Bu dökümantasyonda anlatılan araç; CPP ve C dosyalarında **değişken**, **sınıf** ve **fonksiyon** isimlendirmelerinin standartlara uyması için aşağıdaki CASE yapılarına uygun olarak yeniden düzenler. Bunu clang-tidy aracına doğal dil işleme tekniklerini python **spiral** kütüpahnesi ile kazandırarak yapmaktadır.

- `lower_case`
- `UPPER_CASE`
- `camelBack`
- `CamelCase`
- `camel_Snake_Back`
- `Camel_Snake_Case`
- `aNy_CasE`
- `Leading_upper_snake_case`

Bu aracın diğer linter araçlarından farkı isimlendirilen değikenler arasında geliştirici hatasından kaynaklanan tek öbek isimlendirmeyi düzeltmesidir.

Örneğin `CamelBack` formatında **`ThisIsAClass`** şeklide adlandırılması gereken bir class var ancak geliştirici bunu **`this_is_a_class`** olarak adlandırmış, bu durumda çoğu linter aracı underscore'lara veya büyük-küçük harflere göre öbeklere ayırarak [this-is-a-class] kelimelerini `CamelBack` formatına rahatlıkla çevirebilmektedir. Ancak; eğer bu yanlış isimlerndirme "**thisisaclass"** şekilde veya "**This_is_aClass**" şeklinde tek öbek veya olması gerekenden farklı öbeklendirerek oluşturulursa linter araçlarının hiçbiri olması gerektiği gibi refaktör edemez. Çünkü **thisisaclass** liter aracı için tek bir kelimedir ve bunun `CamelBack` formatı hatalı bir şekilde **Thisisaclass** şeklinde olacaktır.

Geliştirilen araç Python tabanlı olup, **clang** ve **spiral** kütüphanelerini kullanarak bu sorunu çözmektedir. **Clang** sayesinde C/CPP dosyalarındaki, istenilen declarasyonlar (değişken, fonksiyon, sınıf) alınır. Bu isimler içerisindeki underscore gibi özel karakterler atılır ve küçük harfere dönüştürülür. Ardından **spiral** kütüphanesinden yararlanılarak doğal dil işleme teknikleri ile kelimelere ayırılır. Diğer linter uygulamalarının rahatlıkla anlayabilemsi adına `lower_case` yapısına dönüştürülür. İstenilen case yapısına dönüşüm ise `clang-tidy` aracı kullanılarak yapılmış olur.

Dönüşüm süreci aşağıdaki gibidir:

This_is_aClass → ThisisaClass → thisisaclass → [this,is,a,class] → this_is_a_class → ThisIsAClass

Kurulum ve Kullanım

Klasör içerisinde word_parser.py ve install.sh dosyaları bulunmaktadır.

1. sudo ./install.sh
2. source ~/.bashrc
3. conda activate NLP_refactoring

```
refactor \
--src=./src/ \
--include=./include/ \
--config '{"var":"UPPER_CASE","func":"camelBack","cls":"CamelCase"}'
--include-paths /usr/include/c++/11 /usr/local/cuda/includes
```

4. --src: C/C++ dosyalarının dizini verilmeli. Default: ./
5. --include: Header dosyalarının dizini verilmeli. Default: ./
6. --config: ‘{“var”:CASE, “cls”:CASE, “func”:CASE}’
7. --include-paths: Ek kütüphanelerin dosya dizinleri belirtilir. Ör: /usr/include/c++/11 /usr/include/x86_64-linux-gnu/c++/11 /usr/local/cuda/includes