan Pascal.

2. Lex dan Yacc

Program Lex berfungsi menghasilkan *scanner*. *Input* yang diterima Lex adalah berupa spesifikasi *token* dari bahasa C. Sesuai dengan definisi bahasa C dari [KER87], *token* bahasa C ada 32 buah, yaitu : `auto`, `break`, `case`, `const`, `continue`, dst, hingga `while` (urut alfabet). Berikut adalah contoh isi berkas masukan untuk Flex yang akan mengenali sejumlah token bahasa C

```
[0-9]+          return DIGIT;
[_a-zA-Z]+      return ALPHA;
[_a-zA-Z0-9]+ return ALNUM;
"auto"          return STORAGE;
"break"         return BREAK;
"case"          return CASE;
"char"          return CTYPE;
...
...
"void"          return VOID;
"while"         return WHILE;

"+"             return PLUS;
"++"            return INCOP;
"-"             return MINUS;
"--"            return INCOP;
"*"             return STAR;
"/"             return DIVOP;
"%"             return DIVOP;

"&&"            return ANDAND;
"||"            return OROR;
"!"             return NOT;
```

Flex adalah *freeware* yang dapat diperoleh di [FLE07]. Sedangkan program Yacc berfungsi menghasilkan *parser*. Yacc menerima *input* berupa definisi *grammar* bahasa C. Definisi *grammar* untuk ANSI C dapat dilihat pada [KER87]. Berikut adalah contoh berkas masukan untuk Yacc yang mendefinisikan *grammar* bahasa C

```
%term ICON
%term FCON
%term LC RC
%token SEMI
%token ELLIPSE
%left COMMA
%left OROR
%left ANDAND
```

program : ext_def_list;

ext_def_list : ext_def_list ext_def ;

ext_def :

opt_specifier ext_decl_list SEMI | opt_specifier SEMI |

opt_specifier funct_decl def_list compound_stmt ;

Sebagaimana Lex, Yacc/Bison adalah *freeware* dan dapat diperoleh di [BIS07]. Lex dan Yacc

menghasilkan kode untuk komponen *scanner* dan *parser* dalam bahasa C.

Keuntungan membuat *scanner* dan *parser* menggunakan kakas otomatis seperti Lex dan Yacc adalah jauh lebih hemat tenaga, waktu, meminimalisasi *error* dan mempermudah penanganan *error*. Meski sering diklaim bahwa *scanner* dan *parser* yang dibuat dengan kakas memiliki performa yang kurang dibanding dengan versi *hand-coded*, namun keuntungan menggunakan kakas jauh lebih bermakna daripada sekedar masalah performansi.

3. Code Generator

Code generator merupakan subsistem yang terdiri dari beberapa komponen, berupa sekumpulan struktur data dan subrutin yang digunakan secara intensif untuk menerjemahkan kode dan menulis kode hasil terjemahan ke keluaran. Struktur data yang paling penting ada dua : yaitu tabel simbol dan AST (*Abstract Syntax Tree*). Tabel simbol berfungsi menyimpan semua referensi obyek dalam kode sumber ke memori. Tabel simbol yang diimplementasikan dalam tugas akhir ini adalah tabel simbol yang berbasis fungsi hash. Spesifikasi fungsi hash yang diambil dari [GRU00] didefinisikan sebagai H(x) fungsi Hash pada *string* x

$H_0=0$

$H_i=k*H_{i-1} + C_i$ dimana $1 \leq i \leq n$

$H = BITS (H_n, 30) \bmod N$

Dimana :

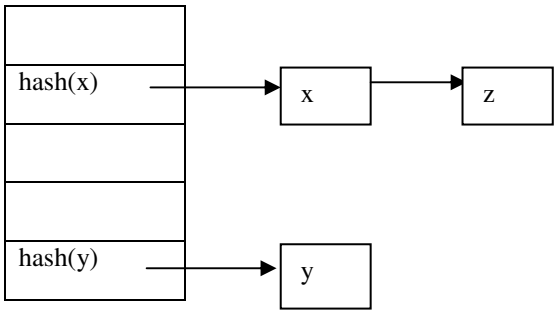
k = 613 N = 1008

n = panjang *string* x

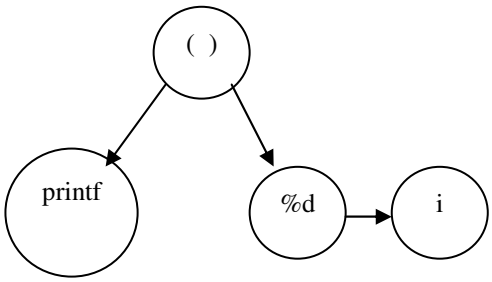
C_i = karakter ke-i dari *string* x

BITS = fungsi yang mengambil nilai 30 bit dari LSB (Least Significan Byte) dari suatu integer

Nilai N pada definisi fungsi hash diatas dipakai sebagai ukuran tabel simbol. Karenanya, ukuran tabel simbol bersifat statik. Tabel simbol diimplementasikan sebagai *array of pointer to linked list*. *Linked list* berisi rangkaian simbol yang memiliki nilai hash yang sama dan sering disebut dengan istilah teknis *collision chain*, karena merupakan bentuk resolusi konflik ketika tabel hash harus mencari suatu *string*, namun nilai hash dari string tersebut juga dipakai oleh *string* lain, sehingga harus dilakukan penelusuran sepanjang list untuk mencari elemen *string* yang dimaksud. Berikut adalah contoh tabel simbol yang menyimpan simbol x, y dan z dimana nilai hash(x) = hash(z)



AST adalah representasi dari pohon *parsing*. AST digunakan untuk menerjemahkan ekspresi dan *statement*. Penerjemahan AST menjadi kode Pascal dilakukan dengan melakukan penelusuran pohon (traversal) dimulai dari simpul akar (*root*) secara DFS (Deep First Search). Berikut adalah contoh AST untuk ekspresi pemanggilan fungsi printf("%d", i);



Karena semua ekspresi bahasa C dapat direpresentasikan sebagai pohon biner [KER87], maka AST juga diimplementasikan sebagai pohon biner. Kumpulan dari semua struktur data dan prosedur ini menghasilkan kode keluaran dengan mengaplikasikan sejumlah aturan terjemahan, hasil dari proses analisis *grammar*.

4. Aturan terjemahan

Dengan cara membandingkan *grammar* C dan Pascal, kita dapat memperoleh aturan penerjemahan dari dua bahasa tersebut. Berikut adalah contoh aturan terjemahan C ke Pascal

a. Tipe dasar

```
int → integer
char → char
double/float → real
```

b. Deklarasi

```
int c → c : integer
struct s p → p : record...end
typedef int INT →
type INT = integer
```

c. Subrutin

```
int foo() →
function foo : integer;
void bar() →procedure bar;
```

d. Ekspresi

```
A = 1 → A := 1
B = 2 / 1 → B := 2 div 1
C = 1 / 2 → C := 1 / 2
```

e. Penamaan ulang

```
int i; char I; →
i : integer; I2 : char;

int program; →
program_ : integer;
```

f. Struktur

```
struct { int x; int y } p →

p : record
x : integer;
y : integer
end;
```

g. senarai dan *pointer*

```
int *pi → pi : ^integer;
int a[4]→ a :
      array[0..3] of integer;
```

5. Program c2p

c2p adalah prototipe program translator yang menerjemahkan berkas berisi kode sumber program dalam bahasa C ke Pascal. Program c2p menggunakan seluruh komponen program translator yang telah disebutkan diatas. c2p menangani *preprocessing* dengan memanggil program eksternal berupa GNU cpp. Pemanggilan program eksternal dilakukan dengan menggunakan antar muka sistem UNIX, yaitu fork( ).

Untuk menerjemahkan kode C menjadi kode Pascal utuh, c2p memecah proses menjadi beberapa tahap, masing-masing tahap berinteraksi dengan tahap sebelumnya lewat *temporary file*. Tidak semua aspek dalam bahasa C diterjemahkan. Fitur kompleks seperti *pointer-to-pointer* dan *pointer-to-function* tidak diterjemahkan.

6. Eksperimen

Untuk menguji validitas output hasil terjemahan program translator c2p, dilakukan pengujian. Berikut adalah contoh proses terjemahan untuk program helloworld [KER87]

```
#include <stdio.h>

main() {

printf( “helloworld\n” );

}
```

Hasil terjemahan oleh c2p adalah

```
program tes;

function main():integer;

begin

writeln( ‘helloworld’ );

main := 0;{ default value  }

end;

begin

main();

end.
```

Kasus uji yang kompleks mengalami kegagalan, misal dalam kasus *structure assignment.* Namun untuk sebagian besar kasus uji fitur sederhana bahasa C, program mampu menghasilkan *output* yang valid.

7. Kesimpulan dan saran

Kemiripan C dan Pascal memungkinkan kode C diterjemahkan ke Pascal secara *fully translation*, dengan menggunakan fitur-fitur kompilator Pascal modern. Masalah paling sulit dalam proses translasi adalah ketika program harus men-*generate* nama baru untuk simbol yang mengalami konflik.

Aplikasi translator C ke Pascal memerlukan manajemen tipe yang cukup kompleks. Sebagian besar masalah muncul karena Pascal tidak seluwes C dalam memperlakukan tipe. Desain AST juga perlu dibuat seluwes dan seelegan mungkin, untuk mempermudah pengembangan program translator, karena AST dapat diperlakukan sebagai *intermediate language* untuk bahasa target lain, selain Pascal, seandainya program translator ini akan dikembangkan lagi di waktu yang akan datang.

8. Daftar pustaka

[BIS07] http://ftp.gnu.org/gnu/bison, tanggal terakhir akses 7 desember 2007

[FLE07] http://flex.sourceforge.net, tanggal terakhir akses 7 desember 2007

[KER87] Kernighan, Brian. Dennis Ritchie.1987. *The C Programming Language*. Prentice Hall

[HOL92] Hollub, Allen I. 1992. *Compiler Design in C*. Prentice Hall.

[GRU00] Grune, Dick dkk.2000.Modern *Compiler Design*. John Wiley & Son's

[STR87] Stroustroup, Bjarne. 1987. The C++ Programming Language. Addison Wesley.

Referensi lainnya :

[1] http://wikipedia.org/c_pascal. *comparing C and Pascal*, tanggal terakhir akses september 2007

[2] Peter, hipson D.1992. *Advanced C*. Sams Publishing

[3] Strastroup, Bjarne. 1987. *The C++ programming language.* Addison Wessley

[4] Brand, Kolman W. 1994. *Problem solving with Pascal*. Nassau college.

[5] Aho, Alfred I, Ravi Sethi, Jeffrey D. Ullmann. 1986. *Compiler Principles, Techniques, and Tools*. Prentice Hall.

[6] Natan.2007.Pengembangan *Compiler* Sederhana Untuk Microcontroller jenis Attiny 2313. Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.