

# C++ PROGRAMLAMA

Yrd. Doç. Dr. ESEN YILDIRIM

# BİLGİSAYAR

- **hardware** – Bilgisayarın fiziksel parçaları
  - Printer, Monitör, Klavye, anakart vs
- **software** – Bilgisayar tarafından kullanılan programlar
  - Word, Excel, Turbo Pascal vs

# DONANIM

- Bir bilgisayar sisteminde 4 önemli bölüm bulunur:

- 1. Input devices**

Bilgisayara bilgi göndermek için kullanılan araçlardır.

- Klavye, mouse vs.

- 2. Output devices**

Bilgisayardan kullanıcıya bilgi iletimi için kullanılır.

- Monitör, printer vs

# DONANIM

## 3. CPU

- CPU (Central Processing Unit – Merkezi İşletim Ünitesi veya işlemci)
  - bilgisayarın beyni olarak bilinir,
    - aritmetiksel ve mantıksal işlemleri yapan aygıtır.
      - Aritmetiksel işlemler : toplama, çıkarma, çarpma, bölme
      - mantıksal işlemler : karşılaştırma büyüklük, küçüklük, eşitlik tespiti gibi işlemlerdir. C
      - PU aynı zamanda bilgisayar birimlerin çalışması ve bu birimler arasındaki veri akışını da kontrol eder.
  - CPU bir programdaki talimatları alıp program tarafından istenen işlemleri gerçekleştiren araçtır.
    - Tipik bir CPU komutu 0 ve 1 leri sayı olarak algılar
      - hafıza birimi 17deki sayıyı al, 19 daki hafıza biriminden aldığı sayıya ekle, sonucu 55teki hafıza birimine yaz” gibi düşünülebilir.

# DONANIM

- CPU 3 ana parçadan oluşur: CU, ALU, ve registers
  - **CU (Kontrol Birimi)**
    - CPU da komutların gerçekleştirilme sırasını kontrol eden birimdir.
  - **ALU (Arithmetic Logic Unit - Aritmetik Lojik Birim)**
    - Bir bilgisayarda genel anlamda aritmetik ve mantıksal işlemlerin yapıldığı birime ALU adı verilir.
  - **Registers**
    - Saklayıcılar modern işlemci tasarımının merkezidir.
    - İşlemci üzerinde bulunan ve hesaplamalarda geçici depo görevi gören yüksek hızlı hafıza birimleridir.

# DONANIM

## 4. Memory: 2 tip hafıza bulunmaktadır: Birincil hafıza ve ikincil hafıza

### • Birincil hafıza (Primary Memory)

- Birincil hafıza bilgisayarın rastgele erişilebilir belleğidir (**RAM -Random Access Memory**).
- En temel fonksiyonu işlemcinin (Merkezi işlem birimi – CPU ) program çalıştırırken geçici olarak verileri sakladığı ve sırası geldikçe bu verileri kullandığı alan olmasıdır.
- Bir işlemci (CPU) çok basit toplama çıkarma seviyesinde işlemler yaparak programları çalıştırır. Komplike bir programın ise bu nevi basit işlemlere indirgenmesi mümkün olsa da bu indirgenme sonucunda çok sayıda işlemin yapılması gerekir. CPU anlık olarak bu işlemlerden sadece birisini yapabilir. Geri kalan işlemler ise sırasını beklemek ve bu sırada bir yerde saklanmak zorundadır.
- RAM bilgisayarda çalışan her programın CPU'da çalışmak için bekletildiği ve o ana kadar çalışması sonucunda biriken verilerinin saklandığı hafıza ünitesidir.



Resim 1.1: RAM bellek

# DONANIM

- **Birincil hafıza (Primary Memory)**

- Harddisklere göre daha hızlı (ve dolayısıyla daha pahalı) olan veri ünitelerinin içinde bulunan bilgiler elektrik kesilmesi sonucu kaybolur.
  - Bu yüzden **ikincil hafıza (secondary memory)** ismi verilen **sabit disklere** her zaman ihtiyaç vardır ve bilgisayarın kapanıp açılması sonucu verilerin korunması için RAMdeki bilgiler diske yazılabilir.
- RAM'ler birbirinden tamamen bağımsız hücrelerden oluşur.
  - Bu hücrelerin her birinin kendine ait sayısal bir adresi vardır.
    - RAM'deki herhangi bir bellek hücresine istenildiği anda diğerlerinden tamamen bağımsız olarak erişilebilir.
      - Rastgele erişimli bellek adı da buradan gelmektedir. RAM'de istenen kayda ya da hücreye anında erişilebilir.
- RAM adresler ve içerikleri tablosu olarak da görülebilir.
- Bellek kapasitesi byte cinsinden belleğin kapasitesini verir.
  - **Byte; bellek ölçü birimidir, 8 bitten oluşur. Bit ise "1" veya "0" sayısal bilgisini saklayan en küçük hafıza birimidir.**

# DONANIM

## – İkincil hafıza (Secondary Memory)

- Sabit disk olarak da bilinir.
- Verilerin uzun ömürlü ve en güvenilir şekilde saklanabilmesini sağlayan depolama aygıtlarıdır.



(A)

(B)

(A) 3.5 inch'lik bir sabit diskın üstten görünüşü, (B) Sabit diskın disk plakaları ve elektronik kartı söküldükten sonraki görünüşü



# DONANIM

## — İkincil hafıza (Secondary Memory)

- Bilgisayarın çevre birimleriyle uyumlu bir şekilde çalışması için gerekli işletim sistemi sabit disk üzerine kurulur.
- Sürekli çalışması gereken yazılımlar da sabit disk üzerinde tutulur.
- Sabit diskler kapasiteleri ve devir sayılarıyla anılır.
  - Sabit disklerin kapasiteleri en küçük anlamlı bilgi kümesi olan Byte'ın üst katlarıyla belirtilmektedir.
  - Devir sayısı **RPM** (Rotation Per Minute – dakika başına dönüş sayısı) olarak isimlendirilir ve sabit disk plakalarının dakikadaki dönüş sayısını belirtir.
  - 90'lı yıllardan günümüze 3600rpm, 5400rpm, 7200rpm ve 10000rpm disk dönüş hızlarına sahip sabit disk sürücüleri üretilmiştir.
    - Yüksek rpm değerine sahip diskler daha fazla ısınacağından içine monte edilecek kasada hava sirkülasyonunun çok iyi sağlanması gerekir.

# YAZILIM

- Donanım birimlerini istenen işleme yöneltip verimli bir çalışma içerisinde kullanımını sağlayan bir dizi komut, prosedür gibi sanal tanımlardır.
- Bilgisayar ile kullanıcı arasındaki iletişimi sağlayan görünen öğedir.
- Bilgisayarın çalışmasını işlem yapmasını sağlayan bütün programlar yazılım gurubunu oluşturur
- En temel olarak, bilgisayarın çalışmasını sağlayan en önemli ve vazgeçilmez unsurlardan biri de yazılımdır (İşletim Sistemi).

# YAZILIM

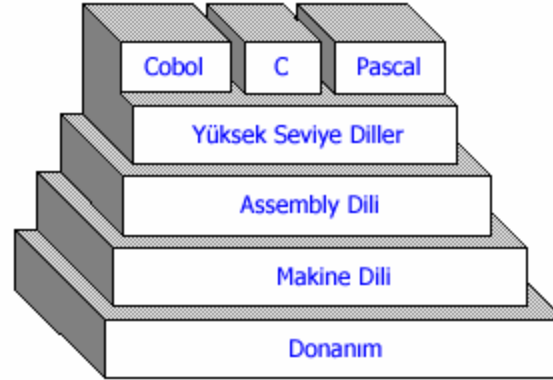
- Yazılımlar genel olarak bir kaç alt grupta incelenebilir.
  - Sistem Yazılımları
    - İşletim Sistemi
    - Bellek ve Disk hizmet yazılımları
    - Derleyiciler
    - Veri Tabanı Yönetim Yazılımları
    - İletişim ve Haberleşme Yazılımları
    - Ağ Yönetim Yazılımları
  - Uygulama Yazılımları
    - Ticari Paket yazılımları
    - Ofis Otomasyonu Yazılımları
    - Diğer özel ve genel tüm yazılımlar (Çizim, Tasarım, Dizgi vb.)
- En önemli Yazılım İşletim Sistemidir. Çünkü bu yazılım olmaksızın bilgisayarımız açılmayacaktır

# Programlama

- **Her şeyden önce herkes bir programlama dilini öğrenebilir.**
  - Bilgisayar programlama yüksek bir zekâ ve matematik bilgisi gerektirmez. Sadece asla vazgeçmeme sabrı ve öğrenme isteği yeterlidir.
  - Programlama bir hünerdir. Bazı insanlar doğal olarak diğerlerinden daha iyidir, ama herkes pratik yaparak iyi olabilir.
- Başaramamaktan korkmak yerine, kendinizi bu maharete vererek, öğrenmek için uğraşın. Programlama eğlencelidir, fakat yanlış çalışma yöntemleriyle sinir bozucu da olabilir ve zamanınızın boşa geçmesine neden olabilir.

# Programlama

- Programlama dili: İnsan-makina ve makina- makina arasındaki iletişimi sağlar.
- Programlama dilleri kullanım yapısına göre genel olarak üç seviyede incelenir.
  - Düşük seviyeli diller (makina ve assembly dilleri)
  - Orta seviyeli diller (C/C++ programlama dili)
  - Yüksek seviyeli diller (Basic, Fortan, Pascal. vb.)



**Dillerin genel görünümleri**

# Yüksek Seviyeli Diller (High-Level Languages)

- Pascal, C/C++, COBOL, ve FORTRAN gibi diller derlenmesi gereken dillerdir.
  - Kodlar kolayca yazılıp, okunabilen ve hata ayıklaması yapılabilen bir şekilde yazılırlar. Bu KAYNAK KODU olarak adlandırılır.
  - Compiler denilen program ise bu kaynak kodunu nesne kodu denilen makine diline çevrilmiş kodu saklayan dosyaya çevirmek için kullanılır.
  - Makine dili, program çalıştırıldığı zaman CPU tarafından yürütülen basit yönergelerin ikili sayılardan (0 ve 1) oluşan kodlanmış versiyonlarıdır.
- Bu derste genel amaçlı kullanıma uygun olan dillerden C++ programlama dilini öğreneceğiz.

# NEDEN C++

- C++ dili günümüzde çok yaygın olarak kullanılmaktadır
  - C++ dili makineye en yakın dil olarak görülmektedir
    - Bundan dolayı sistem programları daha hızlı ve randımanlı olarak yapılıyor
  - C sadece yapısal programlamaya izin verirken, C++ nesne yönelimli programlamaya imkan verir.
- Unutmayın !!! Herhangi bir programlama dilini iyi kavramanız ve programlama kavramını anlamanız durumunda herhangi bir dili öğrenmek çok kolaydır.

# Problem

## ● Problem Nedir?

- Bir işlemin, otomasyonun ya da bilimsel hesaplamaların bilgisayarla çözülmesi fikrinin ortaya çıkmasına problem denir.
- Bu tip fikirlerde insanların bu sorunları beyinle çözmeleri ya imkansızdır ya da çok zor ve zaman alıcıdır.
- Bu tip bir sorunu bilgisayarla çözebilme fikrinin ortaya çıkması bir bilgisayar probleminin ortaya çıkmasına neden olmuştur.
- Bazen de bir işletme veya yönetimin otomasyonunu sağlamak amacı ile bu tip problemler tanımlanır.



# Problem Çözümü

- Problemi Çözebilmek için öncelikle sorunun çok net olarak programcı tarafından anlaşılması gerekir.
- Tüm ihtiyaçlar ve istekler belirlenmelidir. Gerekiyorsa bu işlem için birebir görüşmeler planlanmalı ve bu görüşmeler gerçekleştirilmelidir.
- Problemin Çözümüne ilişkin zihinsel alıştırmalar yapılır.
  - Bu alıştırmaların Bilgisayar çözümüne yakın olması hedeflenmelidir.
  - Bir sorunun tabii ki birden fazla çözümü olabilir.
    - Bu durumda bilgisayar ile en uygun çözüm seçilmelidir.
    - Çünkü bazen pratik çözümler bilgisayarlar için uygun olmayabilir.
- Oluşturulan çözüm **Algoritma** dediğimiz adımlarla ifade edilmelidir.
- Bu algoritmanın daha anlaşılabilir olması için Akış Çizgesi oluşturulmalıdır.
- Uygun bir programlama dili seçilmeli ve oluşturulan algoritma ve akış şeması bu programlama dili aracılığı ile bilgisayar ortamına aktarılmalıdır.
- Oluşturulan program bir takım verilerle test edilmelidir.
  - Oluşabilecek sorunlar ilgili kısımlar tekrar gözden geçirilerek düzeltilir. Bu adımlar defalarca gerçekleştirilmek zorunda kalınabilir.

# Program ve Programlama

- **Program Nedir?**

- Problem Çözümü kısmında anlatılan adımlar uygulandıktan sonra ortaya çıkan ve sorunumuzu bilgisayar ortamında çözen ürüne Program denir.
- Bazı durumlarda bu ürüne yazılım denebilir.

- **Programlama Nedir?**

- Problem Çözümünde anlatılan adımların tümüne birden programlama denilebilir.
- Çoğunlukla ***çok iyi tanımlanmış bir sorunun*** çözümüne dair adımlar ile çözümün oluşturulup bunun bir programlama dili ile bilgisayar ortamına aktarılması Programlama diye adlandırılabilir.

# ALGORİTMA VE AKIŞ ŞEMALARI

- Bilgisayarlar aptal makinelerdir.
  - Sadece ona yapmasını söylediğiniz komutları uygular
  - Bilgisayar kullanarak soru çözmek için sonuca giden yolun **tam olarak** belirlenmesi gerekir.
    - Bir sorunu çözerken, atması gereken her adımı yapması gereken her işi (görmek duymak dahil) söylemeniz gereken bir makine olarak düşünün
  - Aynı soru için değişik çözüm yolları geliştirilebilir.
  - Eğer bilgisayara verilen çözüm yanlıştır, çıkan sonuç yanlış çözüm doğru ise çıkan sonuç da doğrudur.

# Algoritma

- Bir sorunu çözebilmek için gerekli olan sıralı mantıksal adımların tümüne **Algoritma** denir.
- Doğal dille yazılabileceği için fazlaca formal değildir.
- Bir algoritma için aşağıdaki ifadelerin mutlaka doğrulanması gereklidir.
  - Her adım *son derece belirleyici* olmalıdır. Hiç bir şey şansa bağlı olmamalıdır.
  - Belirli bir sayıda adım sonunda algoritma sonlanmalıdır.
  - Algoritmalar karşılaşılabilecek tüm ihtimalleri ele alabilecek kadar genel olmalıdır.

# Akış Çizgeleri

- Bir algoritmanın şekillerle görsel gösterimidir.
  - Algoritma **doğal dille** yazıldığı için herkes tarafından anlaşılabilir
  - Ancak akış çizgelerinde her bir şekil standart bir anlam taşıdığı için farklı yorumlanıp anlaşılabilmesi mümkün değildir.
  - Her bir ifade için ayrı bir sembol kullanılmaktadır.

# İzlenecek Adımlar

- Problem iyice anlaşılır
  - Öncelikle algoritma kurulmalı
  - Problemin hangi dil kullanılarak çözüleceği belirlenir.
  - Belirlenen dilin sentaksına uygun şekilde kaynak kodları yazılır
- Kaynak kodlarını makine diline çevirecek olan derleyici kullanılır
- Çalıştırılabilir dosya oluşturulur
- Program çalıştırılır

- **Programlama Dili Nedir?**

- Bir Problemin Algoritmik çözümünün Bilgisayara anlatılmasını sağlayan, son derece sıkı-sıkıya kuralları bulunan kurallar dizisidir.

- **Derleyici Nedir?**

- Bir programlama dili ile bilgisayara aktarılan programın bilgisayarın anlayabileceği Makine Diline çevirmeyi sağlayan ve yazılan programda söz dizim hatalarının olup olmadığını bulan yazılımlardır.

- *Her Programlama dili için bir derleyici olması gerekmektedir.*

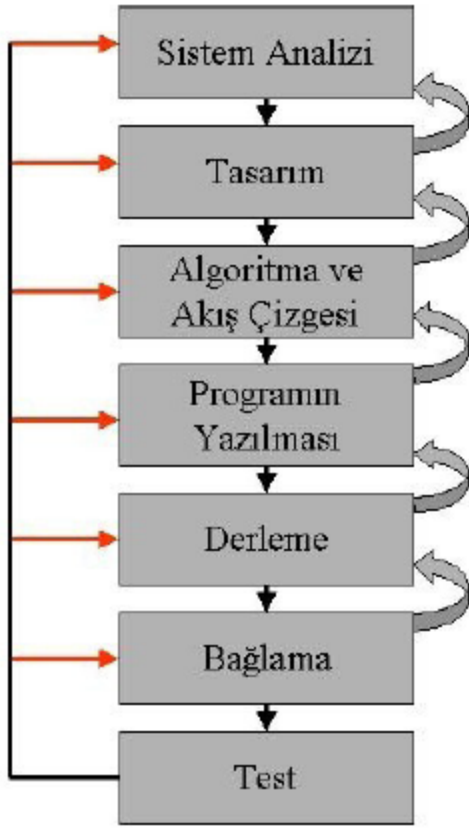
- PASCAL, C/C++, Delphi örnek olarak verilebilir.

- **Yorumlayıcı Nedir?**

- Derleyici gibi çalışan ancak yazılmış programları o anda Makine diline çeviren yazılımlardır.

- Bu tür bir yazılımda Programın Makine dili ile oluşturulmuş kısmı bilgisayarda tutulmaz. Programın her çalıştırılmasında her adım için Makine dili karşılıkları oluşturulur ve çalıştırılır.

- ASP, JavaScript, MATLAB gibi programlar örnek olarak verilebilir.



**Sistem Analizi:** Sorunun çözülebilmesi için tamamen anlaşılmasını sağlayan çalışmalardır.

**Tasarım:** İsteklerle ilgili olarak belirlenen bir takım çözümlerin tanımlanmasıdır.

**Algoritma :** Çözümün adımlarla ifade edilmesidir.

**Akış Çizgesi :** Algoritmanın şekillerle ifade edilmesidir.

**Programlama Dili Seçimi :** Çözümün netleşmesinden sonra yapılacak işlemleri kolay bir şekilde bilgisayar ortamına aktaracak dilin seçilmesidir. Önemli olan bu dilin özelliklerinin programcı tarafından iyi bilinmesidir.

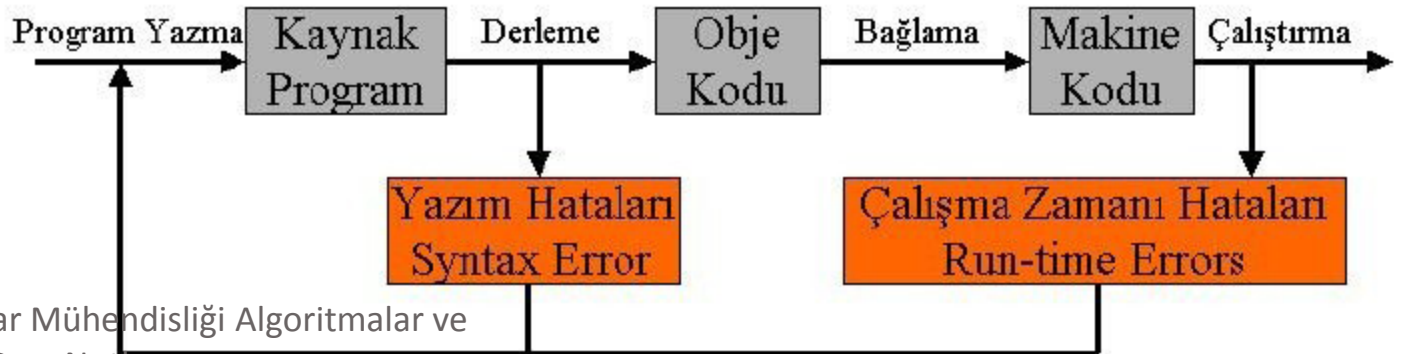
**Programın Yazılması:** Seçilen Programlama dilinin kuralları kullanılarak program yazılmaya başlanır. Bu amaçla çoğunlukla sade bir metin editör kullanılır. Bazı durumlarda Syntax highlighting denilen bir özelliğe sahip olan daha akıllı editörler de kullanılabilir. Bazen de editör ile Programlama dilinin derleyicisinin, bağlayıcısının hatta hata ayıklayıcısının iç içe bulunduğu IDE (Integrated Development Environment) denilen türde derleyiciler kullanılır.

**Derleme :** Programlama Dili ile yazılmış programın yazım hatalarının olup olmadığını kontrol edilmesini ve ara kod olarak Obje kodun üretilmesini sağlama adımıdır.

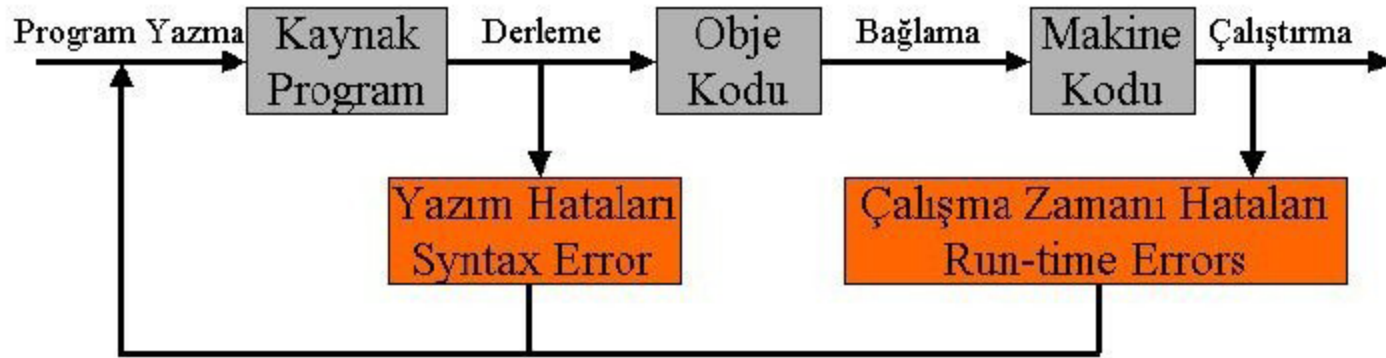
**Bağlama :** Derlenmiş ara kod diğer kütüphane ve parça programlarla birleştirilerek Makine dilinde programın oluşturulması adımıdır. Ancak bazı IDE ortamlarda ve derleyicilerde Derleme ve Bağlama bir bütündür ve beraberce halledilirler. Programcının ayrıca bir bağlama işlemi yapması gerekmez işlemi yapması gerekmez.

**Çalıştırma :** Oluşturulan Makine dili Programının çalıştırılması adımıdır. Yukarıdaki adımların hepsi yolunda gittiyse program sorunsuz olarak çalışabilmelidir.

**Test :** Programın Mantıksal olarak test edilmesini sağlar ve içerik olarak her ihtimal için doğru sonuçlar üretilmediğini kontrol etmenizi sağlar.

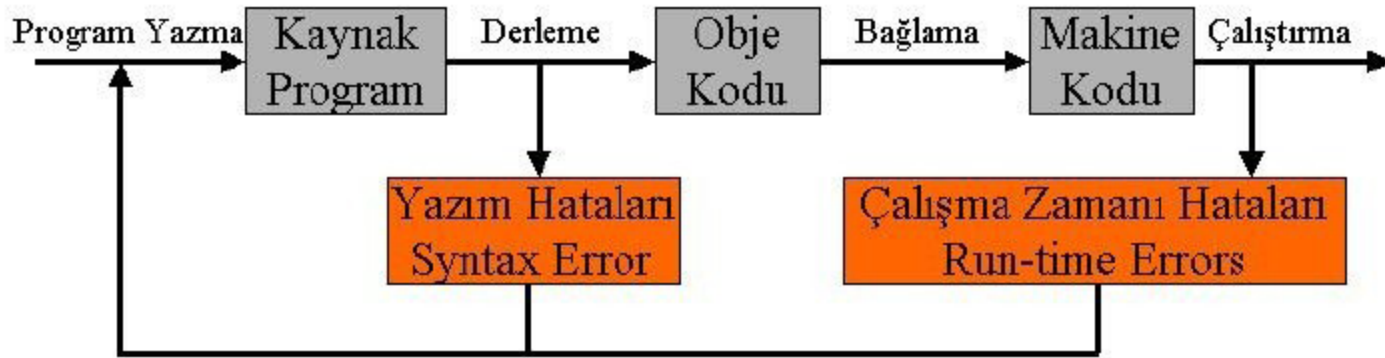






## • Syntax Error :

- Yazılan programda programlama dili kurallarına aykırı bir takım ifadelerden dolayı karşılaşılabilecek hatalardır.
- Düzeltilmesi son derece basit hatalardır.
- Hatanın bulunduğu satır derleyici tarafından rapor edilir. Hatta bazı derleyiciler hatanın ne olduğunu ve nasıl düzeltilmesi gerektiğini dahi bildirebilirler.
- Eğer bir derlemede Syntax Error alındı ise obje kod üretilmemiştir demektir.



- **Run-time Error :**

- Programın *çalıştırılması sırasında* karşılaşılan hatalardır.
- Programcının ele almadığı bir takım aykırı durumlar ortaya çıktığında programın işletim sistemi tarafından kesilmesi ile ortaya çıkar.
- Bu tip hatalarda hata mesajı çoğunlukla çalışan işletim sisteminin dili ile verilir.
- Eğer bu tip hataları kullanıcı ele almışsa, program programcının vereceği mesajlarla ve uygun şekilde sonlandırılabilir.
- Bu tip hataların nerelerde ve hangi şartlarda ortaya çıkabileceğini bazen kestirmek zor olabilir.
- Çoğunlukla işletim sistemi ve donanım kaynakları ile ilgili sorunlarda bu tip hatalar ortaya çıkar.
  - Örneğin olmayan bir dosya açmaya çalışmak, var olan bir dosyanın üzerine yazmaya çalışmak, olmayan bir bellek kaynağından bellek ayırtmaya çalışmak, olmayan bir donanıma ulaşmaya çalışmak

# ALGORİTMA VE AKIŞ ŞEMALARI

- Bilgisayar programı düzensel olarak tanımlanmış bir dizi komuttan oluşur.
- **Örnek bir Algoritma**
  - Örneğimiz bir insanın evden çıkıp işe giderken izleyeceği yolu ve işyerine girişinde ilk yapacaklarını tanımlamaktadır.
    - Evden dışarıya çık
    - Otobüs durağına yürü
    - Durakta gideceğin yöndeki otobüsü bekle
    - Otobüsün geldiğinde otobüse bin
    - Biletini bilet kumbarasına at
    - İneceğin yere yakınlaştığında arkaya yürü
    - İneceğini belirten ikaz lambasına bas
    - Otobüs durunca in
    - İşyerine doğru yürü
    - İş yeri giriş kapısından içeriye gir
    - Mesai arkadaşlarıyla selamlaş
    - İş giysini giy
    - İşini yapmaya başla.

# ALGORİTMA VE AKIŞ ŞEMALARI

- Başarılı bir programın algoritması için
  - Her adım *son derece belirleyici* olmalıdır. Hiç bir şey şansa bağlı olmamalıdır.
  - Belirli bir sayıda adım sonunda algoritma sonlanmalıdır. (Bir şekilde program sonlanabilmelidir)
  - Algoritmalar karşılaşılabilecek tüm ihtimalleri ele alabilecek kadar genel olmalıdır.
  - Algoritmada algoritmanın genel işleyişini etkileyebilecek hiç bir belirsizlik olmamalıdır.
  - Algoritmada bazı adımlar yer değiştirebilir . Ancak bir çok adımın kesinlikle yer değiştiremeyeceğini bilmeliyiz.
    - Yanlış sıradaki adımlar algoritmanın yanlış çalışmasına neden olacaktır.

# ALGORİTMA VE AKIŞ ŞEMALARI

- Bilgisayara çözdürmek istediğiniz bir probleminiz var
  - Karşınızda bir makine olduğunu düşünün
  - İşlemi çözmek için çok basit adımlar belirleyin.
    - Hiçbir adımı atlamayın

# ALGORİTMA VE AKIŞ ŞEMALARI

- Örnek: Bilgisayara verilecek iki sayıyı toplayıp sonucu ekrana yazacak bir program için algoritma geliştirmek istiyoruz
  1. BAŞLA
  2. A sayısını oku
  3. B sayısını oku
  4.  $TOPLAM = A + B$  işlemini yap
  5. TOPLAM değerini ekrana yaz
  6. SON

# ALGORİTMA VE AKIŞ ŞEMALARI

- Örnek: Klavyeden girilecek iki sayıdan büyük olanından küçük olanını çıkarıp sonucu ekrana yazacak program için bir algoritma geliştiriniz.
  1. BAŞLA
  2. A sayısını oku
  3. B sayısını oku
  4. Eğer A büyüktür B  $SONUC=A-B$   
Değilse  $SONUC=B-A$
  5. SONUC değerini ekrana yaz
  6. SON

# ALGORİTMA VE AKIŞ ŞEMALARI

- Örnek: 1den klavyeden girilen bir n değerine kadar sayıları toplayan ve sonucu ekrana yazan bir algoritmayı geliştirelim.

1. BAŞLA
2. N OKU
3.  $T=0$
4.  $X=1$
5.  $T=T+X$
6.  $X=X+1$
7. EĞER  $X \leq N$  İSE 5. ADIMA GİT
8. T YAZ



# ALGORİTMA VE AKIŞ ŞEMALARI

- **Akış Çizgeleri**

- Bir algoritmanın şekillerle görsel ifadesidir.
  - Algoritma **doğal dille** yazıldığı için herkes tarafından anlaşılabilir
  - Ancak akış çizgelerinde her bir şekil standart bir anlam taşıdığı için farklı yorumlanıp anlaşılabilmesi mümkün değildir.
  - Her bir ifade için ayrı bir sembol kullanılmaktadır.

Algoritmanın başladığını ya da sona erdiğini belirtmek için kullanılır.

Klavye aracılığı ile giriş ya da okuma yapılacağını gösterir.

Yazıcı aracılığı ile çıkış yapılacağını gösterir.

Kart okuyucu aracılığıyla giriş yapılacağını gösterir.

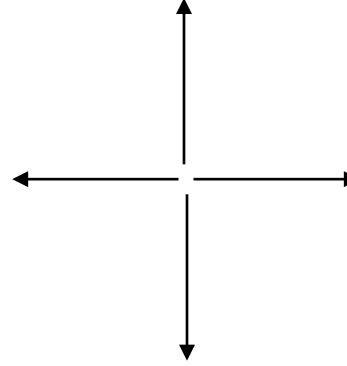
Yapılacak işler birden fazla sayıda yinelenecek İse, diğer bir deyişle iş akışında çevrim (döngü) var ise bu sembol kullanılır.

Araç belirtmeden giriş ya da çıkış yapılacağını gösterir.

Hesaplama ya da değerlerin değişkenlere aktarımını gösterir.

Aritmetik ve mantıksal ifadeler için karar verme ya da karşılaştırma durumunu gösterir.

Diskten okuma veya diskete yazmayı gösterir.



**Oklar için akış yönünü gösterir.**

# ALGORİTMA VE AKIŞ ŞEMALARI

- Bilgisayara verilecek iki sayıyı toplayıp sonucu ekrana yazacak bir program için algoritma geliştirmek istiyoruz

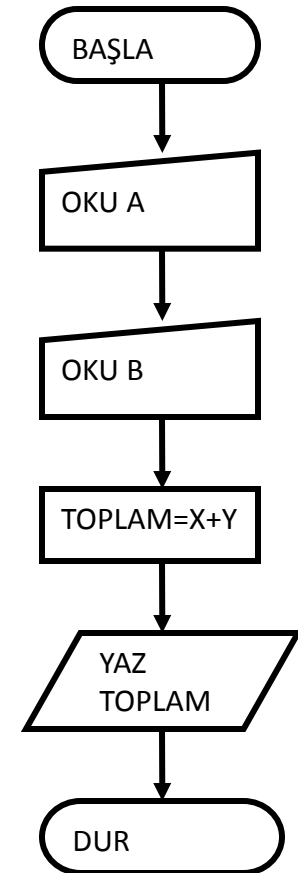
## ALGORİTMA

1. BAŞLA
2. A sayısını oku
3. B sayısını oku
4. TOPLAM=A + B işlemini yap
5. TOPLAM değerini ekrana yaz
6. SON

## C++ Program Kodu

```
#include<iostream>
using namespace std;
int main(){
    int A,B, TOPLAM;
    cin>>A; cin>>B;
    TOPLAM=A+B;
    cout<<TOPLAM;
}
```

## Akış Diyagramı

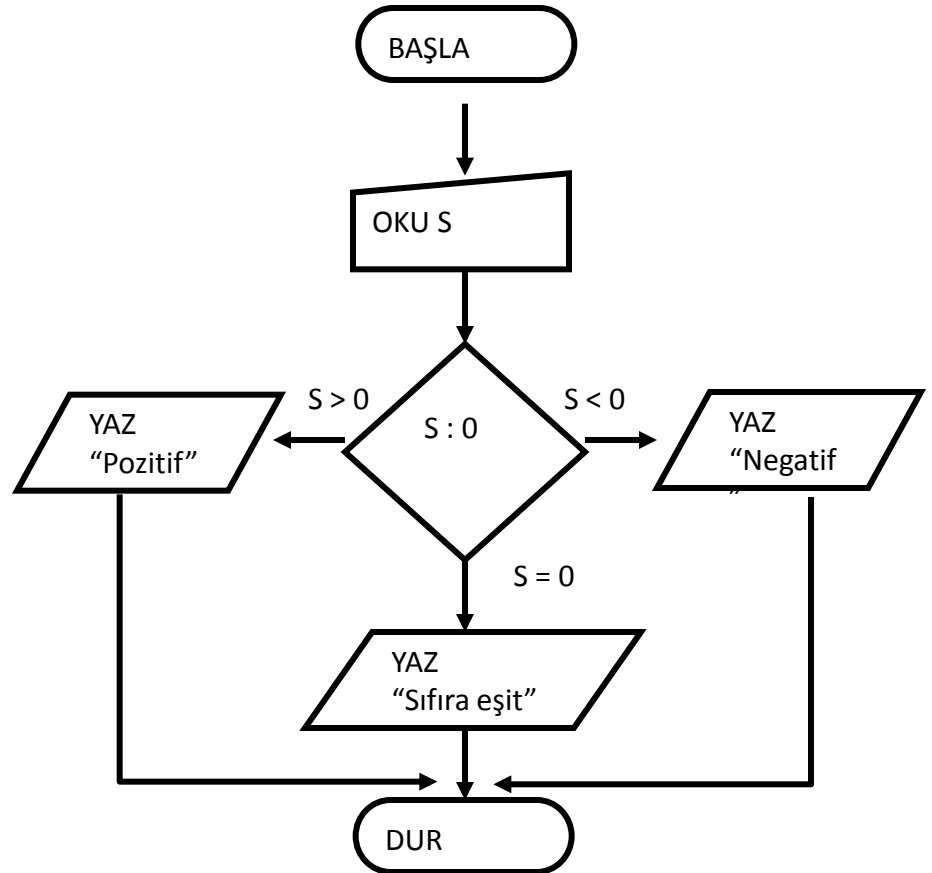


# ALGORİTMA VE AKIŞ ŞEMALARI

- Klavyeden girilen bir sayının pozitif, negatif veya sıfıra eşit olma durumunu hesaplayıp yazdıran algoritma ve akış şemasını hazırlayalım

## ALGORİTMA

- 1 : Başla
- 2 : Oku S
- 3 : Eğer  $S > 0$  ise "Pozitif" yaz,
- 4 : Eğer  $S < 0$  ise "Negatif" yaz,
- 5 : Eğer  $S = 0$  ise "Sıfıra eşit" yaz,
- 6 : Dur



# ALGORİTMA VE AKIŞ ŞEMALARI

- Klavyeden girilen bir yazıyı 5 kez yazdıran algoritma ve akış şemasını oluşturunuz.

( Y : Yazı, S : Sayaç )

1 : Başla

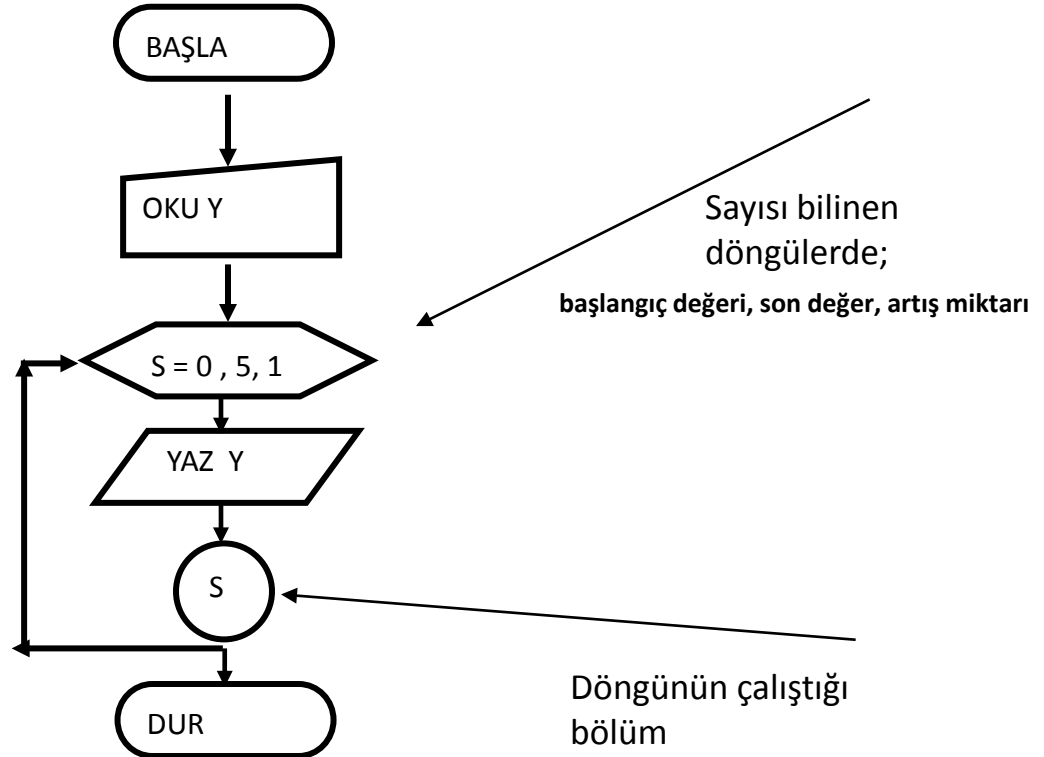
2 : Oku Y

3 : Yaz Y

4 :  $S = S + 1$

5 : Eğer  $S < 5$  ise A3 e git

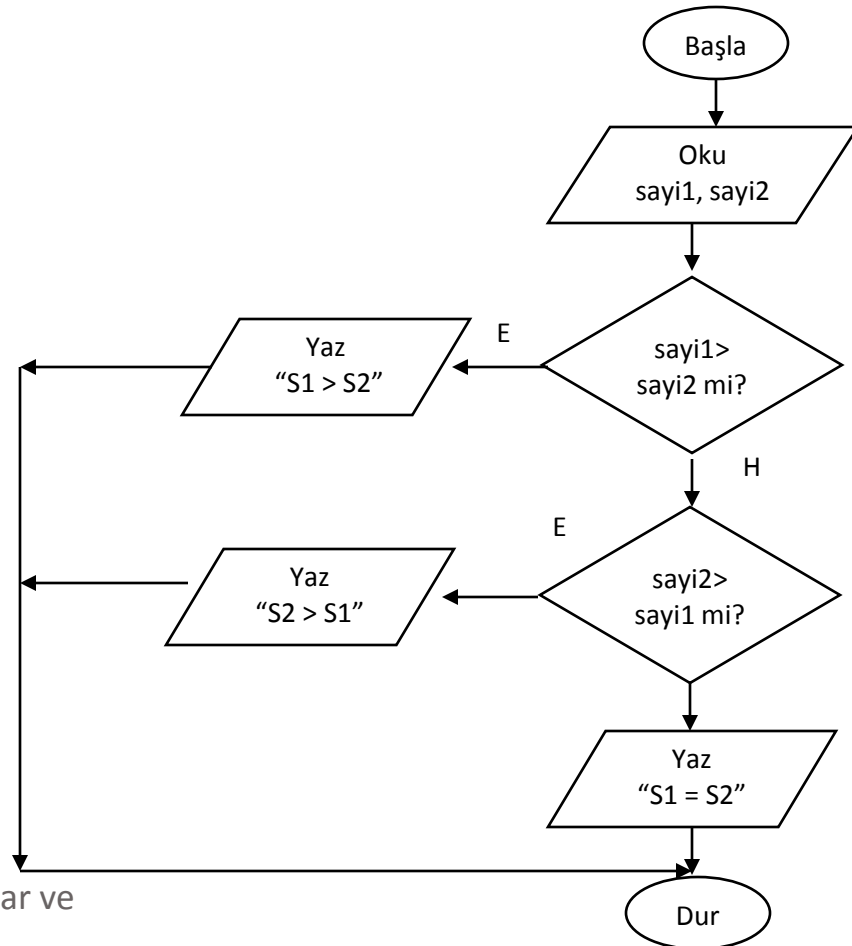
6 : Dur



# ALGORİTMA VE AKIŞ ŞEMALARI

- Klavyeden girilen 2 sayıyı karşılaştırıp sonucu ekrana yazdıran algoritma ve akış şeması

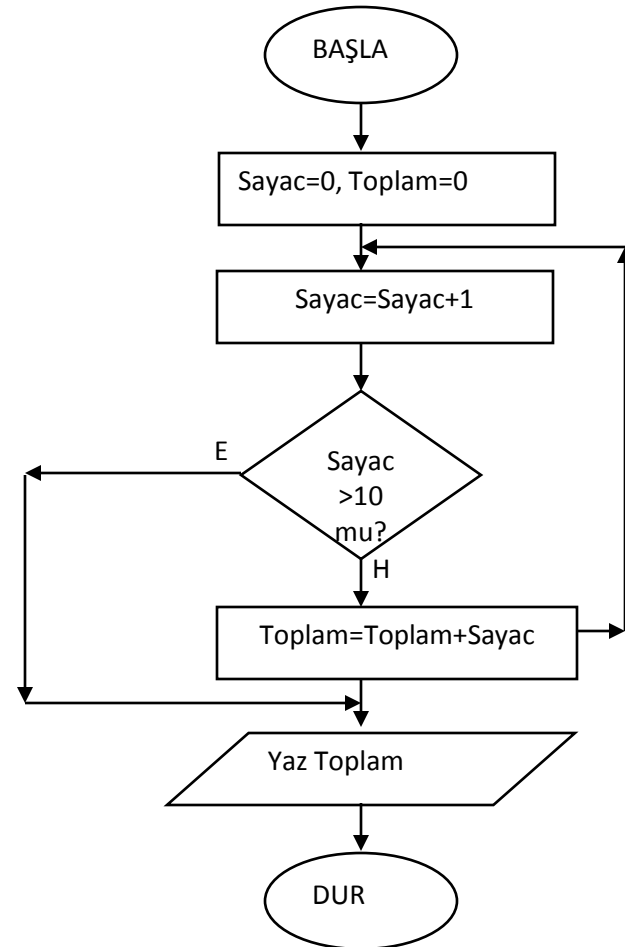
1. BAŞLA
2. OKU sayi1,sayi2
3. EĞER sayi1>sayi2 İSE  
YAZ "sayi1 sayi2'den büyüktür"
4. Değilse EĞER sayi2>sayi1 İSE  
YAZ "sayi2 sayi1'den büyüktür"
5. DEĞİL İSE  
YAZ "sayi1 sayi2'ye eşittir"
6. DUR



# ALGORİTMA VE AKIŞ ŞEMALARI

- 1-10 arasındaki tamsayıların toplamını bulan programın algoritma ve akış şemasını yazın?

1. BAŞLA
2. Sayac=0, Toplam=0
3. Sayac=Sayac+1
4. EĞER Sayac>10 İSE GİT 7
5. Toplam=Toplam+Sayac
6. GİT 3
7. YAZ "1-10 Arası Sayıların Toplamı=",Toplam
8. DUR

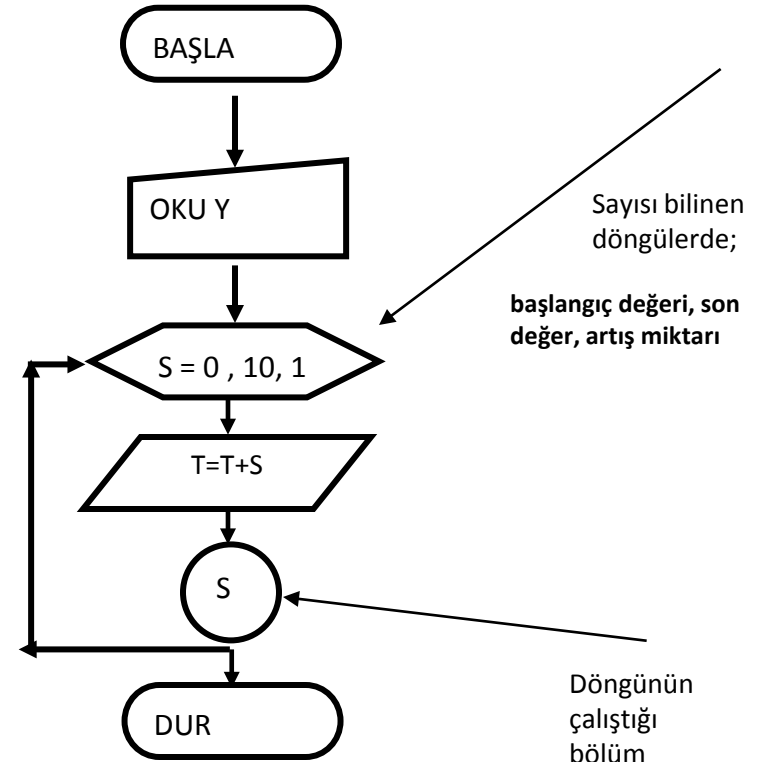


# ALGORİTMA VE AKIŞ ŞEMALARI

- 1-10 arasındaki tamsayıların toplamını bulan programın algoritma ve akış şemasını yazın

1. BAŞLA
2.  $S=0, T=0$
3.  $S=S+1$
4.  $T=T+S$
5. EĞER  $S \leq 10$  İSE GİT 3
7. YAZ "1-10 Arası Sayıların Toplamı=", $T$
8. DUR

S	T
0	0
$0+1=1$	$0+1=1$
$1+1=2$	$1+2=3$
$2+1=3$	$3+3=6$
$3+1=4$	$6+4=10$
$4+1=5$	$10+5=15$
$5+1=6$	$15+6=21$
$6+1=7$	$21+7=28$
$7+1=8$	$28+8=36$
$8+1=9$	$36+9=45$
$9+1=10$	$45+10=55$
$10+1=11$	





# ALGORİTMA VE AKIŞ ŞEMALARI

- N sayısını ekrandan okutarak faktöriyelini hesaplayan ve yazan programın algoritma ve akış şemasını yazın.

N=5 olduğunu varsayalım. İşlem şu şekilde yapılmalı:

$$\text{Faktöriyel} = 1*2*3*4*5$$

**Bilgisayar her problemi adım adım basit parçalar halinde çözer. O halde**

Sürekli üzerine çarpma yapabileceğim bir değişkenim olmalı.

Çarpmada etkisiz elemanla başlamalıyım. O halde:  $F=1$  olsun

**Diğer önemli nokta: Nereye kadar çarpma devam edecek ?**

Cevap N (5) e kadar çarpma devam edecek. O zaman bir kontrol mekanizması gerekiyor.

Birer birer artan ve N değerine ulaşıncaya kadar duran bir değişken.

F=1, Sayac=1 (Sayac = N oldu mu) Hayır

Sayac = Sayac+1 (Değer 2 oldu) F=F\*Sayac (  $F=1*2=2$  oldu) (Sayac = N oldu mu) Hayır

Sayac = Sayac+1 (Değer 3 oldu) F=F\*Sayac (  $F=2*3=6$  ) (Sayac = N oldu mu) Hayır

Sayac = Sayac+1 (Değer 4 oldu) F=F\*Sayac (  $F=6*4=24$  oldu) (Sayac = N oldu mu) Hayır

Sayac = Sayac+1 (Değer 5 oldu) F=F\*Sayac (  $F=24*5=120$  oldu) (Sayac = N oldu mu) **EVET**

# ALGORİTMA VE AKIŞ ŞEMALARI

- N sayısını ekrandan okutarak faktöriyelini hesaplayan ve yazan programın algoritma ve akış şemasını yazın.

Adım 1-Başla

Adım 2-N'i klavyeden oku

Adım 3-NFAK=1

Adım 4-ISAYI=1

Adım 5-ISAYI=ISAYI+1

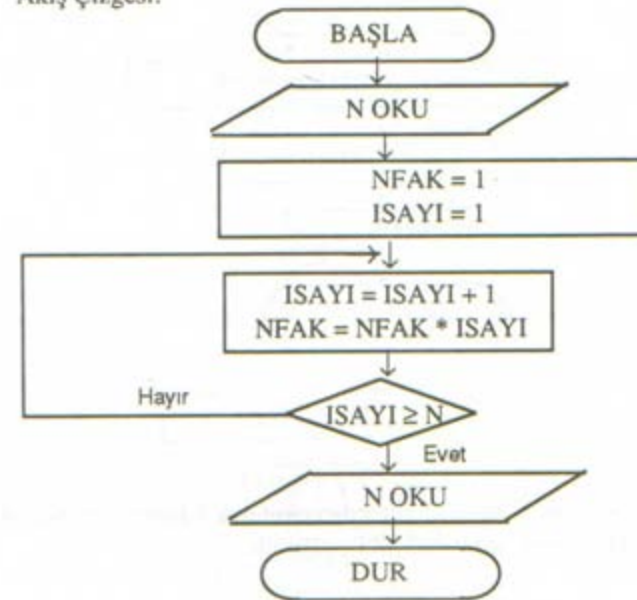
Adım 6-NFAK=NFAK\*ISAYI

Adım 7-Eğer ISAYI<N ise Adım 5e git

Adım 8-NFAK yaz

Adım 9-Dur

Akış Çizgesi:



# ALGORİTMA VE AKIŞ ŞEMALARI

- N sayısını ekrandan okutarak faktöriyelini hesaplayan ve yazan programın algoritma ve akış şemasını yazın.

Adım 1-Başla

Adım 2-N'i klavyeden oku

Adım 3-NFAK=1

Adım 4-ISAYI=1

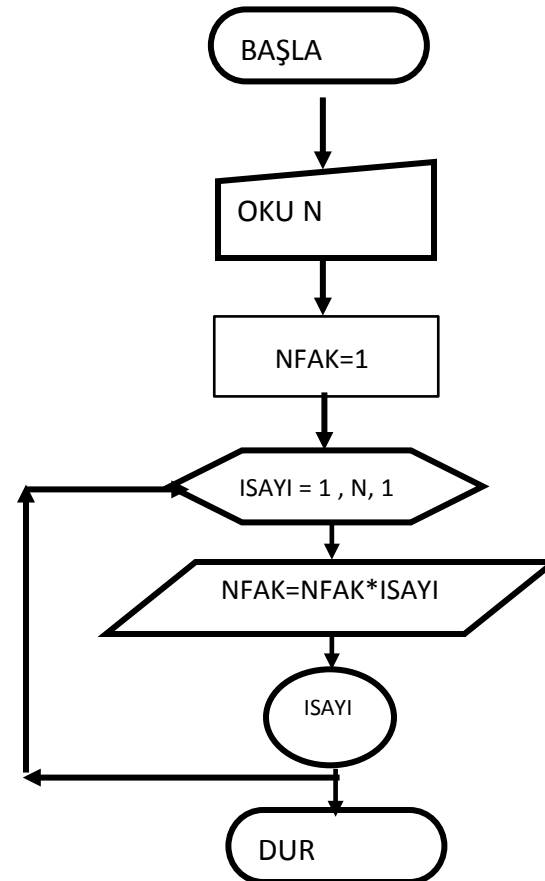
Adım 5-ISAYI=ISAYI+1

Adım 6-NFAK=NFAK\*ISAYI

Adım 7-Eğer ISAYI<N ise Adım 5e git

Adım 8-NFAK yaz

Adım 9-Dur



# ALGORİTMA VE AKIŞ ŞEMALARI

- 3 öğrencinin bir sınavdan aldıkları notların ortalamasını bulan ve yazan bir programın algoritma ve akış şeması

Yapılacak işlem:

3 kişinin notları klavyeden girilecek, toplanacak ve toplam 3 e bölünecek.

Öğrencilerin notları bana ayrı ayrı gerekiyor mu? **HAYIR**

Eğer notlar geldikçe üst üste toplarsam notlara ayrı ayrı ihtiyaç duymuyorum. O halde notlar tek bir hafıza alanı ayırmam ve bu alana sırasıyla notları almam mümkün. Tabii ki notların toplamı için de bir hafıza alanı ayırmak zorundayım.

Kontrol gerektiren diğer nokta öğrenci sayısı için kurmam gereken sayaç.

**Bu durumda kullanılacak değişken listesi**

ONOT:Öğrencinin notunu,  
INOT:Notların toplamını,  
NORT:Notların ortalamasını,  
ISAYI:Öğrenci sayısını gösterebilirsin.

# ALGORİTMA VE AKIŞ ŞEMALARI

- 3 öğrencinin bir sınavdan aldıkları notların ortalamasını bulan ve yazan bir programın algoritma ve akış şeması

## Algoritma

Adım 1-Başla

Adım 2-INOT=0

Adım 3-ISAYI=0

Adım 4-ONOT oku

Adım 5-INOT=INOT+ONOT

Adım 6-ISAYI=ISAYI+1

Adım 7-ISAYI<3 ise Adım 4'e git

Adım 8-NORT=INOT/3

Adım 9-NORT YAZ

Adım 10-DUR

Adım 2 ve 3'te INOT ve ISAYI ismi ile iki değişken için bellekte yer ayrılmış, ayrılan yerlere de ilk değer olarak sıfır atanmıştır.

Adım 4'te herhangi bir öğrencinin sınavdan almış olduğu notun değeri okutulmaktadır

Adım 5'te notların toplamının bulunması işlemi yer almaktadır. İşlemden ONOT, INOT ile toplanmakta ve sonuç notların toplamının saklandığı INOT'a aktarılmaktadır.

Adım 6'da öğrenci sayısını gösteren ISAYI değişkeninin değeri, "bir öğrenci için işlem yapıldı" anlamında 1 arttırılmaktadır.

Adım 7'de ISAYI'nın değeri 3 (toplam öğrenci sayısı) ile karşılaştırılmaktadır. Eğer sayı < 3 ise not okuma işlemi bitmediği için Adım 4'e dönmektedir. Eğer işlem 3 öğrenci için de yapılmışsa yani ISAYI=3 ise ya da ISAYI>3 ise ortalamasının hesaplandığı Adım 8'e geçilmektedir. Adım 9 da, bulunan ortalamasının yazılması ile ilgilidir.

# ALGORİTMA VE AKIŞ ŞEMALARI

- 3 öğrencinin bir sınavdan aldıkları notların ortalamasını bulan ve yazan bir programın algoritma ve akış şeması

## Algoritma

Adım 1-Başla

Adım 2-INOT=0

Adım 3-ISAYI=0

Adım 4-ONOT oku

Adım 5-INOT=INOT+ONOT

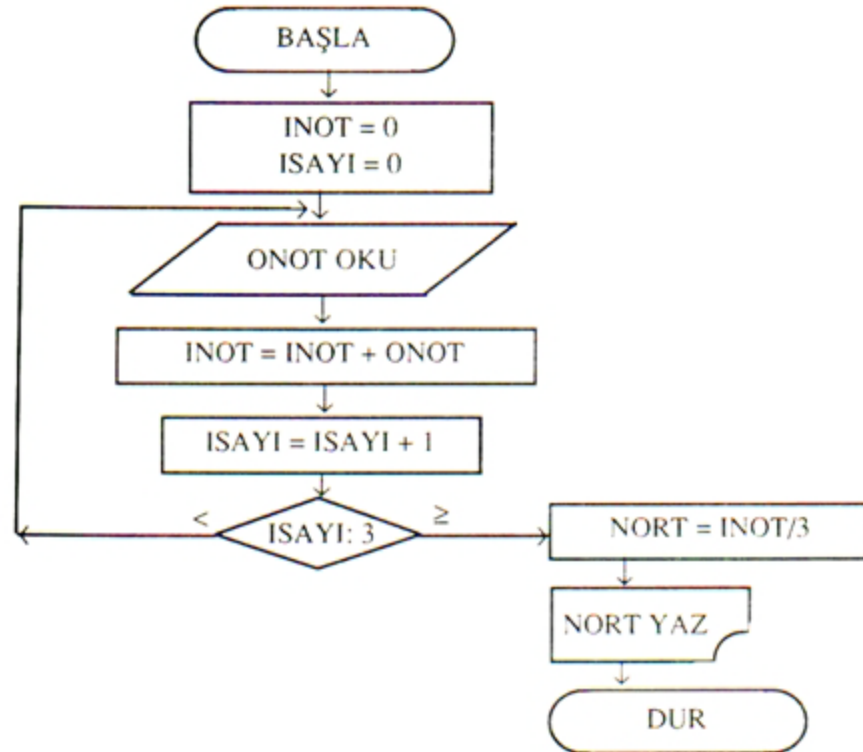
Adım 6-ISAYI=ISAYI+1

Adım 7-ISAYI<3 ise Adım 4'e git

Adım 8-NORT=INOT/3

Adım 9-NORT YAZ

Adım 10-DUR



# ALGORİTMA VE AKIŞ ŞEMALARI

- 3 öğrencinin bir sınavdan aldıkları notların ortalamasını bulan ve yazan bir programın algoritma ve akış şeması

## Algoritma

Adım 1-Başla

Adım 2-INOT=0

Adım 3-ISAYI=0

Adım 4-ONOT oku

Adım 5-INOT=INOT+ONOT

Adım 6-ISAYI=ISAIY+1

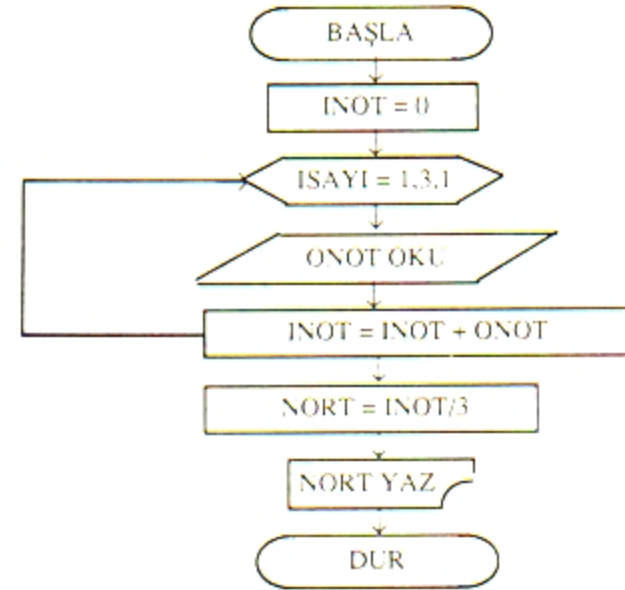
Adım 7-ISAYI<3ise Adım 4'e git

Adım 8-NORT=INOT/3

Adım 9-NORT YAZ

Adım 10-DUR

## DÖNGÜ KULLANARAK



# ALGORİTMA VE AKIŞ ŞEMALARI

- Klavyeden girilen, bir öğrencinin numarasını, ismini ve bilgisayar programlama dersinin 3 vize sınavından aldığı notları okuyan, bu notların aritmetik ortalamasını bulan, eğer ortalaması 50'ye eşit veya 50'den büyükse yazıcıya numara, isim, notlar ve vize notlarının ortalamasını, küçükse numara, isim ve "tekrar" mesajı yazan programın algoritma ve akış şeması

## Değişkenler

INO: öğrencinin numarası,

AD: öğrencinin ismi,

VIZE1: 1. vize sınavı,

VIZE2: 2. vize sınavı,

VIZE3: 3. vize sınavı,

VIZORT: sınavların aritmetik ortalaması

VIZTO: üç vizenin toplamı

## Algoritma

Adım 1-Başla

Adım 2-INO,AD,VIZE1,VIZE2,VIZE3,oku

Adım 3-VIZTO=(VIZE1+VIZE2+VIZE3)

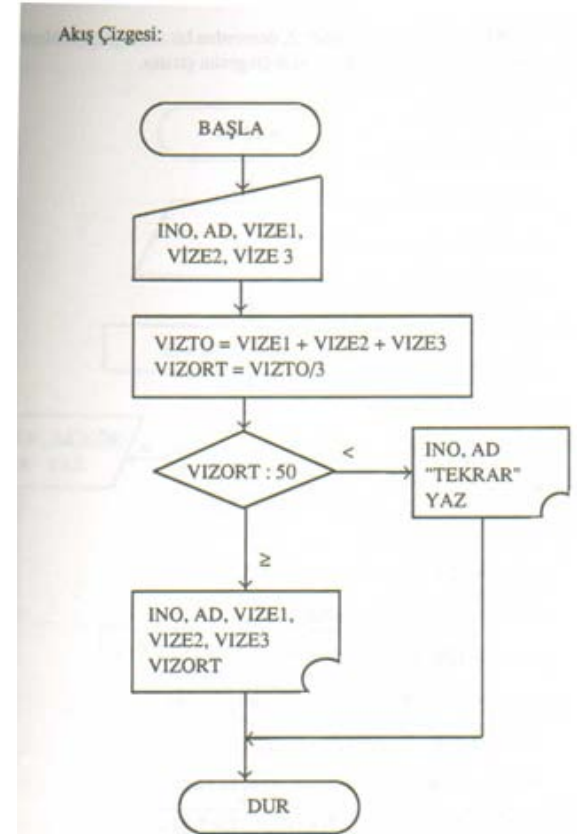
Adım 4 -VIZORT=VIZTO/3 bul.

Adım 5-Eğer VIZORT $\geq$ 50 ise Adım 7'ya git.

Adım 6-INO,AD ve "TEKRAR" yaz ve Adım 8'ye git.

Adım 7-INO,AD,VIZE1,VIZE2,VIZE3,VIZORT yaz

Adım 8-DUR





# ALGORİTMA VE AKIŞ ŞEMALARI

- 300 elemanlı bir veri grubunda bulunan pozitif,sıfır ve negatif değerlerin sayısını bulup yazan programın algoritması ve akış şeması

## Değişkenler

SS:Okunan sayı adedi

PSS:pozitif sayı adedi

NSS:Negatif sayı adedi

SSS:Sıfır sayı adedi

## Algoritma

Adım 1-Başla

Adım 2-SS=PSS=NSS=SSS=0

Adım 3-Sayı oku

Adım 4-SS=SS+1

Adım 5-Eğer SS>300 ise dur

Adım 6-Eğer sayı<0 ise adım 9'a git

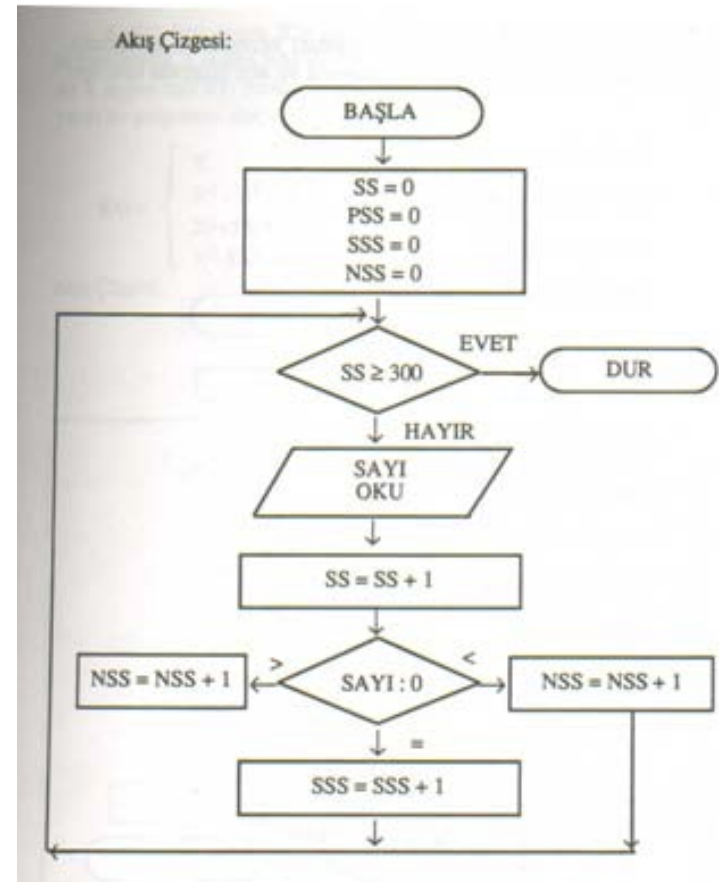
Adım 7-Eğer sayı=0 ise adım 10'a git

Adım 8-PSS=PSS+1 hesapla,adım 3'e git

Adım 9-NSS=NSS+1 hesapla,adım 3'e git

Adım 10-SSS=SSS+1 hesapla,adım 3'e git

MKÜ - Bilgisayar Mühendisliği Algoritmalar ve Programlama Ders Notları



# ALGORİTMA VE AKIŞ ŞEMALARI

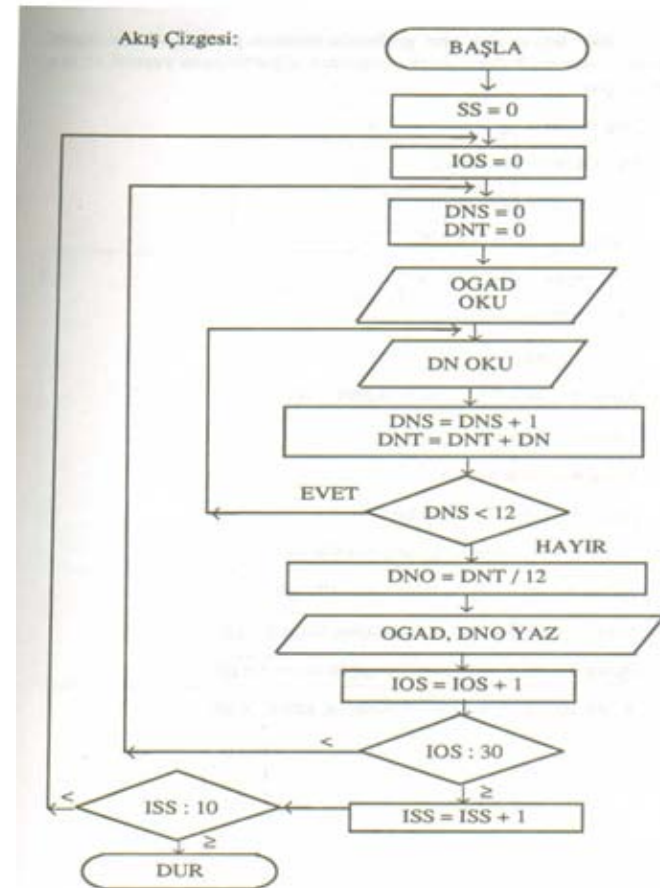
- Bir okulda bulunan 10 sınıftaki 30'ar öğrencinin herbirinin 12 şer dersten aldıkları notların ortalamasını bulan ve öğrenci ismi ile not ortalamasını yazan programın, algoritması ve akış şeması

## Değişkenler

ISS: Sınıf sayacı, IOS:Öğrenci sayacı,  
DN:Ders notu, DNS:Ders notu sayacı,  
DNT:Ders notlarının toplamı,  
DNO:Ders notlarının ortalaması,  
OGAD:Öğrencinin adı

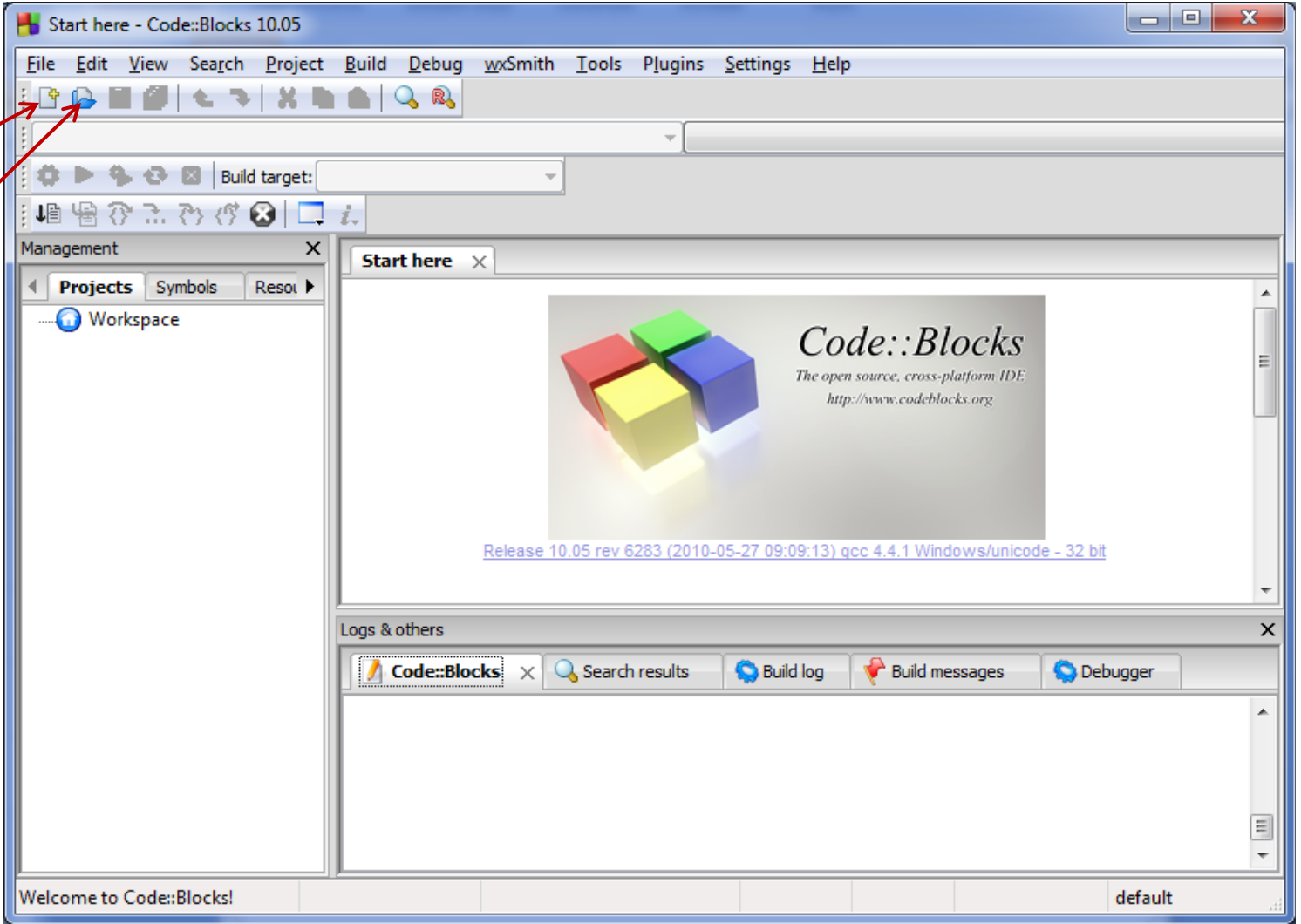
## Algoritma

- Adım 1-Başla
- Adım 2-SS=0
- Adım 3-IOS=0
- Adım 4-DNS=DNT=0
- Adım 5-OGAD oku
- Adım 6-DN oku
- Adım 7-DNS=DNS+1 (ders notu sayacı 1 artır.)
- Adım 8-DNT=DNT+DN (notları topla)
- Adım 9-Eğer DNS<12 ise adım 6'ya git
- Adım 10-DNO=DNT/12(ortalama hesapla)
- Adım 11-OGAD,DNO yaz
- Adım 12-IOS=IOS+1 (öğrenci sayacını bir artır)
- Adım 13-Eğer IOS<30 ise adım 4'e git
- Adım 14-ISS=ISS+1(sınıf sayacını bir artır)
- Adım 15-Eğer ISS<10 ise adım 3'e git.
- Adım 16-Dur



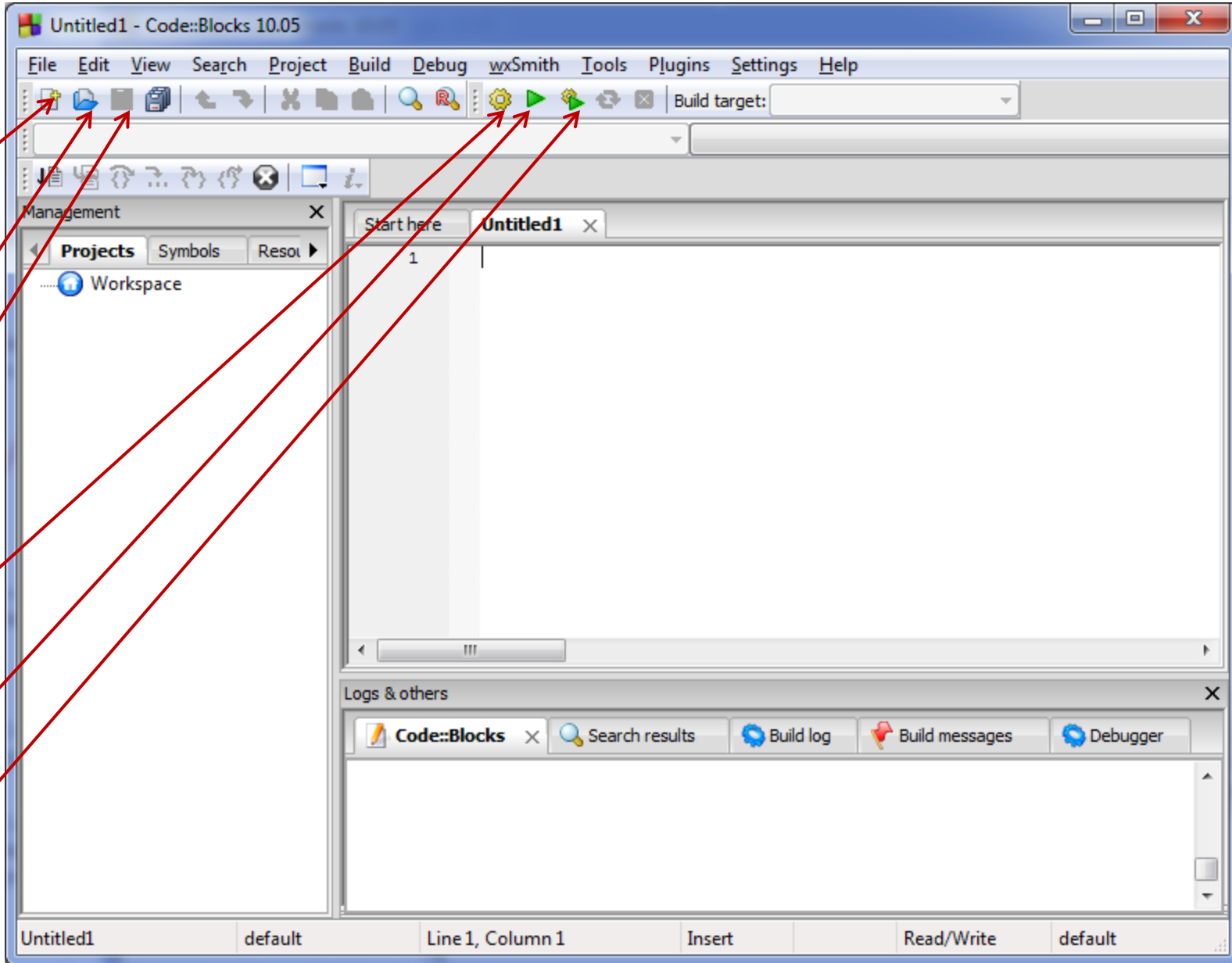
# C++

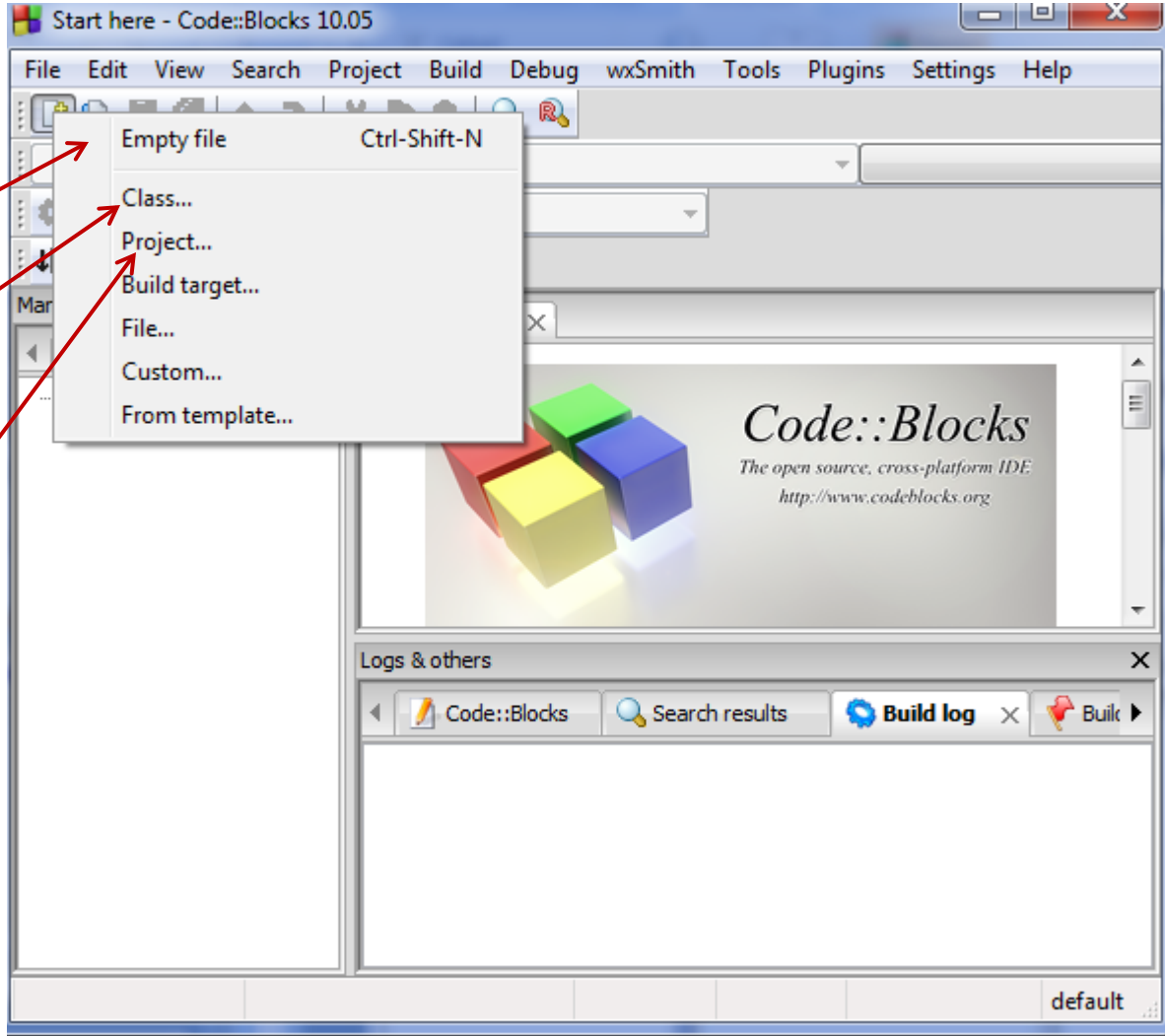
- **C++derlenen bir dildir**
  - **Editör ve Compiler (Derleyici) gerekir**
    - Sıradan bir notepad bile editör olabilir.
    - C++ için özel geliştirilmiş olan editörler hatalarımızı kolay bulmamızı sağlar
  - **İkisini birden sunan yazılımlar vardır.**
    - Örneğin Dev-C++ ya da CodeBlocks



Yeni bir C++  
dosyası açmak  
için

Önceden yaratılmış bir  
C++ dosyasını açmak için

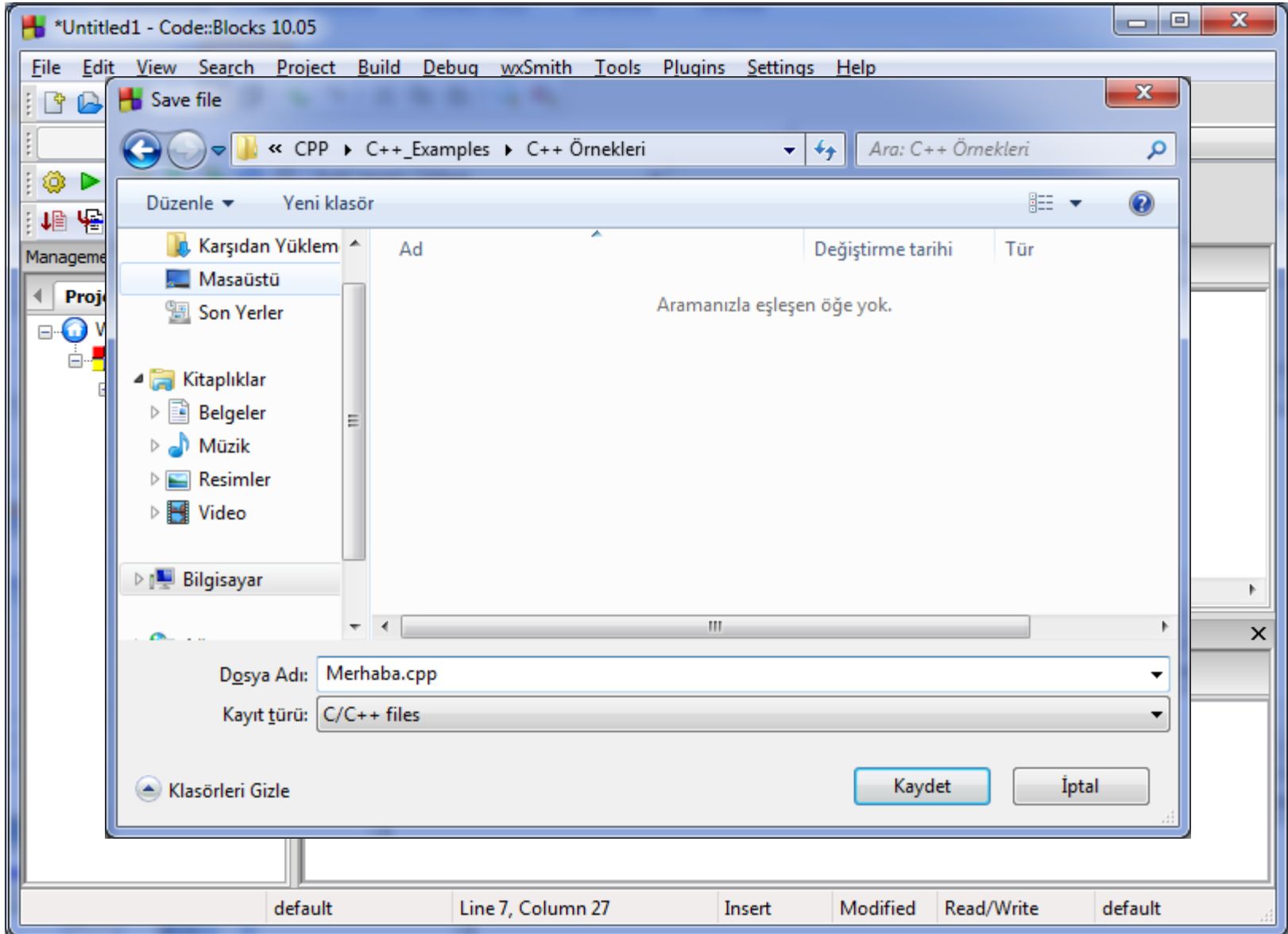




Boş bir C++ dosyası açmak için

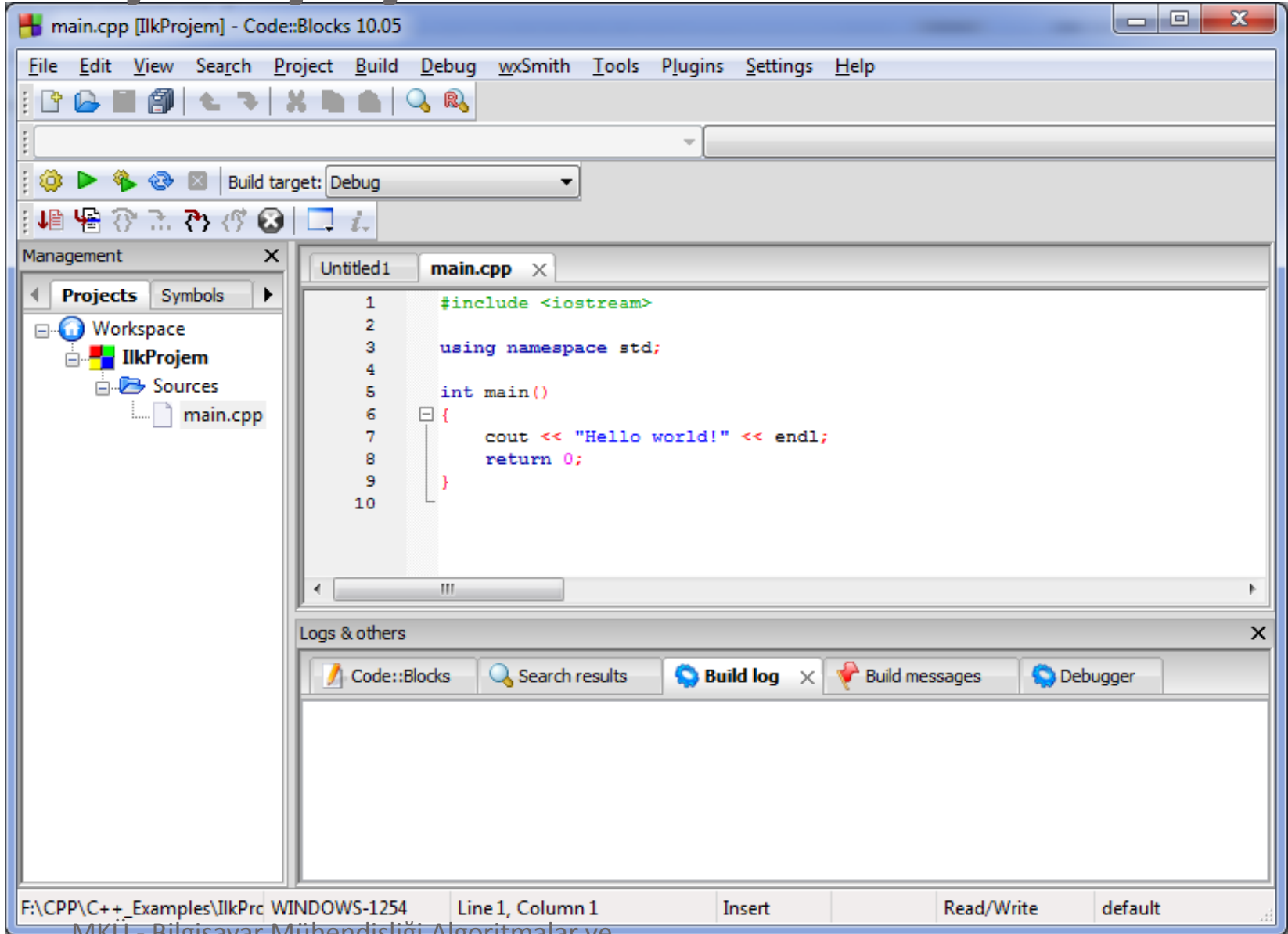
Boş bir C++ sınıfı açmak için

Boş bir C++ projesi açmak için



MKU - Bilgisayar Mühendisliği Algoritmalar ve Programlama Ders Notları

# Proje İle Çalışmak



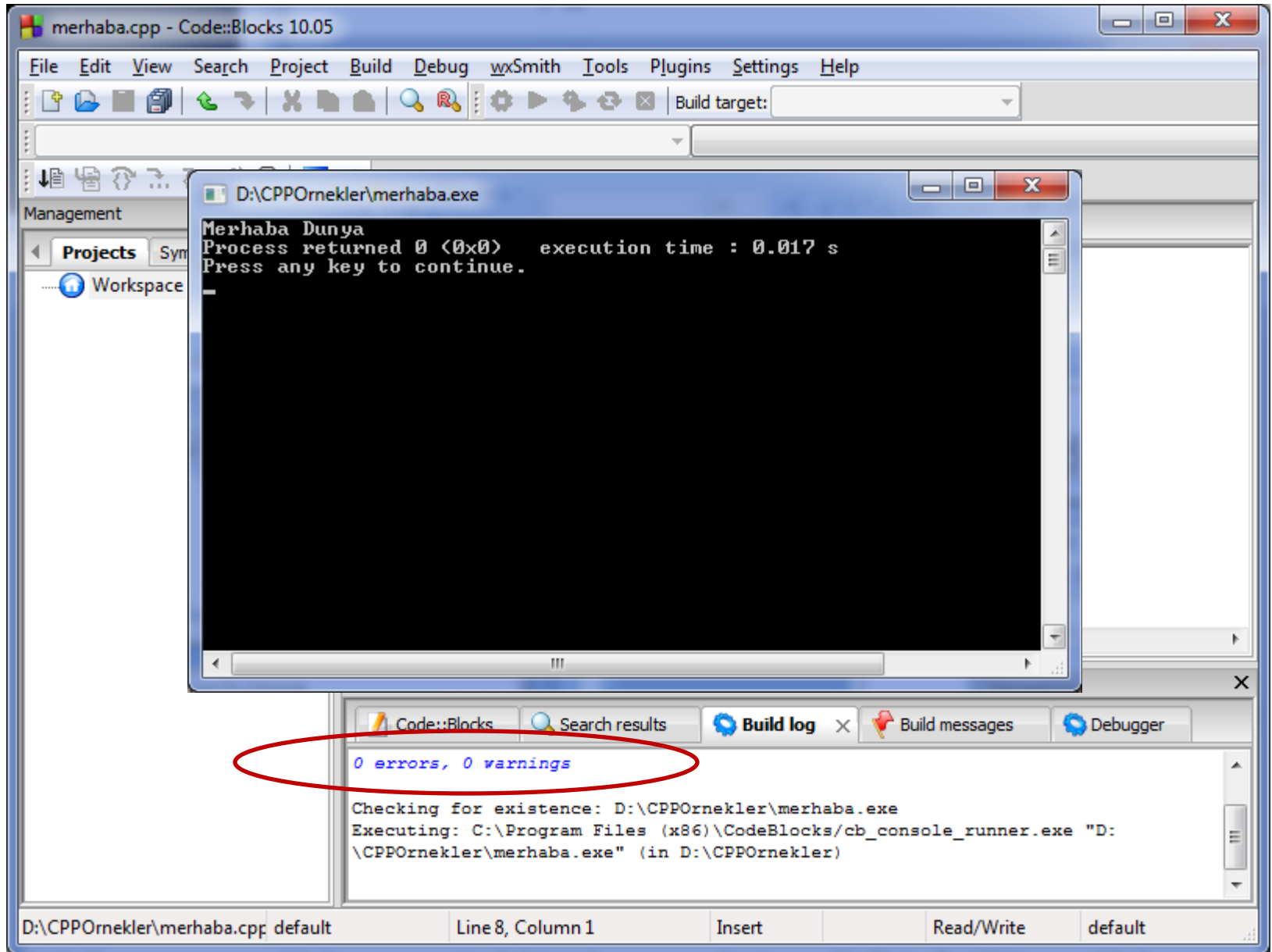


# Adım Adım Örnek Uygulama

The screenshot shows the Code::Blocks 10.05 IDE interface. The main window is titled '\*Untitled2 - Code::Blocks 10.05'. The menu bar includes File, Edit, View, Search, Project, Build, Debug, wxSmith, Tools, Plugins, Settings, and Help. The toolbar contains various icons, with the 'Build' icon (a green play button) circled in red. A 'Save file' dialog box is open, showing the current directory as 'Yerel Disk (D:) > CPPOrnekler'. The dialog displays a list of files and folders, with 'merhaba.cpp' selected. The file type is set to 'C/C++ files'. The 'Kaydet' (Save) button is highlighted. The status bar at the bottom shows 'Untitled2', 'default', 'Line 8, Column 1', 'Insert', 'Modified', 'Read/Write', and 'default'.

```
Compiling: G:\ESEN\KullanilanDosyalar\DOSYALARIM\DERSLER\CPP\C++_Examples\Ornekler\merhaba.cpp
Linking console executable: G:\ESEN\KullanilanDosyalar\DOSYALARIM\DERSLER\CPP\C++_Examples\Ornekler\merhaba.exe
Process terminated with status 0 (0 minutes, 0 seconds)
0 errors, 0 warnings
```

# Adım Adım Örnek Uygulama



# Hatalı Durumlarda!!!!

The screenshot shows the Code::Blocks IDE with a C++ program named `merhaba.cpp`. The code is as follows:

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     cout<<Merhaba Dunya;
6     return 0;
7 }
8
```

The IDE's **Logs & others** window shows the following error messages:

File	Line	Message
D:\CPPOrnekler...		In function 'int main()':
D:\CPPOrnekler...	5	<u>error: 'Merhaba' was not declared in this scope</u>
D:\CPPOrnekler...	5	<u>error: expected ';' before 'Dunya'</u>
Build finished: 2 errors, 0 warnings		

The status bar at the bottom of the IDE indicates the current state: `default`, `Line 5, Column 1`, `Insert`, `Read/Write`, and `default`.

# C++ PROGRAMLAMA DİLİ YAPISI

- Bir C++ programı en **basit** şekilde aşağıdaki gibidir. Bu program **hiçbir şey yapmaz**, ancak **hata vermez**.

```
int main(){  
    return 0;  
}
```

# C++ PROGRAMLAMA DİLİ YAPISI

- Bir C++ programı *en genel anlamda* şu şekildedir

```
//Eklemek istediğiniz yorumlar
```

```
Eklenmesi gereken Kütüphaneler;
```

```
tanımlanması gereken kullanıcı fonksiyonları;
```

```
int main(){
```

```
    icra Bloğu;
```

```
    return 0;
```

```
}
```

```
//İlk C++ Programım
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout<<"Merhaba Dünya";
```

```
    return 0;
```

```
}
```

# C++ PROGRAMLAMA DİLİ YAPISI

//İlk C++ Programım

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout<<"Merhaba Dünya";
```

```
    return 0;
```

```
}
```

cout komutu için kullanmamız zorunludur.

cout komutunu bu şekilde kullanabilmemiz için gereklidir. Eklenmemesi halinde programın çalışabilmesi için bu satırı:  
*std::cout<<"Merhaba Dünya";*  
şeklinde yamamız gerekirdi.

# C++ PROGRAMLAMA DİLİ YAPISI

- **Eklemek istediğiniz yorumlar :**
  - Bir C++ programının herhangi bir bölümünde // ile başlayarak program hakkında bilgi yazabilirsiniz
- **Eklenmesi gereken Kütüphaneler**
  - # işareti ile başlayan satırlar direk olarak önışlemciye yollanır.
    - Örneğin #include <iostream> komutu önışlemciye program için iostream standart dosyasının eklenmesi gerektiğini söyler
      - Böylelikle standart input/output komutlarını programımızda kullanabilme imkanımız olur.
      - Ekrana yazdırma, klavyeden okuma yapma v.s.
- **using namespace std**
  - Standart C++ kütüphanelerinin tüm elemanları bir isim uzayında (namespace) tanımlıdır.
  - Kullanmak istediğimiz komutları belli bir alan içerisinde kullanmamız için gereklidir
  - Kullanılmaması durumunda tüm komutların kullanım alanı program içerisinde tek tek belirtilmelidir (ki bu çok zahmetli bir iştir)
  - Bu satırı pek çok (belki de tüm) yazılan programlarda kullanacağız.

# C++ PROGRAMLAMA DİLİ YAPISI

- **int main(){**
  - Bu satır C++programının ana fonksiyonunun başladığını gösterir
    - Programda ilk işlenecek olan fonksiyon budur
      - Programdaki yerinin önemi yoktur
      - **Hangi satırda yazılmış olursa olsun ilk işlenecek bölümdür**
      - Her program main() fonksiyona sahip olmalıdır.
  - main, bir *fonksiyon* olduğu için () tarafından izlenir
  - Fonksiyon içerisindeki kodlar { } arasına yazılır.



# C++ PROGRAMLAMA DİLİ YAPISI

- **cout<<"Merhaba Dünya";**
  - Bu satır C++ ifadesidir (statement).
    - Statement: Bir sonuç üretilmesini sağlayan basit bir deyimdir.
    - Bu satır ekrana bir bilgi yazılmasını sağlar.
    - cout: C++ için standart output komutudur.
      - Amacı output'a istenilen karakterleri yazdırmayı sağlamaktır
        - Bu örnekte ekrana yazılacak olan karakterler: **Merhaba Dünya**
      - Satırın sonuna eklenen ; karakteri bu komut satırını bittiğini yeni ifadeye geçildiğini gösterir.
      - **En sık rastlanan hata, ifadelerin sonuna ; konulmamasıdır.**
- **return 0;**
  - Bu statement main fonksiyonun bittiğini gösterir.
    - Bu komuttan sonra main program ı bitirileceği için bu noktadan sonra yazılan komutların hiçbir önemi yoktur.

# cout (Biraz Daha Bilgi)

- cout programımızdan output'a bilgi akışını sağlayan nesnedir.
  - Kullanılan operatör : **>>**
  - Her türlü output'a bilgi yazdırmak amacıyla kullanılması **zorunlu** olan bir komuttur.
  - **using namespace std** kullanıldığında, cout standart (**std**) output'a (ekran) bilgi yazar.
  - **ostream** (output stream) sınıfına ait bir nesnedir
    - Program içinde kullanılabilmesi için bu sınıfın programa eklenmesi gereklidir:
      - **#include<ostream>**
      - **iostream** sınıfı **istream** ve **ostream** sınıflarının tüm özelliklerini kalıtım yoluyla aldığı için, **iostream** sınıfı dahil edilerek **istream** ve **ostream** sınıfları dahilindeki tüm komutlar kullanılabilir.

# C++ PROGRAMLAMA DİLİ YAPISI

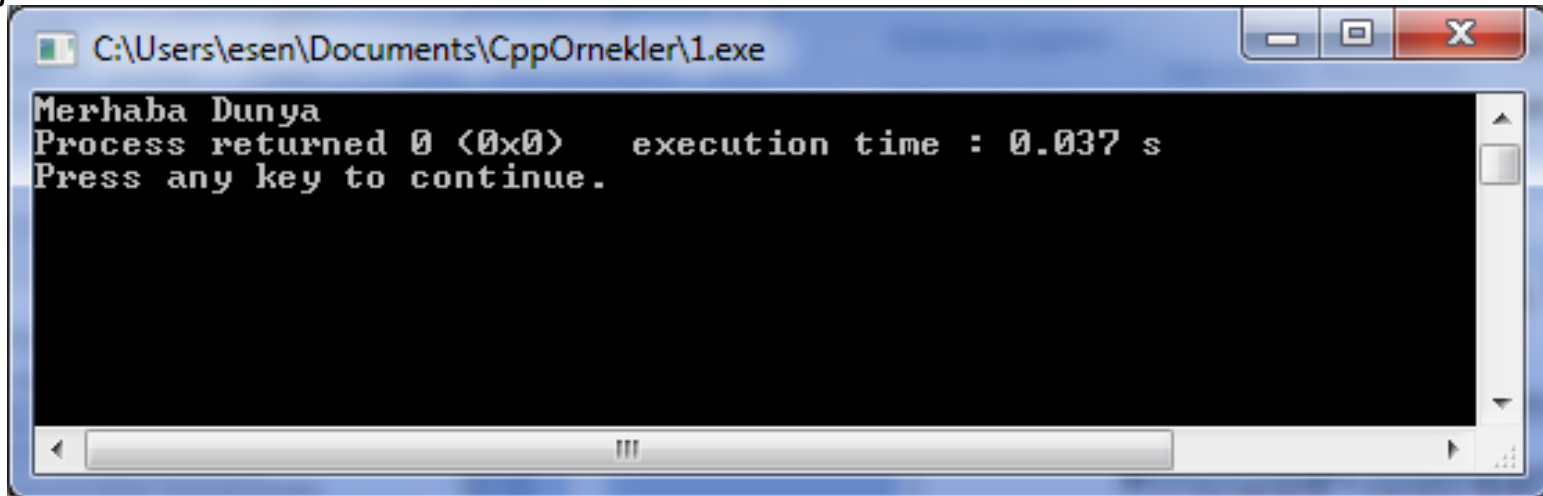
```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main()
{
    cout<<"Merhaba Dünya";
    return 0;
}
```

```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main(){ cout<<"Merhaba Dünya"; return 0;}
```

; komutun bittiğini gösterir, bu nedenle diğer ifadeyi yazmak için alt satıra geçme zorunluluğu yoktur.

# C++ PROGRAMLAMA DİLİ YAPISI

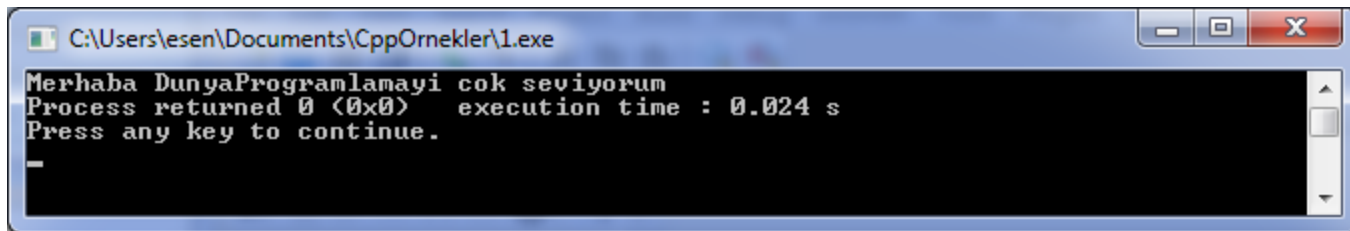
```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main()
{
    cout<<"Merhaba Dunya";
    return 0;
}
```



```
C:\Users\esen\Documents\CppOrnekler\1.exe
Merhaba Dunya
Process returned 0 (0x0) execution time : 0.037 s
Press any key to continue.
```

# C++ PROGRAMLAMA DİLİ YAPISI

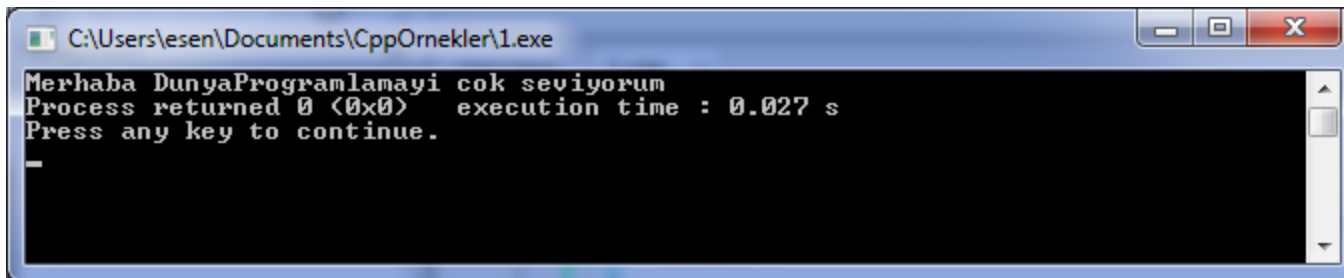
```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main()
{
    cout<<"Merhaba Dunya";cout<<"Programlamayı cok seviyorum";
    return 0;
}
```



```
C:\Users\esen\Documents\CppOrnekler\1.exe
Merhaba DunyaProgramlamayı cok seviyorum
Process returned 0 (0x0) execution time : 0.024 s
Press any key to continue.
-
```

# C++ PROGRAMLAMA DİLİ YAPISI

```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main()
{
    cout<<"Merhaba Dunya"<<"Programlamayı çok seviyorum";
    return 0;
}
```

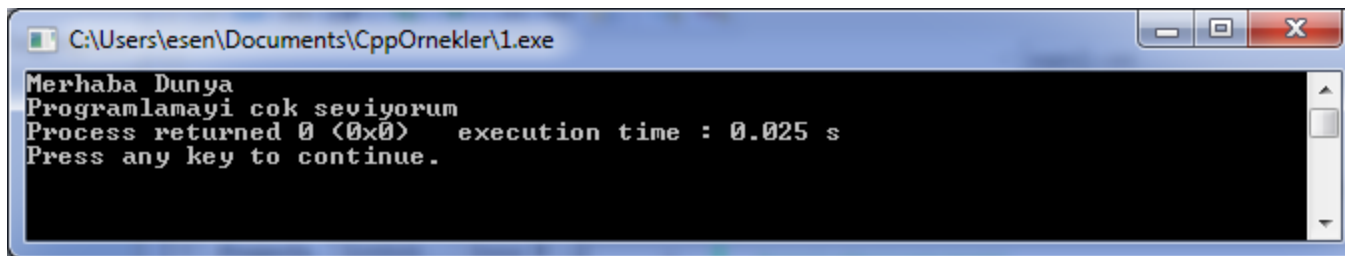


The screenshot shows a Windows command prompt window titled "C:\Users\esen\Documents\CppOrnekler\1.exe". The output of the program is displayed as follows:

```
Merhaba DunyaProgramlamayi çok seviyorum
Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.
-
```

# C++ PROGRAMLAMA DİLİ YAPISI

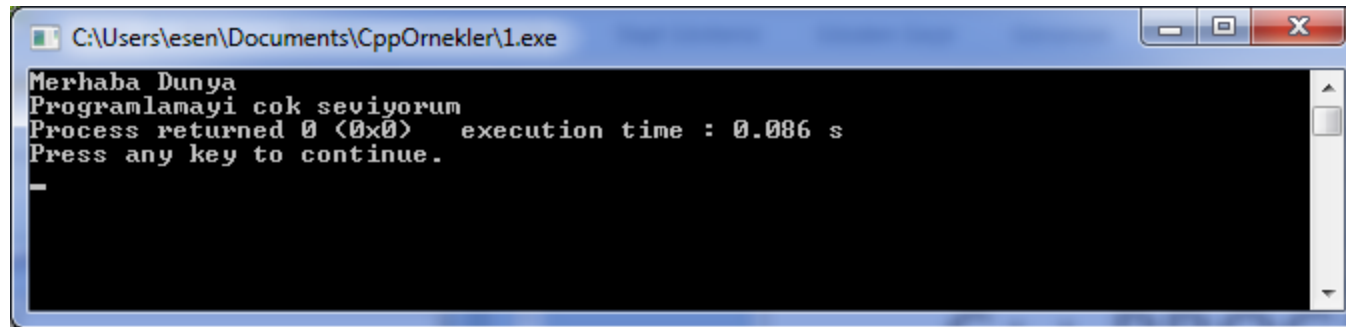
```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main()
{
    cout<<"Merhaba Dunya"<<endl;cout<<"Programlamayı çok seviyorum";
    return 0;
}
```



```
C:\Users\esen\Documents\CppOrnekler\1.exe
Merhaba Dunya
Programlamayı çok seviyorum
Process returned 0 (0x0) execution time : 0.025 s
Press any key to continue.
```

# C++ PROGRAMLAMA DİLİ YAPISI

```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main(){
    cout<<
    "Merhaba Dunya"
    <<endl;
    cout
    <<"Programlamayi cok seviyorum";
    return 0;
}
```



```
C:\Users\esen\Documents\CppOrnekler\1.exe
Merhaba Dunya
Programlamayi cok seviyorum
Process returned 0 (0x0) execution time : 0.086 s
Press any key to continue.
```



# C++ PROGRAMLAMA DİLİ YAPISI

```
//ilk C++ Programım
#include<iostream>
using namespace std;
int main(){
    cout<< "Merhaba Dunya"; //Ekрана Merhaba Dunya yaz
    cout<< endl;          //Alt satıra geç
    cout<<"Programlamayi cok seviyorum";
    return 0;
}
```

```
C:\Users\esen\Documents\CppOrnekler\1.exe
Merhaba Dunya
Programlamayi cok seviyorum
Process returned 0 (0x0) execution time : 0.028 s
Press any key to continue.
```

# Klavyeden Okuma : cin deyimi

- cout ekrana bilgi yazmayı sağlar
- Dışarıdan programa bilgi akışı sağlayabilmek için **cin** deyimi kullanılması **zorunludur**.
  - **using namespace std** kullanıldığında, **cin** standart (**std**) inpt'tan (klavye) bilgi okumamızı sağlar.
  - **istream** (input stream) sınıfına ait bir nesnedir ve kullanılabilmesi için **#include<istream>** kütüphanesinin programın başına eklenmesi gereklidir.
  - **iostream** sınıfı **istream** ve **ostream** sınıflarının tüm özelliklerini kalıtım yoluyla aldığı için, **iostream** sınıfı dahil edilerek **istream** ve **ostream** sınıfları dahilindeki tüm komutlar kullanılabilir.

# Klavyeden Okuma : cin deyimi

- cin ile klavyeden bilgi girişi yapılabilir
  - kullanılacak operatör : **>>**
- Örnek:
  - cin >> yas;
  - yas adlı değişkene klavyeden alınacak bilginin atanmasını sağlar
  - Bu şekilde kullanım için değişkenin öncelikle tanımlanması gerekir.
- Ayrıntılı kullanım ilerleyen haftalarda..

← C++ Programı

```
1 #include <iostream>
2 #include <string>
3 #include <sstream>
4
5 using namespace std;
6
7 int main() {
8     string input = "";
9
10    // How to get a string/sentence with spaces
11    cout << "Please enter a valid sentence (with spaces):\n";
12    getline(cin, input);
13    cout << "You entered: " << input << endl << endl;
14
15    // How to get a number.
16    int myNumber = 0;
17
18    while (true) {
19        cout << "Please enter a valid number: ";
20        getline(cin, input);
21
22        // This code converts from string to number safely.
23        stringstream myStream(input);
24        if (myStream >> myNumber)
25            break;
26        cout << "Invalid number, please try again" << endl;
27    }
28    cout << "You entered: " << myNumber << endl << endl;
29
30    // How to get a single char.
31    char myChar = {0};
32
33    while (true) {
34        cout << "Please enter 1 char: ";
35        getline(cin, input);
36
37        if (input.length() == 1) {
38            myChar = input[0];
39            break;
40        }
41
42        cout << "Invalid character, please try again" << endl;
43    }
44    cout << "You entered: " << myChar << endl << endl;
45    cout << "All done. And without using the >> operator" << endl;
46
47    return 0;
48 }
49
50 }
```

Programı düzelt

Hatalı program ya da istenmeyen sonuçlar



Kodlayıcı  
(yani siz)

## Kullanıma hazır program

```
1 #include <iostream>
2 #include <string>
3 #include <sstream>
4
5 using namespace std;
6
7 int main() {
8     string input = "";
9
10    // How to get a string/sentence with spaces
11    cout << "Please enter a valid sentence (with spaces):\n>";
12    getline(cin, input);
13    cout << "You entered: " << input << endl << endl;
14
15    // How to get a number.
16    int myNumber = 0;
17
18    while (true) {
19        cout << "Please enter a valid number: ";
20        getline(cin, input);
21
22        // This code converts from string to number safely.
23        stringstream myStream(input);
24        if (myStream >> myNumber)
25            break;
26        cout << "Invalid number, please try again" << endl;
27    }
28    cout << "You entered: " << myNumber << endl << endl;
29
30    // How to get a single char.
31    char myChar = {0};
32
33    while (true) {
34        cout << "Please enter 1 char: ";
35        getline(cin, input);
36
37        if (input.length() == 1) {
38            myChar = input[0];
39            break;
40        }
41        cout << "Invalid character, please try again" << endl;
42    }
43    cout << "You entered: " << myChar << endl << endl;
44    cout << "All done. And without using the >> operator" << endl;
45    return 0;
46 }
```

Programdan kullanıcıya  
bilgi akışı:  
output (**cout**)



Kullanıcı

Kullanıcıdan programa bilgi akışı:  
input (**cin**)

# Değişkenler

- Daha önce yazdığımız "merhaba dünya" programı sadece ekrana bilgi yazdırmak içindi
- Asıl programların hemen hemen tümünde değişkenlerle çalışmak zorundayız.
- Programlarımızda işlemlerimizi yaparken verileri kullanırız.
- Mesela herhangi iki sayıyı toplarız veya iki tane karakter dizisini (string) karşılaştırırız.
- Bu işlemler için kullandığımız verilerimizi değişkenler içinde tutarız.
- Değişkenler bilgisayar hafızasında verileri depolayan ve isimleri olan, programlamanın en temel elementleridir.
- Değişkenlere ulaşabilmemiz için isimlerinin olması gerekir.
  - Ancak gerçekte değişkenler sadece hafızada belli bir adreste tutulan bilgilerdir.
  - Değişken isimleri, bizim bilgiye ulaşmak için bakmamız gereken adreslere ulaşmamızı sağlayan araçlardır.
- C++ dilinde bir değişkeni kullanmadan önce onu tanımlamak zorundayız.
  - Tanımlamayı değişkene uygun bir isim verme ve değişkenin hangi tipten olduğunu bildirmeye yaparız.

# Değişkenler

- Değişken isimlerini verirken C++'ın bir takım sıkı kurallarına uymamız gerekir.
  - Değişkenlerin isimleri, alfabede bulunan karakterlerle başlamalı.
    - İlk harf hariç diğer karakterler sayı olabilir.
  - İçerisinde **Türkçe karakterler bulunmamalı**
  - C++ büyük ve küçük harf duyarlıdır. Yani **Sayi**, **sayi** ve **SAYI** hepsi ayrı değişken olarak algılanırlar.
  - Değişkenlerin isimleri **!, ?, {, }** ve **boşluk gibi karakterler içeremezler.**
    - **\_** değişken isimlerinde kullanılabilir
  - C++'ın anahtar kelimelerini de değişken isimleri olarak kullanamayız.

# Değişkenler

- **sayi, tamsayi1, toplam, Fark, KullaniciAdi, isim, \_Adres, sinif\_ortalaması**, kurallara göre adlandırılmış doğru değişken isimleridir.
- Diğer taraftan **1.sayi, tamsayi 1, fark!, 3.sinif\_ortalaması** geçersiz değişken isimleridir.
- Yanlış adlandırılmış değişkenleri içeren programlar derlenmez!
- Anahtar kelimeler C++ dilinde bulunan komutların isimleridir.
  - Bunları değişken ismi olarak kullanamayız.
  - Ayrıca alt çizgi ile başlayan değişken tanımlamadan kaçınmalıyız.
    - Çünkü genelde C++ kütüphanelerini yazan programcılar değişkenlerini alt çizgi ile başlayan isimler verirler. Bu da isimler arasında çakışma yaratabilir.



# Veri Türleri

- Verileri bilgisayarda program çalışırken bellekte (RAM) depolanır.
- Bilgisayar belleği bitlerden oluşmuştur.
  - Bir bit temel olarak 1 veya 0 değerini alır.
  - Sekiz tane bit bir byte eder.
  - Bilgisayarın hafızasında verilerin kapladıkları alanları byte türünden ifade ederiz
  - Değişkenleri ihtiyacımıza göre değişik tiplerde tanımlarız kullanırız.

# Veri Türleri

- C++ dilinde hazır bulunan temel veri tipleri şunlardır:

Değişken	Boyut*	Açıklaması	Değer Aralığı
char	1	karakter veya 8 bit uzunluğunda tamsayı	signed: -128 ile 127 arasında unsigned: 0 ile 255
short int (short)	2	16 bit uzunluğunda tamsayı	signed: -32768 to 32767 unsigned: 0 to 65535
long int (long)	4	32 bit uzunluğunda tamsayı	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
int	4	32 bit uzunluğunda tamsayı	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	4	Kesirli sayı.	+/- 3.4e +/- 38 (~7 basamak)
double	8	Geniş ve fazla duyarlıklı kesirli sayı.	1.7e +/- 308 (15 basamak)
bool	1	true(doğru) veya false(yanlış) değerini alır. Eski derleyiciler bu türü desteklemeyebilir. Yeni ANSI C++ standardında eklenmiştir.	
wchar_t	2	char tipinden geniş olur Unicode tipinde değişkenleri destekler.	

# Değişkenlerin Deklarasyonu

- C++ programında bir değişkeni kullanabilmek için öncelikle onu tanımlamalıyız (**declaration**).
  - Data tipini ve değişken ismini belirtiriz
  - Örnekler:
    - **int a;**
      - a adında bir *tamsayı* tanımladık
    - **float sayi;**
      - sayi adında bir *float* (kesirli sayı) tanımladık
  - Birden fazla aynı tipte değişken tanımlamak için:
    - **int a, b, c;**
      - her birini ayrı ayrı tanımlamak ile aynı
        - **int a; int b; int c;**

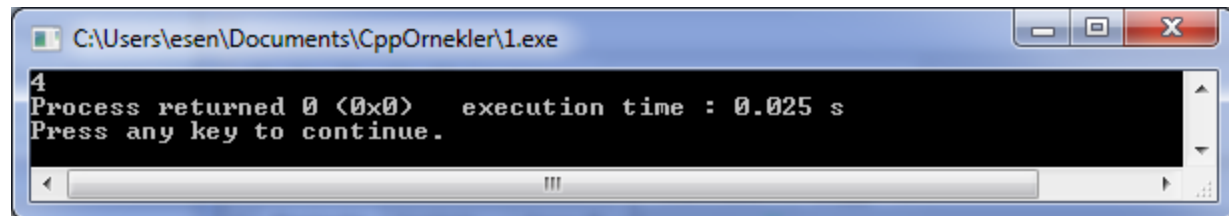
# Değişkenler ile çalışmak

```
#include <iostream>
using namespace std;
int main ()
{
    // değişken deklarasyonu:
    int a, b;
    int result;

    // işlem:
    a = 5;
    b = 2;
    a = a + 1;
    result = a - b;

    // sonucu ekrana yaz:
    cout << result;

    //programı bitir:
    return 0;
}
```



```
C:\Users\esen\Documents\CppOrnekler\1.exe
4
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```

# Değişkenlerin Limitlerini Görüntüleme

```
#include <iostream>
```

```
#include
```

```
using na
```

```
int main
```

```
{
```

```
cout <<
```

```
cout <<
```

```
cout <<
```

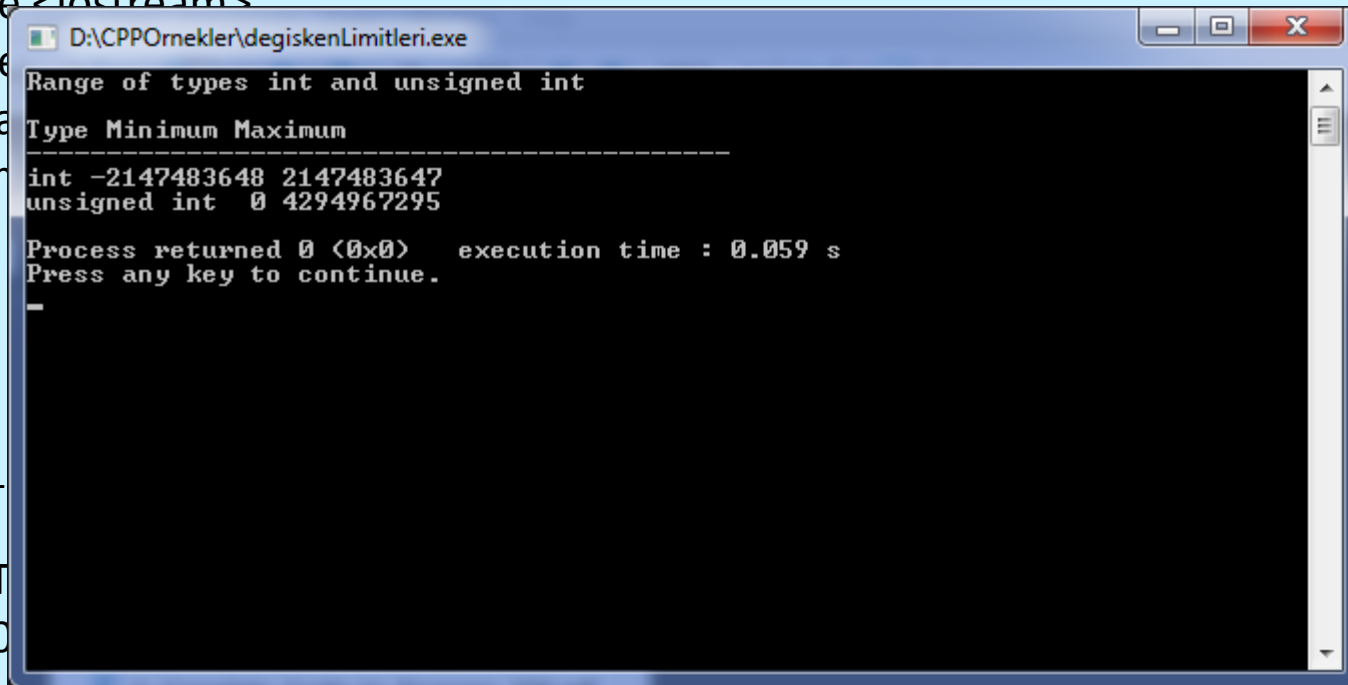
```
<< INT_
```

```
cout <<
```

```
<< UINT
```

```
return 0
```

```
}
```

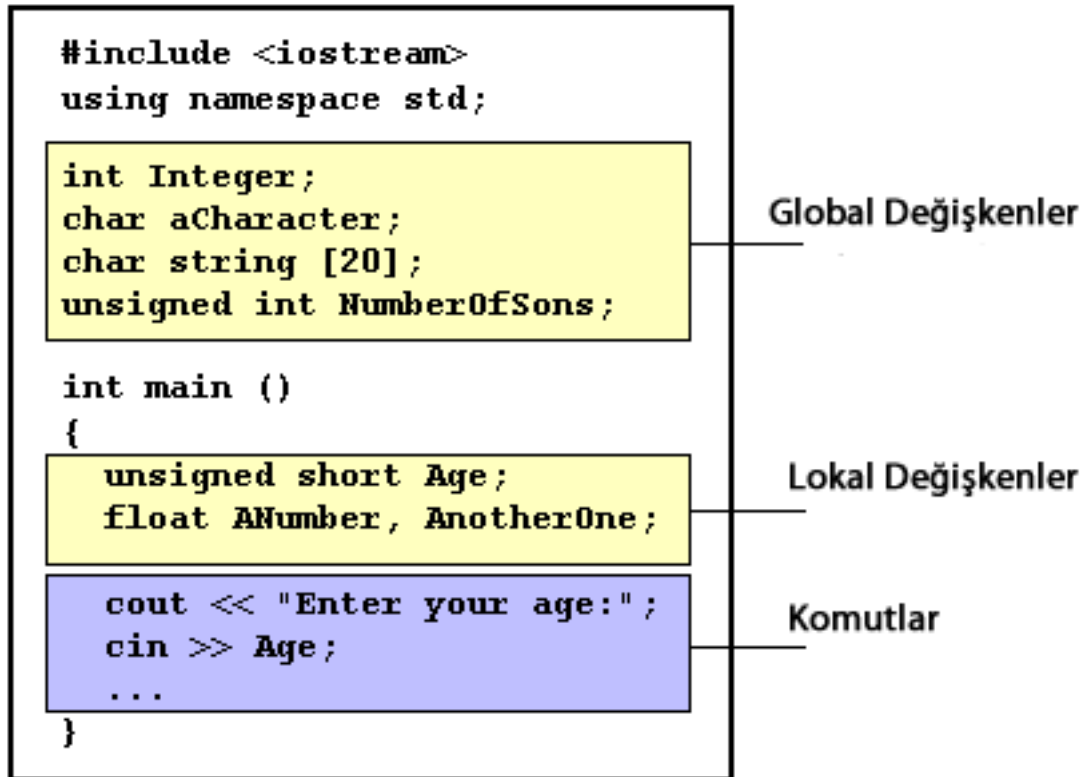


```
D:\CPPOrnekler\degiskenLimitleri.exe
Range of types int and unsigned int
Type Minimum Maximum
-----
int -2147483648 2147483647
unsigned int 0 4294967295
Process returned 0 (0x0) execution time : 0.059 s
Press any key to continue.
-
```

# Değişken Alanları

- Tüm değişkenler kullanıldıkları noktadan önce tanımlanmalıdır
- Bir değişken **global** ya da **lokal** olabilir
  - Global Değişken: Programın ana gövdesinde tanımlanır.
    - Tüm fonksiyonların dışında (main de dahil)
  - Lokal değişken: Bir fonksiyonun gövdesi içinde tanımlanır ve sadece bu fonksiyon içerisinde kullanılabilir.
- Global değişkenler programın herhangi bir yerinde çağırılabilirler.
  - Tanımlanmasından sonra tüm fonksiyonlardan çağırılabilirler
- Lokal değişkenler içinde tanımlandıkları `{}` ile sınırlıdır. Ancak bu aralıkta kullanılabilirler
  - Bu aralık bir kod bloğu da olabilir bir fonksiyon da
    - Bir fonksiyon içinde tanımlanan değişkene diğer fonksiyonlardan ulaşılamaz

# Değişken Alanları



# Değişkenler: Karakterler ve Stringler

- Karakterler tek tırnak içine
  - `char a='z'`
  - `char b='p'`
- Stringler çift tırnak içine
  - `string a= "Hello world"`
  - `string str="How do you do?"`



# Özel Karakterler

- Kod sayfasında bazı özel karakterler kullanılabilir
  - `\n` : Alt satıra geç
  - `\t` : Bir tab kadar boşluk bırak
  - Bu karakterler tırnak içinde yazılsalar da yorumlanarak işlem görürler

```
#include <iostream>  
#include <string>  
using namespace std;  
int main () {  
    string str1 = "Merhaba Dunya\n";  
    String str2 = "Nasilsin";  
    cout << str1<<str2;  
    return 0;  
}
```

## Program Çıktısı

Merhaba Dünya  
Nasilsin

# Özel Karakterler

<code>\n</code>	Yeni Satır
<code>\r</code>	Yeni Satır
<code>\t</code>	Tab
<code>\v</code>	Dikey Tab
<code>\b</code>	Sola Doğru Bir Karakter Sil
<code>\f</code>	Sayfa Sonu
<code>\a</code>	Alarm
<code>\'</code>	Tek Tırnak (')
<code>\"</code>	Çift Tırnak (")
<code>\?</code>	Sioru İşaret (?)
<code>\\</code>	Ters Bölü (\)

# String Değişkenleri

- 1 karakterden daha uzun alfa nümerik değer tutan değişkenlerdir
- Standart string sınıfı ile stringler üzerinde işlemler yapabilmemizi sağlar
  - Deklerasyon diğer değişken türlerinden biraz daha farklıdır.
  - Bu değişken tipindeki değişkenler üzerinde bazı standart işlemleri yapabilmek için <string> kütüphanesi programa eklenmelidir.

```
#include <iostream>  
#include <string>  
using namespace std;  
int main () {  
    string mystring = "Merhaba Dunya";  
    cout << mystring;  
    return 0;  
}
```

## Program Çıktısı

Merhaba Dunya

# String Değişkenleri

- String deklarasyonu farklı şekilde yapılabilir:
  - `string mystring = "This is a string";`
  - `string mystring ("This is a string");`
- Değeri daha sonraki aşamalarda değiştirilebilir

```
#include <iostream>  
#include <string>  
using namespace std;  
int main () {  
    string mystring;  
    mystring = "İlk deger";  
    cout << mystring << endl;  
    mystring = "Son deger";  
    cout << mystring << endl;  
    return 0;  
}
```

## Program Çıktısı

İlk deger  
Son deger

# String Değişkenleri

- String tanımlarken farklı stringleri birleştirebiliriz.
  - string “Merhaba””Dunya”;

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
    string str1 = "Merhaba Dünya\n";
    string str2 = "Nasilsin";
    string str3= "Merhaba" "Bilgisayar"
    "Bolumu ogrencileri\nMerhaba Sizlere";
    cout << str1<<str2<<endl<<str3<<endl<<endl;
    return 0;
}
```

## Program Çıktısı

```
MerhabaBilgisayarBolumu
ogrencileri
Merhaba Sizlere
```

# Sabitler

- Veriler ya nesnelerin içerisinde ya da doğrudan sabit biçiminde bulunurlar.
- Sabitler nesne biçiminde olmayan, programcı tarafından doğrudan girilen verilerdir.
- Sabitlerin sayısal değerleri derleme zamanında tam olarak bilinmektedir.
- Değişkenlerin türleri olduğu gibi sabitlerin de türleri vardır.
- Değişkenlerin türleri daha önce gördüğümüz gibi bildirim yapılırken belirlenir.
- Sabitlerin türlerini ise derleyici, belirli kurallar dahilinde sabitlerin yazılış biçimlerinden tespit eder.

# Sabitler

- Sabitlerin türlerini bilmek zorundayız, çünkü C/C++ dilinde sabitler, değişkenler ve operatörler bir araya getirilerek (kombine edilerek) ifadeler (expressions) oluşturulur.
- C++'ta sabitler 2 türlü tanımlanabilir
  - `#define` yardımı ile
    - `#define PI 3.14159`
    - `#define NEWLINE '\n'`
  - `const` yardımı ile
    - `const int X= 100;`

# Define ile Tanımlanan Sabitler

- İşlemci #define ile başlayan satırlarda yapılan tanımlamalardaki isimlerin program içerisinde geçen kopyalarını o satırda verilen değer ile değiştirir ve program bundan sonra compile edilir.

```
#include <iostream>
using namespace std;
#define PI 3.14159
#define YENISATIR '\n'
int main (){
    double r=5.0;    // yarıcap
    double cember;
    cember = 2 * PI * r;
    cout << cember;
    cout << YENISATIR;
    return 0;
}
```

31.4159



# Deklare Edilen Sabitler

- `const` kullanılarak program içerisinde sabit tanımlanabilir:
  - `const int X= 100;`
  - `const char tab= '\t';`
- Sıradan değişken gibi değer alırlar.
- Program içerisinde değerleri değiştirilemez

# Değişkenlere Değer Atama

- En klasik alanda atama
  - `int a;a=5;`
  - `int a=5;`
- Farklı atamalar:
  - `int a=b=c=5;`
  - `int a,b;b=2 + (a=5);`
    - `int a,b;`
    - `a=5;`
    - `b=2+a;`

# Değişkenlere Değer Atama

```
#include <iostream>
using namespace std;
int main ()
{
    int a=8;
    int b,c;
    b=2+(a=5);
    cout << "a = " << a << endl << endl
         << "b = " << b << endl << endl
         << "c = " << c << endl << endl;
    cout << "-----" << endl << endl;
    a=b=c= -100;
    cout << "a = " << a << endl << endl
         << "b = " << b << endl << endl
         << "c = " << c << endl << endl;
    return 0;
}
```

a = 5

b = 7

c = 2130567168

-----  
a = -100

b = -100

c = -100

# OPERATÖRLER

- Tüm programlama dillerinde olduğu gibi operatörler yapılan işlem türüne göre çeşitlilik gösterirler
  - aritmetik
  - mantıksal
  - karşılaştırma operatörleri

# OPERATÖRLER

- **Aritmetik Operatörler**
  - Dört işlem için kullandığımız operatörler
  - iki sayının bölümünden kalanı veren %

Operatör	İşlem
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Kalan Bulma İşlemi

# Bileşik Atama Operatörleri

İfade	Eşit İfade
<code>a += b;</code>	<code>a = a + b;</code>
<code>a -= b;</code>	<code>a = a - b;</code>
<code>a *= b;</code>	<code>a = a * b;</code>
<code>a /= b;</code>	<code>a = a / b;</code>

İfade	Eşit İfade
<code>a++</code>	<code>a = a + 1;</code>
<code>a--;</code>	<code>a = a - 1;</code>
<code>a=3; b = a++;</code>	İşlem sonunda <code>a=4; b=3;</code>
<code>a=3; b = ++a;</code>	İşlem sonunda <code>a=4; b=4;</code>

# İlişkisel Operatörler

==	Eşittir
!=	Eşit değildir
>	Büyüktür
<	Küçüktür
>=	Büyük ya da eşit
<=	Küçük ya da eşit

# İlişkisel Operatörler

`(7 == 5) // işlem sonucu false`

`(5 > 4) // işlem sonucu true`

`(3 != 2) // işlem sonucu true`

`(6 >= 6) // işlem sonucu true`

`(5 < 5) // işlem sonucu false`



# Mantısal Operatörler

a	b	!a	a && b	a    b
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

```
!(5 == 5) // işlem sonucu false çünkü (5 == 5) doğrudur
```

```
!(6 <= 4) // işlem sonucu true çünkü (6 <= 4) yanlıştır
```

```
!true // işlem sonucu false
```

```
!false // işlem sonucu true.
```

```
((5 == 5) && (3 > 6)) // işlem sonucu false ( true && false ).
```

```
((5 == 5) || (3 > 6)) // işlem sonucu true ( true || false ).
```

# Şartlı Operatör

- $sonuc = \text{şart} ? sonuc1 : sonuc2;$ 
  - Eğer şart doğru ise  $sonuc = sonuc1$
  - Eğer şart yanlış ise  $sonuc = sonuc2$

İşlem	Dönüş Değeri
$7==5 ? 4 : 3$	3
$7==5+2 ? 4 : 3$	4
$5>3 ? a : b$	a
$a>b ? a : b$	a ve b değişkenlerinden büyük olan

# , Operatörü

- İki işlemi tek hamlede yapmaya yarar:
  - $a = (b=3, b+2);$ 
    - Öncelikle  $b=3$  işlemi yapılır
    - daha sonra  $a = b + 2$  işlemi yapılır
    - Sonuç olarak  $b$ 'ye 3  $a$ 'ya 5 değerleri atanmış olur.

# Tip Dönüşümleri ve sizeof

- Bir tipteki değişkeni başka tipe dönüştürmek için
  - `int i;`
  - `float f = 3.14;`
  - `i = (int) f;`
    - ya da
      - `i = int ( f );`
- `sizeof` fonksiyonu
  - bir tipin byte olarak kapladığı yer bilgisini yollar
    - `a = sizeof (char)`
      - *1 cevabını verecektir*

# Output Formatlama

- Kayan Nokta Hassasiyeti
  - Output işlemlerinde kullanılacak digit sayısını belirler
  - Float ve double değişkenlerinin yazımında kullanılır
- **<iomanip>** kütüphanesi kullanılır
- Kullanım şekli: `setprecision(n)`
  - n: noktadan sonra yazdırılacak **maximum** digit sayısı
- **fixed** ile kullanıldığında tam olarak ekrana yazdırılacak digit sayısı belirtilir.

```
// setprecision örneği
```

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main () {
```

```
    double f =3.14159;
```

```
    cout << setprecision (5) << f << endl;
```

```
    cout << setprecision (9) << f << endl;
```

```
    cout << fixed;
```

```
    cout << setprecision (5) << f << endl;
```

```
    cout << setprecision (9) << f << endl;
```

```
    return 0;
```

```
}
```

**OUTPUT:**

→ 3.1416

→ 3.

→ 3.14159

→ 3.141590000

Noktadan sonra tam olarak 9 basamak

```

#include <iostream>
#include <iomanip>
using namespace std;
int main (){
    double a,b,c;
    a = 3.1415926534;
    b = 2006.0;
    c = 1.0e-10;
    cout.precision(5);
    cout<< a << '\t' << b << '\t' << c << endl;
    cout << fixed << a << '\t' << b << '\t' << c << endl;
    cout << scientific << a << '\t' << b << '\t' << c << endl;
    return 0;
}

```

## OUTPUT:

```

3.1416 2006 1e-010
3.14159 2006.00000 0.00000
3.14159e+000 2.00600e+003 1.00000e-010

```

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main (){
```

```
    double a,b,c;
```

```
    a = 3.1415926534;
```

```
    b = 2006.0;
```

```
    c = 1.0e-10;
```

```
    cout.precision(8);
```

```
    cout<< a << '\t' << b << '\t' << c << endl;
```

```
    cout << fixed << a << '\t' << b << '\t' << c << endl;
```

```
    cout << scientific << a << '\t' << b << '\t' << c << endl;
```

```
    return 0;
```

```
}
```

## OUTPUT:

**3.1415927**

**2006**

**1e-010**

**3.14159265**

**2006.00000000**

**0.00000000**

**3.14159265e+000**

**2.00600000e+003**

**1.00000000e-010**



# Doldurma Karakteri

- `setw(n)`
  - output yazdırılırken ne kadar uzunlukta bir bölüme (n) yazdırılacağını belirler.
  - `<iomanip>` kütüphanesi ile kullanılır
  - Eğer yazdırılacak olan bilgi verilen n uzunluğundan kısa ise o bilginin önüne yeterince boşluk karakteri konulur
- Boşluk karakteri yerine başka bir karakter kullanmak için `setfill(c)` kullanılır
  - c karakteri boşluklar yerine yazılır.

```
// setw örneği
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
    cout << setw (10);
    cout << 77 << endl;
    return 0;
}
```

**OUTPUT**

77

```
//Formatlı Yazdırma Ornek
```

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main (){
```

```
    long pop1=12000000,pop2=8000000,pop3=2000000;
```

```
    cout << setw (8)<<"SEHIR"<<setw (12)<<"NUFUS"<<endl;
```

```
    cout << setw (8)<<"ISTANBUL"<<setw (12)<<pop1<<endl;
```

```
    cout << setw (8)<<"ANKARA"<<setw (12)<<pop2<<endl;
```

```
    cout << setw (8)<<"ADANA"<<setw (12)<<pop3<<endl;
```

```
    return 0;
```

```
}
```

## OUTPUT

SEHIR	NUFUS
ISTANBUL	12000000
ANKARA	8000000
ADANA	2000000

```
// setfill örneği
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
    cout << setfill ('x') << setw (10);
    cout << 77 << endl;
    return 0;
}
```

## OUTPUT

```
xxxxxxxx77
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
    int saat,dakika,saniye;
    cout<<"Saati giriniz :";
    cin>>saat;
    cout<<"Dakika giriniz : ";
    cin>>dakika;
    cout<<"saniye giriniz : ";
    cin>>saniye;
    cout<<setfill ('0') << setw (2)<<saat<<":";
    cout<<setfill ('0') << setw (2)<<dakika<<":";
    cout<<setfill ('0') << setw (2)<<saniye<<endl;
    return 0;
}
```

## OUTPUT

```
Saati giriniz :2
Dakika giriniz : 23
saniye giriniz : 2
02:23:02
```

```

#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
    cout << setw(6) << "N" << setw(14) << "square root"
        << setw(15) << "fourth root\n";
    double root;
    for (int n = 10; n <=100; n += 10)
    {
        root = sqrt(double(n));
        cout << setw(6) << setfill('.') << n << setfill(' ')
            << setw(12) << setprecision(3) << root
            << setw(14) << setprecision(4) << sqrt(root)
            << endl;
    }
    return 0;
}

```

N	square root	fourth root
10	3.16	1.778
20	4.47	2.115
30	5.48	2.34
40	6.32	2.515
50	7.07	2.659
60	7.75	2.783
70	8.37	2.893
80	8.94	2.991
90	9.49	3.08
100	10	3.162

# Kararlar

- Program içindeki bir karar deyimin değerine bağlı olarak programın farklı bir bölümüne bir kerelik atlayışa neden olur
- Farklı yapılar mevcuttur:
  - if yapısı
    - En basit karar ifadesidir
    - Koşul sınaama sonucu doğru ise if'i takip eden ifadeler **bir kez** işlenir
    - Yanlış ise bu ifadeler işlenmez, if sonrasında kodlar işlenir
  - if – else yapısı
    - İki alternatif içerisinden seçim yapılır

# Kararlar

- if – else yapısı
  - İki alternatif içerisinde seçim yapılır
  - Sınama sonucu doğru ise ilk ifadeler, yanlış ise else içerisindeki ifadeler işlenir.
- if – elseif – elseif ... -else yapısı
  - İki'den fazla alternatif olması durumunda kullanılır



# Karar Yapıları

- if ifadesi

```
if (Koşul){  
    ifadeler  
}
```

- Eğer sınıma sonucu işlenecek **tek** ifade varsa { } kullanılması gerekli değildir.

```
if (Koşul)  
    ifade
```

# Karar Yapıları

```
if ( x > 100)
```

```
    cout << "x 100den büyük bir değere sahiptir. ";
```

```
if (hiz <= 60){
```

```
    cout << "Hiziniz makul degerler icerisindedir";
```

```
    cout << "Ivmeyi hesaplamak icin gecen sureyi giriniz : ";
```

```
    cin >> sure;
```

```
    cout << ....
```

```
}
```

← Noktalı virgül olmadığına dikkat ediniz.

# switch – case Yapısı

- Switch Case deyimi işlev bakımından if deyimine çok benzemektedir.
- Çok sayıda if işlem blokları kullandığımızda programın okunurluğu azalacak ve programı izlemek zorlaşacaktır.
- Programımızın bir değerini bir çok değerle karşılaştırmak gerektiğinde switch komutunu kullanacağız.
- Switch seçeneği ile değişkenin durumuna göre bir çok durum içersinden bir tanesi gerçekleştirilir.
- Switch in yaptığı iş kısaca, ifadenin değerini sırayla sabitlerle karşılaştırarak ve her satiri işlemektir.

# switch – case Yapısı

- Genel yapısı:

```
switch( Kontrol Degiskeni )  
{  
    case Sabit1 : komut1;  
    case Sabit2 : komut2;  
    .  
    .  
    .  
    default : Komutson;  
}
```

## **AMAÇ:**

Bir koşulun alabileceği bir çok *sabit değer* arasından seçim yaparak gerekli işlemleri yapmaktır.

# switch – case Örnek

```
#include <iostream.h>
main(){
    int i;
    cout<< " 1 ile 4 arasi sir sayi giriniz:";
    cin>>i;
    switch(i) {
    case 1 :
        cout<<"1 Girdiniz";
        break;
    case 2 :
        cout<<"2 Girdiniz";
        break;
    case 3 :
        cout<<"3 Girdiniz";
        break;
    case 4 :
        cout<<"4 Girdiniz";
        break;
    default:
        cout<<"1 ile 4 ten farkli";
    }
    return 0;
}
```

- Program i değişkenine bağlı olarak isliyor.
- Case'lerinin aldığı değere göre kendinden sonra gelen komutları işliyorlar.
- Burada daha önce görmediğimiz **break** komutunu gördük.
  - Programımızda değişkene 1 değerini verdiğimizizi farz edelim. Case 1 adli satiri geçip ondan sonraki komut dizisini işleme soktuk. Bu işlemin tamamlanması için break komutu kullanılıyor.
  - Yazılımda break komutu goto gibi işlev görür ve derleyiciye switch komutundan çıkması için talimat verir.
  - Sorunu ortadan kaldırmak için her durum sonunda break deyimini eklemeliyiz.
- Bir çok karşılaştırma olduğunda switch kullanımını uygun olur..
- Karşılaştırmaların hiç biri olmadığı anda da ortaya default'tan sonraki satirin islenmesi kalıyor.

# switch – case Örnek

```
#include <iostream.h>
main(){
    int i;
    cout<< " 1 ile 4 arasi sir sayi giriniz:";
    cin>>i;
    switch(i) {
    case 1 :
        cout<<"1 Girdiniz";
        break;
    case 2 :
        cout<<"2 Girdiniz";
        break;
    case 3 :
        cout<<"3 Girdiniz";
        break;
    case 4 :
        cout<<"4 Girdiniz";
        break;
    default:
        cout<<"1 ile 4 ten farkli";
    }
    return 0;
}
```

Aynı programın if kullanılarak yazılışı:

```
#include <iostream.h>
main()
{
    int i;
    cout<< " 1 ile 4 arasi sir sayi giriniz:";
    cin>>i;
    if(i==1)
        cout<<"1 Girdiniz";

    else if(i==2)
        cout<<"2 Girdiniz";

    else if(i==3)
        cout<<"3 Girdiniz";

    else if(i==4)
        cout<<"4 Girdiniz";
    else
        cout<<"1 ile 4 ten farkli";
    return 0;
}
```

# switch – case Örnek

```
//ucgen.cpp
// Program girecegimiz ölçülere göre üçgenin Alan, Yükseklik ve Tabanini bulur
// switch-case örneğimiz.
```

```
#include <iostream>
using namespace std;
int main()
{
    char secenek;
    float alan, yukseklik, taban;

    cout << "Program girecegimiz ölçülere göre üçgen'in Alan,"
        << "Yükseklik ve Tabanini bulur!\n" << endl
        << " A ---> Alan : Bulmak için yükseklik ve tabani girecegiz:" << endl
        << " h ---> Yükseklik : Bulmak için alan ve tabani girecegiz:" << endl
        << " t ---> Taban : Bulmak için alan ve yüksekligi girecegiz:" << endl
        << endl << endl;
    cout<< "Seçeneginiz? ---> A, h, t :";
    cin>> secenek;
```

```
switch(secenek)
{
    case 'a':
    case 'A':
    {
        cout<< endl <<endl <<"Yükseklik: ";
        cin>> yukseklik;
        cout<<endl << "Taban: ";
        cin >> taban;
        alan = 0.5 * taban * yukseklik;
        cout<<endl << endl << "Alani: " << alan << endl;
        break;
    }
    case 'h':
    case 'H':
    {
        cout<< endl << endl <<"Alani: ";
        cin>> alan;
        cout<<endl << "Tanabi: ";
        cin >> taban;
        yukseklik = 2.0 * alan / taban;
        cout << endl << endl << "Yükselik: " << yukseklik << endl;
        break;
    }
    case 't':
    case 'T':
    {
        cout << endl <<endl <<"Alani: ";
        cin >> alan;
        cout << endl << "Yüksekligi: ";
        cin >> yukseklik;
        taban = 2.0 * yukseklik / alan;
        cout << endl << endl <<"Tabani: " << taban << endl;
        break;
    }
}
return 0;
}
```

# Döngüler

- Programlama konusunda -hangi dil olursa olsun- en kritik yapılardan biri döngülerdir.
- Döngüler, bir işi, belirlediğiniz sayıda yapan kod blokları olarak düşünülebilir.
- Ekrana 10 kere "Merhaba Dünya" yazan bir programda,
  - "Merhaba Dünya" yazdıran kodu aslında tek bir defa yazarsınız
  - döngü burada devreye girip, sizin için bu kodu istediğiniz sayıda tekrarlar.



# Döngüler

- Bütün döngüler temelde iki aşamayla özetlenebilir.
  - 1. Aşama,
    - döngünün devam edip etmeyeceğine karar verilen mantıksal sorgu kısmıdır.
      - Örneğin, ekrana 10 kere "Merhaba Dünya" yazdıracaksanız, kaçınıcı seferde olduğunu, koşul kısmında kontrol edersiniz.
      - !!!!!Döngünün devam edip etmeyeceğine karar verilen aşamada, hatalı bir mantık sınaması koyarsanız, ya programınız hiç çalışmaz ya da sonsuza kadar çalışabilir.
  - 2. Aşama,
    - döngünün ne yapacağını yazdığınız kısımdır.
      - Yani ekrana "Merhaba Dünya" yazılması döngünün yapacağı iştir.

# Döngüler

- C/C++ programlama diline ait bazı döngüler;
  - while
  - do while
  - for yapılarıdır.
  - Bunlar dışında, goto döngü elemanı olmasına rağmen, kullanılması pek tavsiye edilmemektedir.

# for Döngüsü

- for döngüsü formatı:
  - for(döngü kontrolü)  
{  
    ifadeler; // for döngüsünden yapılması istenenler
  - }
- **döngü kontrolü**
  - for döngüsünün ne kadar devam etmesi gerektiği bu bölümde bildirilir
  - Genel formatı
    - (döngü değişkeni başlangıç değeri; Sınama Koşulu; döngü değişkeni artış miktarı)
    - Ör: `i=1;i>10;i++`

# for Döngüsü

- Aşama 1
  - Döngü değerine başlangıç değeri verilir
    - Bu bölüm sadece 1 kere işlenir
- Aşama 2
  - Koşul kontrol edilir.
    - Eğer koşul doğru ise döngü devam eder
    - Değilse döngü sona erer ve ifadeler işlenmez
- Aşama 3
  - İfadeler işlenir
    - if mekanizmasında olduğu gibi
      - işlenecek tek komut da olabilir
      - {} arasına yazılmış birden fazla ifade de olabilir
- Aşama 4
  - Son olarak döngü kontrolü kısmının son bölümü işlenir (döngü değişkeni artış miktarı)
    - Aşama 2'ye geri dönülür

# for Döngüsü Örnek

- **Örnek:**

```
for(int i=1;i<5;i++)  
    cout<<"Merhaba Dünya"<<endl;
```

- i değişkenine 1 değeri atanır
- Koşul kontrol edilir.
  - 1<5 ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=2)
- Koşul kontrol edilir.
  - 2<5 ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=3)
- Koşul kontrol edilir.
  - 3<5 ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=4)
- Koşul kontrol edilir.
  - 4<5 ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=5)
- Koşul kontrol edilir. 5<5 ?
  - Yanlış olduğu için döngü sona erdirilir

i	ŞART
1	true
2	true
3	true
4	true
5	false

## OUTPUT:

Merhaba Dünya  
Merhaba Dünya  
Merhaba Dünya  
Merhaba Dünya

# Döngü Değişkeni

- Döngü değişkeni 2 şekilde tanımlanabilir
  - döngü öncesinde
    - Değişken değeri içinde tanımlandığı fonksiyon içerisinde herhangi bir yerde kullanılabilir
    - `int i;`
    - `for (i=0;i<3;i++)`
      - `{ifadeler;}`
      - `cout<<i;// Hata vermez`
  - Döngü içerisinde
    - Döngü değişkeni for içerisinde tanımlanmışsa döngü dışında kullanılamaz

# Döngü Değişkeni

```
int i;  
for (i=0;i<3;i++)  
    {ifadeler;}  
cout<<i;// Hata vermez
```

- Döngü içerisinde
  - Döngü değişkeni for içerisinde tanımlanmışsa döngü dışında kullanılamaz

```
for (int i=0;i<3;i++)  
    {ifadeler;}  
cout<<i;// Program hata verir, derlenmez
```

# Döngü Değişkeni

- Birden fazla döngü değişkeni kullanılabilir

```
int main()
{
    int j=9;
    for(int i=0,j=3; i<3&& j>0; i++,j--)
        cout<<i<<'\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

for döngüsü için lokal değişkenler i ve j

## OUTPUT:

```
0 3
1 2
2 1
```

```
9
```

for döngüsü içinde tanımlı lokal değişkenlerin değerleri yazdırılıyor

main fonksiyonda tanımlı j değişkeni kullanılıyor



# Döngü Değişkeni

- Döngünün devamını sağlayan asıl nokta şart bölümüdür.
  - Diğer noktalar yazılmasa da döngü çalışabilir.

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; i++,j--)
        cout<<i<<'\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

**OUTPUT:**

```
1 9
2 8
7
```

!!!!Bu kısmın boş olduğuna dikkat ediniz !!!!!

# Döngü Değişkeni

- Döngünün devamını sağlayan asıl nokta şart bölümüdür.
  - Diğer noktalar yazılmasa da döngü çalışabilir.
  - Ancak değişkenlerin değiştirildiği bölüme dikkat etmeliyiz.

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; )
        cout<<i<<'\\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

Aşağıdaki program ise sorunsuz çalışır:

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; )
        cout<<i<<'\\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

!!!!Böyle bir program sonsuz döngüye girmiştir. !!!!

Durdurulmadığı sürece ekrana **1 9** yazacaktır.

# Döngü Değişkeni

- Aşağıdaki program ise sorunsuz çalışır:

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; ){
        i++;
        cout<<i<<'\\t'<<j<<endl;
    }
    cout<< endl <<i<<endl;
    return 0;
}
```

**OUTPUT:**

```
2    9
3    9
3    9
```

# Döngü Değişkeni

- **!!DİKKAT!!** Şart bildirmezseniz sonsuz döngüye girersiniz..

```
int main()
{
    int j=9;
    int i=1;
    for( i=2,j=5 ; ; i++,j-- )
    {
        cout<<i<<'\t'<<j<<endl;
    }
    return 0;
}
```

**OUTPUT:**

```
2    5
3    4
4    3
5    2
6    1
7    0
8    -1
.
.
.
```

# Örnek

- Klavyeden girilen bir değerin faktoriyelini alan C++ programı

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact=1;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= j;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 9

9 sayisinin faktoriyeli = 362880

# Örnek

- Klavyeden girilen bir değerin faktoriyelini alan C++ programı
  - **Muhtemel hatalar**

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= j;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 5

5 sayisinin faktoriyeli = 322411200

fact değişkeninin ilk değerinin olmadığına dikkat ediniz. Bu durumda

**fact \*= j;**

işleminde ilk değer olarak rastgele bir değer alınacaktır

# Örnek

- Klavyeden girilen bir değerın faktoriyelini alan C++ programı
- **Muhtemel hatalar**

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact=1;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= i;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 5

5 sayisinin faktoriyeli = 3125

Döngü içerisinde

**fact \*= i;**

ifadesine dikkat ediniz. Bu durumda i (Bu örnekte 5 değeri girilmiş) ilk değeri 1 olan fact değişkeninin üzerine 5 defa çarpılacaktır. Diğer bir deyişle  $i^i$  (örnek için  $5^5$ ) işlemi yapılmış oldu.

# while Döngüsü

- while(koşul) ifadesinde "koşul sağlandığı sürece" anlamı vardır.
- while döngüsü, döngü sayısının belli olmadığı zamanlarda kullanılır.
- İşlenecek tek komut varsa:
  - **while ( koşul )**  
Komut;
- While döngüsü, içinde bulunan ifade doğru olduğu sürece altındaki komut veya komut bloğu yürütülür. Eğer yanlış ise kontrol bir sonraki komut veya komut bloğuna geçer.



# while Döngüsü

- While döngüsü daha genel şekliyle:

```
while ( ifade )
```

```
{
```

```
    komut;
```

```
    komut;
```

```
    komut;
```

```
    .
```

```
    .
```

```
    .
```

```
}
```

# while döngüsü örnek

while döngüsü kullanarak klavyeden girilen sayıdan 100 e kadar olan sayıları toplayan C++ programını yazınız

```
#include <iostream>
using namespace std;
main()
{
    int x, y;
    y= 0;
    cout<< "10den küçük bir Sayi Giriniz : ";
    cin>>x;
    while (x< 101)
    {
        y =y+x;
        x =x+1;
    }
    cout<< "Toplam= "<< y;
    return 0;
}
```

Aynı programın for kullanılarak yazılışı:

```
#include <iostream>
using namespace std;
main()
{
    int x, y,x1;
    y= 0;
    cout<< "10den küçük bir Sayi Giriniz : ";
    cin>>x;
    for (int i=x; i<=100; i++)
        y += i;

    cout<< "For ile Toplam= "<< y<<endl;

    return 0;
}
```

# while döngüsü örnek

while döngüsü kullanarak klavyeden girilen sayıdan 100 e kadar olan sayıları toplayan C++ programını yazınız

```
#include <iostream>
using namespace std;
main()
{
    int x, y;
    y= 0;
    cout<< "10den küçük bir Sayı Giriniz : ";
    cin>>x;
    while (x< 101)
    {
        y =y+x;
        x =x+1;
    }
    cout<< "Toplam= "<< y;
    return 0;
}
```

- while döngüsü aşamaları:
  - Koşul kontrol edilir:  $x < 101$  ?
    - Doğru ise
      - sırayla  $y = y + x$ ; ve  $x = x + 1$ ; işlemleri yapılır
      - Koşul kontrolüne geri dönülür
    - Yanlış ise
      - Döngü sona erdirilir
      - `cout<< "Toplam= "<< y`; işlemine geçilir

# while döngüsü

- Döngünün verilen ifade veya koşula göre sağlanması döngülerin en önemli konusudur.
- Eğer bir döngüden çıkılmazsa o döngü sonsuza gider.
  - Buna da "sonsuz döngü" denir.
- Döngüler konusunda en çok rastlayacağımız hata da budur.
- Program siz onu sonlandırıncaya kadar (Ctrl + C ye basarak sona erdirebilirsiniz) devam edecektir!!!!

# Sonsuz Döngü Örnek

```
#include <iostream>
using namespace std;
int main()
{
    int x=1;
    while(x)
        cout<< "x= "<< x++<< endl;
    return 0;
}
```

# Örnek

```
#include <iostream>
using namespace std;
int main()
{
    int x=10;
    while(x)

        cout<< "x= " << x--<< endl;
    return 0;
}
```

## OUTPUT:

```
x= 10
x= 9
x= 8
x= 7
x= 6
x= 5
x= 4
x= 3
x= 2
x= 1
```

Bu program sonsuz döngüye girmedi. Çünkü 0 değeri koşullar için yanlış temsil eder.  $x=0$  değerine ulaşıldığında while (x) döngüsü sona erecektir.

!!!! 0 değeri **yanlış** simgeler, diğer tüm değerler **doğru** anlamına gelir.

# for Döngüsü Örnek

- **Örnek:**

```
for(int i=1;i<5;i++)  
    cout<<"Merhaba Dünya"<<endl;
```

- i değişkenine 1 değeri atanır
- Koşul kontrol edilir.
  - $1 < 5$  ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=2)
- Koşul kontrol edilir.
  - $2 < 5$  ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=3)
- Koşul kontrol edilir.
  - $3 < 5$  ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=4)
- Koşul kontrol edilir.
  - $4 < 5$  ? Doğru olduğu için ifade işlenir
    - Ekrana Merhaba Dünya yazdırılır
- i++ işlemi yapılır (i=5)
- Koşul kontrol edilir.  $5 < 5$  ?
  - Yanlış olduğu için döngü sona erdirilir

i	ŞART
1	true
2	true
3	true
4	true
5	false

## OUTPUT:

Merhaba Dünya  
Merhaba Dünya  
Merhaba Dünya  
Merhaba Dünya

# Döngü Değişkeni

- Döngü değişkeni 2 şekilde tanımlanabilir
  - döngü öncesinde
    - Değişken değeri içinde tanımlandığı fonksiyon içerisinde herhangi bir yerde kullanılabilir
    - `int i;`
    - `for (i=0;i<3;i++)`
      - `{ifadeler;}`
      - `cout<<i;// Hata vermez`
  - Döngü içerisinde
    - Döngü değişkeni for içerisinde tanımlanmışsa döngü dışında kullanılamaz



# Döngü Değişkeni

```
int i;  
for (i=0;i<3;i++)  
    {ifadeler;}  
cout<<i;// Hata vermez
```

- Döngü içerisinde
  - Döngü değişkeni for içerisinde tanımlanmışsa döngü dışında kullanılamaz

```
for (int i=0;i<3;i++)  
    {ifadeler;}  
cout<<i;// Program hata verir, derlenmez
```

# Döngü Değişkeni

- Birden fazla döngü değişkeni kullanılabilir

```
int main()
{
    int j=9;
    for(int i=0,j=3; i<3&& j>0; i++,j--)
        cout<<i<<'\t'<<j<<endl;
    cout<<endl<<j<<endl;
    return 0;
}
```

for döngüsü için lokal değişkenler i ve j

## OUTPUT:

```
0 3
1 2
2 1
```

```
9
```

for döngüsü içinde tanımlı lokal değişkenlerin değerleri yazdırılıyor

main fonksiyonda tanımlı j değişkeni kullanılıyor

# Döngü Değişkeni

- Döngünün devamını sağlayan asıl nokta şart bölümüdür.
  - Diğer noktalar yazılmasa da döngü çalışabilir.

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; i++,j--)
        cout<<i<<'\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

**OUTPUT:**

```
1    9
2    8
7
```

!!!!Bu kısmın boş olduğuna dikkat ediniz !!!!!

# Döngü Değişkeni

- Döngünün devamını sağlayan asıl nokta şart bölümüdür.
  - Diğer noktalar yazılmasa da döngü çalışabilir.
  - Ancak değişkenlerin değiştirildiği bölüme dikkat etmeliyiz.

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; )
        cout<<i<<'\\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

Aşağıdaki program ise sorunsuz çalışır:

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; )
        cout<<i<<'\\t'<<j<<endl;
    cout<< endl <<j<<endl;
    return 0;
}
```

!!!!Böyle bir program sonsuz döngüye girmiştir. !!!!

Durdurulmadığı sürece ekrana **1 9** yazacaktır.

# Döngü Değişkeni

- Aşağıdaki program ise sorunsuz çalışır:

```
int main()
{
    int j=9;
    int i=1;
    for( ; i<3&& j>0; ){
        i++;
        cout<<i<<'\\t'<<j<<endl;
    }
    cout<< endl <<i<<endl;
    return 0;
}
```

**OUTPUT:**

```
2    9
3    9
3    9
```

# Döngü Değişkeni

- **!!DİKKAT!!** Şart bildirmezseniz sonsuz döngüye girersiniz..

```
int main()
{
    int j=9;
    int i=1;
    for( i=2,j=5 ; ; i++,j-- )
    {
        cout<<i<<'\t'<<j<<endl;
    }
    return 0;
}
```

**OUTPUT:**

```
2    5
3    4
4    3
5    2
6    1
7    0
8    -1
.
.
.
```

# Örnek

- Klavyeden girilen bir değerin faktoriyelini alan C++ programı

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact=1;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= j;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 9

9 sayisinin faktoriyeli = 362880

# Örnek

- Klavyeden girilen bir değerin faktoriyelini alan C++ programı
  - **Muhtemel hatalar**

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= j;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 5

5 sayisinin faktoriyeli = 322411200

fact değişkeninin ilk değerinin olmadığına dikkat ediniz. Bu durumda

**fact \*= j;**

işleminde ilk değer olarak rastgele bir değer alınacaktır



# Örnek

- Klavyeden girilen bir değerin faktoriyelini alan C++ programı
- **Muhtemel hatalar**

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    long int fact=1;
    cout<<"Faktoriyeli alınacak sayiyi giriniz: ";
    cin>>i;
    for(int j=1; j<=i; j++)
        fact *= i;
    cout<<endl<<i<<" sayisinin faktoriyeli = "
        <<fact<<endl<<endl;
    return 0;
}
```

## OUTPUT:

Faktoriyeli alınacak sayiyi giriniz: 5

5 sayisinin faktoriyeli = 3125

Döngü içerisinde

**fact \*= i;**

ifadesine dikkat ediniz. Bu durumda i (Bu örnekte 5 değeri girilmiş) ilk değeri 1 olan fact değişkeninin üzerine 5 defa çarpılacaktır. Diğer bir deyişle  $i^i$  (örnek için  $5^5$ ) işlemi yapılmış oldu.