BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
Bilgisayar Mühendisliği Bölümü

Programlama Dilleri Dersi Dönem Projesi

Syntax Highlighting

Hasna Şahinoğlu

22360859005

1. Giriş

Bu proje, C programlama dili için gerçek zamanlı sözdizimi vurgulama ve analiz sistemi geliştirmeyi amaçlamaktadır. Python dili ve PyQt6 arayüz kütüphanesi kullanılarak oluşturulan uygulama; kullanıcıdan alınan C kodunu analiz eder, ardından token bilgilerini kullanarak editör üzerinde renklendirme yapar ve sözdizimsel yapıyı kullanıcıya görsel olarak sunar. Arka planda durum diyagramı ve tablo implementasyonu yöntemini temel alan bir lexer, recursive descent yöntemiyle çalışan bir parser ve bu yapıların çıktısını işleyip GUI'ye entegre eden analiz modülü birlikte çalışır.

2. Language

Projede incelenecek dil olarak C dili, uygulama geliştirme dili olarak Python seçilmiştir.

- **Neden C Dili?**
  C dili, temel yapıları net, sözdizimi zengin ve eğitimde sık kullanılan bir dil olduğu için tercih edilmiştir.
- **Neden Python?**
  Python, hızlı geliştirme imkanı ve PyQt6 kütüphanesi sayesinde gelişmiş arayüz bileşenleriyle gerçek zamanlı sözdizimi vurgulamasını görsel olarak sunmamıza olanak tanımaktadır.

C programlama dilinin sözdizimsel yapısı, büyük ölçüde bağlamdan bağımsız (context-free) kurallara dayanmaktadır ve bu yapı, derleyici tasarımı literatüründe sıkça referans alınan biçimsel gramerlerle uyumludur. Bu çalışmada, C dilinin temel sözdizimsel bileşenleri — örneğin değişken bildirimleri, fonksiyon tanımları, anahtar kelimeler, sabitler ve operatörler — sistematik olarak ele alınmış ve bu yapılar üzerinde sözdizimsel analiz gerçekleştirilmiştir.

3. Syntax Analysis Process

Geliştirilen analiz sistemi, girdi olarak verilen kaynak kodu öncelikle bir lexer aracılığıyla token'lara ayırmakta, ardından bu token dizisini recursive descent yaklaşımıyla çalışan bir parser yardımıyla ayrıştırarak soyut sözdizim ağacı (AST) oluşturmaktadır. AST, analiz edilen kodun yapısal temsilini sunar ve hem hataların tespitinde hem de kullanıcıya geri bildirim sağlanmasında kullanılır. Bu süreç, RealTimeAnalyzer sınıfı ile GUI katmanına entegre edilmiştir ve her metin değişiminde otomatik olarak tetiklenmektedir.

Sistem, C dilinin temel yapılarını tanıyacak şekilde tasarlanmıştır. Desteklenen bazı sözdizim kuralları aşağıdaki gibidir:

program ::= declaration*

variable_declaration ::= type_specifier identifier [array_specifier] [initializer] ';'

type_specifier ::= 'int' | 'float' | 'char' | 'void' | 'double' | 'long' | 'short' | 'signed' | 'unsigned'

array_specifier ::= '[' integer ']'

initializer ::= '=' expression

function ::= type_specifier identifier '(' parameter_list ')' (';' | compound_statement)

parameter_list ::= parameter (',' parameter)* | ε

parameter ::= type_specifier identifier

if_statement ::= 'if' '(' expression ')' statement ['else' statement]

while_statement ::= 'while' '(' expression ')' statement

for_statement ::= 'for' '(' [init_statement] ';' [condition] ';' [increment] ')' statement

init_statement ::= variable_declaration | expression

return_statement ::= 'return' [expression] ';'

compound_statement ::= '{' statement* '}'

expression_statement ::= expression ';'

primary ::= identifier | integer | float | character | string | '(' expression ')'

postfix ::= primary | postfix '[' expression ']' | postfix '(' argument_list ')'

argument_list ::= expression (',' expression)* | ε

unary ::= postfix | unary_operator unary

unary_operator ::= '+' | '-' | '!'

multiplicative ::= unary | multiplicative ('*' | '/' | '%') unary

additive ::= multiplicative | additive ('+' | '-') multiplicative

relational ::= additive | relational ('<' | '<=' | '>' | '>=') additive

equality ::= relational | equality ('==' | '!=') relational

logical_and ::= equality | logical_and '&&' equality

logical_or ::= logical_and | logical_or '||' logical_and

assignment ::= logical_or | logical_or '=' assignment

Parser aynı zamanda çok sayıda hatayla başa çıkabilecek şekilde tasarlanmış, eksik veya bozuk yapılar için senkronizasyon (synchronization) ve toparlama (recovery) mekanizmaları eklenmiştir. Bu sayede sistem, hatalı kodda dahi analiz işlemini sürdürebilmekte ve kullanıcıya geri bildirim sunabilmektedir.

4. Lexical Analysis Details

C dili için geliştirilen lexer, **tablo yönlendirmeli (table-driven)** bir yaklaşımla tasarlanmıştır. Giriş karakterlerine ve mevcut duruma göre geçişleri yöneten bir **durum geçiş tablosu (state transition table)** kullanılmıştır.

Lexer, her karakteri sırayla okuyarak geçerli duruma ve karakter tipine göre bir sonraki duruma geçiş yapar. Bu geçişler sırasında bazı karakterler buffer'a eklenir, bazıları ise token üretimi (emit) ile sonuçlanır. Kullanılan token türleri lexer_base dosyasında görülebilir.

5. Parsing Metodology

Genel Yapı

Bu projede kullanılan parser, **recursive descent** yöntemiyle yazılmıştır ve lexer tarafından üretilen token listesini girdi olarak alarak C kodunun sözdizimsel analizini yapar. Parser, token akışını sıralı şekilde tarar ve yapıya uygun bir **Soyut Sözdizim Ağacı (AST - Abstract Syntax Tree)** oluşturur.

AST (Abstract Syntax Tree)

Parser, her yapısal öğe (örneğin değişken tanımı, fonksiyon bildirimi) için bir ASTNode nesnesi oluşturur. Bu düğümler ağaç yapısında birbirine bağlanır.

Hata Yönetimi

Parser, hatalı yapılarla karşılaştığında:

- Hata mesajı eklenir (add_error)
- Bir sonraki anlamlı yapıya atlayarak analiz sürecini kesintisiz devam ettirir Ayrıca, olası sonsuz recursive çağrılar için maksimum derinlik kontrolü yapılır

6. Highlighting Scheme

Syntax highlighting işlemi, CustomSyntaxTextEditor adında özelleştirilmiş bir QTextEdit sınıfı ile gerçekleştirilmiştir. Bu sınıf, metin değişimlerini dinleyip lexer + parser analizine göre gerçek zamanlı vurgulama uygular.

Çalışma Mantığı

- **Gerçek Zamanlı Dinleme**
  textChanged sinyali üzerinden metin her değiştiğinde on_text_changed() fonksiyonu çağrılır.Performans açısından gereksiz tekrarları önlemek için 100ms gecikmeli (QTimer) çalışır.
- **Syntax Analizi**
  apply_syntax_highlighting() metodu çalıştığında, metni analiz etmek için RealTimeAnalyzer kullanılır. Bu yapı içsel olarak lexer ve parser'ı çağırarak token bilgilerini elde eder.
- **Token Formatlama**
  Analizden gelen token listesi içinde her token'ın:
  - Satır ve sütun bilgisi
  - Değeri (örneğin "int" veya "42")
  - Token tipi (KEYWORD, STRING, IDENTIFIER vs.)
    bulunur. Her bir token, metin içinde uygun konuma denk gelecek şekilde QTextCursor ile seçilir ve QTextCharFormat üzerinden biçimlendirilir.
- **Renkler ve Stiller**
  Token türlerine göre renkler belirlenir (örneğin anahtar kelimeler mavi, yorumlar gri).
  - KEYWORD → Kalın (bold)
  - STRING, CHARACTER, COMMENT → İtalik
  - PREPROCESSOR → Kalın ve renkli

7. GUI Implementation

Uygulamanın arayüzü, LabCodeApp adlı bir QMainWindow sınıfı ile tasarlanmış olup, içerisine C kod editörü, analiz sonuçları ve durum bilgisi gibi bileşenler entegre edilmiştir. Arayüz; PyQt6 altyapısıyla, modern ve bölmeli (splitter-based) bir yapı kullanılarak oluşturulmuştur.

Ana Bileşenler

- **Sol Panel – Kod Editörü**
  CustomSyntaxTextEditor bileşeni, QTextEdit sınıfı üzerine kurulu ve C dili için gerçek zamanlı sözdizimi vurgulaması yapabilmektedir. Kullanıcı kod yazdıkça RealTimeAnalyzer sınıfı ile analiz yapılır ve renklendirme uygulanır.
- **Sağ Panel – Bilgi Alanı**
  - **Token Info**: Lexer tarafından üretilen token'ların listesi.
  - **Parse Tree**: Parser tarafından oluşturulan AST (Abstract Syntax Tree) görünümü.

    - **Status Info**: Gerçek zamanlı durum ve hata mesajları.

- **Durum Çubuğu**
  Editördeki satır ve karakter sayısı gibi bilgileri anlık gösterir.

- **Tema Desteği**
  GUI, isteğe bağlı olarak harici qss dosyasından koyu tema stilini yükleyebilecek şekilde yapılandırılmıştır.

Gerçek Zamanlı Entegrasyon

- textChanged sinyaliyle metin her değiştiğinde analiz tetiklenir.
- Analiz tamamlandığında analysisCompleted sinyali gönderilir ve GUI güncellenir.
- Token ve parse bilgileri, kullanıcıya okunabilir şekilde gösterilir.

8. Özet

Bu proje kapsamında, C programlama diline yönelik gerçek zamanlı sözdizimi vurgulama ve analiz sistemi geliştirilmiştir.

- **Lexer**, tablo yönlendirmeli bir yapıyla tasarlanmış; C dilinin çeşitli token türlerini doğru şekilde ayrıştırabilecek kapsamda geliştirilmiştir.
- **Parser**, recursive descent yaklaşımı ile oluşturulmuş; fonksiyon tanımı, değişken bildirimi, parametre listesi gibi temel gramer yapıları desteklenmiştir.
- **PyQt6 tabanlı kullanıcı arayüzü**, yazılan C kodunun hem renklendirmesini hem de sözdizimsel analiz sonuçlarını gerçek zamanlı olarak sunarak etkileşimli bir deneyim sağlamıştır.
- Eksik veya hatalı kodlara karşı toparlayıcı (error recovery) mekanizmalarla analiz sürecinin kesintisiz devam etmesini mümkün kılmıştır.