



**Yıldız Teknik Üniversitesi**  
**Elektrik Elektronik Fakültesi**  
**Bilgisayar Mühendisliği**

BLM3021

Algoritma

Analizi

GR:2

DOÇ.DR. MEHMET AMAÇ GÜVENSAN

Ödev-3

İsim: Muhammed Taha Güneş

No: 21011017

Video Linki: [https://youtu.be/KpU8hR\\_Rn5M](https://youtu.be/KpU8hR_Rn5M)

E.posta: [taha.gunes@std.yildiz.edu.tr](mailto:taha.gunes@std.yildiz.edu.tr)

# 1- Problemin Çözümü:

Kullanıcıdan ilk başta dosya ismi istenilmiştir. Sonrasında normal mod için 1, debug mod için 2 girilmesi istenmiştir. Akabinde dosya açılmıştır. Noktalı virgül haricinde olan sembollere göre stringler parçalanmıştır ve token dizisine atılmıştır. Token dizisi gezilmiştir. Değişken tipi görüldüğü yerde start\_index ile indis değeri saklanmıştır. Bu değer deklare işleminin başladığı satırın başının indisi saklanmıştır. Noktalı virgül ile karşılaşılan indiste satır sonuna gelindiği anlaşılmıştır ve finish\_index bu indis değeri saklanmıştır. Bu iki değer eğer start\_index sıfır değilse finish\_index – start\_index işlemi yapılarak tanımlanan değişken sayısı bulunmuştur ve toplam değişken sayısına eklenmiştir. Sonrasında deklare işleminin bittiğini anlayabilmek için start\_index değeri sıfırlanmıştır. Toplam değişken sayısı bulunduktan sonra bu boyutta bir sembol tablosu oluşturulmuştur. Sonraki adımda üstteki işlemlere benzer şekilde değişkenler bulunmuştur ve içinde noktalı virgül olan değişkenlerden noktalı virgül kaldırma işlemi uygulanmıştır. Lookup fonksiyonunda hash fonksiyonları ve horner methodu kullanılarak değişkenlerin tablodaki yerleri bulunmuştur ve o değerler parametrede gelen değerle aynı ise 1 değeri döndürülmüştür. Tüm boyutu kadar gezildikten sonra değişken tabloda bulunamazsa 0 değeri döndürülmüştür. Değişkenler lookup fonksiyonundan 0 değeri dönerse tabloya eklenmiştir. 1 değeri dönerse 'önceden zaten tanımlanmıştır' hatası verilmiştir. finish\_declare değişkeni ile deklare işlemlerinin bittiği yer bulunmuştur. Ondan sonrasından başlanarak içerisinde '\_' barındıran stringler yani değişkenler bulunmuştur. İçerisinde yine noktalı virgül barındıranlar varsa noktalı virgülü kaldırma işlemi yapılmıştır. Tekrardan lookup fonksiyonuna bakılmıştır ve 0 değeri dönüyorsa 'değişken önceden tanımlanmamıştır' hatası verilmiştir. Normal mod bu şekilde çalışmaktadır. Debug modda ise ekrana istenilen değerler (tablonun son durumu vs.) yazılmıştır. Gerekli diziler serbest bırakılmıştır.

## 2- Karşılaşılan Sorunlar:

İlk karşılaşılan problem değişkenlerin neye göre sayılacağını bulmak olmuştur. Bunun için çözüm noktalı virgülü barındıran değişken indisini satır sonu düşünmek ve satır başlarında değişken tipi görürsek başlangıç adresini saklamak olmuştur. Bunlar arasında fark bulunarak değişken sayısı hesaplanmıştır. 2.problem tüm declare işlemlerinin nerede biteceğini bulmak olmuştur. Bunun için de satır başı indisi 0 yani herhangi bir değişken tipi görülmemişse, tüm deklare işlemi hangi indiste bittiği finish\_declare değişkeninde saklanmıştır. 3.sorun ise değişken isimleri uzun olduğunda tabloda yerleşmesi gereken yer hesabı yani hash hesaplarında sapıtma olduğu fark edilmiştir. Bunun sebebinin horner methodunda key değerinde çok büyük sayılar birikmesi sonucu int yani 4 byte sığmaması olduğu anlaşılmıştır. Bundan dolayı değişken isimleri kısa kullanılmalıdır.

## 3- Karmaşıklık Analizi:

### 1. Dosyadan Token Okuma (readFile)

Bu fonksiyon, dosyadan satır satır veri okumuştur ve her satırı tokenize etmiştir.

- Her satırı okuma işlemi  $O(L)$ , burada  $L$ , dosyadaki toplam satır sayısıdır.
- Satırdaki tokenları parçalama işlemi  $O(n)$ , burada  $n$ , toplam karakter sayısıdır. Tokenize edilen karakterlerin tamamı işlenmiştir, bu yüzden bu fonksiyonun toplam karmaşıklığı:  $O(n)$   $n$ , dosyadaki toplam karakter sayısıdır.

## 2. Asal Sayı Hesaplama (is\_prime ve next\_prime)

next\_prime fonksiyonu, verilen bir sayıdan sonra gelen asal sayıyı bulmuştur.

- Asallık kontrolü is\_prime ile yapılır ve her sayı için  $O(k^{1/2})$  yani karekök k işlem gerekmiştir.
- Bir sonraki asal sayıyı bulmak için k artarak kontrol edilmiştir. Ortalama  $O(k \cdot k^{1/2})$  olmuştur.

Tablo uzunluğu hesaplanırken, tablo boyutunun iki katından sonraki asal sayı bulunmuştur. Pratikte bu işlem oldukça hızlıdır ve sabit büyüklükteki tablolarda  $O(1)$  'e yakınsamıştır.

## 3. Horner Anahtar Hesaplama (horner\_key)

Bir stringin Horner yöntemiyle hash değerini hesaplamak, stringin uzunluğu kadar işlem gerektirmiştir.

- $O(s)$ , burada s, stringin uzunluğudur.

Bu işlem her değişken için bir kez yapılmıştır. Toplamda tüm tokenların boyutuna bağlı olarak  $O(n)$ 'dir.

## 4. Tabloya Değişken Ekleme (insert)

Değişkenin tabloya yerleştirilmesi için:

- İlk olarak h1 ve h2 hash fonksiyonları hesaplanmıştır:  $O(1)$ .
- Double hashing yöntemiyle boş bir yer aranmıştır. Tablonun boyutu m olduğundan, en kötü durumda tüm tabloyu dolaşmak gerekmiştir. Ancak, hash fonksiyonları genellikle yük faktörü  $\alpha$  ile sınırlandırıldığından pratikte  $O(1)$  ortalama zaman karmaşıklığı vardır.

Kötü bir durumda,  $O(m)$ .

## 5. Değişken Arama (lookup)

Bir değişkenin var olup olmadığını kontrol etmek için:

- Hash fonksiyonları hesaplanmıştır:  $O(1)$ .
- En kötü durumda tüm tablo  $O(m)$  zaman alabilir. Pratikte,  $O(1)$  'dir.

## 6. Tokenları İşleyerek Tabloya Ekleme (insert\_table)

Bu fonksiyon, tüm tokenları sırasıyla işleyerek değişkenlerin tabloya eklenmesini ve hataları kontrol etmiştir.

- Token sayısı n'dir ve her token için:
  - lookup: Ortalama  $O(1)$ , kötü durumda  $O(m)$ .
  - insert: Ortalama  $O(1)$ , kötü durumda  $O(m)$ .

Pratikte, bu fonksiyonun toplam karmaşıklığı:  $O(n)$

## 7. Debug Modu (debugMode)

Debug modunda:

- Tabloyu tamamen dolaşmıştır  $O(m)$  ve değişkenlerin adreslerini hesaplamıştır  $O(1)$ . Tablo boyutuna bağlıdır, bu yüzden karmaşıklık:  $O(m)$

### Toplam Karmaşıklık

Kodun genel işleyişini incelersek:

1. Dosyadan tüm tokenları okuma:  $O(n)$
2. Tablonun boyutunu hesaplama ve asal sayı bulma:  $O(k^{3/2})$  (genelde küçük ve sabit bir maliyet).
3. Tokenları tabloya ekleme:  $O(n)$ ,
4. Debug modu (isteğe bağlı):  $O(m)$ .

Tablo boyutu genelde token sayısına bağlı olarak  $O(n)$ 'dir. Yani toplam karmaşıklık:

$O(n)$  (Pratikte lineer çalışır)

Kötü durumda (hash çakışmaları ve tamamen dolu bir tablo için), karmaşıklık:

$O(n+m)$  (Tablonun tamamen dolaşılması dahil)

## 4- Ekran Çıktıları:

### Örnek1:

```
int main()
{
    int _aa, _bb, _cc;
    char _aa;
    char _x;
    _aa = 5;
    _xx = 9;
    _bb = _aa + _dd;
}
```

Normal Mod:

```
Dosyanizin ismini giriniz: ornek1.txt
Normal mode için 1, debug mode için 2 giriniz.
1
Hata: '_aa' degiskeni daha once deklare edilmistir.
Hata: '_xx' degiskeni deklere edilmemistir.
Hata: '_dd' degiskeni deklere edilmemistir.

-----
Process exited after 5.653 seconds with return value 0
Press any key to continue . . .
```

Debug Mod:

```

Dosyanizin ismini giriniz: ornek1.txt
Normal mode icin 1, debug mode icin 2 giriniz.
2
Hata: '_aa' degiskeni daha once deklare edilmistir.
Hata: '_xx' degiskeni deklere edilmemistir.
Hata: '_dd' degiskeni deklere edilmemistir.


Tablonun son goruntusu:
0
1
2
3
4
5
6 int _cc
7 int _bb
8 int _aa
9 char _x
10

Deklere edilmiş degisken sayisi:4
Sembol tablosunun uzunlugu:11

_cc degiskenin hashtable uzerinde hesaplanan ilk adresi:6   Sonda yerlestirilen adresi:6
_bb degiskenin hashtable uzerinde hesaplanan ilk adresi:7   Sonda yerlestirilen adresi:7
_aa degiskenin hashtable uzerinde hesaplanan ilk adresi:8   Sonda yerlestirilen adresi:8
_x degiskenin hashtable uzerinde hesaplanan ilk adresi:7   Sonda yerlestirilen adresi:9
-----
Process exited after 6.359 seconds with return value 0

```

## Örnek 2:

 ornek2 - Not Defteri

```

Dosya  Düzen  Biçim  Görünüm  Yardım
|
int main()
{
int _aa, _bb, _cc;
char _aa, _t;
char _x;
_aa = 5;
_xx = 9;
_bb = _aa / _dd;
if(_aa==_c){
    _aa=6;
}
while(_bb=_c){
    _bb=_bb+_k;
}
for(_i=0; _i<_b; _i++){
    _i=_i+_k;
}
}

```

Debug Mod:

```
Hata: '_aa' degiskeni daha once deklare edilmistir.  
Hata: '_xx' degiskeni deklere edilmemistir.  
Hata: '_dd' degiskeni deklere edilmemistir.  
Hata: '_c' degiskeni deklere edilmemistir.  
Hata: '_c' degiskeni deklere edilmemistir.  
Hata: '_k' degiskeni deklere edilmemistir.  
Hata: '_i' degiskeni deklere edilmemistir.  
Hata: '_i' degiskeni deklere edilmemistir.  
Hata: '_b' degiskeni deklere edilmemistir.  
Hata: '_i' degiskeni deklere edilmemistir.  
Hata: '_i' degiskeni deklere edilmemistir.  
Hata: '_i' degiskeni deklere edilmemistir.  
Hata: '_k' degiskeni deklere edilmemistir.
```

Tablonun son goruntusu:

```
0  
1  
2  
3  
4  
5 int _cc  
6 int _aa  
7  
8 char _t  
9  
10 char _x  
11  
12 int _bb
```

Deklere edilmiş degişken sayısı:5  
Sembol tablosunun uzunluğu:13

```
_cc degişkeninin hashtable üzerinde hesaplanan ilk adresi:5 Sonda yerleştirilen adresi:5  
_aa degişkeninin hashtable üzerinde hesaplanan ilk adresi:6 Sonda yerleştirilen adresi:6  
_t degişkeninin hashtable üzerinde hesaplanan ilk adresi:6 Sonda yerleştirilen adresi:8  
_x degişkeninin hashtable üzerinde hesaplanan ilk adresi:10 Sonda yerleştirilen adresi:10  
_bb degişkeninin hashtable üzerinde hesaplanan ilk adresi:12 Sonda yerleştirilen adresi:12
```

```
-----  
Process exited after 5.374 seconds with return value 0  
Press any key to continue . . . █
```