|  |  |  |
| --- | --- | --- |
| Description: C:\Users\Optimus\Desktop\tr200t.gif | Karadeniz Teknik Üniversitesi  Bilgisayar Mühendisliği Bölümü  Veri Yapıları |  |

## 2020-2021 Güz Dönemi

C Kodlama Dosyasının Girintilerinin Ayarlanması Raporu

|  |  |  |
| --- | --- | --- |
| Öğretim | 1. Öğretim | 1. Öğretim |
| Numara | Ad ve Soyad | |
| 358623 | Fırat Kaan BİTMEZ | |
| 330199 | Berk KUYUMCU | |
| 314042 | Osman Selçuk KARSLI | |
| Ders Sorumlusu: | Öğr. Gör. Dr. Zafer YAVUZ | |

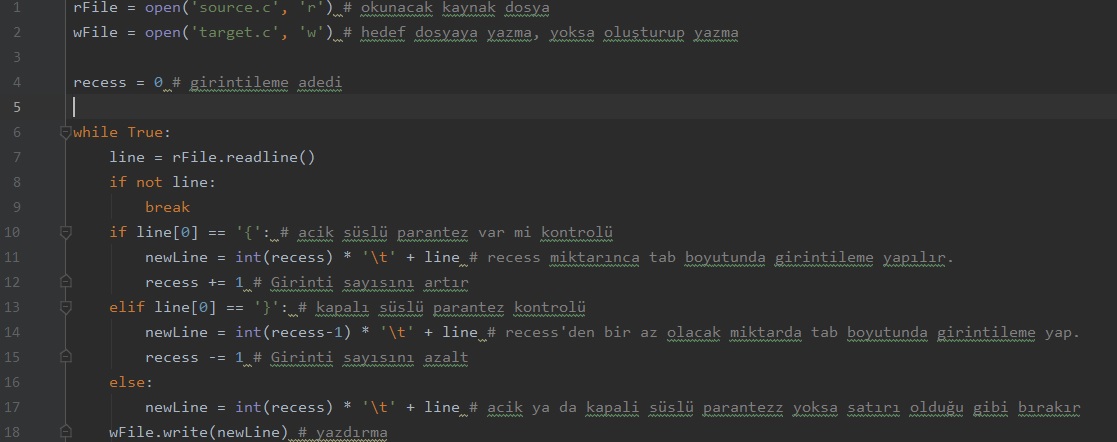
**C Kodlama Dosyasının Girintilerinin Ayarlanması**

**Problem Tanımı**

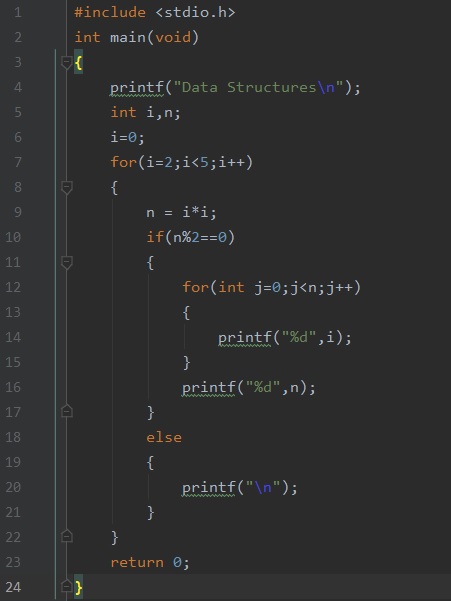
Python programlama dilinde girintiler önemli bir yer kaplamaktadır. C dilinde yazılmış bir kodu python'ın girini hassasiyetinden ötürü görsel manada güzelleştirmek ve her süslü parantez açıldığında girinti oluşturup kapatıldığında da girintiyi geri alan bir kod yazdık. kaynak olarak verilen dosyayı isteğimize uygun düzenleyip çıkışa aktardık.

**Kodlama**

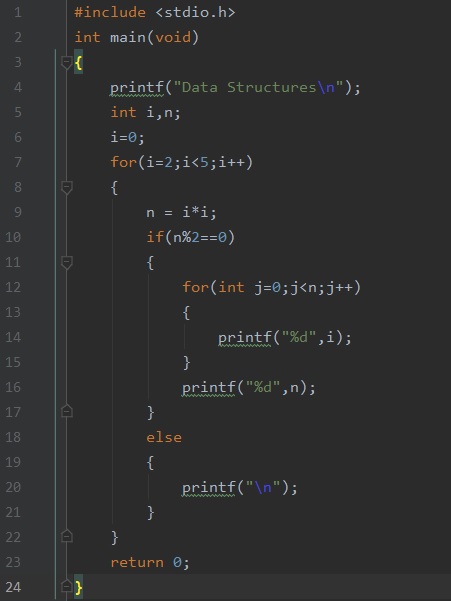
main



source

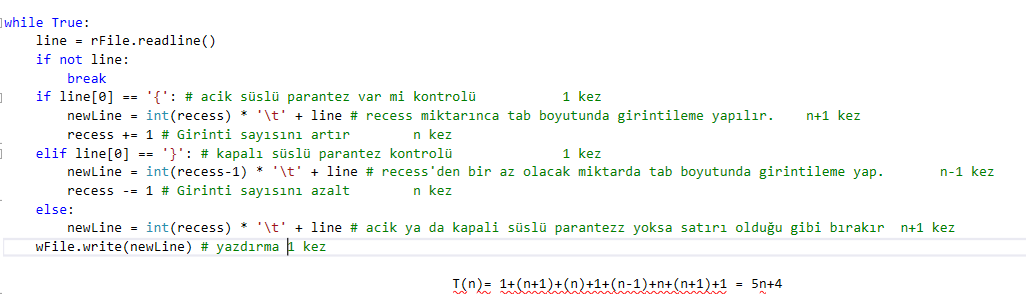


target



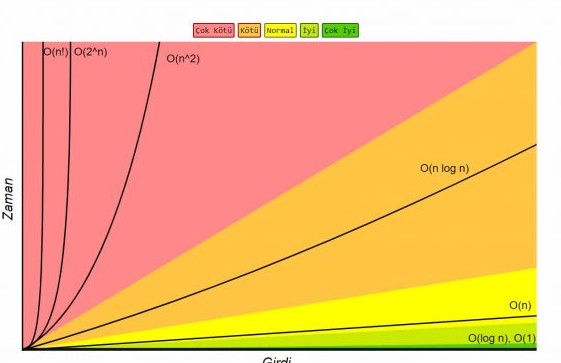
1. **ZAMAN KARMAŞIKLIĞI (TIME COMPLEXITY)**

Time Complexity , bir programın ya da fonksiyonun işlevini tam anlamıyla yerine getirebilmesi için her işlemden kaç kere yapması gerektiğini gösteren bir bağıntıdır.

****

**Zaman karmaşıklığını hesapladığımızda T(n) = 5n+4 bulduk.**

1. **BIG O NOTASYONU**

****

  Big o notasyonu fonksyonların input size’ı büyüdükçe oluşacak karmaşıklığı göstermek için kullanılır. Fonksyonun input size’ı büyüdükçe küçük kalan big o notasyonu makbüldür. Eğer input size ile beraber big o da çok fazla büyüyorsa var ise alternatiflerini kullanmamız gerekir.  
  
   Big o bulunurken time complexity bağıntısındaki en büyük artışı yöneten terim alınır diğerleri göz ardı edilir çünkü input size sonsuza giderken bu büyümeyi yönetecek olan en büyük değişkenli derimdir.

Time complexity’de büyümeyi yöneten terimi seçeceğimi için bağıntı içerisinde en çok karşımıza çıkan değişken terimlerin sıralamasını aşağıda hazır olarak verdim. Eğer ki bu terimler aynı bağıntı içerisindeyse yapmanız gereken seçim en büyük olanı seçmektir. Diğer bir kural ise seçilen en büyük derime ait çarpan, bölen, toplanan, çıkarılan sabitleri yok saymaktır çünkü bunlar da sonsuza giden büyüme sırasında anlamını kaybedecektir.

***n ! > 2^n > n^3 > n^2 > nlogn > n >logn > 1***

**Sabit Zamanlı : O (1)**

İfadenin çalışma zamanı input size N’e bağlı değildir. Örneğin;

* Dizi üzerinden elemana ulaşmak
* Sabit boyutlu stackten push ve pop işlemi yapmak
* Aritmatik operasyonlar
* Kıyaslamalar

**Yaptığımız araştırmalar neticesinde bizim kodumuzun BigO notasyonunu O(1) olarak bulduk.**

KAYNAKLAR

<http://cagataykiziltan.net/programin-calisma-hizi-ve-algoritma-verimliligi/zaman-karmasikligi-ve-buyuk-o-notasyonu-time-complexity-and-big-o-notation/>

<https://www.ridvantulemen.com/buyuk-o-notasyonu-nedir-big-o-notation/>

<https://python-ogren.readthedocs.io/en/latest/file_write.html>