# Programlama ve Veri Yapılarına Giriş

JAVA, C, C++ dilleri ile

Yazan: Şadi Evren ŞEKER

Yayın tarihi: Şubat 2009

**Programlama ve Veri Yapılarına Giriş**
JAVA, C, C++ dilleri ile

**Yazan:**
Şadi Evren ŞEKER

**Yayın Tarihi:**
Şubat 2009



**İrtibat Bilgileri:**
**Şadi Evren ŞEKER**
Atatürk Cad. No 53/9 Kozyatağı
Kadıköy
İstanbul
Tel: 0216 302 3042
E-Posta: kitap@sadievrenseker.com

**Baskı - Cilt:**
**Erkam Matbaası**
İkitelli OSB. Turgut Özal Cd.
Çelik Yenal Endüstri Merkezi
No: 117/4 Küçükçekmece /İstanbul
Tel: 0212 671 07 07
E-Posta: info@erkammatbaasi.com

# İçindekiler

# 1. Giriş ve Kitabın Kullanılışı

Bu bölümün amacı kitapta kullanılan sembollerin açıklanması ve kitap hakkında tanıtıcı bilgi vermektir.

# 1 Giriş ve Kitabın Kullanılışı

Bu kitabın amacı programlama bilgisi olmayan giriş seviyesindeki bir kişiye bilgisayar mühendisliğinin iki temel dersi olan “programlamaya giriş” ve “veri yapılarına giriş” derslerinin bilgisini kazandırmaktır. Bahsi geçen bu dersler, sonuca yönelik ve yapılan işlerin gerçek hayatta birebir kullanımı olan konulardan ziyade teorik ve diğer konulara temel teşkil eden içeriktedir. Dolayısıyla bu kitabın okuyucusu kitaptan edindiği bilgileri olmazsa olmaz birer kazanım olarak görmeli ve herhangi bir programlama konusunda uzmanlaşmadan önce bu konularda kendisini yeterli hale getirmelidir.

## 1.1 Kitabın konulara göre bölüm ve alt bölümlere ayrılışı

Bu kitap, bütün konuları üç ayrı programlama dilinde ele almaktadır. Dolayısıyla her bölümün en az dört alt bölümü bulunmaktadır. Bunlar aşağıdaki şekilde sıralanabilir:

1. Konunun açıklandığı ve teorik anlatımın bulunduğu ilk alt bölüm
2. C dilinde örneklerle anlatıldığı ikinci alt bölüm.
3. C++ dilinde anlatılan üçüncü alt bölüm.
4. JAVA dilinde anlatılan dördüncü alt bölüm.

Genel olarak her bölümde yukarıdaki yapı izlenmesine karşılık bazı giriş seviyesi bölümlerde üç dilde de aynı özellikler bulunduğu için birleştirilmiş olan alt bölümler bulunmaktadır. Ayrıca kullanılan dile özel olarak hazır teknoloji var olduğunda bu teknolojiyi açıklayan özel alt bölümler bulunmaktadır. Örneğin bağlı liste (linked list) konusunu JAVA dilinde sıfırdan yazmak mümkündür. Aynı zamanda JAVA dili içerisindeki “collection” sınıfında (class) hazır olarak yazılmış bir bağlı liste bulunmaktadır ve okuyucu ihtiyaç duyduğunda bu sınıfı kullanabilir. Bu durumda sıfırdan yazılışı açıklayan bir alt bölüm ile birlikte hazır sınıfların kullanılmasını açıklayan ikinci bir alt bölüm bu konuya eklenecektir.

Kitabın üç farklı dil ile programlamayı anlatıyor olmasından dolayı okuyucunun tek dile yoğunlaşma ihtimali düşünülerek her dilin ayrı ayrı okunduğunda da anlaşılabilmesi hedeflenmiştir. Bu yüzden kitapta bazı

konular her dil için tekrar edilmektedir. Ancak kitabın üç farklı dilde programlamayı anlatıyor oluşu, programlamayı öğrenmenin yanında bu üç dili karşılaştırma imkanı doğurması da okuyucu için bir avantajdır ve okuyucunun kitabı üç dil için de okuması tavsiye edilir.

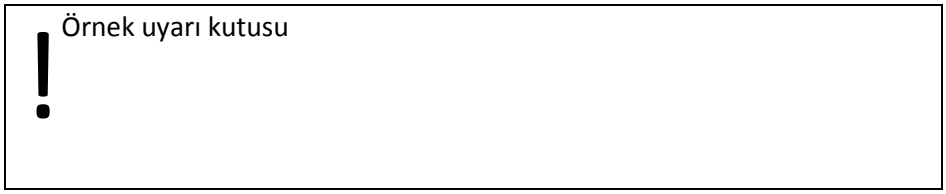
## 1.2 Kitabın dili ve terminoloji

Kitap şimdiye kadar anlaşılacağı üzere Türkçe olarak neşredilmiştir. Ancak gelişen iletişim teknolojileri ve bilhassa İnternet sayesinde bilgisayar teknolojilerindeki İngilizcenin rolü Türk dilinde de hissedilmektedir. Okuyucunun araştırma yapabilmesi ve aynı terimin Türkçeye farklı şekillerde çevrilmesinden kaynaklanan karmaşanın giderilmesi için bu kitapta Türkçe terminolojinin yanında İngilizce terimler parantez içersinde verilecektir. Örneğin:

“Bağlı liste (linked list)” şeklinde

Kitapta bulunan terminoloji hakkında ansiklopedik bilgiye ihtiyaç duyulması halinde yine kitabın yazarı Şadi Evren ŞEKER tarafından hazırlanan www.bilgisayarkavramlari.com adresinden yararlanılabilir. Kitap konusu dışında kalan kavramlara kitapta değinilmeyecektir dolayısıyla okuyucu bu konudaki araştırmalarını verilen bu adresten karşılayabilir.

Ayrıca bu kitapta konusu geldikçe önemli olan noktalar aşağıdaki şekilde:




Uyarı kutularında anlatılacaktır. Bu anlatılanlar ya yaşanan bir tecrübeye dayalı uyarılar olacak ya da konu için büyük öneme sahip noktalar olacaktır. Okuyucunun bu uyarı kutularını dikkate alması tavsiye edilir.

Ayrıca bilgilendirme kutuları da aşağıdaki şekilde kitap içerisinde yer bulmaktadır.

Örnek bilgilendirme kutusu

Bu kutların içerisinde genel olarak anlatılan konuyu açıklayan bilgiler bulunmaktadır.

## 1.3 Kitaptaki örnek ve kodların kullanılması

Kitapta bulunan örnek kodların kullanılması için öncelikle bu kodların kullanılabildiği bir programlama ortamına (programming environment) ihtiyaç duyulmaktadır. Kitabın EK1 ekinde ilgili programlama ortamının kurulumu ile ilgili resimli anlatım bulunmaktadır. Okuyucu bu bölümden faydalanarak istediği programlama dilinin ortamını kurabilir.

Programlama ortamı kurulduktan sonra kitaptaki örnekleri www.sadievrenseker.com/kitap adresinden indirebilir ve denemeye başlayabilirsiniz. Buradaki örneklerin hepsi kitapta açıklanan programlama ortamlarında test edilmiş ve çalışan örneklerdir. Bu programlama ortamları:

C dili için Dev-CPP (C dili olarak kullanılacak)

C++ dili için Dev-CPP (C++ dili olarak kullanılacak)

JAVA dili için JDK

Şeklinde sıralanabilir. Okuyucu bu ortamlar dışında ortamları da kullanabilir ancak özellikle C dili için ortamlara göre farklılıklar olduğu bilinmelidir.

Kitapta bulunan kodlar anlatımın yazı tipinden farklı olarak verilmiş bu sayede okuyucunun kodu anlatımdan kolaylıkla ayırması hedeflenmiştir. Örneğin:

*printf(“Merhaba Dünya”);*

satırında olduğu gibi.

Ayrıca çalışan kodların ekran çıktıları aşağıdaki şekilde gösterilecektir:

## 1.4 Konu sonu alıştırmaları

Kitabın sonunda bulunan çalışma soruları, kitabın bir ders kitabı olarak kullanılması hedeflendiği için eklenmiştir. Okuyucu konuları okuyarak öğrendiği bilgileri bu alıştırmalar ile pekiştirerek geliştirebilir. Ayrıca ders kitabı olarak okutulduğu ortamlarda ödev olarak da bu sorulardan istifade edilebilir. Konu sonunda bulunan soruların çözümleri kitabı okutmaya karar veren hocalara talep edilmesi durumunda verilecektir. Bunun için www.sadievrenseker.com/kitap adresinde bulunan hoca formunun doldurulması gerekmektedir.

Konu sonu sorularının yanlarında bulunan işaretlerin anlamları şu şekilde sıralanabilir:

* Diğer sorulara göre zor sorular

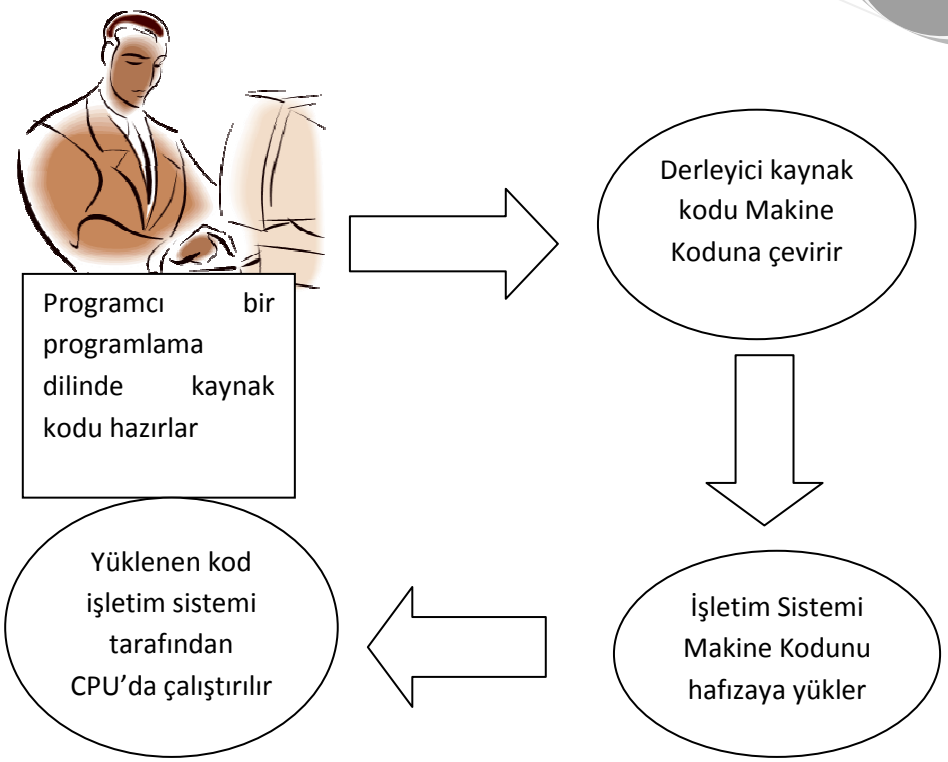
A Okuyucunun kitaptaki bilgilere ilave olarak araştırma yapması gereken sorular

J Java dilinde kodlanabilir sorular

C C dilinde kodlanabilir sorular

CPP C++ dilinde kodlanabilir sorular

Okuyucu bu soruları karar verdiği dile göre veya vakit durumuna göre seçerek çözebilir.




Bir programın nasıl çalıştığını daha detaylı anlamak için daha ileri seviyede işletim sistemi (operating systems) ve derleyici tasarımı (compiler design) bilgisine ihtiyaç vardır. Bu konular bu kitabın konusu dışında olduğu için şimdilik bu giriş bilgisi ile yetiniyor ve sırasıyla C, C++ ve JAVA dillerinde programlamaya giriş yapıyoruz.

## 2.3 C dilinde programlamaya giriş

C dili bu kitapta üzerinde uygulama geliştirilen en eski dildir. Gerek C++ gerekse JAVA, C dilinden sonra tasarlanmış ve büyük ölçüde C dilinden esinlenmiştir. Genel olarak C yazılım şekli (C-syntax) ismi verilen ve C dilinde kullanılan yazım kuralları güncel pek çok dili etkilemiştir. (JSP, PHP, C#, C++, JAVA, D, Vala, JavaScript, Cyclone bu dillerden bazılarıdır)

C dilinde basit bir uygulama ile başlayalım:

```
1. #include <stdio.h>
2. #include <conio.h>
3. int main(){
4.         printf("Merhaba Dünya");
5.         getch();
6.         return 0;
7. }
```

Yukarıdaki kod hemen her programlama diline başlarken yazılan ve ekrana "Merhaba Dünya" (Hello World) yazdırmaya yarayan koddur. Şimdi bu kodu inceleyelim.

Koddaki ilk 2 satır "#" işaretleri ile başlamaktadır. Bunun anlamı derleme öncesi (pre-compile)birer satır olduklarıdır. Yani derleyicimiz (compiler) bu satırları derleme işlemi öncesinde çalıştırır. Bu yüzden bu satırlarda bir hata olması durumunda çoğu derleyici (Dev-CPP de dahil olmak üzere) hata vermez.

İlk 2 satırda "#include" komutu yazılmıştır. Bu kelime Türkçeye dahil etmek, içermek olarak da çevrilebilir. Anlamından da anlaşılacağı üzere yanında yazan kütüphaneyi koda dahil etmeye yarar. Örneğin

```
1. #include <stdio.h>
```

Satırında "stdio.h" isimli dosya koda dahil edilmektedir. C dilinde kütüphaneler ".h" uzantılıdır bu uzantı ismi başlık manasına gelen "header" kelimesini ilk harfinden gelmektedir. Bu dosyalara başlık dosyaları (header file) ismi de verilmektedir.

Başlık dosyalarında kodda kullanılacak bazı fonksiyon, işlem, tip ve komutların tanımı bulunur. Örneğin birinci satırda "stdio" kütüphanesi yüklenerek C dilinin en temel işlevleri kodumuza dahil edilmiştir. İsmini standart giriş çıkış anlamına gelen Standard input output kelimelerinden alan bu dosya C dilinin en temel dosyasıdır ve bu kitaptaki konular boyunca her C dosyasının başında yerini alacaktır. Ancak çok özel durumlarda bu dosyanın kullanılmaması mümkündür.

İkinci satırımızda dahil edilen “conio.h” dosyası ise DOS ve dolayısıyla Windows ortamlarına özgü bir kütüphanedir. Linux veya DOS dışında farklı ortamlarda kod geliştiren okuyucular bu dosyayı dahil ettiklerinde hata mesajı alabilirler. Bunun yerine örneğin Linux üzerinde “ncurses.h” kütüphanesini kullanabilirler.

Aslında conio.h dosyasını kullanmamızın tek sebebi kodda bulunan getch() fonksiyonunu çağırabilmektir. Çünkü bu dosya stdio.h kütüphanesi içinde yer almaz ve bu fonksiyonu kullanabilmek için conio.h kütüphanesi koda dahil edilmelidir.

Şimdilik bu iki dosyayı yazacağımız C kodlarının çalışması için bir ihtiyaç olarak düşünebilirsiniz. Kodumuza farklı kütüphaneleri eklemeye ileri bölümlerde değinilecektir.

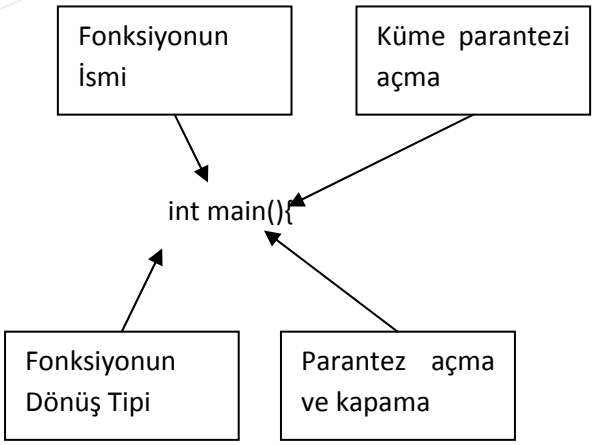
Kodumuzun 3. Satırında :

3. *int main(){*

yazılıdır. Bu satır da C kodlarında bizim seviyemiz için olmazsa olmaz olarak kabul edilebilir. Kısaca “main” isminde bir fonksiyon tanımlanmaktadır. C dilinde fonksiyon tanımı ve kullanılması ileri bölümlerin konusu olduğu için burada detaya girilmeyecektir ancak bu tanımlamanın özel bir durumu vardır. “main” fonksiyonları C dilinde önceliğe sahiptir ve kod çalıştırılmaya bu fonksiyondan başlar. Yani kodumuzda yüzlerce fonksiyon bulunabilir ama bir tanesinin ismi main olmalıdır ve bu fonksiyon C tarafından ilk çalıştırılan fonksiyondur.

Kodumuzda başka fonksiyon olmadığı ve basit bir kod olduğu için C, çalıştırmaya buradan başlayacak ve bu fonksiyon içerisindeki işlemleri yapacaktır.

Satırı parçalayarak inceleyecek olursak:




Yukarıdaki şekilde de açıklandığı üzere "main" fonksiyonunun dönüş tipi "int" olarak tanımlanmıştır. C99 standardı olarak bir main fonksiyonunun dönüş değeri int olmalıdır. Ancak bazı derleyiciler dönüş tipi olarak void'de kabul eder. Yani burada fonksiyonun başına void yazılması da mümkündür. İşlev olarak bir main fonksiyonu işletim sistemine bir değer döndürür. Bir önceki alt bölümde açıklandığı üzere programı çalıştıran işletim sistemidir. Dolayısıyla program işletim sistemine bir sinyal olarak bir değer döndürebilir. İşte main bu yüzden int tipinde değer döndürmelidir ve başına int yazılır. Hemen ardından fonksiyonumuzun ismi geliyor ki bu kodda fonksiyon ismimiz main'dir.

! C99 güncel olan ve en son C dilinin standartlarının belirlendiği C standardıdır. Sonundaki 99, 1999 yılında yayınlanmış olduğu anlamındadır. C dilinin yaygın kullanımı ve farklı alanlarda kullanılıyor olması dil üzerinde zamanla farklılaşmalar doğurmuş bunun bir neticesi olarak da C dilinde her derleyici (compiler) ve geliştirme ortamında ufak tefek farklılıklar çıkmaya başlamıştır. Bu sorunun giderilmesi için C dilinin standartları, uluslar arası standartlar enstitüsü (International Standards Orgainzation) tarafından ISO/IEC JTC1/SC22/WG14 numarası ile yayınlanmıştır. Bu standartlar daha sonra Mayıs 2000 yılında ANSI (American National Standards Institute, Amerikan ulusal standartlar enstitüsü) tarafından da kabul edilmiştir.

Bugün ANSI-C veya C99 ismi ile kast edilen standartlar bunlardır.

C dilinde fonksiyonları belirtmek için parantezler kullanılır. Yani parantezden önce gelen kelimeler fonksiyon isimleridir. Normalde matematikteki fonksiyonlar gibi parantezlerin içine fonksiyonun parametreleri yazılır. Yukarıdaki örnekte fonksiyonumuzun parametresi bulunmamasına rağmen parantez açılıp kapatılmıştır. Bunun sebebi main'in bir fonksiyon olduğunu belirtmektir.

Son olarak bu satırda bir küme parantezi (curved bracket) sembolü bulunmaktadır. Bu sembolün anlamı fonksiyonun içinin yazılmaya başladığıdır. Genel olarak küme parantezleri C dilinde geçerlilik alanı (scope) belirlemeye yarar. Yani fonksiyonun geçerliliği, değişkenin geçerliliği, döngünün geçerliliği gibi belirlemeler için küme parantezleri kullanılır. Yukarıdaki örnekte de main fonksiyonunun geçerlilik alanının başladığını ve bu satırdan sonra yazacaklarımızın main fonksiyonunun içerisinde olduğunu belirten küme parantezi görülmektedir.

main fonksiyonunun tanımlandığı bu satırı inceledikten sonra 4. Satırı inceleyebiliriz:

*4.* *printf("Merhaba Dünya");*

Yukarıdaki bu satırda main fonksiyonuna benzemektedir. Yukarıda anlatıldığı üzere parantezler C dilinde fonksiyon belirtmektedir. Dikkat edilirse burada da "printf" yazısından sonra bir parantez açılmıştır. Bunun anlamı printf'in de bir fonksiyon olduğudur. Gerçekten de "printf" daha önce dahil ettiğimiz (include) stdio.h kütüphanesinde tanımlı ve ekrana verilen yazıyı yazmaya yarayan bir fonksiyondur.

Basitçe printf parametere olarak verilen değeri ekrana basar. Burada parametre olarak iki parantez arasında verilen değer "Merhaba Dünya" yazısıdır.

Dolayısıyla C, bu satırı çalıştırdığında ekrana "Merhaba Dünya" yazısının çıkmasını beklemekteyiz.

Ardından gelen satırlarda:

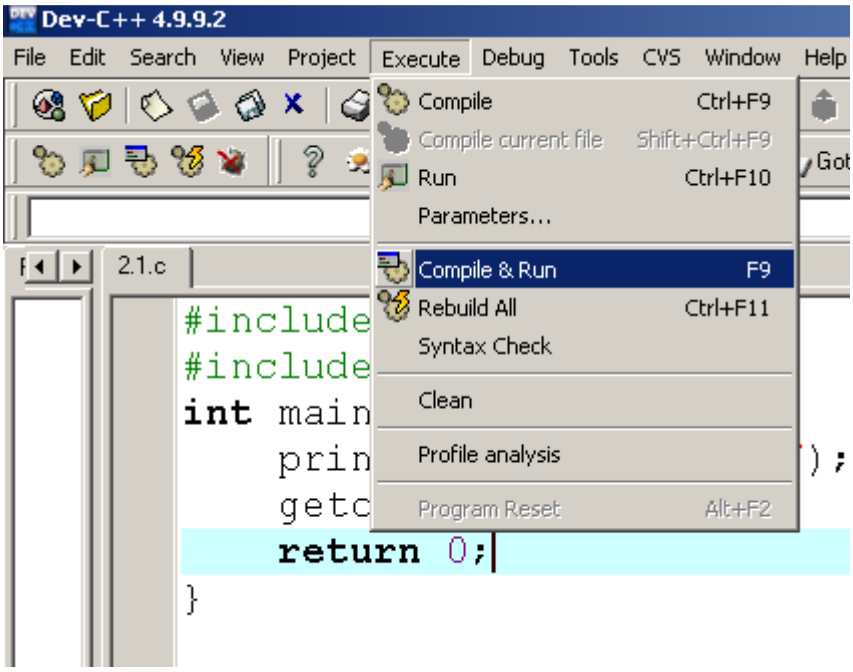
*5.* *getch();*
*6.* *return 0;*
*7.* *}*

Satırları bulunmaktadır. Buradaki getch anlaşılacağı üzere yine bir fonksiyondur ve daha önce de ifade edildiği üzere conio.h kütüphanesinde tanımlıdır. Bu fonksiyonu burada çağırmamızın tek sebebi ekranda çıkan yazıları kaybetmeden görebilmektir. Yani C yukarıdaki "Merhaba Dünya" yazısını yazdıktan sonra program başka bir işi olmadığı için kapanacaktır. Dolayısıyla programın çalıştığında ne yaptığını görmemiz mümkün olmayacaktır. Çözüm olarak 5. Satırda bir getch fonksiyonu çağrılmakta ve C'den bir karakter okuması istenmektedir. Aslında okunan karakterin bir önemi yoktur sadece program çalıştığında kullanıcı bir karakter girene kadar (klavyeden bir tuşa basana kadar) program duracak ve bekleyecektir. Bu sayede çalışan programın sonucu ekranda görülebilecektir.

6. satırda yer alan "return 0;" saıtırı ise daha önce de belirtildiği üzere main fonksiyonunun işletim sistemine bir değer döndürmesine yarayan satırdır.

Son olarak 7. Satırda bir küme parantezi kapatılmıştır. Bu kapatılan parantez 3. Satırda main fonksiyonu tanımlanırken açılan parantezdir.

Kodun çalıştırılması için geliştirme ortamı olan Dev-CPP üzerinde kod yazılarak derlenir. Ardından bu geliştirme ortamında bulunan çalıştır özelliği ile kod çalıştırılabilir.



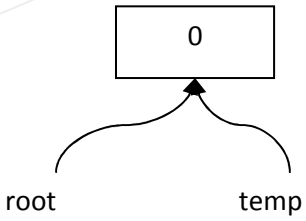

Yukarıdaki şekilde yazılan kodun çalışıtırılması için menülerden "Execute" altındaki "Compile & Run" seçeneği gösterilmektedir.

> **!** Yukarıda yazılan kod çalıştırıldığında Merhaba Dünya yazısındaki ü harfi Türkçe karakter olduğu için düzgün görülmeyebilir.

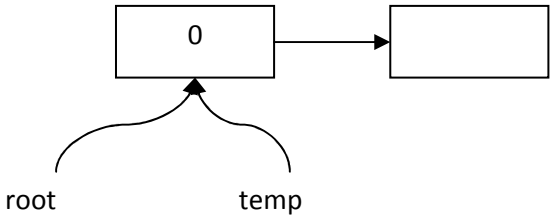
## 2.4 C++ dilinde programlamaya giriş

C++ dili temel olarak C dilinin bütün özelliklerini taşımaktadır. Yani teorik olarak C dilinde yazılan her şey C++ dilinde de çalışmalıdır. C++'ın C'den en büyük farkı nesne yönelimli (object oriented) bir dil olmasıdır. Bunun anlamını ilerleyen konulardan oluşum (composition) başlığı altında detaylıca göreceğiz. Şimdilik basit bir C++ uygulaması ile ekrana "Merhaba Dünya" yazdırmaya çalışalım.

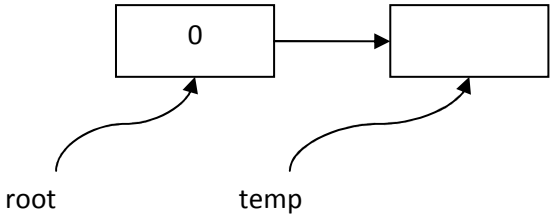
1. *#include <iostream>*
2. *#include <conio.h>*
3. *using namespace std;*



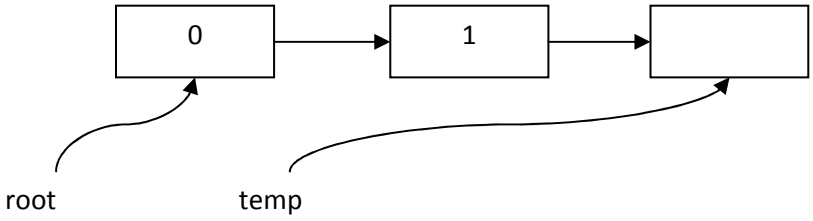

Ardından 14. Satırda temp'in gösterdiği kutunun next göstericisine yeni bir kutu malloc fonksiyonu ile atanıyor.




Döngünün içerisindeki son satırda da temp bir önceki satırda oluşturulan yeni kutuyu gösteriyor.




Örneğin döngünün bir dönüş sonrasında oluşan bağlı liste aşağıdaki şekildedir:

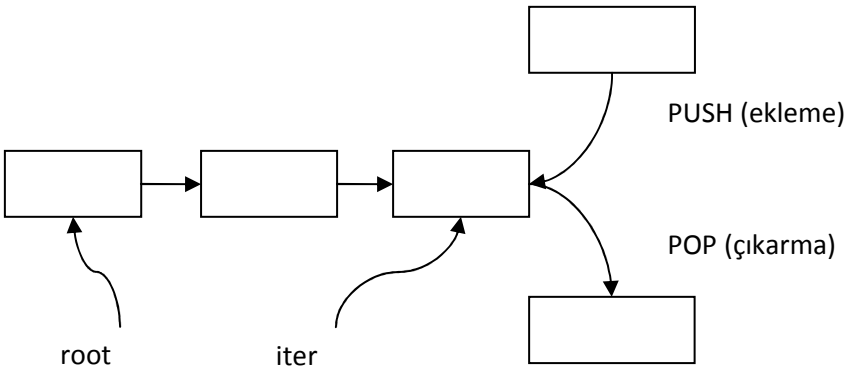



Döngü 10' kadar döndüğü için, kodun bitmiş halinde 11 kutu olacak ve 10 tanesinin içerisinde sayı bulunacaktır.

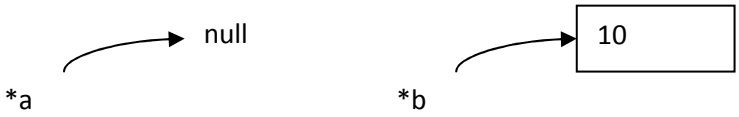
```
40.     printStack();
41.     push(30);
42.      printStack();
43.      printf("\npop> %d",pop());
44.      printStack();
45.      printf("\npop> %d",pop());
46.      printStack();
47.      printStack();
48.      printf("\npop> %d",pop());
49.      printStack();
50.      printf("\npop> %d",pop());
51.      printStack();
52.     getch();
53.     return 0;
54. }
```

Yukarıdaki kodda bağlı liste kullanılarak yığın (stack) kodlaması yapılmıştır. Kodda bastiçe bir bağlı liste oluşturulmuş. Bu işlem için daha önceden gördüğümüz düğüm (node) yapısını içeren bir yapı (struct) kullanılmıştır.

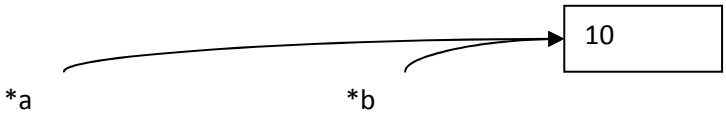
Temel olarak bir yığın içerisinde tutulan bilgileri bağlı liste ile modellerken iki uçtan yaklaşım olabilir. Birinci yaklaşımda bağlı listenin başına eklenip başından çıkarılması mümkünken, ikinci yaklaşımda (yukarıdaki kodda da bu yaklaşım izlenmiştir) bağlı listenin sonuna eleman ekleyip sonundan çıkarmak mümkündür.




Kodun 35. satırında, sıraya bir eleman konulması ile hafızada bir dizi oluşturulur. Dizi yukarıdaki kodda 7. satırda bulunan malloc fonksiyonu ile oluşturulmaktadır. Ancak bu dizi oluşturma işlemi "a" dizisinin gösterdiği alanda yapılmak yerine, yeni bir gösterici olan "b" göstericisinin gösterdiği alanda yapılmaktadır.




İkinci dizi hafızada tanımlanıp içerisine eklenmesi istenen sayı konulduktan sonra "a" göstericisi de "b" göstericisinin gösterdiği yeri işaret etmektedir.




Ardından fonksiyondan çıkıldığı için b göstericisinin geçerlilik alanı sona erecek ve dolayısıyla kaldırılacaktır. Sonuçta yeni bir eleman eklenmiş bir diziyi gösteren bir a göstericisi ile işlem tamamlanmış olacaktır. Ayrıca kodun 13. satırında size değişkeni bir arttırılmaktadır. Dolayısıyla ilk başta 0 olan değer bu satırın çalışması ile birlikte 1 olarak değişecektir. Gerçekten de şu anda sırada 1 eleman bulunmaktadır ve dizimizin boyutu da 1'dir.

Kodun 36. satırındaki ikinci enqueue komutuyla birlikte yeniden yeni bir dizi oluşturulacak ve bu yeni dizinin içerisine eski dizideki veriler kopyalanacaktır (kodun 9-11 satırları arası). Dikkat edilirse yeni oluşan dizi,