CEN 324 - Proje Bölümü II
Son Gönderme Tarihi: 17 Mayıs 2019, 23:50

Hedef için montaj kodu oluşturmak üzere Proje Bölüm I'de yazdığınız ayrıştırıcıyı (**JAVA'da**) genişletme Aşağıdaki dil gramerini için **DERS NOTLARI 8'de** açıklandığı gibi makine . Kullanabilirsiniz montaj kodunu oluşturmaya yardımcı olmak için üç adresli ara kod (**DERS NOTLARI 9**). **Geri kalan talimatlardan bazıları kırmızı olanlar hariç hemen hemen aynıdır (Proje PI için olduğu gibi) .** Genişletilmiş ayrıştırıcı komut satırından bir dosya (dilde yazılmış bir program) kabul etmelidir: **java -jar compiler.jar <dosyaadı>** .

```
program -> stmts
stmts -> stmt stmts | stmt-
stmt -> if-stmt
              | tekrarlamak-stmt eos
              | atamak-stmt eos
              | okuma-stmt eos
              | yazmak-stmt eos

if-stmt -> ifr eğer sonra stmts biter
                    | sonra başka stmts sonu stmts İfade eğer
tekrarla-stmt -> ifade edilene kadar stmts tekrarla
assign- stmt -> id assignop expr
read-stmt -> okuma kimliği
yazma-stmt -> yazma

ifade- > basit-ifade- compop basit-ifade | Basit-expr
simple-expr -> terim { addop terim}
terim -> faktör { mulop faktörü}
faktörü -> id | num

kimlik -> harf +
num -> digit +

mektup -> [a-zA-Z]
basamak -> [0-9]
eos -> ';'
compop -> '<' | '='
addop -> '+' | '-'
mulop -> '*' | '/'
assignop -> ': ='
```

Tüm ayrılmış kelimelerin altı çizilidir ve tüm terminal olmayanlar **kalın harflerle gösterilmiştir** . Giriş / çıkış ifadeler ayrılmış sözcüklerin **okunup yazılmasıyla başlar** . Read deyimi yalnızca birini okuyabilir

her seferinde değişken (id) ve bir write ifadesi bir seferde yalnızca bir ifade yazabilir.

Hedef makine (TM) için doğru montaj kodunu oluşturduğunuzdan emin olun, yani çalışması gerekir ve girdi ve çıktıyı, derlenmekte olan programdaki kodla aynı şekilde üretmek. Okumak Ders notlarından TM'ye ilişkin talimatlar ve örnekler 8. Bunları anladığınızdan emin olun. Bu proje üzerinde çalışmaya başlamadan önce. Diğer çıktılara ek olarak (Proje PI için) derleyici bir derleme dosyası oluşturmalı, örneğin <dosyaadı> .tm. Bu derleme dosyası TM'de çalıştırılacak derleyicinizi test etmek için simülatör.

Anlamsal kurallar:

Değişkenler kullanımla örtülü olarak bildirilir. Tüm değişkenler tamsayı veri türüne sahiptir. İç içe geçmiş kapsam yok, yalnızca küresel kapsam Bu nedenle, sembol tablosunun herhangi bir kapsam bilgisini korumasına gerek yoktur. Var sadece iki basit tip, tamsayı ve boolean. Yalnızca boolean değerler, iki tamsayı değerinin karşılaştırılması. Sadece if-stmt veya repeat-test ifadesinde görünebilirler. stmt-. Bir Boolean değeri write-stmt kullanılarak verilemez.

Aşağıdaki program için tipik bir **sembol tablosu** :

```
x okumak;
y = x + 10;
x yazmak;
y yazın;
x = x + y;
x yazmak;
```

```
Değişken adı Konum                        Satır numaraları
-------------------------------------------------- -
x                          0                 0 1 2 4 4 5
y                          1                 1 3 4
```

Kod oluşturma sırasında değişkenlerin hafıza yerlerine ayrılması gerekecektir, bunu saklıyoruz sembol tablosundaki bilgiler. Şimdilik konumlar, tam sayıdaki basit tam sayı endeksleri olarak görülebilir. her yeni bir değişkenle karşılaşıldığında artar. Ayrıca satırın çapraz referans listesini saklıyoruz değişkenlere erişilen sayılar. Aynı değişkene yapılan birden fazla referansı not edin (bu durumda x in Satır 4) aynı satırda sembol tablosunda bu satır için birden fazla giriş oluşturur.

Yukarıdaki gramer kullanılarak yazılmış programlardan biri (**sample.txt**) aşağıdaki gibidir. İlk bölüm FIBONACCI sayılarını giriş numarasına (n) ve diğer parçalara kadar hesaplar ve yazdırır 32'den az ise, giriş numarasının FAKTÖRÜ (x).

```
      okunan n;
      sonraki: = 0;
      ilk: = 0;
      ikinci: = 1;
      c: = 0;
      tekrar et
            c <1 öyleyse
```

```
                  sonraki: = c;
            son
            c = 1 ise
                  sonraki: = c;
            Başka
                  sonraki: = birinci + ikinci;
                  birinci: = ikinci;
                  saniye: = sonraki;
            son
            c: = c + 1;
            sonraki yaz;
      c = n'ye kadar;

      x okumak;
      eğer x <32 o zaman
            gerçek: = 1;
            tekrar et
                  olgu: = olgu * x;
                  x: = x - 1;
            x = 0 olana kadar;
            gerçeği yaz;
      son
```

Genişletilmiş çözümleyici, **Kaynak Kodunu (ek açıklamalar - belirteçler vb.) Sözdizimi Ağacı olarak** vermelidir. ve programın **Sembol Tablosu** . Diğer hatalara ek olarak (sözdizimi, dilbilgisi, vb.) genişletilmiş ayrıştırıcı türleri denetlemeli ve tür denetimi başarısız olursa, aynı zamanda bir hata da yazmalıdır hat numarası ile mesaj Örnek olarak, açıklamalı kaynak kod, sözdizimi ağacı ve Yukarıdaki programın sembol tablosu aşağıda listelenmiştir:

Kaynak kodu: sample.txt

```
  1: n'yi okuyun;
        1: ayrılmış sözcük: oku
        1: Kimlik, ad = n
        1:;
  2: sonraki: = 0;
        2: Kimlik, isim = sonraki
        2:: =
        2: NUM, val = 0
        2:;
  3: ilk: = 0;
        3: Kimlik, ad = ilk
        3:: =
        3: NUM, val = 0
        3:;
  4: saniye: = 1;
        4: Kimlik, ad = saniye
        4:: =
        4: NUM, val = 1
        4:;
```

```
  5: c: = 0;
        5: Kimlik, isim = c
        5:: =
        5: NUM, val = 0
        5:;
  6: tekrarla
        6: ayrılmış sözcük: tekrar
  7: eğer c <1 ise
        7: ayrılmış sözcük: eğer
        7: Kimlik, isim = c
        7: <
        7: NUM, val = 1
        7: ayrılmış sözcük: o zaman
  8:                sonraki: = c
        8: Kimlik, isim = sonraki
        8:: =
        8: Kimlik, isim = c
  9: son;
        9: ayrılmış sözcük: son
        9:;
 10: eğer c = 1 ise
        10: ayrılmış sözcük: eğer
        10: Kimlik, isim = c
        10: =
        10: NUM, val = 1
        10: ayrılmış sözcük: o zaman
 11:                sonraki: = c
        11: Kimlik, isim = sonraki
        11:: =
        11: Kimlik, isim = c
 12: başka
        12: ayrılmış sözcük: else
 13:                sonraki: = birinci + ikinci;
        13: Kimlik, isim = sonraki
        13:: =
        13: Kimlik, isim = ilk
        13: +
        13: Kimlik, isim = saniye
        13:;
 14:                birinci: = ikinci;
        14: Kimlik, isim = ilk
        14:: =
        14: Kimlik, isim = saniye
        14:;
 15:                saniye: = sonraki
        15: Kimlik, isim = saniye
        15:: =
        15: Kimlik, isim = sonraki
 16: son;
```

```
        16: ayrılmış sözcük: son
        16:;
 17: c: = c + 1;
        17: Kimlik, isim = c
        17:: =
        17: Kimlik, isim = c
        17: +
        17: NUM, val = 1
        17:;
 18: sonraki yaz
        18: ayrılmış sözcük: yazma
        18: Kimlik, isim = sonraki
 19: c = n'ye kadar;
        19: ayrılmış sözcük: kadar
        19: Kimlik, isim = c
        19: =
        19: Kimlik, isim = n
        19:;
 20:
 21: x oku;
        21: ayrılmış sözcük: oku
        21: Kimlik, isim = x
        21:;
 22: eğer x <32 ise
        22: ayrılmış sözcük: eğer
        22: Kimlik, isim = x
        22: <
        22: NUM, val = 32
        22: ayrılmış sözcük: o zaman
 23: gerçek: = 1;
        23: Kimlik, isim = gerçek
        23:: =
        23: NUM, val = 1
        23:;
 24: tekrarla
        24: ayrılmış sözcük: tekrar
 25:                olgu: = olgu * x;
        25: Kimlik, isim = gerçek
        25:: =
        25: Kimlik, isim = gerçek
        25: *
        25: Kimlik, isim = x
        25:;
 26:                x: = x - 1
        26: Kimlik, isim = x
        26:: =
        26: Kimlik, isim = x
        26: -
        26: NUM, val = 1
```

```
 27: x = 0'a kadar;
        27: ayrılmış sözcük: kadar
        27: Kimlik, isim = x
        27: =
        27: NUM, val = 0
        27:;
 28: gerçekleri yaz
        28: ayrılmış sözcük: yazma
        28: Kimlik, isim = gerçek
 29: son
        29: ayrılmış sözcük: son
        30: EOF
```

Sözdizimi ağacı: sample.txt

```
  Oku: n
  Atama: sonraki
    Const: 0
  Atama: ilk
    Const: 0
  Atama: saniye
    Const: 1
  Atama: c
    Const: 0
  Tekrar et
    Eğer
      Op: <
        Id: c
        Const: 1
      Atama: sonraki
        Id: c
    Eğer
      Op: =
        Id: c
        Const: 1
      Atama: sonraki
        Id: c
      Atama: sonraki
        Op: +
          Id: ilk
          Id: ikinci
      Atama: ilk
        Id: ikinci
      Atama: saniye
        Id: sonraki
    Atama: c
      Op: +
        Id: c
        Const: 1
    Yazmak
```

```
      Id: sonraki
    Op: =
      Id: c
      Id: n
  Oku: x
  Eğer
    Op: <
      Id: x
      Const: 32
    Atama: gerçek
      Const: 1
    Tekrar et
      Atama: gerçek
        Op: *
          Id: gerçek
          Id: x
      Atama: x
        Op: -
          Id: x
          Const: 1
      Op: =
        Id: x
        Const: 0
    Yazmak
      Id: gerçek
```

Sembol tablosu: samples.txt

```
Değişken Adı Yer Satır Numaraları
------------- -------- ------------
n                 0             1 19
Sonraki           1             2 8 11 13 15 18
ilk               2             3 13 14
ikinci            3             4 13 14 15
c                 4             5 7 8 10 11 17 17 19
x                 5            21 22 25 26 26 27
gerçek            6            23 25 25 28

Türler Denetleniyor ...

Tip kontrol bitmiş
```

Kaynak koddaki 22. satır şu şekilde değiştirilirse:

```
      eğer x + 32 ise
```

daha sonra kontrol yazın yazdırmalısınız:

```
Türler Denetleniyor ...
22. satırda hata yazın: eğer test Boole değilse

Tip kontrol bitmiş
```

KURALLAR :

1. Onur kod ilkelerine uyun.
2. Ödevinizi dikkatlice okuyun ve G / Ç formatı ile ilgili yönergeleri izleyin (veri dosyası adları, dosya biçimleri, vb.) ve gönderim biçimleri kesinlikle. Bu direktiflerden herhangi birinin ihlal edilmesi cezalandırılır.
3. Kodlama kurallarına uyun.
4. **Online gönderiminiz** aşağıdaki dosyayı ve HİÇBİR ŞEY içermemelidir (veri dosyası içermemeli, nesne dosyaları, vb.):
   P_2 <Firstname> _ <Lastname> _ <öğrenci numarası> _compiler.zip.
5. İngilizce olmayan karakterleri ödevinizin hiçbir bölümünde (beden, dosya adı vb.) Kullanmayın.

ÖNEMLİ

Herkesin kodunu farklı aşamalara bölmesini istiyorum (en azından: sözcüksel, anlamsal ve kod gen) derleyici Her aşama için farklı bir alt dizin (alt paket) yapın.

Bir Makefile gönderin (*tüm* komutun kendiliğinden çalıştırılabilir bir jar dosyası oluşturması gerekir; örneğin, compiler.jar) ve bildirim dosyası. Her üretim kuralını kodlamanız önerilir (if-stmt, repeat-stmt, read-stmt, write-stmt, assign-stmt, expr, simple-expr, terim, vb.) ayrı bir dosya / sınıf / nesne.

En azından kaynak kodunuz en az **dört** dosya içermelidir (örneğin: main.java, lexical.java, semantic.java ve codegen.java) ve **üç** alt dizin (örneğin: derleyici / sözcüksel, derleyici / semantik ve derleyici / kodlayıcı, derleyici kök dizin / pakettir).

Üç java dosyası altında veya Makefile olmadan yapılan başvurular **işaretlerin% 50'sini kaybedecektir** . Bu Seni cezalandırmak değil, hayatında sana yardımcı olacak çok önemli bir pratik beceri öğrenmeni sağlamak için yazılım mühendisi olarak.

Daha fazla açıklama yapmanız gerekiyorsa, bana e-posta yoluyla veya ofiste başvurun.

Notlar: Linux veya cygwin'de geliştirin (www.cygwin.com) Windows'ta GNU markasını kullanmak Yarar.

İYİ ŞANSLAR.