BAB III

METODOLOGI PENELITIAN

3.1. Analisis Masalah

Pada analisis masalah ini, penulis akan menjabarkan mengenai aturan terjemahan yang diberlakukan pada suatu Teks yang dikenali sebagai Pascal.

Idealnya, suatu teks akan dikenali sebagai pernyataan Pascal jika teks tersebut telah sukses melalui proses *compile* pada Turbo Pascal 7.0. Pernyataan Pascal tidak bisa secara langsung diterjemahkan ke bentuk C. Proses penerjemahan pun harus memiliki kualifikasi yang baik dari sudut lama waktu proses yang dibutuhkan. Maka harus diterapkannya beberapa fungsi pengenalan, diantaranya:

1. Penyesuaian algoritma Boyer-Moore pada setiap kasus pencocokan.
2. Proses pencocokan string terhadap teks, dimaksudkan agar setiap kata dapat dikenali oleh translator dan dapat diterjemahkan.
3. Aturan terjemahan, dimaksudkan sebagai aturan yang diterapkan jika ditemukannya sebuah bentuk pernyataan di dalam teks.
4. Mengenali jenis ekspresi yang dioperasikan dan menemukan bentuk hasil operasi dari Pascal ke C.
5. Proses *handling* untuk setiap bentuk penyataan yang tidak memiliki aturan penerjemahan.
6. Metode pemberian pesan kesalahan.

3.2. Analisis Translator

Berikut pemaparan kebutuhan primer/utama translator sebagai alat alih bahasa Pascal ke C.

3.2.1. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum.

Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan *pattern*. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokkan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

3.2.1.1. Cara Kerja Algoritma Boyer-Moore

Dimisalkan ada sebuah usaha pencocokan yang terjadi pada teks[i..i + n − 1], dan anggap ketidakcocokan pertama terjadi di antara teks[i + j] dan *pattern*[j], dengan 0 < j < n. Berarti, teks[i + j + 1..i + n − 1] = *pattern*[j + 1..n − 1] dan a = teks[i + j] tidak sama dengan b = *pattern*[j].

Jika u adalah akhiran dari *pattern* sebelum b dan v adalah sebuah awalan dari *pattern*, maka penggeseran-penggeseran yang mungkin adalah :

1. Penggeseran *good-suffix* yang terdiri dari mensejajarkan potongan teks[i + j + 1..i + n − 1] = *pattern*[j + 1..n − 1] dengan kemunculannya paling kanan di *pattern* yang didahului oleh karakter yang berbeda dengan *pattern*[j]. Jika tidak ada potongan seperti itu, maka algoritma akan

mensejajarkan akhiran v dari teks[i + j + 1..i + n − 1] dengan awalan dari *pattern* yang sama.

2. Penggeseran *bad-character* yang terdiri dari mensejajarkan teks[i + j] dengan kemunculan paling kanan karakter tersebut di *pattern*. Bila karakter tersebut tidak ada di *pattern*, maka *pattern* akan disejajarkan dengan teks[i + n + 1].

Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat mencocokkan string adalah :

1. Algoritma Boyer-Moore mulai mencocokkan *pattern* pada awal teks.
2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi :
   a. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (*mismatch*).
   b. Semua karakter di *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser *pattern* dengan memaksimalkan nilai penggeseran *good-suffix* dan penggeseran *bad-character*, lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks.

3.2.1.2. Pseudocode Algoritma Boyer-Moore

Berikut representasi dari algorima Boyer-Mooyer dengan menggunakan pseudocode :

```
Algoritma Boyer-Moore fase pra-pencocokan, preBmBc.

procedure preBmBc (
  input P : array[0..n-1] of char,
  input n : integer,
  input/output bmBc : array[0..n-1] of integer )

Deklarasi:
  i: integer

Algoritma:
  for (i := 0 to ASIZE-1)
     bmBc[i] := m;
  endfor
  for (i := 0 to m - 2)
     bmBc[P[i]] := m - i - 1;
  endfor
```

Gambar 3.1 Pseudocode algoritma Boyer-Moore preBmBc

Algoritma Boyer-Moore fase pra-pencocokan, preSuffixes.

```
procedure preSuffixes(
  input P : array[0..n-1] of char,
  input n : integer,
  input/output suff : array[0..n-1] of integer )

Deklarasi:
  f, g, i: integer

Algoritma:
  suff[n - 1] := n;
  g := n - 1;
  for (i := n - 2 downto 0) {
    if (i > g and (suff[i + n - 1 - f] < i - g))
      suff[i] := suff[i + n - 1 - f];
    else
      if (i < g)
        g := i;
      endif
      f := i;
      while (g >= 0 and P[g] = P[g + n - 1 - f])
        --g;
      endwhile
      suff[i] = f - g;
    endif
  endfor
```

Gambar 3.2 Pseudocode algoritma Boyer-Moore preSuffix

```
Algoritma Boyer-Moore fase pra-pencocokan, preBmGs.

procedure preBmGs(
  input P : array[0..n-1] of char,
  input n : integer,
  input/output bmBc : array[0..n-1] of integer )

Deklarasi:
  i, j: integer
  suff: array [0..RuangAlpabet] of integer

Algoritma:
  preSuffixes(x, n, suff);

  for (i := 0 to m-1)
      bmGs[i] := n
  endfor
  j := 0
  for (i := n - 1 downto 0)
    if (suff[i] = i + 1)
      for (j:=j to n - 2 - i)
        if (bmGs[j] = n)
            bmGs[j] := n - 1 - i
        endif
      endfor
    endif
  endfor
  for (i = 0 to n - 2)
      bmGs[n - 1 - suff[i]] := n - 1 - i;
  endfor
```

Gambar 3.3 Pseudocode algoritma Boyer-Moore preBmBc

Tabel untuk penggeseran bad-character dan good-suffix dapat dihitung dengan kompleksitas waktu dan ruang sebesar $O(n + \sigma)$ dengan σ adalah besar ruang alfabet.

```
Algoritma Boyer-Moore fase pencocokan, Boyer MooreSearch.

procedure BoyerMooreSearch(
  input m, n : integer,
  input P : array[0..n-1] of char,
  input T : array[0..m-1] of char,
  output ketemu : array[0..m-1] of boolean )

Deklarasi:
  i, j, shift, bmBcShift, bmGsShift: integer
  BmBc : array[0..255] of interger
  BmGs : array[0..n-1] of interger

Algoritma:
  preBmBc(n, P, BmBc)
  preBmGs(n, P, BmGs)
  i:=0
  while (i<= m-n) do
      j:=n-1
    while (j >=0 n and T[i+j] = P[j]) do
        j:=j-1
    endwhile
    if(j < 0) then
        ketemu[i]:=true;
    endif
    bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
    bmGsShift:= BmGs[j]
    shift:= max(bmBcShift, bmGsShift)
    i:= i+shift
  endwhile
```

Gambar 3.4 Pseudocode algoritma Boyer-Moore Search

Pada fase pencarian, algoritma Boyer-Moore membutuhkan waktu sebesar O(mn), pada kasus terburuk, algoritma ini akan melakukan 3n pencocokkan karakter, namun pada performa terbaiknya algoritma ini hanya akan melakukan O(m/n) pencocokkan.

3.2.2. Analisis Aturan Penerjemahan

Aturan penerjemahan adalah hasil hubungan sintaksis pernyataan Pascal terhadap C. Aturan penerjemahan digunakan untuk mengenali dan memecah pernyataan Pascal sehingga dapat diterjemahkan dan disusun kembali ke bentuk

C. Aturan ini diperoleh dari hasil analisis kedua sintaksis. Berikut penjabaran dari kedua sintaksis.

3.2.2.1. Analisis Sintaksis Pernyataan Pascal

Analisis dikaji berdasarkan sintaksis Pascal yang telah ditetapkan. Sedangkan pengaplikasian pernyataan, dikaji berdasarkan aturan di Turbo Pascal 7.0. Pengkajian pertama yaitu Pascal tidak *case sensitif.*

3.2.2.1.1. Pernyataan / *Statement*

Pernyataan Adalah instruksi atau gabungan instruksi, yang menyebabkan komputer melakukan aksi. Ragam pernyataan dalam Pascal terdiri atas :

1. Pernyataan sederhana, setiap jenis pernyataan ini harus diakhiri oleh ";".
   a. Menandai sebuah item data ke sebuah variabel (*assignment statement*).
      Analisis bentukan pernyataan, contoh :

```
   X       :=    Y * 4.135   ;
   ↑       ↑         ↑       ↑
Variabel  Operator   Ekspresi   Tutup
```

      Variabel diasumsikan sebagai sebuah pengenal / *identifier* khusus. Pernyataan ini dikenali jika ditemukannya operator ":=" setelah pengenal dan hanya jika pengenal telah terdaftar sebagai variabel. Setelah itu, operator harus diikuti oleh sebuah ekspresi. Nilai keharusan adalah mutlak, artinya jika tidak diikuti oleh sebuah ekspresi, maka penyataan ini dinyatakan salah. Selain itu, ekspresi harus memiliki hasil operasi dengan tipe data yang sama terhadap variabel.
   b. Pemanggilan pernyataan input, output, dan goto.

Analisis bentukan pernyataan, contoh :

```
Write   (Y * 4.135, 'andrik')    ;
  ↑               ↑              ↑
Pengenal          Ekspresi       Tutup
```

Pengenal write tidak harus selalu diikuti oleh sebuah ekspresi. Ekspresi yang dimiliki write harus diapit oleh "(" dan ")" serta ekpresi tidak diperkenankan berupa karakter kosong. Jumlah ekspresi tidak dibatasi, hanya saja masing-masing ekspresi harus dipisahkan oleh ",". Nilai keharusan adalah mutlak. Aturan ini diterapkan pula pada penggunaan pengenal WRITELN.

```
 Read   (Y, Z)    ;
  ↑       ↑       ↑
Pengenal  Variabel  Tutup
```

Aturan penggunaan pengenal read dapat menggunakan aturan penggunaan pengenal write, hanya saja penggunaan ekspresi pada read tidak diizinkan. Pernyataan ini hanya mengenal variabel. Variabel bernilai benar hanya jika variabel tersebut telah terdaftar. Aturan ini diterapkan pula pada pernyataan READLN.

```
Readkey      ;
   ↑         ↑
 Pengenal    Tutup
```

Readkey dikenali sebagai pernyataan benar jika setelah pengenal diikuti ";". Selain itu readkey dinyatakan salah. Karakter *newline*, *tab*, dan *space* diabaikan.

```
Gotoxy  (y, z)   ;
  ↑       ↑      ↑
Pengenal Ekspresi Tutup
```

Penggunaan pengenal gotoxy dapat diterapkan penggunaan pengenal write. Hanya saja, gotoxy harus diikuti sebuah ekspresi dan ekspresi harus selalu diapit oleh "(" dan ")". Sedangkan ekspresi harus berjumlah 2 dan hasil operasi ekspresi harus bertipe bilangan bulat. Keharusan ini bersifat mutlak.

2. Terstruktur:

   a. Pernyataan perulangan (*Repetitive statement*)

   Analisis bentukan pernyataan, contoh:

```
While  Y > 100    Do    Readkey;
  ↑       ↑       ↑       ↑
Pengenal Ekspresi Pengenal Pernyataan
```

Pengenal while harus selalu diikuti oleh ekspresi yang jika dioperasikan akan menghasilkan tipe boolean, dan harus selalu diikuti oleh pengenal do. Tidak ada aturan khusus mengenai penggunaan pernyataan pada while. Aturan penggunaan pernyataan mengikuti kesesuaian jenis pernyataan dan tidak harus digunakan.

```
Repeat  Readkey;  Until  Y > 100;
  ↑        ↑        ↑        ↑
Pengenal  Pernyataan  Pengenal  Ekspresi
```

Pengenal repeat harus selalu diikuti oleh pengenal until setelah pernyataan. Sedangkan penggunaan pernyataan tidak menjadi sebuah keharusan. Aturan khusus pada pengenal until adalah, satu pernyataan sebelum until dapat menggunakan ";" sebagai penutup maupun tidak.

```
 For     Y:=1+Y     To     Y + 100     Do     Readkey;
  ↑        ↑        ↑        ↑        ↑        ↑
Pengenal  P_Assignment  Pengenal  Ekspresi  Pengenal  Pernyataan
```

Pernyataan di atas harus selalu terdiri dari 3 buah pengenal seperti yang terlihat pada contoh. Pengenal to dapat diganti dengan menggunakan downto. Hal terpenting dalam operasi ekspresi pada P_*Assignment* adalah hasil operasi ekspresi harus bertipe bilangan bulat. Penggunaan pernyataan setelah pengenal do merupakan suatu ketidakharusan.

b. Pernyataan kondisi (*Conditional statement*)

Analisis bentukan pernyataan, contoh :

```
  If    Y > 100   Then   Readkey   Else    Read;
  ↑       ↑       ↑       ↑       ↑       ↑
Pengenal  Ekspresi  Pengenal  Pernyataan  Pengenal  Pernyataan
```

Pada penggunaan pernyataan di atas, aturan pertama yaitu pengenal then hanya dianggap benar jika sebelum pengenal tersebut ditemukan pengenal if, dan else hanya dianggap benar jika di sebelum pengenal tersebut ditemukan then. Sedangkan pengenal if hanya akan

dianggap benar jika setelahnya ditemukan pernyataan then. Kondisi khusus untuk else, pernyataan sebelum else harus tidak diikuti oleh ";", keharusan bersifat mutlak. Sedangkan aturan untuk penggunaan pernyataan, mengikuti aturan pernyataan yang berkesesuaian. Untuk ekspresi, hasil operasi ekspresi harus bertipe boolean, keharusan bernilai mutlak.

c. Pernyataan campuran (*Compound statement*), jenis pernyataan ini dapat terbentuk dari berbagai jenis pernyataan. Perbedaan yang mencolok dari jenis pernyataan ini adalah pada penggunaan blok pernyataan. Pernyataan campuran selalu diapit oleh pernyataan BEGIN dan END.

   Analisis bentukan pernyataan, contoh :

```
Begin                ←  Awalan
       read (x) ;
       y := x + 5;   ←  Pernyataan
       write (y)
End;                 ←  Akhiran
```

Bentuk awalan pada pernyataan ini harus selalu dituliskan sebagai begin dengan tanpa penyatuan terhadap karakter lain selain *newline, tab,* dan *space*, baik di awal maupun di akhir. Selain itu, pernyataan ini harus menggunakan end sebagai penutup blok. End harus selalu diikuti oleh ";". Sedangkan penggunaan begin dan end dapat disetarakan dengan penggunaan pernyataan readkey. Terdapat suatu aturan khusus di pernyataan ini, yaitu satu pernyataan sebelum end diizinkan untuk tidak atau menggunakan ";" sebagai penutup.

3.2.2.1.2. Tipe Data

Jenis – jenis tipe data sederhana yang dikenal dalam bahasa Pascal antara lain yaitu :

1. Integer. Tipe data ini terdiri atas integer positif, integer negatif dan nol yang merupakan nilai bilangan bulat. Pada Turbo Pascal 7.0 jenis tipe data ini di bagi atas beberapa bagian, yaitu :

Tabel 3.1 Tipe data integer Pascal

Tipe	Ukuran Memori (Dalam Byte)	Jangkauan Nilai
Byte	1	0..255
Shortint	1	-128..127
Integer	2	-32768..32767
Word	2	0..65535
Longint	4	-2147483648..2147483647

Operator Integer terdiri atas : + , - , * , / , DIV dan MOD.

2. Real. Penulisan untuk jenis data ini selalu menggunakan titik desimal. Nilai konstanta numerik real berkisar dari 1E-38 sampai dengan 1E+38. E menunjukkan nilai 10 pangkat. Contoh :

123.45 | 12E5 | 12E+5 | -12.34 → penulisan benar.

12345. → salah, titik desimal tidak boleh di belakang

.1234 → salah, titik desimal tidak boleh di muka

Pada Turbo Pascal, jenis data ini dibedakan atas :

Tabel 3.2 Tipe data real Pascal

Tipe	Ukuran Memori (Dalam Byte)	Jangkauan Nilai
Single	4	1.5x10E-45..3.4x10E38
Double	8	5.0x10E-324..1.7x10E308
Extended	10	1.9x10E-4951..1.1x10E4932
Comp	8	-2E+63+1..2E+63-1

Operator untuk tipe data ini terdiri atas : + , - , * dan /

3. Karakter dan string. Tipe data karakter adalah karakter tunggal atau sebuah karakter yang ditulis diantara tanda petik tunggal, seperti misalnya ‘A’,’a’,’!’,’5’ dsb. Dasarnya adalah ASCII Character Set. Sedangkan tipe data string merupakan urut-urutan dari karakter yang terletak di antara tanda petik tunggal. Nilai data string akan menempati memori sebesar banyaknya karakter stringnya ditambah dengan 1 byte. Bila panjang dari suatu string di dalam deklarasi variabel tidak disebutkan, maka dianggap panjangnya adalah 255 karakter.
4. Boolean. Jenis data ini mempunyai nilai TRUE atau FALSE. Operator untuk tipe data ini adalah :
   a. Logical Operator, yaitu : NOT, AND dan OR
   b. Relational Operator, yaitu : >, <, >=, <=, <> dan =

3.2.2.1.3. Ekspresi

Sebuah ekspresi dapat dikenali dengan penggunaan tanda “(“ dan “)” maupun tidak. Penggunaan variabel di dalam sebuah pernyataan, dapat dikategorikan sebagai ekspresi.

1. Numerik / aritmatika dan string. Ekspresi aritmatika adalah semua jenis ekspresi yang jika dioperasikan akan menghasilkan nilai dengan tipe data numerik (integer / real). Begitupun sebaliknya dengan ekspresi string.
2. Boolean / logika. Jenis ekspresi yang jika dioperasikan akan menghasilkan tipe data boolean.

Tabel 3.3 Tipe data hasil operasi ekspresi Pascal

Operator	Tipe Data		
	Operand 1	Operand 2	Hasil Operasi
not	-	Boolean	Boolean
and, or, xor, =, <, >, <>, <=, >=	Boolean	Boolean	Boolean
=, <, >, <>, <=, >=	Char	Char	Boolean
=, <, >, <>, <=, >=	Integer	Integer	Boolean
=, <, >, <>, <=, >=	Integer	Real	Boolean
=, <, >, <>, <=, >=	Real	Real	Boolean
=, <, >, <>, <=, >=	String	String	Boolean
=, <, >, <>, <=, >=	String	Char	Boolean
-, +	-	Integer	Integer
shr, shl, <<, >>	Char	Integer	Integer
div, mod, **-, *, +,** or, xor, and	Integer	Integer	Integer
shr, shl, <<, >>	String	Integer	Integer
-, +	-	Real	Real
-, *, /, +	Integer	Real	Real
/	Integer	Integer	Real
-, *, /, +	Real	Real	Real
+	Char	Char	String
+	String	String	String
+	String	Char	String

3.2.2.2. Analisis Sintaksis Pernyataan C

Di dalam penggunaannya, terdapat beberapa perbedaan yang cukup mendasar mengenai bentukan sebuah pernyataan, ekspresi, maupun tipe data C terhadapa Pascal. Pengkajian penggunaan pernyataan C dilandasi dengan sintaksis C yang ditelah dipatenkan, sedangkan penulisan pernyataan dikaji melalui Turbo C 3.0. Pengkajian pertama yaitu C *case sensitif*. Pada penerapannya, sintataksis pernyataan C hanya digunakan sebagai aturan penggabungan hasil translasi. Oleh karena itu pengkajian dibatasi terhadap pernyataan yang memiliki hubungan terhadap sintaksis pernyataan Pascal yang digunakan sebagai kode sumber. Berikut perbedaan yang cukup mendasar antara Pernyataan C dan Pascal :

1. Tipe data integer dan boolean. Pada C, tidak terdapat tipe boolean. C menggunakan bilangan 0 sebagai false dan selain 0 sebagai true.
2. String kontrol (%SC). Diartikan sebagai penentu format dari sebuah tipe data hasil operasi ekspresi yang digunakan untuk membaca ataupun menampilkan sebuah ekspresi.

Tabel 3.4 Daftar string kontrol C

String Kontrol	Keterangan
%c	Sebuah karakter
%s	Sebuah string
%u	Data bilangan tak bertanda (unsigned) dalam bentuk desimal
%d, %i	Bilangan integer bertanda (signed) dalam bentuk desimal
%o	Bilangan bulat tak bertanda dalam bentuk oktal
%x, %X	Bilangan bulat tak bertanda dalam bentuk heksadesimal
%f	Bilangan real
%e, %E	Bilangan real dalam notasi eksponensial
%g, %G	Bilangan real dengan kesesuaian terhadap format %f atau %E
l	Merupakan awalan yang digunakan untuk %d, %u, %x, %X, %o untuk menyatakan long int. Jika diterapkan bersama %e %E, %f, %F, %g atau %G akan memiliki nilai double.
L	Merupakan awalan yang digunakan untuk %f, %e, %E, %g dan %G untuk menyatakan long double
h	Merupakan awalan yang digunakan untuk %d, %i, %o, %u, %x, atau %X untuk menyatakan short int

3.2.2.3. Analisis Hubungan Sintaksis Pascal dan C

Pada tahapan analisis ini, hubungan antara Pascal dan C akan dikaji berdasarkan hubungan bentukan per pernyataan, ekspresi, maupun tipe data yang apabila dieksekusi memiliki nilai operasi yang sama. Dengan batasan pengkajian bahwa nilai kebenaran masukan kode sumber pernyataan hanya dinilai berdasarkan sintaksis pernyataan Pascal, sedangkan sintaksis pernyataan C hanya sebagai aturan bentukan penggabungan hasil terjemahan.

3.2.2.3.1. Hubungan Pernyataan *Assignment*

```
Pascal → variable := ekspresi ;
   C → variable  =  ekspresi ;
```

Dari kedua bentukan ditemukan hubungan bahwa pernyataan *assignment* dibedakan oleh bentuk operator.

3.2.2.3.2. Hubungan Pernyataan Output

```
Pascal → write(ekspresi);
   C → printf("%sc", ekspresi);
```

Dari kedua bentukan ditemukan hubungan bahwa pernyataan output write pada Pascal memiliki nilai eksekusi yang sama terhadap printf pada C, hanya saja perbedaan muncul pada penyusunan ekspresi Pascal terhadap C. C memungkinkan sebuah string tidak dikategorikan sebagai bagian dari ekspresi. Sedangkan bagian %SC, mewakili setiap ekspresi yang ada secara terurut. Contoh lain untuk pernyataan ini yaitu :

```
Pascal → writeln(ekspresi, 'andrik');
   C → printf("%sc andrik\n", ekspresi);
```

Pada contoh di atas terlihat bahwa pengenal Pascal mengalami perubahan, tetapi tidak pada C. Perubahan C hanya terjadi pada bagian %SC, dimana di akhir penggunaan %SC terjadi penambahan "\n" yang mewakili fungsi "ln" pada pengenal writeln.

3.2.2.3.3. Hubungan Pernyataan Input

```
Pascal →  read(var1, var2);
    C →  scanf("%sc1 %sc2", &var1, &var2);
```

Dari contoh di atas terlihat bahwa read-scanf mempunyai karakteristik yang hampir sama dengan write-printf. Terlihat pula bahwa read-scanf hanya menerima variabel sebagai ekspresi dan penambahan operator alamat "&" sebagai awalan pada bagian variabel. Sedangkan penerapan readln pada C, disimpulkan sama.

```
Pascal →  readkey;
    C →  getch();
```

Readkey mempunyai persamaan fungsi terhadap getch, sedangkan getch harus diikuti oleh ekspresi kosong yaitu "()". Dapat diartikan bahwa readkey selalu di akhiri oleh ";" sedangkan getch di akhiri oleh "();".

3.2.2.3.4. Hubungan Pernyataan Gotoxy

```
Pascal →  gotoxy(ekspresi1, ekspresi2);
    C →  gotoxy(ekspresi1, ekspresi2);
```

Pada dasarnya, aturan penulisan pernyataan gotoxy pada Turbo C tidak berbeda dengan aturan Tubo Pascal. Jika pernyataan Pascal telah benar, maka aturan Pascal dapat diterapkan langsung ke C tanpa perubahan sedikitpun kecuali pada bentuk ekspresi.

3.2.2.3.5. Hubungan Pernyataan Perulangan

```
Pascal →  while ekspresi_logika do penyataan;
    C →  while (ekspresi_logika) pernyataan;
```

Pada contoh di atas, terlihat bahwa ekspresi while C harus selalu diapit oleh "(" dan ")" dan ekspresi harus menghasilkan tipe integer. Selain itu, while C hanya mengenal satu pengenal yaitu while, sedangkan while-do Pascal harus terdapat dua pengenal yaitu while dan do.

```
Pascal →  repeat pernyataan until ekspresi_logika;
    C →  do pernyataan while(not (ekspresi_logika));
```

Pada contoh telihat bahwa ekspresi do-while pada C menggunakan operator not, operator not difungsikan sebagai penyesesuaian sintaksis Pascal dan C. Diketahui bahwa repeat-until pada Pascal akan terus melakukan perulangan jika bernilai false, dan sebaliknya pada do-while C.

```
Pascal →  for p_assignment to ekspresi do pernyataan;
    C →  for (p_assignment; var >= ekspresi; var++)
         pernyataan;

Pascal →  for p_assignment downto ekspresi do
         pernyataan;
    C →  for (p_assignment; var <= ekspresi; var--)
         pernyataan;
```

Dari dua contoh di atas, disimpulkan bahwa pengenal to/downto pada for pascal mempunyai nilai yang sama terhadap *increment/decrement* variabel yang terdapat pada pernyataan *assignment* for Pascal.

3.2.2.3.6. Hubungan Pernyataan Kondisi (*Conditional)*

```
Pascal →  if e_logika then pernyataan;
    C →  if (e_logika) pernyataan;

Pascal →  if e_logika then pernyataan else pernyataan;
    C →  if (e_logika) pernyataan; else pernyataan;
```

Dari kedua contoh di atas, terlihat dua perbedaan sintaksis yaitu pada penggunaan ";" di sebelum pengenal else dan penggunaan ekspresi logika.

3.2.2.3.7. Hubungan Pernyataan Campuran

```
Pascal →  begin pernyataan; end;
    C →  { pernyataan; }
```

Dari contoh di atas terlihat bahwa begin-end memiliki nilai kesamaan terhadap {-}. Sedangkan aturan pernyataan, mengikuti aturan hubungan pernyataan yang berkesesuaian.

3.2.2.3.8. Hubungan Tipe Data

Berikut merupakan tipe data C yang mempunyai nilai tipe yang sama atau mendekati terhadap tipe data Pascal.

Tabel 3.5 Persamaan Tipe Data Pascal ke C

Tipe Data		C String Kontrol
Pascal	C	
boolean	unsigned char	%c
bytebool	unsigned char	%c
longbool	signed char	%c

Tipe Data		C String Kontrol
Pascal	C	
qword	unsigned int	%u
shortint	signed char	%c
smallint	short int	%d

Tabel 3.5 Persamaan Tipe Data Pascal ke C (lanjut)

Tipe Data		C String Kontrol
Pascal	C	
wordbool	int	%d
char	char	%c
byte	unsigned char	%c
cardinal	unsigned int	%u
int64	unsigned int	%u
integer	int	%d
longint	int	%d
longword	unsigned int	%u

Tipe Data		C String Kontrol
Pascal	C	
word	unsigned short int	%u
comp	float	%f
currency	float	%f
double	long double	%Lf
extended	long double	%Lf
real	float	%f
single	double	%lf
string	char	%s

3.2.2.3.9. Hubungan Ekspresi

Pada dasarnya, sebuah ekspresi tersusun oleh operand dan operator. Berikut representasi hubungan ekspresi Pascal ke C melalui keterkaitan operan dan operator.

Tabel 3.6 Persamaan Operator Pascal ke C

Hirarki	Operator	
	Pascal	C
1	not	!
2	*	*
2	/	/
2	div	/
2	mod	%
3	-	-
3	+	+
4	<<	<<
4	>>	>>
4	shl	<<

Hirarki	Operator	
	Pascal	C
4	shr	>>
5	<	<
5	<=	<=
5	>	>
5	>=	>=
6	<>	!=
6	=	==
7	and	&
8	xor	^
9	or	|

Dari tabel di atas, terlihat bahwa terdapat beberapa kesamaan bentuk operator pada Pascal dan C. Ketika operator dioperasikan terhadap operand. Hasil operasi dapat menghasilkan tipe data yang sama maupun berbeda terhadap operand yang menjadi sumber operasi.

3.3. Analisis Perangkat Lunak dan Perangkat Keras

Perangkat yang dibutuhkan dalam pembuatan Translator ini terdiri dari 2 bentuk perngkat, yaitu perangkat keras dan perangkat lunak.

1. Perangkat keras yang dibutuhkan yaitu:

   1 Unit komputer lengkap (netbook), prosessor Intel Atom CPU N280, RAM 2 Gb, VGA 256 Mb, tersedia Mouse

2. Perangkat lunak yang dibutuhkan sebagai sistem operasi yang ada pada komputer dan sebagai media pengembangan Translator antara lain:

   a. Sistem Operasi Windows 7

   b. Program Visual Basic 6.0

   c. Program Microsoft Access 2003