Real-Time Grammar-Based Syntax Highlighter

Özet

Bu projede, Python dili kullanılarak geliştirilen ve gerçek zamanlı vurgulama sağlayan metin editörü uygulaması yapılmıştır.
Uygulama, kullanıcıların yazdığı kodu anında analiz ederek hataları bildirir ve doğru sözdizimine sahip ifadeleri renklendirerek kullanıcıyı bilgilendirir.
Projede, lexical ve parser analiz teknikleri bir araya getirilmiş, kullanıcı arayüzü için ise Python'ın standart kütüphanelerinden biri olan Tkinter kullanılmıştır.

1. Giriş

Editörlerin önemli özelliklerinden biri de sözdizimi vurgulamasıdır. Gerçek zamanlı vurgulayıcılar, kullanıcıya yazım anında geri bildirim sağlayarak, hataların erken tespitini mümkün kılar.
Bu çalışmada, Python diliyle geliştirilmiş, karakter karakter analiz yapan, 11 farklı token türünü tanıyan bir syntax vurgulayıcı tanıtılmaktadır.

2. Yöntem

2.1 Yazılım Mimarisi

Proje iki temel bileşenden oluşmaktadır:
- Sözcüksel analizör (Lexer)
- Sözdizimsel analizör (Parser)

Kullanıcı arayüzü ile bu bileşenler entegre edilerek gerçek zamanlı çalışan bir uyguluma kurulmuştur.

2.2 Sözcüksel Analiz (Lexer)

Lexer, metni karakter karakter inceleyerek her bir token'ı okur ve tanımı ile eşleştirir.
Tanımlanan token türleri şunlardır:

- TYPE: int, float gibi veri tipleri
- KEYWORD: if, while gibi kontrol ifadeleri
- ID: Değişken isimleri (x,y)

- NUMBER: Sayılar (5, 100)
- OP: Operatörler (+, -, =)
- COMPARISON: Karşılaştırma operatörleri (<, >, ==, !=)
- SEMICOLON: Satır sonu (;)
- BRACE: Süslü parantez ({,})
- PAREN: Normal parantez (( ))
- UNKNOWN: Tanımlanamayan karakterler
- COMMENT: Yorum satırları (//)

2.3 Sözdizimsel Analiz (Parser)

Parser, Top-Down (öncelikli) yaklaşım kullanılarak tasarlanmıştır ve parse ağacı önyineleme (preorder) yöntemiyle oluşturulmuştur. Kullanılan gramer kuralları şu şekildedir:

```
program      → stmt*
stmt         → decl_stmt | assign_stmt | if_stmt | while_stmt
decl_stmt    → TYPE ID '=' expr ';'
assign_stmt  → ID '=' expr ';'
if_stmt      → 'if' '(' condition ')' block
while_stmt   → 'while' '(' condition ')' block
condition    → expr COMPARISON expr
block        → '{' stmt* '}'
expr         → term (('+' | '-') term)*
term         → NUMBER | ID
```

2.4 Grafiksel Arayüz (GUI)

Arayüz, Python'ın Tkinter kütüphanesi kullanılarak geliştirilmiştir. Kullanıcı her tuşa bastığında girilen kod anında analiz edilir.
Renklendirme işlemi, etiket (tag) eklenerek gerçekleştirilir.
Arayüzdeki alt bilgi çubuğunda ise analiz sonucu, "✓Syntax OK" ya da "✗Syntax Error" olarak kullanıcı bilgilendirilir.

3. Sonuçlar

Python ile geliştirilen bu söz dizimi vurgulayıcı editör, yazılan kodu gerçek zamanlı olarak analiz ederek anlık bir kullanıcı deneyimi sunmaktadır.
Kod yazımı sırasında hatalar anlık olarak tespit edilmekte ve kullanıcı uyarılmaktadır.
Projenin Java sürümünü tamamlamadaki zorluklar,
Python'un hızlı prototipleme ve kolay GUI geliştirme özellikleri sayesinde aşılmıştır.
Özellikle sözdizimsel analiz kurallarının Python'da uygulanabilirliği,
projenin başarısını önemli ölçüde artırmıştır.