PR 2

Menyadari bahwa setiap aktifitas manusia ada yang memantaunya, maka kami menyatakan bahwa Pekerjaan ini merupakan pekerjaan sendiri, jika ditemukan adanya bentuk kecurangan maka kami bersedia didiskualifikasi dari kuliah ini.

- Daniel Christian Mandolang - 2106630006
- Andrew Jeremy - 2106630082
- Pikatan Arya Bramajati - 2106630031

Link Video: https://youtu.be/bvFfLmtZDWk

Pada tugas ini, kami akan melakukan serangkaian ujicoba untuk membuat variasi program berbasis grammar model kompilator yang telah dipelajari dalam kuliah. Tujuan utama dari tugas ini adalah untuk mengkompilasi dan menjalankan program tersebut, baik yang dirancang secara benar maupun yang sengaja dibuat dengan kesalahan, untuk menguji kemampuan deteksi kesalahan oleh kompilator. Berikut ini adalah hasil ujicoba kami.

1. Deklarasi Variabel

1.1. Deklarasi String

Source code:

```
# tk/test-01-string.txt
{
    var s : string;
    s := "abc"
}
```

Hasil kompilasi:

```
$ java parser < tk/test-01-string.txt
Syntax error at line 2: string
Variable undeclared at line 3: "abc"

Process terminated.
At least 2 error(s) detected.
```

Hasil kompilasi di atas menunjukkan bahwa model compiler tidak mengenali tipe data string. Oleh karena itu, kami mencoba menggunakan array of character sebagai pengganti string sebagai berikut.

Source code:

```
# tk/test-02-arr-of-char.txt
{
    var s[10] : char;
```

```
    s[0] := 'a'
    s[1] := 'b'
    s[2] := 'c'
}
```

Hasil Kompilasi:

```
$ java parser < tk/test-02-arr-of-char.txt
Syntax error at line 2: char
Variable undeclared at line 4: 0

Process terminated.
At least 2 error(s) detected.
```

Berdasarkan hasil kompilasi di atas, kita mengetahui bahwa model compiler juga tidak mendukung tipe data karakter.

1.2. Deklarasi Float

Source code:

```
# tk/test-03-float.txt
{
    var x : float;
    x := 3.14
}
```

Hasil kompilasi:

```
$ java parser < tk/test-03-float.txt
Syntax error at line 2: float
Variable undeclared at line 4: 3

Process terminated.
At least 2 error(s) detected.
```

Source code:

```
# tk/test-04-double.txt
{
    var x : double;
    x := 3.14
}
```

Hasil kompilasi:

```
$ java parser < tk/test-03-float.txt
Syntax error at line 2: double
Variable undeclared at line 4: 3

Process terminated.
At least 2 error(s) detected.
```

Berdasarkan hasil kompilasi di atas, dapat dilihat bahwa model compiler tidak mengenali tipe data float maupun double.

1.3. Deklarasi Integer

Source code:

```
# tk/test-05-int.txt
{
    var x : integer;
    x := 5
}
```

Hasil kompilasi:

```
$ java parser < tk/test-05-int.txt
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 5
13   STORE
14   PUSHMT
15   NAME 0 0
18   SUB
19   POP
20   SETD 0
22   HALT
```

Pada source code di atas tidak terdapat error dan berhasil dikompilasi.

1.4. Deklarasi Boolean

Source code:

```
# tk/test-06-boolean.txt
{
    var x : boolean;
    x := true
}
```

Hasil kompilasi:

```
$ java parser < tk/test-06-boolean.txt
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 1
13   STORE
14   PUSHMT
```

```
15   NAME 0 0
18   SUB
19   POP
20   SETD 0
22   HALT
```

Source code di atas berhasil dikompilasi, artinya tidak terdapat error.

1.5. Kemungkinan Error

1.5.1. Tidak ada titik-koma

Source code:

```
# tk/test-07-no-semicolon.txt
{
    var x : boolean
    x := true
}
```

Hasil kompilasi:

```
$ java parser < tk/test-07-no-semicolon.txt
Syntax error at line 3: x
Syntax error at line 4:

Process terminated for unrecovered error.
2 error(s) found.
```

Dapat dilihat bahwa tk/test-07-no-semicolon.txt sama dengan tk/test-06-boolean.txt dengan menghilangkan titik-koma. Model compiler menangkap error pada line 4 yang menyatakan bahwa variabel x tidak terdeteksi, artinya deklarasi pada line 2 tidak berhasil akibat tidak adanya titik-koma.

1.5.2. Menggunakan simbol assignment yang salah

Source code:

```
# tk/test-08-wrong-assignment.txt
{
    var x : boolean;
    x = true
}
```

Hasil kompilasi:

```
$ java parser < tk/test-08-wrong-assignment.txt
Syntax error at line 3: =

1 error(s) found.
```

Hasil kompilasi menunjukkan bahwa untuk melakukan assignment diperlukan simbol :=. Assignment menggunakan simbol yang salah akan menyebabkan error.

1.5.3. Kelebihan titik-koma

Source code:

```
# tk/test-09-add-semicolon.txt
{
    var x : boolean;
    x := true;
}
```

Hasil kompilasi:

```
$ java parser < tk/test-09-add-semicolon.txt
Syntax error at line 3: ;

1 error(s) found.
```

File tk/test-09-add-semicolon.txt merupakan modifikasi dari file tk/test-06-boolean.txt dengan penambahan titik-koma pada line 3. Hasil kompilasi menunjukkan error pada line tersebut.

2. Conditional Statement

Kita akan coba mengeksplorasi segala hal tentang conditional statement. Untuk itu, kita akan melakukan beberapa modifikasi pada source code berikut:

```
# conditional-base.txt
{
    var x : integer;
    get x
    put x
}
```

Untuk source code awal ini, hasil kompilasinya adalah sebagai berikut:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   READI
12   STORE
13   NAME 0 0
16   LOAD
17   PRINTI
18   PUSH 10
20   PUSH 13
22   PRINTC
```

```
23   PRINTC
24   PUSHMT
25   NAME 0 0
28   SUB
29   POP
30   SETD 0
32   HALT
```

2.1. if

Mari kita lihat apa yang terjadi jika kita tambahkan percabangan bersyarat sederhana pada kode awal kita.

```
# conditional-if.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    end if
    put x
}
```

Program hanya perlu menjalankan isi kode dalam blok if jika kondisi x > 265 terpenuhi. Hasil kompilasi untuk kode ini adalah sebagai berikut:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   READI
12   STORE
13   NAME 0 0
16   LOAD
17   PUSH 265
19   FLIP
20   LT
21   PUSH 30
23   BF
24   NAME 0 0
27   PUSH 265
29   STORE
30   NAME 0 0
33   LOAD
34   PRINTI
35   PUSH 10
37   PUSH 13
```

```
39   PRINTC
40   PRINTC
41   PUSHMT
42   NAME 0 0
45   SUB
46   POP
47   SETD 0
49   HALT
```

Perhatikan bahwa baris 13 sampai 20 berguna untuk mengambil nilai variabel x dan konstanta 265 dan mengomparasikannya untuk mendapatkan status kebenaran dari pernyataan x < 265. Status tersebut digunakan pada operasi BF untuk menentukan apakah perlu melompati beberapa baris atau tidak. Jika x < 265 terpenuhi, maka baris 24 sampai 29 dilakukan, yang merupakan kode di dalam blok if. Jika tidak, maka operasi BF akan membuat jalannya program lompat langsung ke baris 30 (berdasarkan operasi PUSH 30 tepat sebelum BF).

2.2. else

Satu blok if juga bisa berisi blok else pada bagian akhirnya. Kita bisa modifikasi kode menjadi berikut

```
# conditional-if-else.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    else
        x := 0
    end if
    put x
}
```

Jika x > 265 terpenuhi, maka isi kode dalam blok if yang perlu dijalani. Jika x > 265 tidak terpenuhi, maka isi kode dalam blok else yang perlu dijalani.. Hasil kompilasi untuk kode ini adalah sebagai berikut:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   READI
12   STORE
13   NAME 0 0
16   LOAD
```

```
17   PUSH 265
19   FLIP
20   LT
21   PUSH 33
23   BF
24   NAME 0 0
27   PUSH 265
29   STORE
30   PUSH 39
32   BR
33   NAME 0 0
36   PUSH 0
38   STORE
39   NAME 0 0
42   LOAD
43   PRINTI
44   PUSH 10
46   PUSH 13
48   PRINTC
49   PRINTC
50   PUSHMT
51   NAME 0 0
54   SUB
55   POP
56   SETD 0
58   HALT
```

Sama seperti sebelumnya, ada rangkaian operasi yang digunakan untuk mengecek kebenaran pernyataan x < 265, yaitu pada baris 13 sampai 20. Jika pernyataan itu terpenuhi, maka operasi BF akan membuat program tetap lanjut tanpa melompat, sehingga menjalankan operasi-operasi pada baris 24 sampai 29, yang merupakan kode di dalam blok if. Setelah melakukan itu, ada operasi BR yang membuat program melompat ke baris 39. Jika pernyataan itu tidak terpenuhi, maka operasi BF akan membuat program melompat baris 30, sehingga menjalankan baris 32 sampai 38, yang merupakan kode di dalam blok else. Itu berarti, kode dalam blok if hanya dilakukan jika pernyataan terpenuhi, dan kode dalam blok else hanya dilakukan jika pernyataan tidak terpenuhi, sesuai dengan jalan program yang seharusnya.

2.3. Compilation Error

Ada beberapa compilation error yang bisa terjadi pada penulisan percabangan if. Salah satu compilation error yang berhubungan dengan parsing adalah kesalahan pada mulainya dan akhirnya suatu blok, contohnya adalah jika ada blok if yang tidak diakhiri end if.

```
# conditional-no-end-if.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    put x
}
```

Jika kita coba mengompilasi kode di atas, akan muncul pesan error berikut:

```
Syntax error at line 7: }
Syntax error at line 8:

Process terminated for unrecovered error.
```

Pesan error di atas menunjukkan bahwa kurung kurawal tutup yang menutup scope yang dimulai sebelum blok if ditemukan sebelum blok if-nya ditutup oleh end if, sehingga membuat programnya tidak bisa dikompilasi.

Error juga bisa terjadi jika kita menutup suatu blok if yang belum dibuka.

```
# conditional-extra-end-if.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    end if
    end if
    put x
}
```

Baris end if ekstra di akhir tidak berperan dalam menutup blok if mana pun. Jika dikompilasi, hasilnya error-nya seperti berikut:

```
Syntax error at line 7: end

Process terminated.
At least 2 error(s) detected.
```

Ditemukannya perintah end tanpa adanya blok if terdeteksi oleh compiler sehingga program tidak bisa dikompilasi.

Compilation error bisa terjadi jika menggunakan blok else if. Pada kebanyakan bahasa pemrograman, kita bisa menyisipkan nol atau lebih blok else if di antara blok if dan else.

```
# conditional-else-if-incorrect.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    else if x > 26 then
        x := 26
    else if x > 2 then
        x := 2
    else
        x := 0
    end if
    put x
}
```

Seharusnya, blok if paling atas dicek terlebih dahulu. Jika tidak terpenuhi, blok else if di bawahnya dicek. Jika tidak, blok selanjutnya dicek. Dan seterusnya sampai sampai ada kondisi yang terpenuhi atau semua kondisi tidak terpenuhi. Jika semua kondisi tidak terpenuhi, baru masuk ke blok else.

Namun, jika kode di atas dikompilasi, akan muncul error berikut.

```
Syntax error at line 14: }
Syntax error at line 16:

Process terminated for unrecovered error.
```

Mari kita cari tahu mengapa terjadi error. Sebenarnya, else if bukanlah blok yang terdefinisi sendiri. Blok if else yang memiliki satu atau lebih else if sebenarnya terkomposisi dari lebih dari satu blok if else. Jika kode di atas kita tulis ulang dengan mengubah struktur blok if else supaya setiap blok hanya memiliki satu if dan satu else, maka akan menjadi seperti berikut:

```
# conditional-else-if-correct.txt
{
    var x : integer;
    get x
    if x > 265 then
        x := 265
    else
        if x > 26 then
            x := 26
        else
            if x > 2 then
                x := 2
            else
```

```
                x := 0
            end if
        end if
    end if
    put x
}
```

Jika kodenya kita ubah seperti itu, compiler bisa berhasil mengompilasi tanpa masalah. Melihat struktur yang benarnya, bisa dilihat bahwa kode sebelumnya dianggap salah karena tidak semua blok if mempunyai end if yang menutupnya.

3. Looping

3.1. Sintaks dasar

Source code:

```
# tc/repeat.txt
{ var b : integer;
   b := 0
   repeat
      b := b+1
   until b >= 5
}
```

Hasil kompilasi:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 0
13   STORE
14   NAME 0 0
17   NAME 0 0
20   LOAD
21   PUSH 1
23   ADD
24   STORE
25   NAME 0 0
28   LOAD
29   PUSH 5
31   LT
32   PUSH 0
34   EQ
35   PUSH 0
37   EQ
38   PUSH 44
```

```
40   BF
41   PUSH 14
43   BR
44   PUSHMT
45   NAME 0 0
48   SUB
49   POP
50   SETD 0
52   HALT
```

3.2. Tanpa body

Source code:

```
# tc/repeat-with-no-content.txt
{ var a : integer;
   a := 1
   repeat
   until a = 0
}
```

Hasil kompilasi:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 1
13   STORE
14   NAME 0 0
17   LOAD
18   PUSH 0
20   EQ
21   PUSH 0
23   EQ
24   PUSH 30
26   BF
27   PUSH 14
29   BR
30   PUSHMT
31   NAME 0 0
34   SUB
35   POP
36   SETD 0
38   HALT
```

3.3. Dengan scope

Source code:

```
# tc/repeat-with-no-content.txt
{ var b : integer;
    b := 0
    repeat { var a : integer;
      b := b + 1
      a := 5
      if b >= a then
        put a
        exit
      end if
    } until b >= 5
}
```

Hasil kompilasi:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 0
13   STORE
14   NAME 1 0
17   PUSHMT
18   SETD 1
20   PUSH -32768
22   NAME 0 0
25   NAME 0 0
28   LOAD
29   PUSH 1
31   ADD
32   STORE
33   NAME 1 0
36   PUSH 5
38   STORE
39   NAME 0 0
42   LOAD
43   NAME 1 0
46   LOAD
47   LT
48   PUSH 0
50   EQ
51   PUSH 68
53   BF
54   NAME 1 0
57   LOAD
```

```
58    PRINTI
59    PUSH 10
61    PUSH 13
63    PRINTC
64    PRINTC
65    PUSH -32768
67    BR
68    PUSHMT
69    NAME 1 0
72    SUB
73    POP
74    SETD 1
76    NAME 0 0
79    LOAD
80    PUSH 5
82    LT
83    PUSH 0
85    EQ
86    PUSH 0
88    EQ
89    PUSH 95
91    BF
92    PUSH 14
94    BR
95    PUSHMT
96    NAME 0 0
99    SUB
100    POP
101    SETD 0
103    HALT
```

3.4. Tanpa until

3.4.1. Dengan scoping

Source code:

```
# tc/repeat-with-no-until-scope.txt
{ var b : integer;
    b := 0
    repeat { var a : integer;
        get a
    }
}
```

Hasil kompilasi:

```
$ java parser < tc/repeat-with-no-until-scope.txt

Syntax error at line 6: }
Syntax error at line 7:

Process terminated for unrecovered error.%
```

3.4.2. Tanpa scoping

Source code:

```
# tc/repeat-with-no-until.txt
{ var b : integer;
     b := 0
     repeat
       b := b + 1
}
```

Hasil kompilasi:

```
$ java parser < tc/repeat-with-no-until.txt

Syntax error at line 5: }
Syntax error at line 6:

Process terminated for unrecovered error.%
```

3.5. Dengan kondisi until non-expresi

3.5.1. Contoh dengan kondisi false (boolean)

Source code:

```
# tc/repeat-with-non-expression-until.txt
{ var b : integer;
   b := 0
   repeat
      b := b + 1
   until false
}
```

Hasil kompilasi:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 0
13   STORE
```

```
14   NAME 0 0
17   NAME 0 0
20   LOAD
21   PUSH 1
23   ADD
24   STORE
25   PUSH 0
27   PUSH 0
29   EQ
30   PUSH 36
32   BF
33   PUSH 14
35   BR
36   PUSHMT
37   NAME 0 0
40   SUB
41   POP
42   SETD 0
44   HALT
```

3.5.2. Contoh dengan kondisi lainnya

Source code:

```
# tc/repeat-with-non-expression-until-int.txt
{ var b : integer;
   b := 0
   repeat
      b := b + 1
   until 1
}
```

Hasil kompilasi:

```
$ java parser <
tc/repeat-with-non-expression-until-int.txt

Type of boolean expected at line 4: b

1 error(s) found.
```

3.6. Dengan semicolon

Source code:

```
# tc/repeat-with-no-semicolon.txt
 { var b : integer;
    b := 0
```

```
    repeat
      b := b + 1
    until b >= 5;
}
```

Hasil kompilasi:

```
$ java parser < tc/repeat-with-no-until.txt

Syntax error at line 5: }
Syntax error at line 6:

Process terminated for unrecovered error.%
```

3.7. Dengan tanda kurung

Source code:

```
# tc/repeat-with-parenthesis.txt
 { var b : integer;
    b := 0
    repeat
      b := b + 1
    until (b >= 5)
}
```

Hasil kompilasi:

```
0   NAME 0 0
3   PUSHMT
4   SETD 0
6   PUSH -32768
8   NAME 0 0
11   PUSH 0
13   STORE
14   NAME 0 0
17   NAME 0 0
20   LOAD
21   PUSH 1
23   ADD
24   STORE
25   NAME 0 0
28   LOAD
29   PUSH 5
31   LT
32   PUSH 0
34   EQ
35   PUSH 0
37   EQ
```

```
38   PUSH 44
40   BF
41   PUSH 14
43   BR
44   PUSHMT
45   NAME 0 0
48   SUB
49   POP
50   SETD 0
52   HALT
```