BURSA TECHNICAL UNIVERSITY

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BLM0238-PROGRAMLAMA DİLLERİ
BASİT PROGRAMLAMA DİLİ DERLEYİCİSİ

LEVENT KUTAY SEZER
22360859013

1. Giriş

1.1 Projenin Amacı

Bu proje, basit bir programlama dili için lexical analyzer (sözcük çözümleyici) ve parser (sözdizimi çözümleyici) geliştirmeyi amaçlamaktadır.
Proje, kullanıcıların basit programlama yapılarını (değişken tanımlama, kontrol yapıları, aritmetik işlemler) kullanarak kod yazabilmelerini ve bu kodların sözdizimi analizini gerçekleştirebilmelerini sağlar.

1.2 Kullanılan Teknolojiler

- Python 3.x: Ana programlama dili
- Tkinter: Grafiksel kullanıcı arayüzü için
- Regular Expressions (re): Lexical analiz için
- Top-down Parsing: Sözdizimi analizi için

1.3 Projenin Kapsamı

Proje, aşağıdaki temel bileşenleri içermektedir:

- Lexical analyzer (Sözcük çözümleyici)
- Parser (Sözdizimi çözümleyici)
- Grafiksel kullanıcı arayüzü
- Hata yakalama ve raporlama sistemi
- Kod vurgulama sistemi

2. Tasarım

2.1 Token Sınıfı

```python
ANAHTAR_KELIMELER = ["if", "else", "while", "return", "int", "float", "print"]
# Token türleri
TOKEN_TURLERI = {
    "ANAHTAR_KELIME": "ANAHTAR_KELIME",
    "DEGISKEN": "DEGISKEN",
    "SAYI": "SAYI",
    "OPERATOR": "OPERATOR",
    "NOKTALAMA": "NOKTALAMA",
    "BOSLUK": "BOSLUK",
    "STRING": "STRING",
    "BILINMEYEN": "BILINMEYEN"
}
```

- ANAHTAR_KELIME: if, else, while, return, int, float, print
- DEGISKEN: Değişken isimleri
- SAYI: Sayısal değerler
- OPERATOR: +, -, , /, =, >, <, >=, <=, ==, !=
- NOKTALAMA: {, }, (,), ;, ,
- STRING: Metin değerleri
- BOSLUK: Boşluk karakterleri
- BILINMEYEN: Tanımlanamayan karakterler

2.2 Regex Desenleri

```
desenler = [
    (TOKEN_TURLERI["STRING"], r'"[^"\n]*"'),
    (TOKEN_TURLERI["SAYI"], r"\b\d+(\.\d+)?\b"),
    (TOKEN_TURLERI["DEGISKEN"], r"\b[a-zA-Z_][a-zA-Z0-9_]*\b"),
    (TOKEN_TURLERI["OPERATOR"], r"(==|!=|<=|>=|[+\-*/=<>])"),
    (TOKEN_TURLERI["NOKTALAMA"], r"[{}();,]"),
    (TOKEN_TURLERI["BOSLUK"], r"\s+")
]
```

2.3 Parser

```
class Parser:
    def __init__(self, tokenler):
        self.tokenler = tokenler
        self.index = 0
        self.current = self.tokenler[self.index] if self.tokenler else None
        self.has_error = False
        self.hata_mesajlari = []
```

- Değişken tanımlamaları
- Atama ifadeleri
- Aritmetik ifadeler
- Karşılaştırma ifadeleri
- If-else yapıları
- Print ifadeleri

- Token akışı, top-down recursive descent yöntemi ile analiz edilir.

- parse_statement fonksiyonu, gelen token'ları yukarıdaki yapıları kontrol

edecek şekilde tanımlar.

- Her ifade için ayrı hata yakalama sistemi bulunmaktadır.

3. Yöntem ve Uygulama

3.1 Genel Yöntem

Bu projede temel olarak iki aşamalı bir derleyici yaklaşımı uygulanmıştır:

- Lexical Analyzer (lexical.py): Kodları satır satır işleyerek token'lara (sözcük birimlerine) ayırır. Bu işlemde Python'un re modülü (Regular Expressions) kullanılmıştır. Her token bir Token nesnesi olarak saklanır ve türü (ANAHTAR_KELIME, DEGISKEN, SAYI, vb.) ile birlikte konumu ve satır numarası tutulur.
- Parser (topdown.py): Recursive descent (özyineli iniş) yöntemiyle token'ları analiz eder. parse_statement fonksiyonu if, else, while, print, int, float, return gibi ifadeleri algılar ve yapısal doğruluklarını kontrol eder. Hatalı ifadeler satır bazında raporlanır.

3.2 GUI Uygulaması (tkinterapp.py)

- Tkinter Kullanımı: Kod giriş alanı (Text), çalıştırma butonu (Button) ve analiz sonuçlarının gösterildiği etiketler (Label) ile sade bir grafik arayüz oluşturulmuştur.
- Kod Renklendirme: Text widget'ı üzerinden tag_add fonksiyonları kullanılarak ANAHTAR_KELIME, SAYI, DEGISKEN, OPERATOR, NOKTALAMA gibi token türleri için ayrı renkler atanmıştır.
- Gerçek Zamanlı Vurgulama: Kullanıcı her tuş girdiğinde renklendir_metni() fonksiyonu tetiklenir ve sözcükler yeniden analiz edilerek renklendirilir.
- Hata Mesajları: Hatalı girişler kullanıcıya anlamlı açıklamalarla geri bildirilir ve ekranın altında gösterilir.

3.3 Desteklenen Yapılar ve Geliştirmeler

Söz Dizimi Yapıları:

- if (koşul) { ... } else { ... }
- while (koşul) { ... }
- int x = 5;, float y = 3.14;
- print("merhaba");

- Hata Yönetimi: Eksik ;, }, (gibi karakterler için özel hata mesajları hazırlanmıştır.
- Kapsam Genişletme: ANAHTAR_KELIMELER listesi kullanıcıdan gelen talebe göre genişletilmiştir (int, float, print, if, else, return, while).

Bu projede top-down parsing yöntemi tercih edilmiştir çünkü bu yöntem, özellikle dilin sözdizimini açık ve hiyerarşik bir yapıda tanımlamak açısından oldukça elverişlidir. Recursive descent (özyinelemeli iniş) yapısıyla uyumlu olan bu yaklaşım, kodun adım adım analiz edilmesini ve hataların kullanıcıya detaylı şekilde sunulmasını kolaylaştırmıştır. Ayrıca, parser'ın işleyişini modellemek için kullanılan state diagram (durum diyagramı) yöntemi ise, daha önce otomata teorisi dersinde edindiğim bilgi ve alışkanlıklarla uyumludur. Bu diyagram, dilin kurallarını görselleştirerek parser'ın davranışını daha iyi analiz etmeme ve uygulamayı sistematik şekilde geliştirmeme olanak tanımıştır.

4. Sonuç ve Değerlendirme

Bu projede, kullanıcıdan alınan programlama dili benzeri ifadelerin sözdizimsel analizinin yapıldığı, etkileşimli ve görsel geri bildirim sağlayan bir uygulama geliştirilmiştir. Uygulama, lexical analiz ve top-down parsing adımlarını içeren iki aşamalı bir yapı kullanarak çalışmaktadır.
Proje kapsamında:

- Kullanıcının girdiği kod satırları, token'lara ayrılmış, türlerine göre sınıflandırılmış ve anında renklendirilerek vurgulanmıştır.
- Yazım hataları, eksik karakterler (örneğin parantez veya noktalı virgül), desteklenmeyen anahtar kelimeler gibi problemler kullanıcıya detaylı mesajlarla sunulmuştur.
- print, if, else, while, return, int, float gibi temel yapılar desteklenmiş, bu yapılar için hata kontrol mekanizmaları tanımlanmıştır.
- Kullanıcı arayüzü (GUI), basit ve kullanıcı dostu bir şekilde tasarlanmış, hataların ve token'ların ayrı ayrı gösterimi sağlanmıştır.

Uygulamanın testleri sonucunda söz dizimsel olarak doğru girilen örnek kodlar başarılı şekilde analiz edilmiş, hatalı kodlara ise anlamlı hata mesajları verilmiştir. Bu yönüyle uygulama, temel bir sözdizimi analizörü (parser) olarak görevini başarıyla yerine getirmiştir.
Uygulamanın sınırlılıkları arasında şu anda yalnızca temel dil yapılarının destekleniyor oluşu yer almaktadır. İlerleyen çalışmalarda:

- Fonksiyon tanımları,
- Karmaşık mantıksal ifadeler (örn. &&, ||),

- Çok satırlı yorumlar ve daha gelişmiş string işlemleri,