DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

EEM164 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 1

Dr. Öğr. Üyesi Enes KAYMAZ

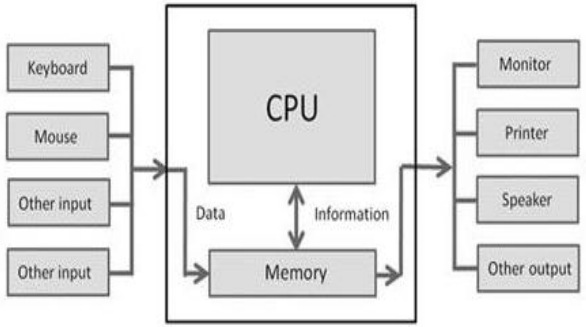
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

Bilgisayar Nedir?

- Bilgisayar en basit olarak üç ana görevi yerine getiren bir makinedir. Girilen bilgiyi alır (INPUT), işler (PROCESSING) ve bu işlenmiş veriden bir sonuç (OUTPUT) çıkarır. Bilgisayar, sadece donanım olarak çalışmaz. Çünkü yazılım olmadan, donanım ne yapacağını bilemez. Bilgisayar donanımına ne yapacağını söyleyecek bir komutlar dizisi gerekir.
- Hardware (Donanım)- Bilgisayarın Fiziksel Parçaları
  - ✓ Printer, Monitör, Klavye, Anakart vs.
- Software (Yazılım) - Bilgisayar tarafından kullanılan programlar
  - ✓ Word, Excel, C, Matlab, Java, Phyton vs.

Bilgisayar Donanımı

1. **Input Devices (Giriş Cihazları)**
   - Bilgisayara bilgi göndermek için kullanılan araçlardir. (Klavye, Mouse vs.)
2. **Output Devices (Çıkış Cihazları)**
   - Bilgisayardan kullanıcıya bilgi iletimi için kullanılır. (Monitör, printer vs.)
3. **CPU (Central Processing Unit- Merkezi İşletim Ünitesi):**
   - Bilgisayarın beyni olarak bilinir
   - Aritmetik ve mantıksal işlemleri yapan birimdir. (Toplama, çıkarma, karşılaştırma, büyüklük, küçüklük vs.)
   - CPU aynı zamanda birimlerin çalışmasını ve veri akışını kontrol eder.
4. **Memory (RAM)**
   - CPU çalışırken geçici olarak verilerin saklandığı ve kullanıldığı alandır.



Bilgisayar Yazılımı

- Bilgisayarın çalışması için donanıma komutlar veren programlara **yazılım** adı verilir.
- Genel olarak üç kısımda incelenebilir
  - ✓ Sistem Yazılımları (Işletim Sistemi – Windows, Unix, Linux vs.)
  - ✓ Program Geliştirme Yazılımları (Programlama Dilleri – Java, C, Pascal, Phython vs.)
  - ✓ Uygulama Yazılımları (MS Word, Excel, Autocad, vs.)
- Yazılım geliştirme sonucu ortaya çıkan ürüne **program** denir.
- Bir problemin bilgisayar tarafından çözülebilmesi için öncelikle **algoritmasının** oluşturulması gerekmektedir.

Bilgisayar Programı

- Belirli bir işi gerçekleştirmek için gerekli **komutların** belirlenmesi ve uygun biçimde kullanılmasıdır.
  - ✓ **Komut:** Bilgisayara belli iş yaptırmaya yarayan emir sözcüğüdür. Bu emir özcüğü **'break'** gibi basit bir komut da olabilir **'for ( )'** döngüsü gibi daha karmaşık bir yapıda da olabilir.

C Language

- **Programlama Dilleri:**
  - Bir programın oluşturulmasında kullanılan komutlar, tanımlar ve kuralların belirtildiği programlama araçlarıdır. Yaygın Programlama Dillerine örnek olarak; ***C , Phyton, Java, Matlab, Visual Basic*** vb. verilebilir.
  - Windows, Linux gibi işletim sistemleri ya da piyasada yaygın olarak kullanılan elektronik aygıt ve bilgisayar kartlarının sürücüleri çoğunlukla C dili ve türevleri kullanılarak yazılmıştır.

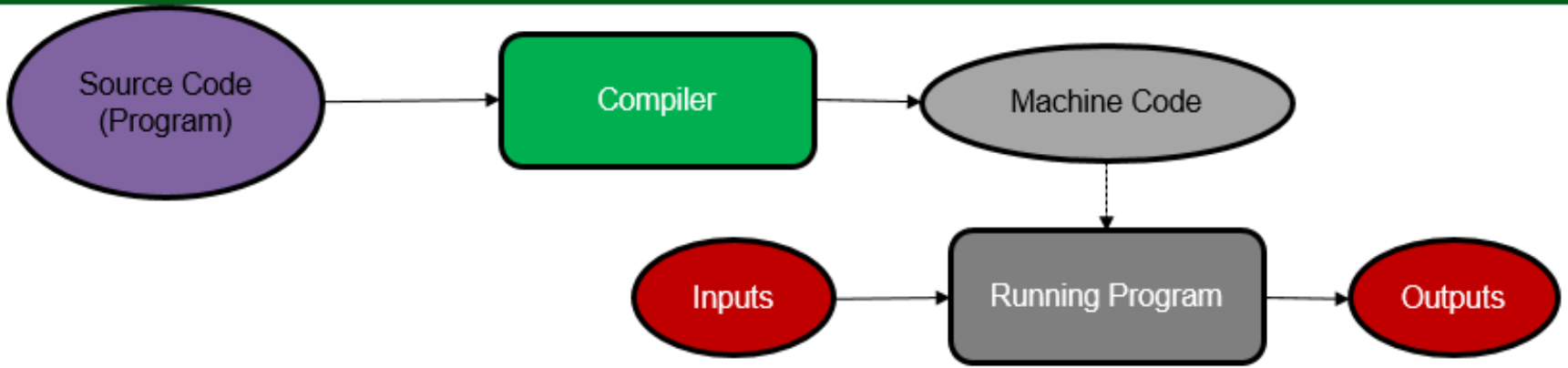
python

Java

MATLAB & SIMULINK

Bilgisayar Programı

- ✓ **Assembly:** Mikroişlemcileri veya mikrodenetleyicileri programlamak için kullanılan programlama dilleridir. Assembly'de program yazmak zor ve zaman alıcıdır fakat donanım seviyesinde işlemler yapılmasına imkan verir. Bu nedenle özellikle robot vb. uygulamalarda sıklıkla tercih edilir.
- ✓ **Makine Dili:** Sadece 0 ve 1'lerden oluşan programlama dilidir. Makine dili ile program yazmak zor ve zaman alıcı olduğundan dolayı fazla tercih edilmez.
- ✓ **Kaynak Kod:** Herhangi bir programalama dili kullanılarak yazılmış metinlere kaynak kod denir. Örneğin;
  - C# ile yazılmış bir programın kaynak kodunun dosya uzantısı .cs
  - Visual Basic ile yazılmış bir programın kaynak kodunun dosya uzantısı .vb
  - Matlab ile yazılmış bir programın kaynak kodunun dosya uzantısı .m şeklinde olur.
- ✓ **Editör:** Kaynak kodları oluşturmak yani kod yazmak için kullanılan yazılımlardır.
- ✓ **Derleyici (Assembler/Compiler):** Belirli bir programlamacı diliyle yazılan kaynak kodların işlemcinin anladığı makine koduna dönüştürülmesini sağlayan yazılımdır.
- ✓ **Amaç (Executable) Program:** Derleme işlemi sonunda ortaya çıkan ve bilgisayar tarafından çalıştırılabilen programdır.




```
#include <stdio.h>
int main()
{
   printf("Hello world!");
   return 0;
}
```

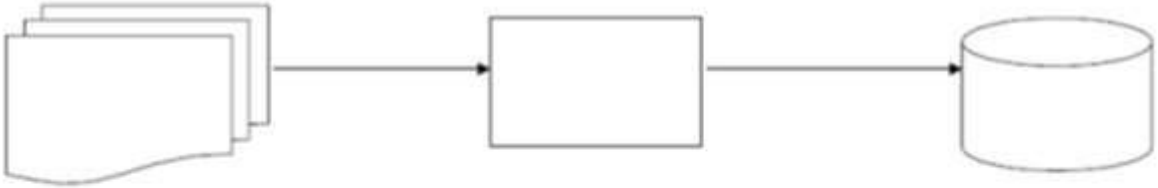
(source code)

Compile & Link

(object code)

```
1110101011001001
0100010101001010
0001001010010101
0101000101001000
100101001
```

(machine code or executable code)



Programlama dili ile yazılan program (kaynak kodu)(**source code**)

Çevirici program

- **Assembler**
- **Compiler**
- **Interpreter**

Makine diline Çevrilmiş kod (**object code**)

CPU tarafından işlenir

Sayı Sistemleri

- Günlük yaşantımızda 10 luk sayı sistemi kullanılır. Ancak, bilgisayar sistemleri 2 lik sayı sistemini kullanılırlar. 10 luk sistemde taban 10, ikilik sistemde taban 2 dir.
- Bilgisayar sistemlerindeki bütün bilgiler ikilik sistemde 1ve 0 ile temsil edilen elektrik sinyalleri ile saklanır.
- İkilik sistemdeki her bir basamağa bit denir.
- Bit nicelik ifade edebilmek için yeterli bir birim değildir. Temel hafıza birimi olarak byte kullanılır. 1 byte = 8 bit
- Bilgisayar sisteminde her bir karakter 8 bit'ten oluşur. Örneğin: A karakteri bilgisayar içinde 0100001 sayısıyla ifade edilir. İşte bu sayının her basamağına 1 Bit denir.

İkilik	Onluk
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
...	...

Sayı Sistemleri

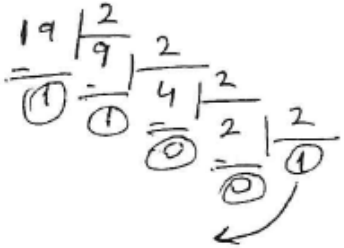
Onlu Sayı Sistemi:

0 – 9 arası rakamlardan oluşur.

$1453 = 1\times10^3 + 4\times10^2 + 5\times10^1 + 3\times10^0$

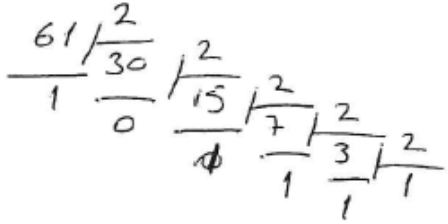
19 sayısının 2'lik karşılığı nedir?

$(10011)_2$



61 sayısının 2'lik karşılığı nedir?

$(111101)_2$



Sayı Sistemleri

İkili Sayı Sistemi:

Elektronik cihazların, bilgisayarların, cep telefonlarının v.b. cihazların kullandığı sayı sistemi 0 ve 1 rakamlarından oluşur. Bu rakamlara BIT (Binary Digit) denir

0 = False = OFF = PASİF = LOW

1 = TRUE = ON = AKTİF = HIGH

$(0101)_2 = (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 4 + 1 = (5)_{10}$

$(10101)_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 16 + 0 + 4 + 0 + 1 = (21)_{10}$

$$123456 = 1 * 10^5 + 2 * 10^4 + 3 * 10^3 + 4 * 10^2 + 5 * 10^1 + 6 * 10^0$$

$$100101 = 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$$

Sayı Sistemleri

- **Bellek Ölçü Birimleri**
  - ✓ 1 Byte = 8 Bit
  - ✓ 1 Kilobyte (KB) = 103 byte= 1.024 byte.
  - ✓ 1 Megabyte (MB) = 106 byte = 1.048.576 byte.
  - ✓ 1 Gigabyte (GB) = 109 byte = 1.073.741.824 byte.
  - ✓ 1 Terabyte (TB) = 1012 byte = 1.099.511.627.776 byte.
  - ✓ 1 Petabyte (PB) = 1015 byte.
  - ✓ 1 Exabyte (EB) = 1018 byte.
  - ✓ 1 Zettabyte (ZB) = 1021 byte.
  - ✓ 1 Yottabyte (YB) = 1024 byte.

Temel Kavramlar ve Matematiksel İşlemler

- **Bilgisayar programları ile gerçekleştirilen işlemler;**
  1) Matematiksel İşlemler
  2) Karşılaştırma (karar) İşlemleri
  3) Mantıksal (lojik) İşlemler
- **Matematiksel İşlemler**
  - Temel aritmetik işlemler
  - Toplama, çıkarma, çarpma, bölme
  - Matematiksel fonksiyonlar
  - Üstel, logaritmik, trigonometrik, hiperbolik) vb

İşlem	Matematik	Bilgisayar
Toplama	a+b	a+b
Çıkarma	a-b	a-b
Çarpma	a.b	a*b
Bölme	a:b	a/b
Üs alma	a^b	a^b

Matematiksel işlemlerin öncelik sırası ?

Sıra	İşlem	Bilgisayar dili
1	Parantezler	((.....))
2	Üs alma	a^b
3	Çarpma ve bölme	a*b ve a/b
4	Toplama ve çıkarma	a+b ve a-b

NOT: Bilgisayar diline kodlanmış bir matematiksel ifadede, aynı önceliğe sahip işlemler mevcut ise bilgisayarın bu işlemleri gerçekleştirme sırası soldan sağa(baştan sona) doğrudur.

Örneğin ; Y=A*B/C
Önce A*B işlemi yapılacak, ardından bulunan sonuç C ye bölünecektir.

Temel Kavramlar ve Matematiksel İşlemler

Matematiksel İşlemler

Matematiksel Yazılım	Bilgisayara Kodlanması
a+b-c+2abc-7	a+b-c+2*a*b*c-7
$a+b^2-c^3$	a+b^2-c^3
$a-\frac{b}{c}+2ac-\frac{2}{a+b}$	a-b/c+2*a*c-2/(a+b)
$\sqrt{a+b}-\frac{2ab}{b^2-4ac}$	(a+b)^(1/2)-2*a*b/(b^2-4*a*c)
$\frac{a^2+b^2}{2ab}$	(a^2+b^2)/(2*a*b)

Karşılaştırma (Karar) İşlemleri

- İki büyüklükten hangisinin büyük veya küçük olduğu,
- İki değişkenin birbirine eşit olup olmadığı gibi konularda karar verebilir.

İşlem sembolü	*Anlamı*
=	*Eşittir*
<>	*Eşit değil*
>	*Büyüktür*
<	*Küçüktür*
>= veya =>	*Büyük eşittir*
<= veya =<	*Küçük eşittir*

Temel Kavramlar ve Matematiksel İşlemler

Mantıksal İşlemler

Mantıksal işlem	Matematiksel sembol	Komut
Ve	.	And
Veya	+	Or
değil	'	Not

"ve,veya,değil " operatörleri hem matematiksel işlemlerde hem de karar ifadelerinde kullanılırlar.

- Bütün şartların sağlatılması isteniyorsa koşullar arasına VE
- Herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA
- Koşulu sağlamayanlar isteniyorsa DEĞİL

  mantıksal operatörü kullanılır.

ALGORİTMA

- Günlük hayatta, matematikte veya bilgisayar ortamında bir problemi çözebilmek için çözüm aşamalarının iyi belirlenmesi gerekir. İyi bir algoritma oluşturmak veya iyi bir kod yazmak için öncelikle aşağıdaki aşamalar takip edilmelidir:
  - Problemin Tanımlanması.
  - Algoritma Oluşturulması. (Akış diyagramı ve sözel kodlarla yazıya dökülmesi önemlidir.)
  - Deneme ve Düzeltme Süreci.

❖ **Algoritmanın Oluşturulması:**

- Bir problemi çözebilmek için gerekli olan sıralı mantıksal adımların tamamına **algoritma** denir. Diğer bir ifadeyle **algoritma**, bir problemin mantıksal çözümünün adım adım nasıl gerçekleştirileceğinin sözlü ifadesidir.

Algoritmanın Özellikleri

- İyi bir algoritmanın temel özellikleri şu şekilde olmalıdır:
- **Kesinlik:** Algoritma içindeki adimlar herkes tarafindan aynı şekilde anlaşılabiliyor olmalı, farklı anlamlara gelebilecek bulanık ifadeler içermemelidir.
- **Sıralı Olma:** Her algoritma için bir başlangiç durumu söz konusudur. Çözüm, bu başlangıç durumu gözönünde bulundurularak gereklestirilir. Adimlarin hangi sırada gerçekleştirileceği çok önemlidir ve net bir şekilde belirtilmelidir.
- **Sonluluk:** Algoritma sonlu sayıda adımdan oluşmali, sınırlı bir zaman diliminde tamamlanmalıdır. Her algoritmanın bir son noktası, bitişi olmalıdır.

Algoritmanın Faydaları

- Programın kodlanmasını kolaylaştırır; algoritmaya bakılarak daha rahat bir şekilde kod yazılabilir.
- Algoritması belli olan programı sadece siz değil, başkaları da yazabilir. Böylece kodlama işi daha hızlı ilerler.
- **Özellikle mantıksal hata yapma ihtimali azalır.** Eğer algoritmanın doğruluğunu daha önce test ettiyseniz programın hatalı sonuç vermesi zorlaşır.
- Algoritmayı hazırlarken, o konu hakkındaki uzmanlığınızı da test etmiş olursunuz. Eğer konu hakkında yeterli bilgiye sahip değilseniz, bilgi eksikliğinizi gidermeli veya uzman kişiden destek almalısınız.

Algoritma Geliştirmek

Şimdi algoritmanın tanımını ve özelliklerini günlük yaşamdan basit bir örnekle pekiştirelim. Diyelim ki araç trafiği olan bir yolda karşıya geçmek istiyoruz. Bu durumda çözmemiz gereken problem (buna yapılması gereken iş de diyebiliriz) karşıya geçmektir. O zaman bu problemin çözümü için bir yol bulmamız gerekiyor.

Şöyle bir ifadeye ne dersiniz?

1) Önce araba var mı kontrol et, ardından yürü!
2) Bu ifade özünde doğrudur. Ancak yeterince açık değildir. Bunu hayatında ilk defa karşıdan karşıya geçecek birine söylersek, kim bilir nasıl anlar?
3) Yolun kenarına park etmiş araba var mı? Evet var. O zaman kaldırımdan yürüyeyim.
4) Yolun kenarına park etmiş araba var mı? Hayır yok. O zaman yolun ortasından yola paralel yürüyeyim.
5) Sol taraftan gelen araba var mı? Hayır yok. O zaman sola bakarak karşıya yürüyeyim.
6) Yolun karşısına geçtin.

Algoritma Geliştirmek

➢ Algoritma ile oluşturulacak çözümler sözel olarak ifade edilir. Örneğin, sabah kalktığımızda kahvaltı yapılacağı zaman kahvaltı hazırlama algoritması oluşturulursa:

- ✓ **Adım 1:** Yataktan kalk.
- ✓ **Adım 2:** Mutfağa git.
- ✓ **Adım 3**: Ekmek al.
- ✓ **Adım 4:** Çayı hazırla.
- ✓ **Adım 5:** Dolaptan kahvaltılıkları çıkar.
- ✓ **Adım 6:** Bardağın bitince çayını doldur.
- ✓ **Adım 7:** Karnın doyunca sofradan kalk.
- ✓ **Adım 8:** Kahvaltılıkları dolaba koy.
- ✓ **Adım 9:** Sofrayı temizle.

Algoritma Geliştirmek

Örnek-1

Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.

Burada iki koşul vardır ve bu iki koşulun da doğru olması gerekir. Yani;

Eğer Yaş>23 VE maaş=asgari ücret ise ismi Yaz

Yaz komutu 1. ve 2.koşulun her ikisi de sağlanıyorsa çalışır.

Örnek-2

Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65'in üzerinde not alanların isimleri istenmektedir.

Burada 3 koşul vardır ve Bilgisayar dersinden 65 in üzerinde not almış olmak temel koşuldur.

Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gerekir.

Eğer Bilg.Not>65 VE (TDili Not>65 veyaYDil Not>65) ise ismiYaz

Algoritma Geliştirmek

- Bir problem çözmek üzere geliştirilen algoritma üç şekilde yazılabilir:
  - Satır algoritma: Problemin çözüm adımları düz metin olarak açık cümlelerle yazılır.
  - Akış diyagramı (flow-chart): Problemin çözüm adımları geometrik şekillerle gösterilir.
  - Sözde kod (pseudo-code): Problemin çözüm adımları komut benzeri anlaşılır metinlerle veya kısaltmalarla ifade edilir.
- **Örnek:** Klavyeden girilen iki sayıyı toplayıp ekranda gösteren programın algoritmasının üç değişik yöntemle gösterilmesi:

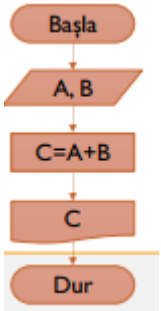
Satır algoritma ile gösterilmesi:

1. Başla
2. Birinci sayıyı (A) klavyeden oku
3. İkinci sayıyı (B) klavyeden oku
4. Girilen sayıları toplayarak sonucu oluştur (C=A+B)
5. Sonucu (C) ekrana yazdır
6. Dur

Sözde kod ile gösterilmesi:

1. Başla
2. A oku
3. B oku
4. C=A+B
5. C yaz
6. Dur

Akış diyagramı ile gösterilmesi:



Algoritma Geliştirmek

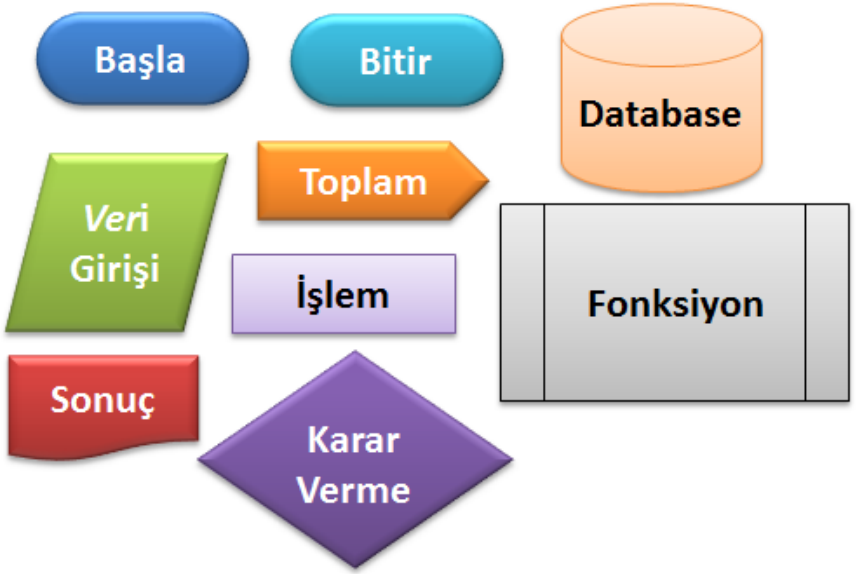
- **Örnek: İki sayıdan büyük olanı bulan programın sözde (pseudo) kodunu yazınız.**

Sözde kod ile gösterilmesi:

1. Başla
2. İlk sayıyı giriniz. (S1)
3. İkinci sayıyı giriniz. (S2)
4. Eğer S1>S2 ise
5. Ekrana ‘S1 büyük’ yaz
6. Akis takdirde ‘S2 büyük’ yaz.

AKIŞ DİYAGRAMLARI

- Çeşitli anlamlar ifade eden ve birbirine oklarla bağlanan şekillerle görsel olarak algoritmanın adımlarını ifade etmektir.
- Akış şemaları Dikdörtgen, Baklava, Elips, Daire gibi özel amaçlı bazı sembollerin çizilmesi ile oluşturulurlar.



Genel Yöntem Algoritması

Genelde basit bir algoritma 3 temel aşamadan oluşur:

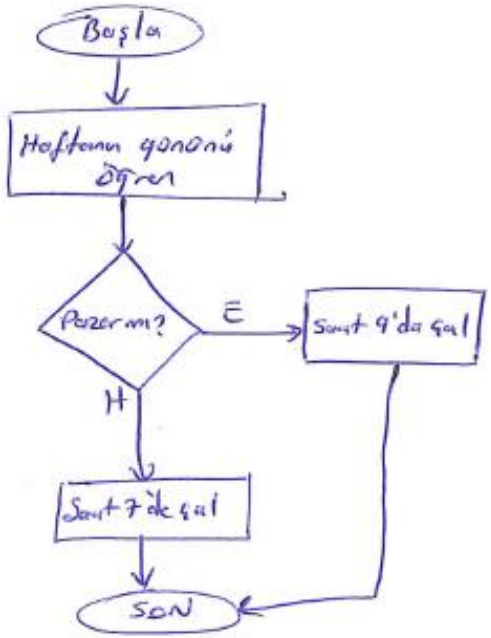
1) Veri girişi: Dışarıdan gerekli verilerin alınması
2) İşlem ve sorgu komutları ile alınan verilerin işlenmesi
3) Veri çıkışı: Elde edilen işlem sonuçlarının ekrana veya kağıda dökülmesi

Günlük Program Algoritması :

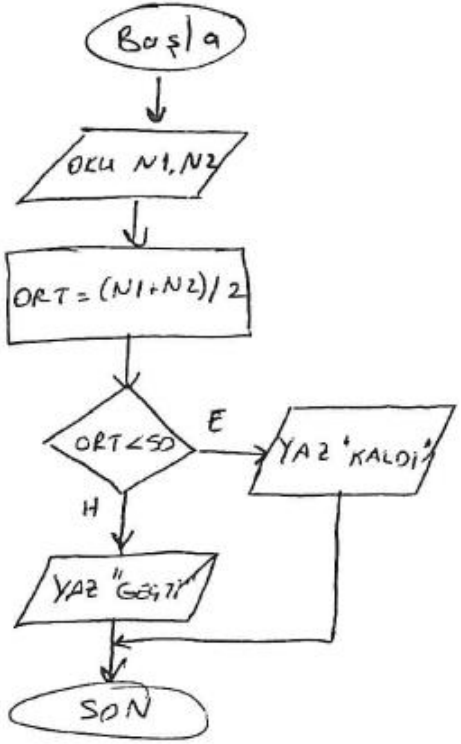
1) Her algoritma "Başla" komutu ile başlar.
2) "6'da uyan" bir iş veya eylem olduğu için dikdörtgen içine yazılmalıdır.
3) Her algoritmanın sonu olmalıdır.

Saat Alarmı Algoritması;

Hafta içi farklı, hafta sonu farklı saatlerde uyandıran bir saat alarmı algoritması. Eğer günlerden pazar ise alarm saat 9'da, diğer günlerde ise saat 7'de çalsın.



Girilen iki ders notunun ortalamasını bulan ve ortalamaya göre öğrencinin geçip geçmeyeceğine karar veren programın akış diyagramı söyledir;



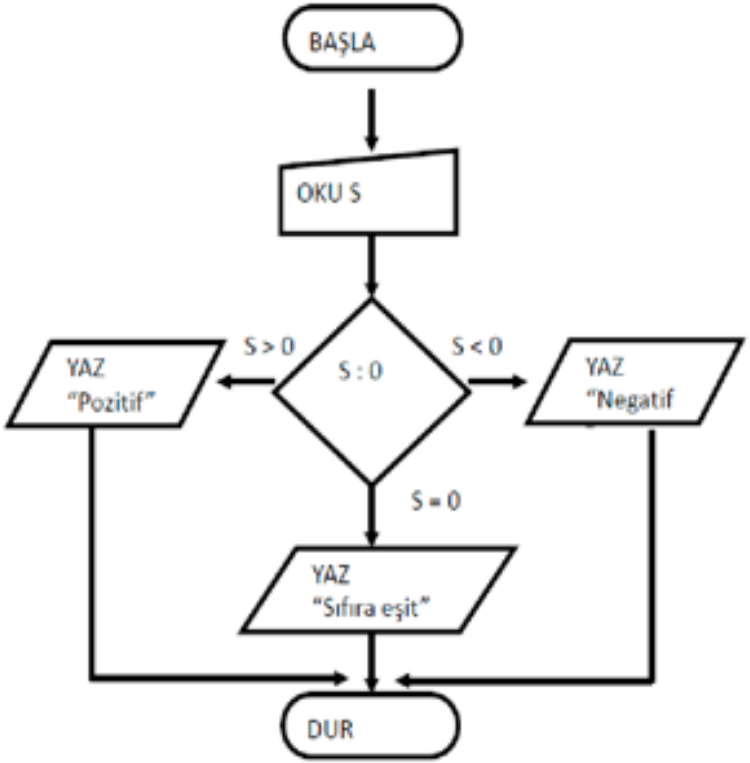
Kod yazarken not defteri gibi basit bir editör de kullanılabilir, visual studio gibi gelişmiş editörde kullanılabilir.

Dev c++ (bazı avantajlardan dolayı) tercih ettik.

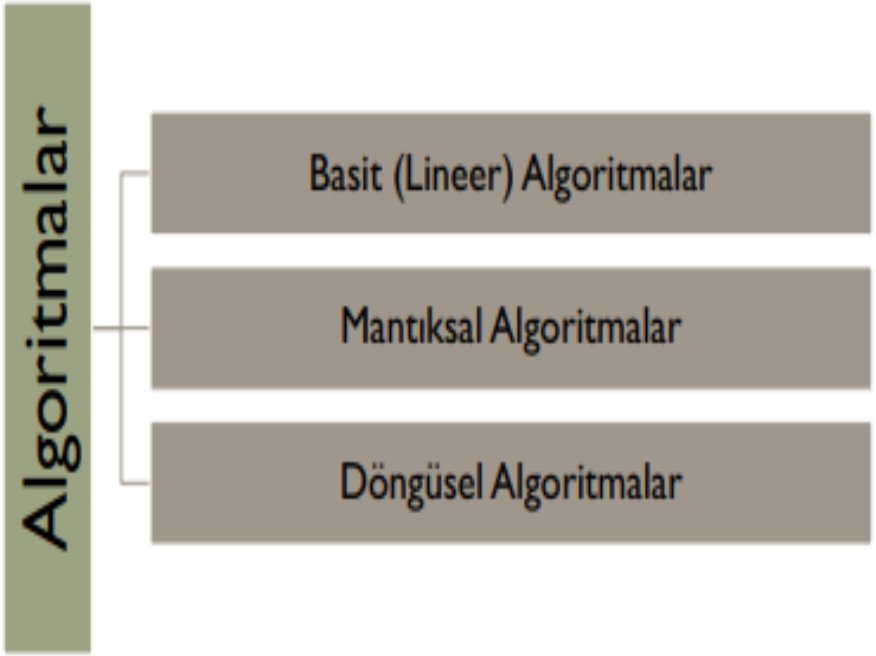
- Klavyeden girilen bir sayının pozitif, negatif veya sıfıra eşit olma durumunu hesaplayıp yazdıran algoritma ve akış şemasını hazırlayalım

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



Algoritmaların Sınıflandırılması



Basit (Lineer) Algoritmalar

- İçerisinde mantıksal ifadelerin yer almadığı, program akış dallanmalarının olmadığı algoritmalardır. Bu algoritmalarda akış düz bir halde baştan sona doğru olacaktır. Çoğunlukla küçük hesaplamaları gerçekleştirmek için kullanılırlar. Bir önceki algoritmaya bakılırsa bir karar yapısının olmadığı görülmektedir.
- **Örnek:**
  - ✓ **Adım 1**: Hesaplanacak kilometre uzunluğunu al; km
  - ✓ **Adım 2:** Girilen değeri 1000 ile çarp; m=km*1000
  - ✓ **Adım 3:** Hesaplanan değeri ekrana yazdır; m

Basit (Lineer) Algoritmalar

- **Örnek:** Dışarıdan girilen üç adet sayısının toplamını, çarpımını ve ortalamasını hesaplayan ve ekrana yazdıran algoritma;
  - ✓ **Adım 1**: Başla
  - ✓ **Adım 2:** Üç adet sayı al; a, b, c
  - ✓ **Adım 3:** Sayıların toplamını hesapla; toplam=a+b+c
  - ✓ **Adım 4:** Sayıların çarpımlarını hesapla; çarpım=a*b*c
  - ✓ **Adım 5**: Sayıların ortalamasını hesapla; ort=toplam/3
  - ✓ **Adım 6:** Sayıların toplamını, çarpımını ve ortalamasını ekrana yazdır; toplam, çarpım, ort.
  - ✓ **Adım 7:** Dur

Mantıksal Algoritmalar

- Algoritma içerisinde mantıksal karşılaştırmaların bulunduğu yapılardır. Mantıksal karşılaştırmalara göre algoritmanın akışı farklı adımlara geçecektir. Bu şekilde oluşturulan algoritmalara Mantıksal Algoritmalar denir.
- İlk oluşturulan algoritma biraz daha ayrıntılanırsa karar yapılarının ortaya çıktığı görülür.

- ✓ **Adım 1:** Başla
- ✓ **Adım 2:** Yataktan kalk
- ✓ **Adım 3:** Mutfağa git
- ✓ **Adım 4:** Eğer ekmek yoksa ekmek al
- ✓ **Adım 5:** Çayı hazırla
- ✓ **Adım 6:** Dolaptan kahvaltılıkları çıkar
- ✓ **Adım 7**: Bardağın bitince çayını doldur
- ✓ **Adım 8:** Karnın doyunca sofradan kalk
- ✓ **Adım 9**: Eğer kahvaltılıklar bitmişse bulaşık makinesine koy
- ✓ **Adım 10**: Eğer kahvaltılıklar bitmemişse kahvaltılıkları dolaba koy
- ✓ **Adım 11:** Sofrayı temizle
- ✓ **Adım 12:** Dur

Döngüsel Algoritmalar

- Program için geliştirilen algoritmada bir işlem birden fazla tekrar ediyorsa döngülü algoritma yapısı kullanılır.
- Döngüsel algoritmalarda mantıksal karşılaştırma yapısı özel olarak kullanılır.
- Eğer algoritma içerisinde kullanılan mantıksal karşılaştırma işlemi sonucunda programın akışı karşılaştırma yapılan yerden daha ileriki bir adıma değil de daha önceki adıma gidiyorsa bu şekilde oluşturulmuş algoritmalara döngüsel algoritma denir.
- Yani döngüsel algoritmalarda mantıksal karşılaştırma sonucunda program daha önceki adımlara gider.

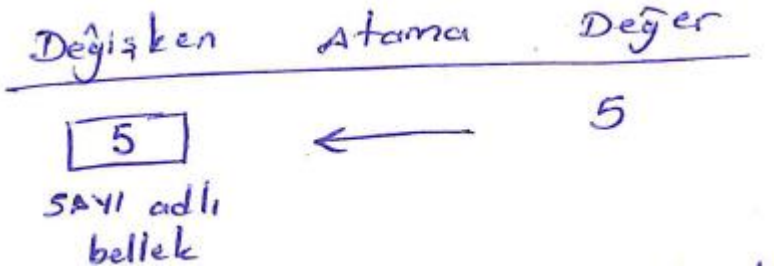
Döngüsel Algoritmalar

Örnek Pasta Yapım Süreci

1. Pastanın yapımı için gerekli malzemeleri hazırla
2. Yağı bir kaba koy
3. Şekeri aynı kaba yağın üzerine koy
4. Yağ ve şekeri çırp
5. Karışımın üzerine yumurtayı kır
6. Tekrar çırp
7. Kıvama geldi mi diye kontrol et
8. a. Kıvamlı ise 9. adıma devam et
   b. Değilse 6. adıma dön.
9. Karışıma un koy
10. Karışıma vanilya, kabartma tozu vb. koy
11. Karışımı Kıvama gelinceye kadar çırp
12. Pastayı Kek kalıbına koy
13. Yeteri kadar ısınan fırına pastayı koy
14. Pişimi diye kontrol et
15. a. Pişmiş ise 16. adıma devam et
    b. Değilse 14. adıma dön
16. Keki fırından çıkart
17. Fırını kapat
18. Kekin ı kapat
19. Kekin soğumasını bekle
20. Keki servis edebilirsin.

❖ DEĞİŞKEN

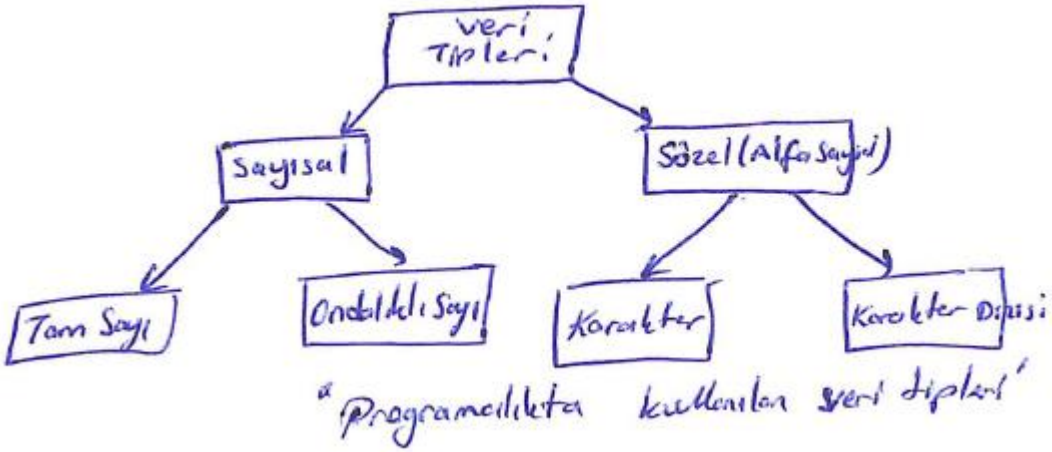
- Dışarıdan alınan veya işlem sonucunda elde edilen verilerin saklandığı bellektir.
- Örneğin bir sayının karesini alacaksınız:
  - ✓ Öncelikle bu sayıyı alıp SAYI veya benzer isimli bir değişkene atamak zorundasınız.



- ✓ 2 + SAYI (2 ile SAYI değişkeninin içindeki değeri topla.
- ✓ 5* SAYI (SAYI değişkeninin içindeki değeri 5 ile çarp.
- ✓ 2*SAYI+1 (SAYI değişkeninin içindeki değeri 2 ile çarp ve 1 ekle.)

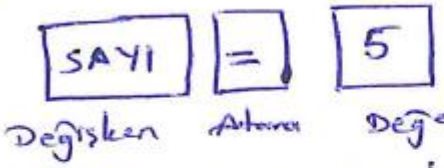
❖ DEĞİŞKEN

➢ Değişkenler farklı veri tipinde olabilir. Örneğin bazı değişkenler; 3, 41, 1453 gibi tam sayı (integer) olurken; bazıları ‘‘bilimsel’’, ‘‘bilgisayar’’ gibi metin (string) tipinde olabilir.



❖ DEĞİŞKEN ATAMA İŞLEMİ

Atama:



(=) ifadesi atama işlemi için kullanılır.

değişken = değişken1 + değişken2 + --- (işlemleri yap ve soldaki değişkene aktar)

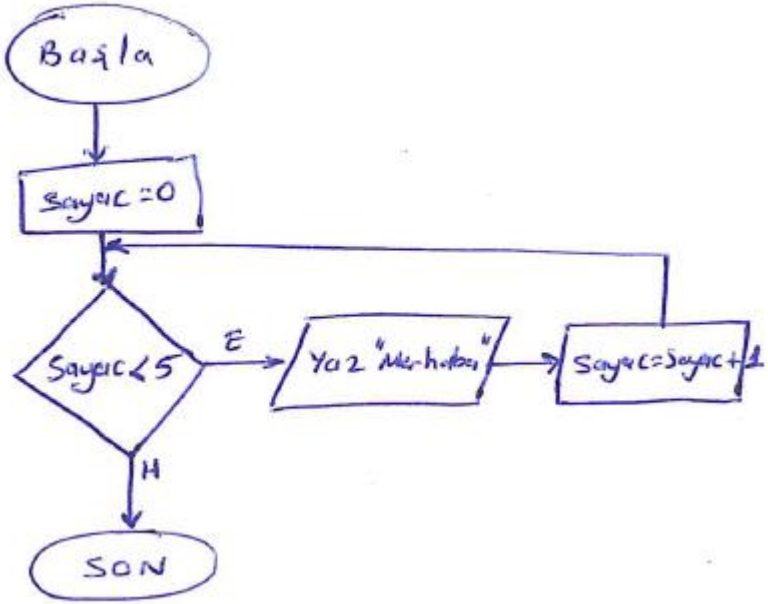
Örnekler:

```
SAYI = 22
SAYI = SAYI + 1
TOPLAM = SAYI + 5
TOPLAM = TOPLAM - 2
```

```
ALAN = PI * R * R
CEVRE = 2 * PI * R
```

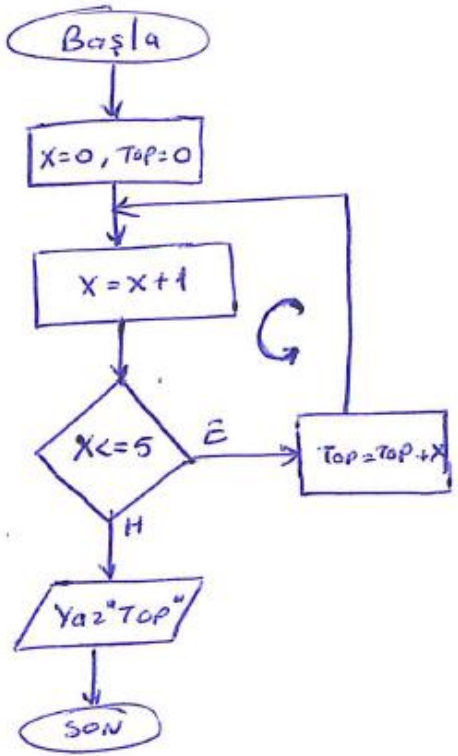
❖ SAYAÇLAR

- Sayaçlar, bir işlemin kaç defa yapıldığını sayan değişkenlerdir. Eğer belirli sayıda bir işlem yapılacaksa veya bir işlemin kaç defa yapıldığını öğrenmek istiyorsanız sayaç kullanmalısınız.
- Örneğin ekrana 5 defa ‘‘ merhaba’’ yazdırılacaksa bir sayaç değişkeni tanımlanır ve sayaç 5 oluncaya kadar ekrana ‘‘ merhaba’’ yazdırılır.



❖ DÖNGÜ

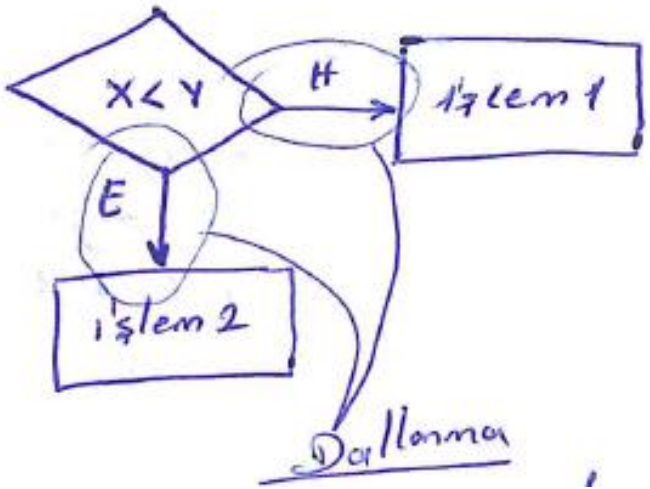
➢ Belirli bir şart sağlandığı sürece veya sağlanana kadar belirli işlemlerin defalarca yapılmasını sağlayan algoritmalardır. Örneğin 1'den 5'e kadar olan sayıların toplamı bulunacaksa burada 5 defa toplama işleminin tekrar edilmesi gerekmektedir.



X	X<=5	TOP=TOP+X
1	evet	1
2	evet	3
3	evet	6
4	evet	10
5	evet	15
6	Hayır	İşlem yok

❖ SORGU İŞLEMİ

- Değişkenlerin durumunun kontrol edilmesi işlemidir.



- Sorgu işlemi sonucunda algoritmanın iki veya daha fazla dala ya da yöne ayrılması durumuna ‘dallanma’ denir .

❖ FONKSİYON

- Kullanıcı tarafından program içinde oluşturabileceği gibi hazır da olabilir.
- Örneğin;
  - ✓ KARE (A) : A değerinin karesini hesaplar.
  - ✓ KOK (A) :A değerinin kökünü hesaplar.

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

EEM164 BİLGİSAYAR PROGRAMLAMA I
Ders Notu 2

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

Değişkenler ve Veri Girişi-Çıkışı

- Açıklama Satırları
- C Dili Yazım ve Noktalama Kuralları
- Değişkenler
- Veri Giriş-Çıkışı
- Değişken Ömrü ve Değişken Kapsamı
- Veri Tipi Dönüştürme
- Uygulamalar

Değişkenler ve Veri Girişi-Çıkışı

❖ **Açıklama Satırları :**

- ✓ " // " tek satırda açıklama veya yorum yapılmasını sağlar.
- ✓ /*…..*/ çoklu satır açıklaması yapılır.

❖ **Noktalı Virgül:** Her komut sonuna noktalı virgül (;) koyulur.

```
int x = 0 , y = 0,  top = 0 ;
x=x+1;
top=top+x;
```

```
printf  (  " yasiniz: "); scanf ( "%d " , & yas); veya
printf  (  " yasiniz: ");
scanf ( "%d " , & yas);
```

Değişkenler ve Veri Girişi-Çıkışı

❖ **Güzel (Süslü) Parantezler {….} ve Blok Kavramı:**

- C dili bloklardan oluşur. Blokların başlangıç ve bitiş yerini de süslü parantezler belirler.
- Örneğin; main ( ) isimli ana programımız **{** işaretiyle başlar ve **}** işareti ile biter.
- **{…….}** işaretleri içinde bulunan komutlara blok denir. Bloklar belirli bir amaç için bir araya getirilmiş olan komutlardan oluşur ve ilgili kodların ayırt edilmesini sağlar. Ayrıca bloklar programın daha rahat okunabilmesine de yardımcı olur.

```
int  main ( )
{ // blok başlangıcı
Komut 1
Komut 2
Komut 3
} // blok bitişi
```

Değişkenler ve Veri Girişi-Çıkışı

❖ **Boşluk Karakteri:**

➢ C dili için boşluk karakterinin bir anlamı yoktur. İstediğiniz kadar boşluk koyabilirsiniz.

```
int  main ( ) { printf ( " merhaba " ); getchar ( ); return 0; }
```

```
#include <stdio.h>
int  main ( )
{
printf ( " merhaba " );
getchar ( );
return 0;
}
```

✓ Yukarıdaki ve yandaki program ekrana " *merhaba* " *yazar ve bekler*. Dev C++ boşlukların düzenli kullanılmasını kendisi halletmeye çalışacaktır.

Değişkenler ve Veri Girişi-Çıkışı

❖ **Boşluk Karakteri:**

➢ C dili büyük ve küçük harfe duyarlı bir dildir. Yani C dili için "ali" ve "Ali " farklı ifadelerdir.

```
int  x=1;
int y=2;
int top =0;
Top=x+y;
printf (Top);
```

✓ Yan tarafta verilen program top ve Top ifadeleri farklı olduğundan dolayı çalışmayacaktır. Bazı editörlerde ***İntelliSense*** adı verilen otomatik kod tanımlama özelliği vardır.

Nesne-Değişken

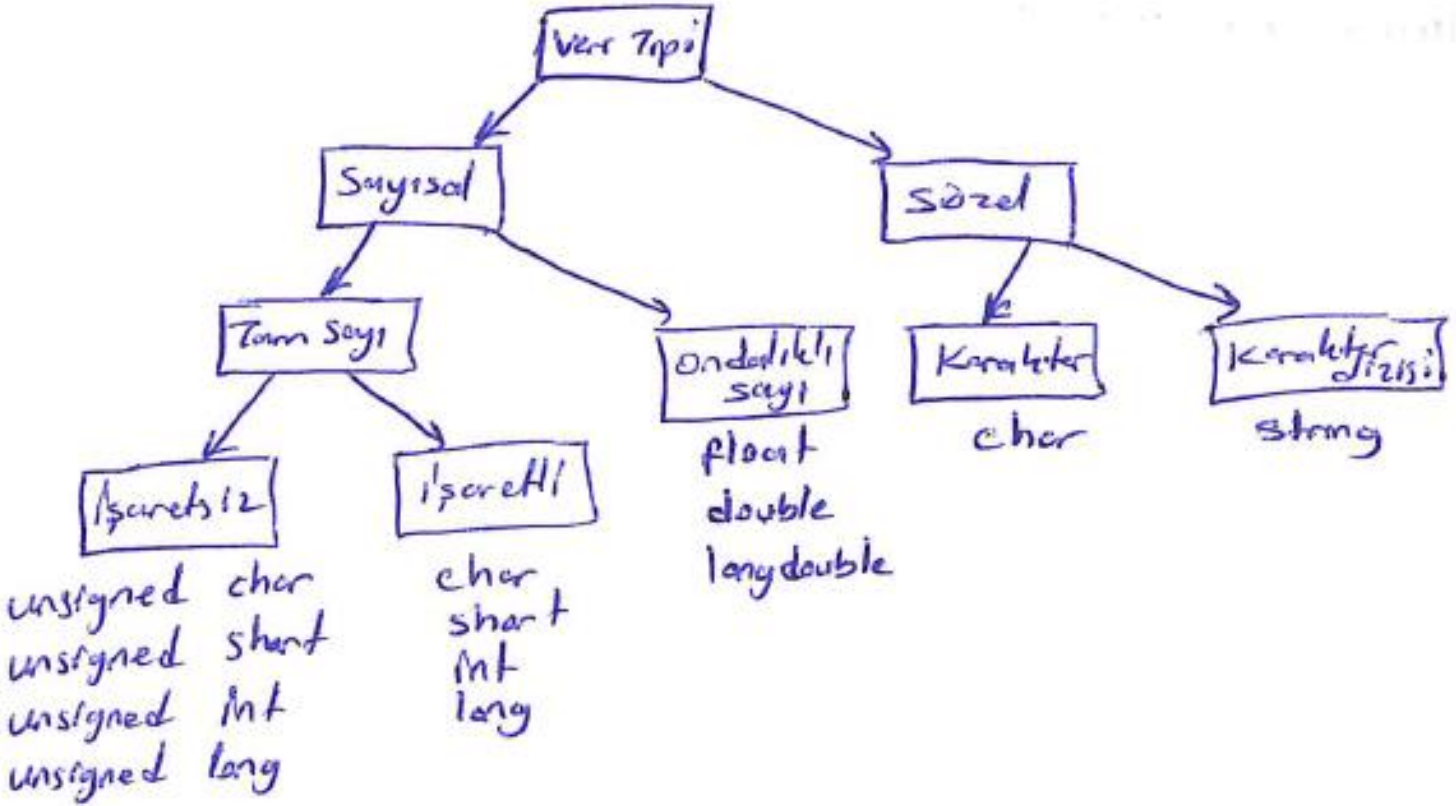
- Değişkenler, dışarıdan alınan veya program içerisinde üretilen değerleri geçici olarak saklayan belleklerdir.
- Bir ifadenin nesne olabilmesi için bellekte yer belirtmesi gerekir.
- a = b + c; d = 100;
- Yukarıdaki ifadelerde a, b, c, d birer nesnedir.
- Nesnelerin özellikleri: İsmi, değeri, türü, faaliyet alanı ve ömrü.
- İsim (name): Nesneyi temsil eden karakterlerdir.
- Değer (value): Nesnelerin tuttuğu bilgidir. İstenildiği zaman değiştirilebilir.
- Tür (type): Nesnenin türü, işleme girdiğinde derleyici tarafından nasıl yorumlanacağını belirleyen özelliktir.
- Programlama dillerinin çoğunda char (karakter), integer (tamsayı) ve float (gerçek sayı) gibi nesne türleri bulunur.

Veri tipleri

- Aşağıdaki tabloda dışarıdan okunabilecek bazı verileri ve değişken veri tiplerini görebiliriz:

Okunan değer	Değişken veri tipi
5	0:255 arası küçük tamsayı (unsigned char)
23500	0:65535 " tamsayı (unsigned short)
-125900	-2 milyar : +2 milyar arası (int)
3,14	ondalıklı sayı (float)
'a'	karakter (char)
"merhaba"	metin (string)

Veri Tipleri



Veri Tipleri

- **Unsigned Char:** RAM bellekte 8 bit yani 1 byte yer kaplar. 0-255 arası değer alır. *Char* kelimesinin önündeki *unsigned* ifadesi *işaretsiz* demektir ve bu durum bu veri tipinin sadece pozitif tam sayıları alabileceği anlamına gelir. Örneğin; 0-100 arası öğrenci notları ile çalışacaksak *short* veya *int* veri tipi yerine bu veri tipi kullanılması daha iyi olur. Çünkü *int* RAM bellekte 4 byte yer kaplarken, *char* veri tipi 1 byte yer kaplar ve bu sayede daha az bellek isteyen ve kaynakları daha iyi kullanan bir program yazmış oluruz.
- **Char:** RAM bellekte 8 bit yani 1 byte yer kaplar. -128 ile 127 arasında değer alır.
- **Unsigned Short:** RAM bellekte 16 bit yani 2 byte yer kaplar. '*ushort*' olarak kullanılır ve u işaretsiz anlamına gelir ve sayının sadece pozitif değerler alabileceğini gösterir. 0 ile 65535 arası değerler alır.
- **Short:** RAM bellekte 16 bit yer kaplar. -32768 ile 32767 arası değerler alır.
- **Unsigned Int:** 32 bit yani 4 byte veri tipidir. Sadece pozitif değerler alır ve 0 ile 4294967295 (0 ile 4 milyar) arası değerler alır.
- **Int :** 32 bitlik veri tipidir. -2147483648 ve 2147483647 arasında değerler alır. (- 2 milyar : + 2milyar)

Veri Tipleri

- **Unsigned Long :** RAM bellekte 64 bit yer kaplar ve çalışılabilecek en büyük tam sayı değerlerini kapsar. (0 ve 18446744073709551615) arasında değerler alır.
- **Long:** 64 bit yer kaplar. En soldaki bit, sayının pozitif veya negatif olma durumunu belirler. (-9223372036854775808 ve 9223372036854775807) arasında değerler alır.
- **Float:** Eğer tam sayılarla değil de ondalık veya kesirli sayılarla çalışılacaksa float veri tipi kullanılabilir. RAM'de 32 bit yer kaplar. (-3.4e +-38 ile +3.4e+- 38)arası değerde çalışır.
- **Double:** RAM bellekte 64 bit yer kaplar. Float veri tipinden daha küçük veya daha büyük sayılarla çalışılmasına olanak sağlar. (-1.7e+-308 ve +1.7e+-308) arasında değer alır.
- **String**: Aslında C dilinde string adında bir veri tipi bulunmamaktadır. Ancak tüm sözel ifadeler bu veri tipi ile saklanır. Saklanan her karakter RAM bellekte 1 byte yer kaplar. C dilinde string değerler " " içerisine alınır.
- **Not:** Genellikle tamsayı değerleri için *'int'*, ondalıklı değerler için *'double'* ve metinler içinde *'char'* veri tipini kullanabiliriz.

Veri tipleri, kapladığı alanlar (size) ve değer aralıkları (range)

ÇIKTI

Windows (32 bit) Turbo C	Windows (32 bit) Salford	Linux (32 bit) GCC	Linux (64 bit) GCC
char : 1 bayt	char : 1 bayt	char : 1 bayt	char : 1 bayt
short : 2 bayt	short : 2 bayt	short : 2 bayt	short : 2 bayt
int : 2 bayt	int : 4 bayt	int : 4 bayt	int : 4 bayt
long : 4 bayt	long : 4 bayt	long : 4 bayt	long : 8 bayt
unsigned char : 1 bayt	unsigned char : 1 bayt	unsigned char : 1 bayt	unsigned char : 1 bayt
unsigned short : 2 bayt	unsigned short : 2 bayt	unsigned short : 2 bayt	unsigned short : 2 bayt
unsigned int : 2 bayt	unsigned int : 4 bayt	unsigned int : 4 bayt	unsigned int : 4 bayt
unsigned long : 4 bayt	unsigned long : 4 bayt	unsigned long : 4 bayt	unsigned long : 8 bayt
float : 4 bayt	float : 4 bayt	float : 4 bayt	float : 4 bayt
double : 8 bayt	double : 8 bayt	double : 8 bayt	double : 8 bayt
long double : 10 bayt	long double : 10 bayt	long double : 12 bayt	long double : 16 bayt

Veri Tipi	Açıklama	Bellekte işgal ettiği boyut (bayt)	Alt sınır	Üst sınır
char	Tek bir karakter veya küçük tamsayı için	1	-128	127
unsigned char			0	255
short int	Kısa tamsayı için	2	-32,768	32,767
unsigned short int			0	65,535
int	Tamsayı için	4	-2,147,483,648	2,147,483,647
unsigned int			0	4,294,967,295
long int	Uzun tamsayı için	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long int			0	18,446,744,073,709,551,615
float	Tek duyarlı gerçel sayı için (7 basamak)	4	-3.4e +/- 38	+3.4e +/- 38
double	Çift duyarlı gerçel sayı için (15 basamak)	8	-1.7e +/- 308	+1.7e +/- 308

Değişken isimlendirme

- Programda kullanılacak değişken isimleri programcı tarafından tanımlanır.
- Değişkenlere isim verirken aşağıdaki kurallara uyulmalıdır.
- C dilinin kendine özgü anahtar sözcükleri, komut veya fonksiyon adları değişken ismi olarak kullanılamaz.
- Değişken isimleri içerisinde, a-z ve A-Z arası İngilizce harfleri, 0-9 arası rakamlar ve özel karakter olarak sadece alt çizgi (_) karakteri kullanılabilir.
- Özel karakterler (+,-,! vs.) ve Türkçe karakterler kullanılmaz.
- Maaş, öğrenci, sınıf -> bunlar değişken adı olarak kullanılamaz.
- Değişken ismi rakam ile başlayamaz.
- 1. vize (yanlış) vize1 (doğru)

Değişken Oluşturma ve Değer Atama

❖ **Atama Operatörü**

✓ C dilinde değişken oluşturma ve değer atama aşağıdaki gibidir:

✓ (Değişken Tipi) Değişken Adı = İlk Değer

✓ Bir değeri bir değişkene atar ve C programlamada ”=“ ile gösterilir.

```
int  a=5;
char harf= 'a ';
double pi=3.14
```

✓ İstenirse, önce değişken oluşturulup sonrasında değer atanabilir.

```
int  b;
b = 20;
char karakter;
karakter= "e";
double pi
pi=3.14
```

Değişken Oluşturma ve Değer Atama

❖ **Atama Operatörü**

Toplam ve Atama:
Toplam = toplam + sayi;

Çarpma ve Atama:
İşlem= işlem sayi;*

Ör: 5!
*5*4*3*2*1=120*

```
int f=1;
int sayi=1;
sayi=sayi+1;
f=f*sayi;
```

Değişken Oluşturma ve Değer Atama

❖ **Tam Sayı Değişkenler**

- ✓ Eğer 2, 547, 9150, -33 gibi tam sayılarla işlem yapılacaksa,boyutuna ve işaretine göre unsigned char, char, unsigned short, short, unsigned int, int , unsigned long veya long gibi tam sayı değişkenleri kullanılması gerekir.
- ✓ Örneğin; 0:100 arası pozitif sayılarla çalışacaksanız ″unsigned char‟ veri tipi kullanılabilir. Bu veri tipi 0:255 arası değerleri saklayabilir ve RAM bellekte sadece 1 byte yer kaplar.

Değişken Oluşturma ve Değer Atama

❖ **Tam Sayı Değişkenler**

❑ **Örnek:**

```
#include <stdio.h>
 int main ( )
{ // Değişkenleri tanımlayalım.
unsigned char s1=1, s2=2;
unsigned top=0;
// s1 ve s2 değişkenlerini toplayalım
top=s1+s2;
// toplamı yazdıralım
printf (" %d ", top);
return 0;
}
```

➢ Aslında 0:100 arası değerleri char ve n tipi de saklayabilir. Yani "unsigned char" yerine "char " veri tipi de kullanılabilir. "short " veya "int " veri tiplerinde de saklayabilirsiniz. Ama "short " veri tipi 2 Byte, "int " veri tipi ise 4 Byte yer kaplar. Bu durum hafızada daha fazla yer kaplanmış olması nedeniyle özellikle büyük programların yavaş çalışmasına yol açar. Eğer -100:+100 gibi tam sayılarla çalışacaksanız "char " veri tipi en mantıklısıdır. Bu veri tipi -128:+127 arası değerleri saklayabilir.

Değişken Oluşturma ve Değer Atama

❖ **Tam Sayı Değişkenler**

➢ Eğer bir işlem sonucunda ondalıklı (kesirli) bir değer ortaya çıkarsa, virgülden sonraki değeri atarlar.

```
#include <stdio.h>
 int main ( )
{
char n1=1, n2=2;
char ort;
ort = (n1+n2) / 2;
printf (" %d ", ort);
return 0;
}
```

✓ (n1+n2) / 2; yani (1+2)/2 = 1,5 ondalık sayıdır. Fakat "ort " değişkeninin veri tipi tam sayı olduğu için virgülden sonraki rakamlar atılır ve ort=1 olur.

Değişken Oluşturma ve Değer Atama

❖ **Ondalıklı Değişkenler**

➢ Tam sayı olmayan, 1.25, 13.4141 gibi sayılara ondalıklı veya kesirli sayı denir. Bu tür sayıları saklayabilmek için ″float ‟ veya ″double‟ veri tipi kullanılır. ″float ‟ veri tipi virgülden sonra 6 karakter ″double ‟ karakteri virgülden sonra 8 karakter saklayabilir.

✓ float pi =3.14;
✓ double r=24.434365;
✓ double z=1.25 e3; // 1.25*10^3
✓ float x=234 e-2; // 234 * 10^-2

Değişken Oluşturma ve Değer Atama

```
#include <stdio.h>
 int main ( )
{
char n1=1, n2=2;
double ort
ort = (n1+n2) / 2;
printf (" %f ", ort);
return 0;
}
```

```
#include <stdio.h>
 int main ( )
{
double n1=1, n2=2;
double ort;
double ort
ort = (n1+n2) / 2;
printf (" %f ", ort);
return 0;
}
```

- (n1+n2)/2 yani (tam sayı 1 + tam sayı 2) /2 ifadesinin sonucu da tam sayı olur. Elde edilen 1 değeri eşitliğin solundaki double veri tipine aktarılır. Tüm değişkenlerin **double veya float** olması gerekmektedir. Ayrıca **%f** yerine **%.2f** yazılırsa virgülden sonra iki karakter yazdırılır.

Değişken Oluşturma ve Değer Atama

❖ **Sözel Değişkenler**

➢ C dilinde sözel verileri saklayabilmek için ″char‟ veri tipi kullanılır. ″char‟ veri tipi sadece tek bir karakter saklayabilir. ″char‟ veri tipindeki değişkenlere değerler tek tırnak ('……') içinde atanmalıdır.

```
char harf= 'a'
char cevap;
cevap= 'e'
```

❖ **Sabitler**

➢ Program içinde değeri hiç değişmeyen değişkenlere sabit denilir. Sabit tanımlamak için ″define‟ komutu kullanılır.

```
#define değişken değer
#include <stdio.h>
#define pi 3.14
int main( )
{
double r, alan=0, cevre=0;
double pi=3.14; // yazılmayabilir.
alan=pi*r*r
cevre=2*pi*r
return 0;
}
```

Veri Çıkışı ve Metin Yazdırma

➢ Program içinde üretilen bir değeri konsola yazdırmak için " printf " kullanılır.

- ✓ \ n : Alt satıra geç
- ✓ \ r : Satır başına git
- ✓ \ t : Tab (sekme)
- ✓ \ ' : Tek tırnak
- ✓ \ ' ' : Çift tırnak
- ✓ \\ : Ters bölü
- ✓ \? : Soru işareti

```
printf (" Merhaba ") ;
printf (" Dünya " )
```

```
Merhaba Dünya
```

```
printf (" Merhaba \ n ")
printf (" Dünya " )
```

```
Merhaba
Dünya
```

Veri Çıkışı ve Metin Yazdırma

- Eğer verileri ve değişkenleri elirli bir biçimde yazdırmak isterseniz " printf " komutunun kalıbı şöyle olur;
- printf (" biçim ifadesi " , değişkenler);
- Veri tipine göre; *% d, % f, % c* gibi ifadeler kullanılır.

Tip karakteri	Tip belirleyici anlamı	Yazdırılacak veri tipi
%c	tek bir karakter	char
%s	karakter dizisi (string)	char
%d	işaretli ondalık tamsayı	int, short
%ld	uzun işaretli ondalık tamsayı	long
%u	işaretsiz ondalık tamsayı	unsigned int, unsigned short
%lu	işaretsiz uzun tamsayı	unsigned long
%f	gerçel sayı	float
%lf	çift duyarlı gerçel sayı	double

Veri Çıkışı ve Metin Yazdırma

Tip karakteri	Tip belirleyici anlamı	Yazdırılacak veri tipi
%c	tek bir karakter	char
%s	karakter dizisi (string)	char
%d	işaretli ondalık tamsayı	int, short
%ld	uzun işaretli ondalık tamsayı	long
%u	işaretsiz ondalık tamsayı	unsigned int, unsigned short
%lu	işaretsiz uzun tamsayı	unsigned long
%f	gerçel sayı	float
%lf	çift duyarlı gerçel sayı	double

```
#include <stdio.h>
int main ( )
{
float s1=1.55, s2=10.200, top=0;
top=s1+s2;
printf ( " sayilarin toplami = %f '‘,top;
return 0;
}
```

Ekran çıktısı:

Sayilarin toplami=10.355

C dilinin bazı Anahtar Sözcükleri

ANSI C (C89)/ISO C (C90) keywords:

• auto	• double	• int	• struct
• break	• else	• long	• switch
• case	• enum	• register	• typedef
• char	• extern	• return	• union
• const	• float	• short	• unsigned
• continue	• for	• signed	• void
• default	• goto	• sizeof	• volatile
• do	• if	• static	• while

Kütüphaneler

- Her standart kütüphane, o kütüphanedeki her fonksiyonun prototiplerinin yer aldığı ve bu fonksiyonlar tarafından kullanılabilecek çeşitli veri tipleriyle, bazı sabitlerin bulunduğu bir öncü dosyaya sahiptir.
- **assert.h**
- Programın hatalarının ayıklanmasında yardımcı olması için eklenen teşhislerin makroları ve bilgilerini içerir.
- **math.h**
- Matematik kütüphane fonksiyonlarının prototiplerini tutar.
- **setjmp.h**
- Fonksiyon çağrıları ve geri dönüşleri arasındaki geçişlere izin veren fonksiyonların prototiplerini tutar.
- **signal.h**
- Programın çalıştırılması esnasında oluşabilecek çeşitli durumları gerçekleştiren makroları ve fonksiyon prototiplerini tutar.
- **stdarg.h**
- Sayısı ve tipleri belli olmayan argüman listesine sahip fonksiyonların çalışmasını sağlayan makroları tutar.
- **stdio.h**
- Standart giriş/çıkış kütüphane fonksiyonlarının prototiplerini ve bunlar tarafından kullanılan bilgileri tutar
- **stdlib.h**
- Sayıları yazılara, yazıları sayılara çeviren, rastgele sayılar üreten, hafıza ayrılmasını sağlayan fonksiyonların prototiplerini tutar
- **string.h**
- String işlemlerini yapan fonksiyonların prototiplerini tutar.
- **time.h**
- Zamanla ve tarihle ilgili işlemler yapan fonksiyonların prototiplerini tutar.

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

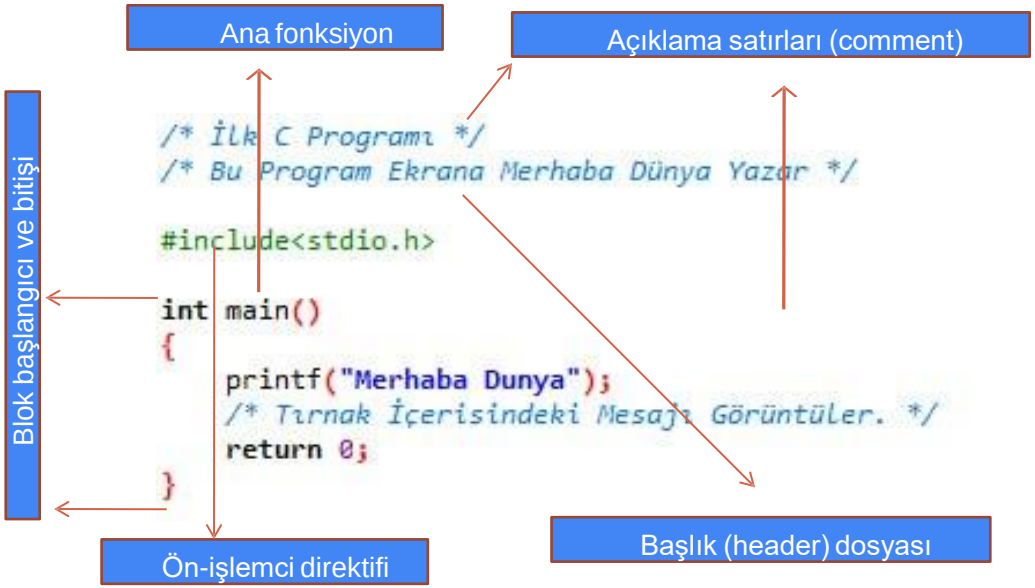
EEM165 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 3

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

BASİT BİR C PROGRAMI



C Dilinin Genel Yazım Kuralları

- C dili birden fazla satırdan oluşan açıklama satırlarına izin verir.
- Bu açıklama satırları programın ne iş yaptığı hakkında bilgi verir.
- Açıklama satırları /* ile başlayıp */ ile sona erer.
- Derleyici bu satırları çalışma anında dikkate almaz.
- C de her bir işletilebilir ifade (komut satırı) ; ile sonlandırılır.
- Bütün anahtar kelimeler ve komutlar küçük harfle yazılır (#define hariç).
- C dili büyük-küçük harf duyarlıdır.
- Yani; “TOPLAM”, “toplam” ve “tOpLaM” kelimelerinin hepsi C derleyicisi tarafından ayrı ayrı algılanır.

Ön-işlemci Direktifleri

- Ön-işlemci direktifleri # işareti ile başlar ve program derlenmeden önce C ön-işlemcisi tarafından işletilir.
- Her bir ön-işlemci direktifinin farklı bir görevi vardır.
- #include ve #define en çok kullanılan direktiflerdir.
- #include direktifi program içerisinde kullanılan fonksiyonlar için gerekli kodları programa dahil etmek için kullanılır.
- I/O fonksiyonları standart input/output *C* Kütüphanesinde tanımlanmış
  - ✓ stdio.h
- stdio.h programın başına eklemeniz gerekiyor.
- Bu eklemeyi #include önişlemci komutuyla yapmanız gerekiyor.
  - ✓ #include <studio.h>
- Önişlemcikomutları # ile başlar.
  - ✓ #define

Main Fonksiyonu

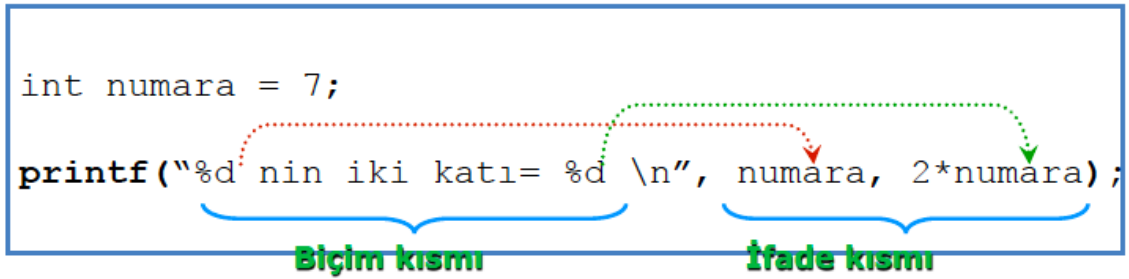
- Hemen hemen bütün C programları birden fazla fonksiyondan oluşur.
- Main ( ) bütün C programlarında bulunması gereken programın ana fonksiyonu yani gövdesidir.
- İlk çalıştırılacak olan fonksiyondur.
- Programda çalıştırılacak ifadeler (kod satırları) { - } küme parantezleri içinde yazılırlar.
- Her parantez çiftinin oluşturduğu yapılara kod blokları denir.
- Bir kod bloğu içerisinde program içerisinde kullanılacak değişkenler ve gerçekleştirilecek işlemleri yerine getirecek komutlar bulunur.

Ön-işlemci Direktifleri

- Örneğin tasarladığımız programda ekrana çıktı yazdırmak istiyoruz.
- Bunun için *C* dilinin standart bir fonksiyonu olan *printf* fonksiyonunu kullanmamız gerekir.
- *printf* (“Örnek Çıktı”);
- Ancak *printf* fonksiyonunun çalışabilmesi için <stdio.h> isimli dosyaya ihtiyacımız olacaktır.
- Bu dosyayı programa dahil etmek için program kodunun en tepesine
- #include <stdio.h> komut satırı yazılır.
- C dilinde .h uzantılı dosyalara başlık dosyası (Header File) adı verilir.
- stdio.h başlık dosyası standart giriş çıkış işlemleri için gerekli kodları içerir.

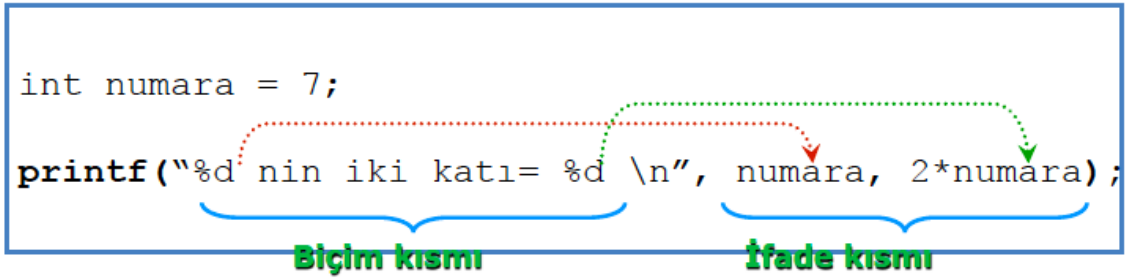
printf ( ) fonksiyonu

- Değişkenlerin değerlerini, hesaplanan sonuçları ya da mesajlar ekranda göstermek için kullanılır.
- *printf ( )* fonksiyonu, fonksiyon ismi ve parantezler içindeki parametreler olmak üzere iki kısımdan oluşur.
- *printf ( )* fonksiyonu, parametre olarak görüntülenecek bilginin hangi biçimde görüntüleneceğini bildiren çıktı metin formatını ve bu formatın içinde yazdırılacak olan değişkenler listesini alır.



printf ( ) fonksiyonu

- *printf* çıktı formatındaki % karakterinin dışındaki tüm karakterleri ekrana yazar.
- % karakterini gördüğünde bunun sağındaki karakteri yazdırılacak değişkenin format karakteri olarak ele alır.
- *printf* çıktı formatındaki ‘ \ ’ karakterine escape karakteri denir.
- Bu karakterden sonra gelen karakter ise escape serisini ifade eder.
- Örneğin \n ifadesi, çıktı ekranında yeni bir satıra geçilmesi gerektiğini ifade eder.



printf ( ) fonksiyonu

- Kontrol: "\" işaretiyle başlayan bu karakterlerin anlamları şu şekildedir:

- \a Ses üretir (alert)
- \b imleci bir sola kaydır (backspace)
- \f Sayfa atla. Bir sonraki sayfanın başına geç (formfeed)
- \n Bir alt satıra geç (newline)
- \r Satırbaşıyap (carriage return)

- \t Yatay TAB(HorizontalTAB)
- \v Dikey TAB(verticalTAB)
- \" Çift tırnak karakterini ekrana yaz
- \’ Tek tırnak karakterini ekrana yaz
- \\\ karakterini ekrana yaz
- %% % karakterini ekrana yaz

Değişken tanımlama yerleri, şekilleri ve ilk değer atama

- Standart C de üç farklı yerde değişken tanımlanabilir: Fonksiyonların üstünde, blokların { } içerisinde ilk sırada ve fonksiyonlarda parametre olarak.

- İlk değer ataması yapılmayan değişkenlerin değerleri (eğer main fonksiyonunun üstünde tanımlanmışsa) sayısal olanlar 0 diğerleri boş olarak belirlenir, eğer main içinde tanımlanmışsa bellekte rastgele değerler olarak belirlenir.

Değişken tanımlama yerleri, şekilleri ve ilk değer atama

```
#include<stio.h>

/* Örnek Değişken tanımlamaları */

int main(){

    int a;
    a=1;

    int a=1;

    int x, y, z; /* Aynı satırda birden fazla değişken tanımlanabilir. */

    char m='k';

    double n=5.05;

}
```

Değişken tanımlama yerleri, şekilleri ve ilk değer atama

```
/*Bu program dairenin alanini bulur*/
#include <stdio.h>
#define PI 3.14
int main(void)
{
  double varicap, alan;
  printf("Yaricapi giriniz:");
  scanf("%lf", &varicap);
  alan = PI * varicap * varicap;
  printf(" Alan = %f", alan);
  return(0);
}
```

açıklama

kütüphane

isim sabiti

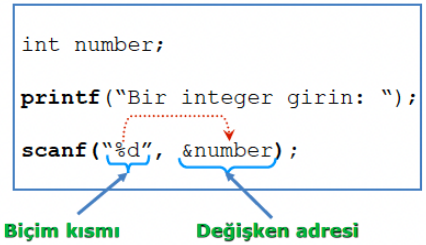
ana fonksiyon

tanımlama komutu

Özel amaçlı sözcükler

scanf ( ) fonksiyonu

- Değişkenlerin içerisine klavyeden değer atamak için kullanılır.
- Fonksiyon ismi ve parametrelerden oluşur.
- Parametre olarak, girilecek değerin hangi formatta olacağını bildiren girdi formatını ve bu formata göre girilecek değişkenler listesini alır.
- *scanf* fonksiyonunda dışarıdan değer girilecek bütün değişkenlerin başına & işareti konur.
- Bu işaret bellek operatörüdür, değişkenlerin tutulduğu bellek hücresinin adresini okur.



Mantıksal operatörler

Ve (&&)

A	B	işlem	sonuç
Hayır	Hayır	A && B	Hayır
Hayır	Evet	A && B	Hayır
Evet	Hayır	A && B	Hayır
Evet	Evet	A && B	Evet

Veya (||)

A	B	işlem	sonuç
Hayır	Hayır	A || B	Hayır
Hayır	Evet	A || B	Evet
Evet	Hayır	A || B	Evet
Evet	Evet	A || B	Evet

Operatörlerde Öncelik Sırası

		YÜKSEK ÖNCELİK	
( )	Soldan sağa		Öncelik op.
! ++ --	Sağdan sola		Aritmetik op.
* / %	Soldan sağa		
+ -	Soldan sağa		
> >= <<=	Soldan sağa		İlişkisel op.
== !=	Soldan sağa		
&&	Soldan sağa		Mantıksal op.
||	Soldan sağa		
=	Sağdan sola	DÜŞÜK ÖNCELİK	Atama op.

return

- *return* ifadesi bir fonksiyondan çıkış yapmak ve program içinde fonksiyon çağrısının yapıldığı işlem satırından bir sonraki işlem satırına geçiş yapmak için kullanılır.
- ***Return 0*** = *0* değeri, programın başarılı bir şekilde yürütüleceği ve yapmak istediği şeyi yapacağı anlamına gelir.
- Eğer bir fonksiyon *void* olmayan bir değer geri döndürecek şekilde tanımlanmış ise, *return* ifadesi fonksiyonun geri döndürdüğü değerle birlikte dönüş yapar. Eğer fonksiyon *void* bir değer geri döndürecek şekilde tanımlanmış ise, *return* değeri de herhangi bir değer geri döndürmez.

ÖRNEKLER

- ***Örnek 1 :*** *" günaydın, merhaba düzce " ifadesini ekrana yazdıracak olan C programlama kodunu yazınız.*

```
#include <stdio.h>
// bu bir c ornegidir
int main( ) {
printf("gunaydin,merhaba duzce");
// printf("gunaydin \n");
// printf("merhaba duzce");
return 0;
}
```

ÖRNEKLER

***Örnek 2** : Aşağıda verilen işlemlerin sonuçlarını işlem öncelik sırasını göz önüne alarak hesaplayınız.*

- *a=8, b=10, c=4, d=2*
- a*b/(c*b)+d+a*b/d;
- a*b/c*b+d+a*b/c;//
- a*b/c*b+(d+a)*b/c;//

```
#include<stdio.h>
int main()
{
int a=8,b=10,c=4,d=2;
double sonuc;
sonuc=a*b/(c*b)+d+a*b/d;
//sonuc=a*b/c*b+d+a*b/c;//
//sonuc=a*b/c*b+(d+a)*b/c;//
printf("%f",sonuc);
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 3*** *: char komutu kullanarak "sehiradi=duzce" ifadesini ekrana yazdırınız.*

```
#include <stdio.h>
int main() {
//parantez ici karakter sayisini belirtir
char sehiradi[10]="duzce";
printf("%s",sehiradi);
return 0;
}
```

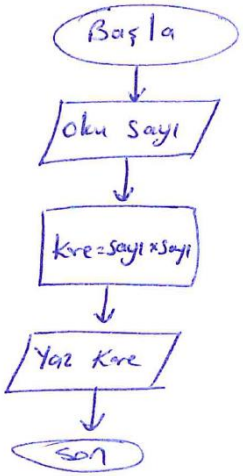
ÖRNEKLER

❖ ***Örnek 4*** *:Kişinin ismini ve soyismini sorarak (scanf), cevapları ekrana yazdırınız.*

```
#include <stdio.h>
int main() {
//parantez ici karakter sayisini belirtir
char isim[10],soyisim[10];
printf("isminizi giriniz:");
scanf("%s",isim);
printf("soyisminizi giriniz:");
scanf("%s",soyisim);
printf("Sayin %s, %s",isim,soyisim);
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 5*** *:Klavyeden girilen bir sayının karesini hesaplayan programın akış diyagramını çiziniz ve C programlama kodunu yazınız.*




```
#include<stdio.h>
int main()
{
int sayi,kare;
printf("sayi giriniz:"); scanf
("%d",&sayi);
kare=sayi*sayi;
printf("sayi= %d karesi=%d",sayi,kare);
return 0;
}
```

ÖRNEKLER

- ***Örnek 6** : sayi1=10, sayi2=20, toplam ve çarpım sonuç ifadelerini ekrana yazdıran C programlama kodunu yazınız.*

```
#include <stdio.h>
int main(){
int sayi1=10,sayi2=20,toplam,carpim;
toplam=sayi1+sayi2;
carpim=sayi1*sayi2;
printf("sayilarin toplami %d'dir: \n",toplam );
printf("sayilarin carpimi %d'dir:",carpim );
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 7*** *: Aşağıda verilen matematiksel ifadelerin C dilindeki kod karşılıklarını yazınız.*

$$C = \sqrt{\frac{A+5}{2+\frac{1}{A^2}}}$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ( ) {
int A=3;
float x;
x = sqrt (( A+5) / (2 + (1 / pow (A,2))));
printf("Islem Sonucu = %.4f" , x );
return 0;
}
```

$$C = \frac{A+B}{B^2+\frac{1}{A^2B}}$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main ( ) {
int A=4,B=8;
float x;
x= (float)(A+B) / (float)((B*B)+(1/(A*A*B)));
printf("Islem Sonucu = %.4f" , x );
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 8*** *: A=3, B=4 ve C=2 için aşağıda verilen denklemin C programlama kodunu yazarak sonucunu hesaplayınız.*

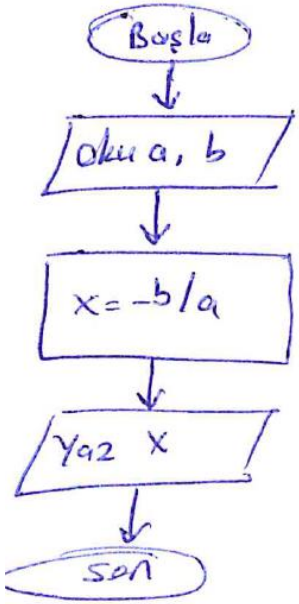
$$x = \frac{A + AB^2 - 3AC}{2ABC}$$

```
#include <stdio.h>
#include <stdlib.h>
int main ( ) {
int A=3, B=4, C=2;
float x;
x = (float) (A+A*(B*B)+3*A*C) / (float) (2*A*B*C);
printf(" Islem Sonucu = %.4f ", x );

return 0;
}
```

ÖRNEKLER

❖ ***Örnek 9*** *: ax+b=0 tek bilinmeyenli denklemi için kullanıcı tarafından klavyeden girilen a ve b değerlerine göre x değişkenini hesaplayan programın akış diyagramını çiziniz ve C programlama kodunu yazınız.*




```
#include<stdio.h>
int main()
{
        double x,a,b;
        printf("a ve b degerlerini giriniz:");
        scanf("%lf %lf",&a,&b);
        x=-b/a;
        printf("x=%lf",x);
        return 0;
}
```

ÖRNEKLER

❖ ***Örnek 10*** *: Bir öğrencinin 3 sınav sonucunun ortalamasını alan programı C programlama dilinde yazınız*

```
#include <stdio.h>
int main() {
float sinav1,sinav2,sinav3,ortalama;
printf("1. sinav sonucunu giriniz=" );
scanf("%f",&sinav1);
printf("2. sinav sonucunu giriniz= ");
scanf("%f",&sinav2);
printf("3. sinav sonucunu giriniz= ");
scanf("%f",&sinav3);
ortalama=(sinav1+sinav2+sinav3)/3;
printf("Ders ortalamaniz %f'dir",ortalama);
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 11*** *: Bir kenar uzunluğu 8 cm olan karenin cevresini ve alanını hesaplayan C programı kodunu yazınız.*

```
#include <stdio.h>
int main() {
int uzunluk=8,cevre,alan;
cevre=uzunluk*4;
alan=uzunluk*uzunluk;
printf("karenin cevresi %d, alani ise %d'dir: \n",cevre,alan );
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 12*** *: Taban ve Yükseklik Değerine göre üçgen alan hesabi yapan algoritmaya ait C programlama kodunu yazınız.*

```
#include <stdio.h>
int main() {
//int komutu sadece sabit degerleri hesaplar.
float taban,yukseklik;
float alan;
printf("taban degerini giriniz: ");
scanf("%f",&taban);
printf("yukseklik degerini giriniz: ");
scanf("%f",&yukseklik);
alan=(taban*yukseklik)/2;
printf("ucgenin alani %.2fdir",alan);
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 13*** *:Aşağıda verilen program parçacığındaki yanlışlıkları bularak nedenlerini belirtiniz. Program yarıçapı klavyeden girilen bir dairenin alanını ve çevresini hesaplamaktadır.*

```
#include<stdio.h>
#include<stdlib.h>
const float PI=3.14;
int main(){
        float r;
        int  alan , cevre;
        printf("Yaricapi Giriniz:");
        scanf("%f",r);
        alan=PI*r*r;
        cevre=2*PI*r;
        printf("Alan=%.2f\n",Alan);
        printf("Cevre=%.2f\n",cevre);
        return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
const float PI=3.14;
int main(){
        float r, alan , cevre;
        printf("Yaricapi Giriniz:");
        scanf("%f",&r);
        alan=PI*r*r;
        cevre=2*PI*r;
        printf("Alan=%f\n",alan);
        printf("Cevre=%f\n",cevre);
        return 0;
}
```

ÖRNEKLER

❖ ***Örnek 14*** *: Aşağıda verilen program parçacığındaki yanlışlıkları bularak nedenlerini belirtiniz. Program koordinatları klavyeden girilen iki nokta arasındaki mesafenin bulunmasını sağlamaktadır..*

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main ( ) {
int x1,y1,x2,y2
float 2uzaklik;
printf("1. Noktanin koordinantlarini giriniz:" );
scanf("%d%d",x1,y1);
printf("2. Noktanin koordinantlarini giriniz: ");
scanf("%d%d",&x2, &y2);
uzaklik=sqrt (pow ( (y2-y1) ^ 2) + pow( (x2-x1) ^ 2) );
printf("\n Mesafe : %.2d", uzaklik);
return 0;
}
```

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main ( ) {
int x1,y1,x2,y2
float uzaklik;
printf("1. Noktanin koordinantlarini giriniz:" );
scanf("%d%d", &x1, &y1);
printf("2. Noktanin koordinantlarini giriniz: ");
scanf("%d%d",&x2, &y2);
uzaklik=sqrt (pow ( (y2-y1) , 2) + pow( (x2-x1) , 2) );
printf("\n Mesafe : %.2f ", uzaklik);
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 15*** *: Aşağıda verilen programın çıktısı ne olur?*

```
#include<stdio.h>
#include<stdlib.h>
int main( )
{
		int a=1, b=10, c=100;
		printf("%d\n",a);
		printf("%d\n",b);
		printf("%d\n",c);
		printf("\n");
		printf("%3d\n",a);
		printf("%3d\n",b);
		printf("%3d\n",c);
		printf("\n");
		printf("%.3d\n",a);
		printf("%.3d\n",b);
		printf("%.3d\n",c);
		return 0;
}
```

```
1
10
100

  1
 10
100

001
010
100
```

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

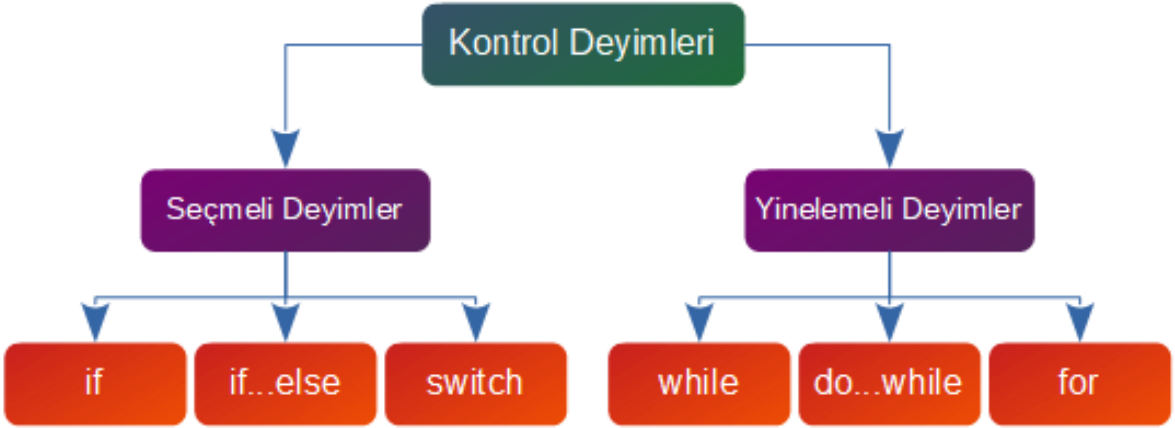
EEM165 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 4

Dr. Öğr. Üyesi Enes KAYMAZ

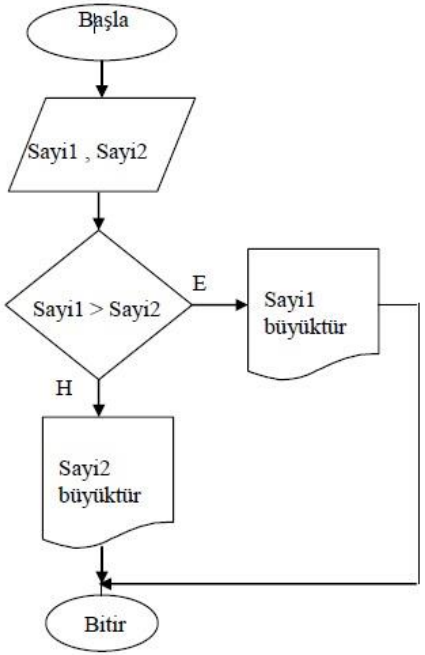
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

KARAR YAPILARI



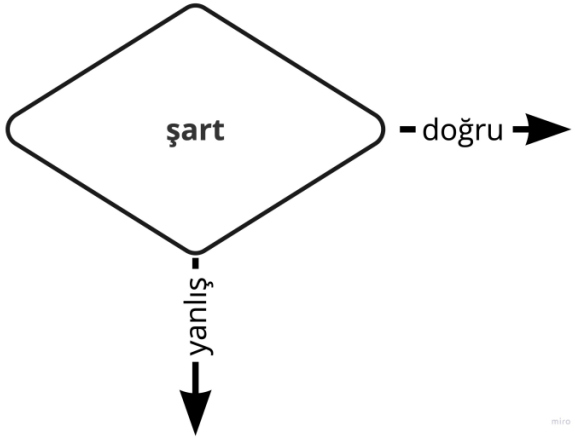
Akış Diyagramı

- **Akış diyagramı:** Algoritmanın grafiksel gösterimidir.
- Özel şekiller birbirine çizgilerle bağlanır ve oklar akış yönünü gösterir.
- **Dikdörtgen şekli (işlem sembolü):** Herhangi bir işlemi gösterir.
- **Oval Şekil:** Programın veya programın bir bölümünün başlangıcını ve sonunu gösterir.
- **Tek-giriş/tek çıkış kontrol yapısı:** Bir kontrol yapısının çıkışı diğerinin girişine bağlanır. Programın yapılandırılmasını kolaylaştırır.



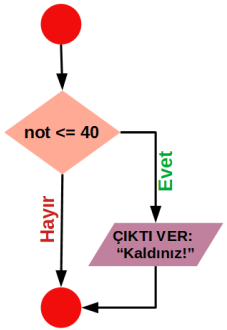
Koşullar

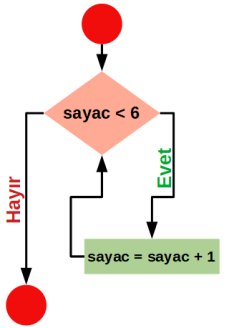
- Akış diyagramında karar yapısı için baklava deseni sembolü kullanılır.
- Bu şeklin içine şart ifadesi yazılır. Karar yapısından iki adet ok çıkar.
- Birisi şartın doğru olması durumunda, diğeri ise yanlış olması durumunda gidilecek yönü belirtir.



Koşullar

- **Seçmeli Yapı (Selection Structure):** Kodun çalışma akışını belli şartlara göre dallandırmaya (branching) yarayan yapılara denir. Programda belli kararlar alınması gerektiğinde kullanılır. Programın akışı 2 veya daha fazla yola ayrılabilir. Daha önce kullandığımız ve bu bölümde anlatacağımız if deyimi de bu yapıdadır.
- **Yinelemeli Yapı (Iteration Structure):** Bir kodun birden fazla olacak şekilde çalışmasını sağlayan yapılardır. Bu yapılarda da yine seçmeli yapıda olduğu gibi koşul ifadeleri kullanılır. Fakat bu sefer koşulun gerçekleşmesi sonucu yapılması gereken işlem birden fazla kez tekrarlanabilir.

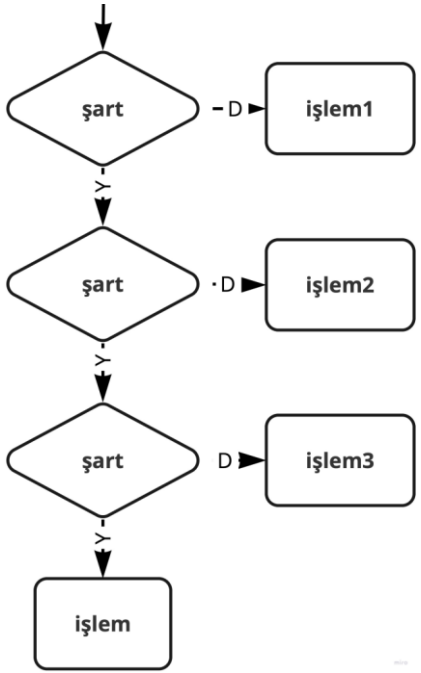




Kademeli Yapılar

- Bazen birden fazla koşulu test etmek isteriz, ta ki bir koşul sağlanana kadar.

eğer *<şart ifadesi1>*
işlem1;
değilse eğer *<şart ifadesi2>*
işlem2;
değilse eğer *<şart ifadesi3>*
işlem3;
…….
değilse
işlem;



if Yapısı

- İşlem gruplarından birini seçmek için kullanılır.
- *if* ifadesi parantez içindeki test ifadesini değerlendirir.
- Test ifadesi true (sıfır olmayan) olarak değerlendirilirse, *if* 'in gövdesi içindeki ifadeler uygulanır.
- Test ifadesi false (0) olarak değerlendirilirse, *if*''in gövdesi içindeki ifadeleri yürütmeden atlanır.
- C boşlukları ve satır sonlarını dikkate almaz.

Expression is true.

```
int test = 5;

if (test < 10)
{
  // codes
}

// codes after if
```

Expression is false.

```
int test = 5;

if (test > 10)
{
  // codes
}

// codes after if
```

- Eğer *if* 'in altında birden çok komut varsa, ayraç işareti (veya küme parantezi) koymamız gerekir. Şayet *if* 'ten sonra, tek komut bulunuyorsa, ayraç koyup-koymamak size kalmıştır. Zorunluluğu yoktur.

Mantıksal Operatörler

- Boolean ifadeler mantıksal operatörler (AND,OR,NOT) kullanılarak birleştirilebilirler.
- C dilinde bunlar için “&&” “||” “!” sembolleri kullanılıyor.

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

- **1: mantıksal AND operatörü kullanarak**

```
if(sayı >=0&&sayı <=100)
cout<<"sayı aralıkiçinde";
else
cout<<"sayı aralıkdısında";
```

- **2: mantıksal AND ve NOT operatörleri kullanarak**

```
if(!(sayı >=0&&sayı <=100))
cout<<"sayı aralıkdısında";
else
cout<<"sayı aralıkiçinde";
```

- **3: mantiksal OR operatörü kullanarak**

```
if(sayı <0||sayı >100)
cout<<"sayı aralıkdısında";
else
cout<<"sayı aralıkiçinde";
```

if – else Yapısı

- ***if*** deyiminin içerisinde tanımlanan koşul haricinde kalan tüm koşullar için ***else*** deyimi kullanılmaktadır.
- Programda ***else*** deyimi kullanıldığında ***if*** deyiminin çalışmadığı yani ***if*** koşulunun gerçekleşmediği tüm koşullarda ***else*** bloğundaki kod çalışmaktadır. Test ifadesi **true** olarak değerlendirilirse,
  – ***if*** ifadesinin gövdesi içindeki ifadeler yürütülür.
  – ***else*** ifadesinin gövdesi içindeki ifadeler yürütmeden atlanır.
- Test ifadesi false olarak değerlendirilirse,
  – ***else*** ifadesinin gövdesi içindeki içindeki ifadeler çalıştırılır.
  – ***if*** ifadesinin gövdesi içindeki ifadeler atlanır.

Expression is true.

```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

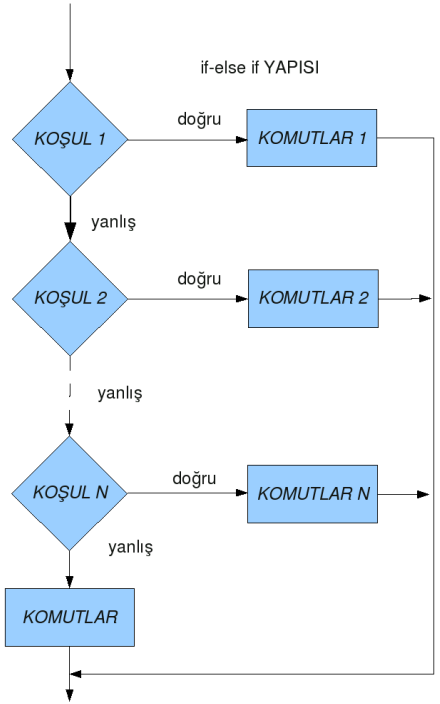
Expression is false.

```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

if , else-if, else Yapısı

- Bazı durumlarda yazılan kodlarda birden fazla koşul programcı tarafından programa eklenmek ister. Bu gibi durumlarda her koşul için sürekli olarak ***if*** yapısı kurmak yerine tek bir ***if*** yapısı içerisinde tüm koşulların yazılması tercih edilmektedir. İlk koşul ***if*** deyimi ile tanımlanırken diğer koşulların hepsi ***else if*** deyimiyle tanımlanmaktadır. Son olarak tüm koşulları kapsamayan koşul ***else*** deyimi ile ifade edilmektedir.



break ve continue Yapısı

- ***break*** ifadesi döngüyü sonlandırırken,
- ***continue*** ifadesi döngünün bir sonraki yinelemesini zorlar.
- Bazen, döngü içindeki bazı ifadeleri atlamak veya test ifadesini kontrol etmeden hemen döngü sonlandırılmak istenebilir.
- Bu gibi durumlarda, ***break*** ve ***continue*** ifadeleri kullanılır.

```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

break ve continue Yapısı

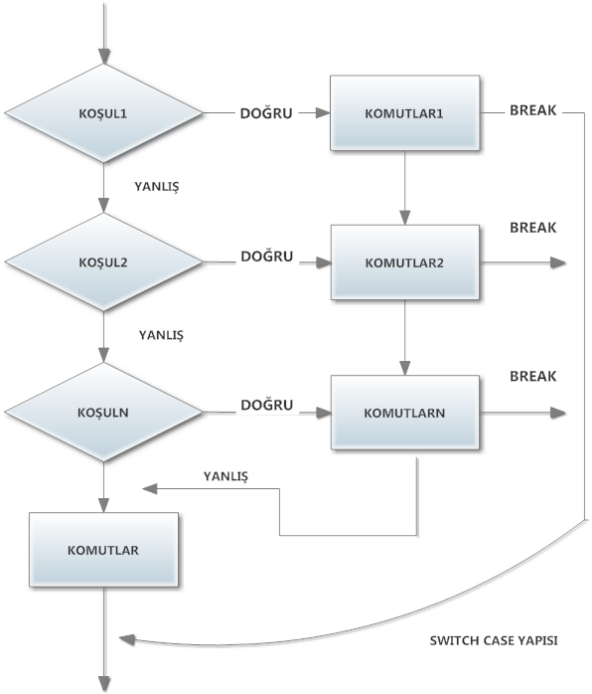
- ***break*** ifadesi döngüyü sonlandırırken,
- ***continue*** ifadesi döngünün bir sonraki yinelemesini zorlar.
- Bazen, döngü içindeki bazı ifadeleri atlamak veya test ifadesini kontrol etmeden hemen döngü sonlandırılmak istenebilir.
- Bu gibi durumlarda, ***break*** ve ***continue*** ifadeleri kullanılır.

```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
} while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

switch-case Yapısı



- Bu yapı bir değişkenin içeriğine bakarak, programın akışını çok sayıda seçenekten birine yönlendirir. ***case (durum)*** yapısından sonra değişkenin durumu belirlenir ve sonraki gelen satırlar işleme girer.
- Aksi söz konu olduğunda gerçekleştirilmesi istenen deyimler ***default*** deyiminden sonraki kısımda bildirilir.
- Kısaca yapının temel amacı; değişkenin değerine göre programın çalışmasına yön vermektir.
- Aynı işlem ***if else*** yapısı ile uygulanabilse de daha kolay okunması nedeniyle programcılar tarafından tercih edilmektedir.

switch-case Yapısı

```
switch( secim )
{
        case 1:
               sonuc = x + y;
               printf("Toplam = %f\n",sonuc);
               break;
        case 2:
               sonuc = x-y;
               printf("Fark = %f\n",sonuc);
               break;
        case 3:
               sonuc = x * y;
               printf("Carpim = %f\n",sonuc);
               break;
        case 4:
               sonuc = x/y;
               printf("Oran = %f\n",sonuc);
               break;
        default:
               puts("Yanlis secim !\a");
}
```

```
if(secim == 1){
               sonuc = x + y;
               printf("Toplam = %f\n",sonuc);
}
else if(secim == 2){
               sonuc = x-y;
               printf("Fark = %f\n",sonuc);
}
else if(secim == 3 ){
               sonuc = x * y;
               printf("Carpim = %f\n",sonuc);
}
else if(secim == 4){
               sonuc = x/y;
               printf("Oran = %f\n",sonuc);
}
else{
               puts("Yanlis secim !\a");
}
```

ÖRNEKLER

❖ ***Örnek 1 :*** *Girilen sayının 10dan büyük olup olmadığını karşılaştıran programı if yapısı kullanarak yazınız.*

```
#include <stdio.h>
int main() {
int sayi;
printf("bir sayi giriniz= ");
scanf("%d",&sayi);
if(sayi>10)
{
printf("girilen sayi 10dan buyuktur");
}
else
{
printf("girilen sayi 10dan kucuktur");
}
}
```

ÖRNEKLER

❖ ***Örnek 2 :*** *Girilen sayının pozitif, negatif veya 0 olduğunu gösteren programı yazınız.*

```
#include <stdio.h>
int main() {
int sayi;
printf("bir sayi giriniz= ");
scanf("%d",&sayi);
if(sayi>0)
{
printf("sayi pozitiftir");
}
if(sayi<0)
{
printf("sayi negatiftir");
}
if(sayi==0)
{
printf("sayi sifira esittir");
}
}
```

ÖRNEKLER

❖ ***Örnek 3 :*** *Girilen 2 sayının toplamının 50'den büyük olup olmadığını karşılaştıran programı yazınız.*

```
#include <stdio.h>
int main() {
int sayi1,sayi2,toplam;
printf("bir sayi giriniz= ");
scanf("%d",&sayi1);
printf("bir sayi giriniz= ");
scanf("%d",&sayi2);
toplam=sayi1+sayi2;
if(toplam>50)
{
printf("toplam sayi 50dan buyuktur");
}
else
{
printf("toplam sayi 50dan kucuktur");
}
}
```

ÖRNEKLER

❖ ***Örnek 4 :*** *Girilen 3 sınavın ortalaması 50'den büyük ise, dersi geçtiğini gösteren ifadeyi yazdıran programın C programlama kodunu yazınız.*

```
#include <stdio.h>
int main() {
int sinav1,sinav2,sinav3,ortalama;
printf("bir sayi giriniz= ");
scanf("%d",&sinav1);
printf("bir sayi giriniz= ");
scanf("%d",&sinav2);
printf("bir sayi giriniz= ");
scanf("%d",&sinav3);
ortalama=(sinav1+sinav2+sinav3)/3;
if(ortalama>50)
{
printf("ortalamaniz %d, Dersi gectiniz",ortalama);
}
else
{
printf("ortalamaniz %d,Dersi tekrardan aliniz",ortalama);
}
}
```

ÖRNEKLER

- ***Örnek 5 :*** *Üç sınav sonucunun ortalamasına göre harf notunu belirleyen programın kodunu C dilinde yazınız.*
- *80-100= A*
- *70-80=B*
- *60-70=C*
- *60 ve aşağısı=D*

```
#include <stdio.h>
int main() {
float sinav1,sinav2,sinav3,ortalama;
printf("bir sayi giriniz= ");
scanf("%f",&sinav1);
printf("bir sayi giriniz= ");
scanf("%f",&sinav2);
printf("bir sayi giriniz= ");
scanf("%f",&sinav3);
ortalama=(sinav1+sinav2+sinav3)/3;

if(ortalama>=80 && ortalama <100)
{
printf("ortalamaniz %f,Harf notunuz=A",ortalama);
}
else if(ortalama>=70 && ortalama <80)
{
printf("ortalamaniz %f,Harf notunuz=B",ortalama);
}
else if(ortalama>=60 && ortalama <70)
{
printf("ortalamaniz %f,Harf notunuz=C",ortalama);
}
else
{
printf("ortalamaniz %f,Harf notunuz=D",ortalama);
}
}
```

ÖRNEKLER

❖ ***Örnek 6 :*** *Girilen sayının 2 veya 5e bölündüğünü gösteren programı (||-or) yapısıyla yazınız*

```
#include <stdio.h>

int main() {

int sayi1;
printf("bir sayi giriniz= ");
scanf("%d",&sayi1);
if(sayi1%2==0 || sayi1%5==0)
{
printf("sayi 2 veya 5e tam bolunur");
}
else
{
printf("sayi 2 veya 5e tam bolunmez");
}
}
```

ÖRNEKLER

❖ ***Örnek 7 :*** *Klavyeden kenar uzunlukları girilen bir üçgenin ne tür bir üçgen olduğunu (eşkenar,ikizkenar,çeşitkenar) bulan C programını yazınız.*

```
#include<stdio.h>
#include<stdlib.h>
int main(){
		int a,b,c;
		printf("Kenar uzunluklarini giriniz:");
		scanf("%d %d %d",&a,&b,&c);
		if((a==b)&&(b==c)){
		printf("Tum kenarlar esit oldugundan eskenar ucgendir.\n");
		}else if((a==b)||(b==c)||(a==c)){
				printf("Iki kenar esit oldugundan ikizkenar ucgendir.\n");
		}else{
				printf("Tum kenarlar farkli uzunlukta oldugundan cesitkenar ucgendir.\n");
		}
		return 0;
}
```

ÖRNEKLER

❖ ***Örnek 8 :*** *İkinci dereceden bir bilinmeyenli bir denklemin köklerini hesaplayan C programlama kodunu yazınız.*

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main(){
        float x1,x2;
        int a,b,c,delta;
        printf ("Denklemin a katsayisi:");
        scanf("%d",&a);
        printf("Denklemin b katsayisi:");
        scanf("%d",&b);
        printf("Denklemin c katsayisi:");
        scanf ("%d",&c);
        delta=pow(b,2)-4*a*c;
        if (delta<0)
        printf("Denklemin kokleri yoktur.\n");
        else if (delta==0){
                printf("Denklemin esit iki koku vardir.\n");
                x1=(-b)/(2*a);
                printf("X1=X2=%.2f",x1);
        }else{
                printf("Denklemin farkli iki koku vardir.\n");
                x1=(-b+sqrt(delta))/(2*a);
                x2=(-b-sqrt(delta))/(2*a);
                printf("X1=%.2f\n X2=%.2f\n",x1,x2);
        }
        return 0;
        }
```

ÖRNEKLER

❖ ***Örnek 9 :*** *Switch yapısıyla girilen rakamı [1-5] tahmin eden programı yazınız.*

```
#include <stdio.h>
int main()
{
int sayi;
printf("Bir sayi giriniz [1-5]\n");
scanf("%d", &sayi);
switch (sayi) {
case 1:
printf("Bir rakamini tusladiniz");
break;
case 2:
printf("İki rakamini tusladiniz");
break;
case 3:
printf("Üç rakamini tusladiniz");
break;
case 4:
printf("Dört rakamini tusladiniz");
break;
case 5:
printf("Beş rakamini tusladiniz");
break;
default:
printf("Hatali giris yaptiniz");
                                    }

 return 0;
}
```

ÖRNEKLER

❖ ***Örnek 10 :*** *Kullanıcı tarafından rakam olarak girilen bir değerin haftanın hangi günü olduğunu gösteren C programlama kodunu switch-case kullanarak yazınız.*

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

int main(){
        int gun;
        printf("Gun degerini giriniz (1-7):");
        scanf("%d",& gun);
        switch(gun){
                case 1: printf("Pazartesi\n");break;
                case 2: printf("Sali\n");break;
                case 3: printf("Carsamba\n");break;
                case 4: printf("Persembe\n");break;
                case 5: printf("Cuma\n");break;
                case 6: printf("Cumartesi\n");break;
                case 7: printf("Pazar\n");break;
                default:
                    printf("Gecersiz bir deger girdiniz.\n");

        }
        return 0;
}
```

ÖRNEKLER

- ***Örnek 11 :*** *Switch yapisiyla 4 islem yapan programı aşağıdaki şekilde yazınız.*
- *Kullanicidan 2 sayi iste, printf ile islem komutlarini gir. Switch komutuyla islemleri sirala.*

```
#include <stdio.h>
main()
{
int sayi1,sayi2;
int islem;

printf("1.Sayiyi giriniz:");
scanf("%d",&sayi1);
printf("2.Sayiyi giriniz: ");
scanf("%d",&sayi2);

printf("\n\n1-Toplama\n");
printf("2-Cikarma\n");
printf("3-Bolme\n");
printf("4-Carpma\n");
printf("\nIslemi seciniz:");
scanf("%d",&islem);
switch(islem){
case 1:
printf("Toplama isleminin sonucu : %d",sayi1 + sayi2);
break;
case 2:
printf("Cıkarma isleminin sonucu : %d",sayi1 - sayi2);
break;
case 3:
printf("Bolme isleminin sonucu : %f", (float) sayi1 / (float) sayi2);
break;
case 4:
printf("Carpma isleminin sonucu : %d", sayi1 * sayi2);
break;
default:
printf("Lutfen gecerli bir sayi giriniz..");
}
}
```

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

EEM165 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 5

Dr. Öğr.Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

Döngüler

- Bir programda bir komut parçasının belli koşullar gerçekleştikçe veya gerçekleşinceye kadar defalarca icra etmesi gerekebilir. Bu defalarca çalışmayı sağlayan komutlarına döngü (çevrim) yapıları denir. Döngü yapıları genelde karşımıza iki tür olarak çıkar. Çalışma sayısının belli olduğu durumlarda sayaç kontrollü döngüler, döngü sayısının değişken olduğu durumlarda ise koşullu döngüler kullanılır. C programlama dilinde sayaç kontrollü döngü yapısına ***for*** döngüsü, koşullu döngü yapısına ***while*** ve ***do-while*** döngüleri örnek olarak verilebilir.

for döngüsü

❖ C programlama dilinde sayaç kontrollü döngü yapısı olarak for döngüsü kullanılmaktadır. Kullanımı aşağıdaki gibidir;

✓ *For (ilk_değer; devam_şartları; ara_işlemler)*

✓ *{*

✓ *Döngü_Bloğu*

✓ *}*

❖ ***ilk_değer :*** Döngüye ilk defa girildiğinde burada belirttiğimiz komutlar icra edilir. Döngünün diğer icralarında bu işlem tekrarlanmaz.

❖ ***devam_şartları :*** Döngünün bir kere daha icra ettirileceği veya döngünün icrasına son verileceği buradaki koşul ifadeleri değerlendirilerek yapılır. Buradaki ifade doğru sonuç verdikçe döngü yenilenir.

❖ ***ara_işlemler :*** Döngünün her icrası sonunda uygulanacak işlemler burada belirtilir.

❖ For döngüsü altında tek bir satırlık kod yazıldığında blok parantezleri { } kullanmak zorunluluğu yoktur, ancak döngü için birden fazla komut yazılacaksa blok parantezleri { } kullanılmalıdır.

for döngüsü

C

```
char ad[100][20];
...
printf("1. kişinin adı :"); scanf("%s", ad[0]);
printf("2. kişinin adı :"); scanf("%s", ad[1]);
printf("3. kişinin adı :"); scanf("%s", ad[2]);
...
...
printf("100. kişinin adı :"); scanf("%s", ad[99]);
```

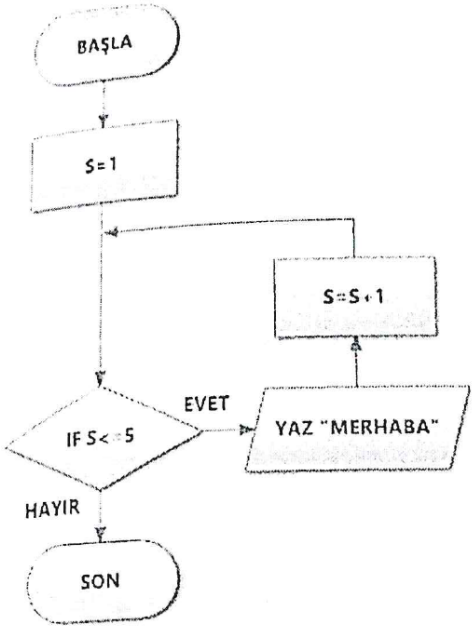
Döngüler olmasaydı, böyle bir durumda 100 satır yazmamız gerekirdi.

```
char ad[100][20];
int i;
...
for (i=0; i<100; i++)
{
   printf("%d. kişinin adı :", i+1);
   scanf("%s", ad[i]);
}
```

Döngüler kullanarak aynı işi daha az satırda yapabiliriz.

ÖRNEKLER

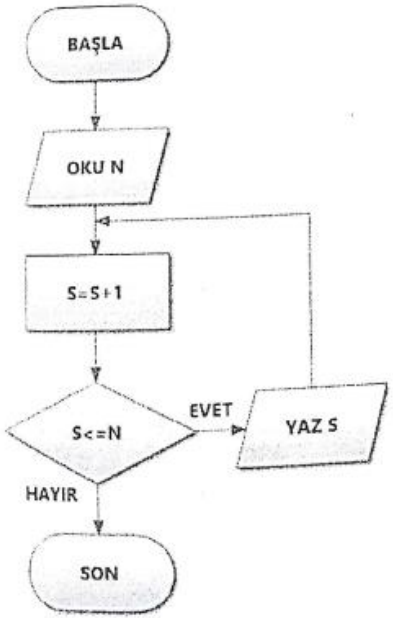
❖ ***Örnek 1 :*** *Ekrana 5 Defa 'Merhaba' yazdıran programın akış diyagramını çiziniz ve C programlama kodunu for döngüsü kullanarak yazdırınız.*




```
#include <stdio.h>
int main() {
int i;
for(i=1;i<=5;i++)
printf("Merhaba");
return 0;
}
```

ÖRNEKLER

- ***Örnek 2:*** *1' den 20'ye kadar olan tamsayıları ekrana yazdıran programın akış diyagramını oluşturunuz ve C programlama dilinde kodunu for döngüsü kullanarak yazınız*




```
#include <stdio.h>
int main() {
for (int i = 0; i <= 20; i++)
{
printf("%i \n", i);
}
printf("\n");
return 0;
}
```

ÖRNEKLER

❖ ***Örnek 3 :*** *12 'den 54'e kadar olan çift sayıların toplamını hesaplayan programı C programlama dilinde for döngüsünü kullanarak kodlayınız.*

```
#include <stdio.h>
int i, sonuc;
main() {
sonuc= 0;
for (i=12; i<54; i=i+2)
{
sonuc= sonuc+ i;
}
printf("12'den 54'e kadar çift sayıların toplamı :%d", sonuc);
return 0;
}
```

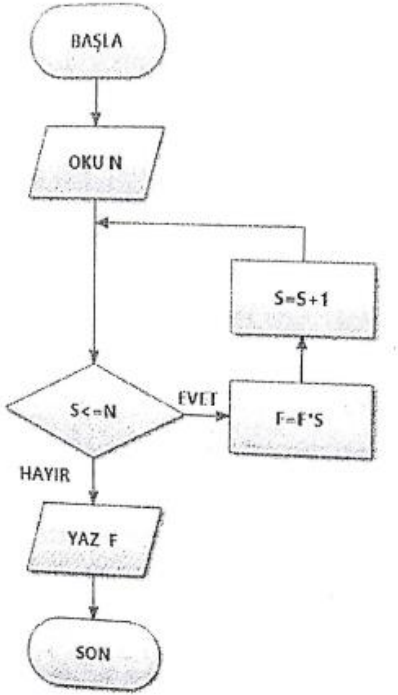
ÖRNEKLER

❖ ***Örnek 4 :*** *Klavyeden girilen N adet tam sayının ortalamasını bulan programı for döngüsü kullanarak C dilinde yazınız.*

```
#include <stdio.h>
#include<stdlib.h>
int main ( ){
        double ortalama=0;
        int i, toplam=0,N,sayi;
        printf("Kac adet sayi toplanacak:");
        scanf("%d",&N);
        for (i=0;i<N;i++){
        printf("%d.sayiyi gir:",i+1);
        scanf("%d",&sayi);
        toplam += sayi;
        }
        ortalama=(float) toplam/N;
        printf("Girilen sayilarin toplami=%d\n",toplam);
        printf("Girilen sayilarin ortalamasi=%f\n",ortalama);
        return 0;
}
```

ÖRNEKLER

❖ ***Örnek 5 :*** *Kullanıcı tarafından girilen bir sayının faktöriyelinin for döngüsüyle hesaplandığı ve ekrana yazdırıldığı programın akış diyagramını çiziniz ve C programlama dilinde kodunu yazınız.*




```
#include <stdio.h>
int main() {
int c, n;
int faktoriyel = 1;
printf("Faktoriyel Alinacak Sayiyi Giriniz\n");
scanf("%d", &n);
for (c = 1; c <= n; c++)
faktoriyel = faktoriyel * c;
printf("%d", faktoriyel);
printf("\n");
return 0;
}
```

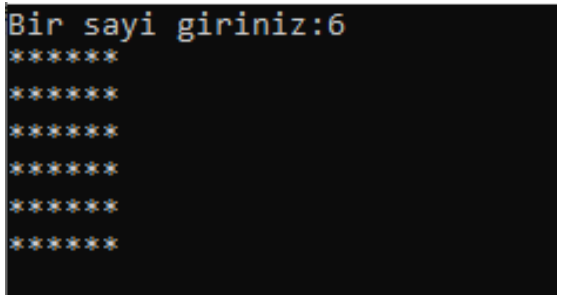
ÖRNEKLER

❖ ***Örnek 6 :*** *Klavyeden girilen bir N pozitif tam sayısına kadar 7'ye tam bölünen sayıları listeleyen C programını yazınız.*

```
#include <stdio.h>
#include<stdlib.h>
int main ( ){
        int i,N;
        printf("Bir sayi giriniz:");
        scanf("%d",&N);
        for(i=0;i<=N;i++){
                if(i % 7==0){
                printf("%d\n",i);
                }
        }
        return 0;
}
```

ÖRNEKLER

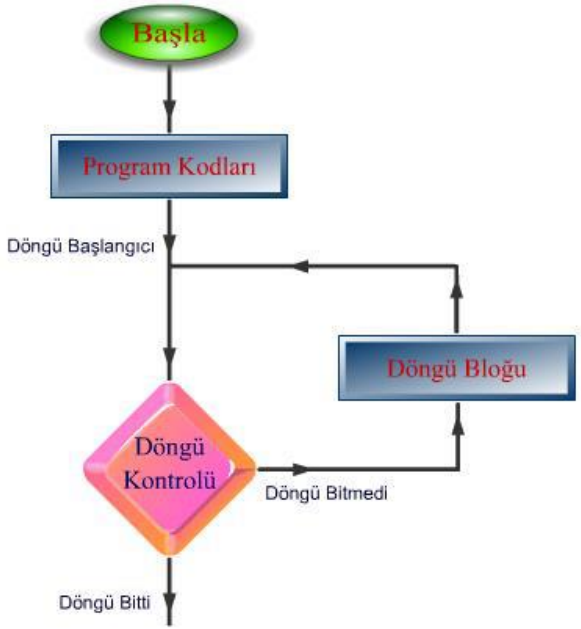
❖ ***Örnek 7 :*** *Aşağıda ekran görüntüsü verilen şekildeki gibi klavyeden girilen bir sayı adeti kadar ekrana * karakteri bastıran C programını yazınız.*




```
#include <stdio.h>
#include<stdlib.h>
int main ( ){
        int sayi, i, j;
        printf("Bir sayi giriniz:");
        scanf("%d",&sayi);
        for(i=0;i<sayi;i++){
                for(j=0;j<sayi;j++){
                        printf("*");
                }
                printf("\n");
        }
        return 0;
}
```

Koşullu Döngüler

- Eğer program içerisinde belirtilen bloklar belli bir koşula göre tekrar edecekse koşullu döngü yapısı kullanılır. Koşulun döngü bloğundan önce veya sonra kontrolüne göre koşullu döngü yapıları ikiye ayrılır. Verilen koşul ifadesi doğru olduğu sürece, döngü bloğun tekrar icra görmesini sağlayan döngü yapısının akış şeması şeklindeki gösterimi yandaki gibidir;
- Döngünün kaç defa çalışacağı bilinmediği durumlarda kullanılır



while döngüsü

- C programlama dilinde ön kontrollü koşullu döngü yapısı olarak while döngüsü kullanılmaktadır. While döngüsü döngü sayısının belli olmadığı durumlarda kullanılır. belirtilen şart sağlandığı sürece döngü çalışmaya devam eder.
- Kullanımı aşağıdaki gibidir;

```
while (devam_şartları)
{
Döngü_Bloğu
}
```

- ***devam_şartları :*** Döngünün bir kere daha icra ettirileceği veya döngünün icrasına son verileceği buradaki koşul ifadeleri değerlendirilerek yapılır. Buradaki ifade doğru sonuç verdikçe döngü yenilenir.

- While döngüsü altında tek bir satırlık kod yazıldığında blok parantezleri { } kullanmak zorunluluğu yoktur, ancak döngü için birden fazla komut yazılacaksa blok parantezleri { } kullanılmalıdır

ÖRNEKLER

❖ ***Örnek 1 :*** *While döngüsü kullanarak ekrana 10 defa 'while dongusu' yazdıran programın C kodunu yazınız.*

```
#include <stdio.h>
int main(){
int i=0;
while(i<10){
printf("%d \n",i);
 i++;
}
}
```

ÖRNEKLER

❖ ***Örnek 2 :****While döngüsü ile, sayı 0 değilse tekrardan sayı girmesini isteyen programın C kodunu yazınız.*

```
#include <stdio.h>
int main() {
int c =1;
while ( c != 0 )
{
printf("\nBir Sayi Girin:");
scanf("%i", &c);
}
printf("\nDonguden Cikildi. ");
printf("\n");
return 0;
 }
```

ÖRNEKLER

❖ ***Örnek 3 :*** *While döngüsü ile, 0'dan 20'ye kadar olan tek sayıları ekrana yazdıran programın C kodunu yazınız.*

```
##include <stdio.h>
 int main(){
int i=0;
while(i<20){
if(i%2 == 1){ //Eğer 2 ile modu alındığında sonuç 1 ise sayı tektir.
printf("%d \n",i);
}
 i++;
}
}
```

ÖRNEKLER

❖ ***Örnek 4 :****Klavyeden girilen bir N pozitif tam sayısından sıfıra (0) kadar olan tamsayıları listeleyen C programını while döngüsü kullanarak yazınız.*

```
#include <stdio.h>
#include <stdlib.h>
int main( ) {
int N,i;
printf(" Bir Sayi Giriniz:");
scanf (" %d" ,&N);
i=N;
while ( i>=0){
printf(" %d \n", i);
i--;
}
return 0;
 }
```

Do-While döngüsü

- **Do-While döngüsü**, döngü bloğunun tekrar icrası verilen koşul ifadesi gerçekleştiği sürece gerçekleşen döngü yapısıdır.
- Bazı durumlarda döngü bir kere çalıştıktan sonra devam edip etmemeye karar vermek isteriz. Bu durumlarda **"do while"** döngüsü kullanılır.
- Döngünün gövdesi en az bir kere çalışmaktadır.

```
do
{
Döngü_Bloğu
} while (devam_şartları)
```

- **Do-While** döngüsü içindeki döngü bloğu her durumda en az bir kere çalışır. Tekrar çalışıp çalışmayacağını ise şart belirler.
- **devam_şartları :** Döngünün bir kere daha icra ettirileceği veya döngünün icrasına son verileceği buradaki koşul ifadeleri değerlendirilerek yapılır. Döngü sonuna gelindiğinde buradaki ifade doğru sonuç verdikçe döngü yenilenir.

Do-While döngüsü

```
#include <stdio.h>
int main()
{
int a = 10;
do{
printf("Kod Bloklari");
}
while(a > 100);
}
```

- Sol tarafta yer alan kodda, *while* komutunun içindeki koşulun yanlış olmasına rağmen döngü 1 defa çalışır. Bunun sebebi *do-while* döngüsü şarta bakmaksızın önce *'do'* bloğunu çalıştırır daha sonra şartı kontrol eder.

```
#include <stdio.h>
int main()
{
int a = 0;
do{
printf("Kod Bloklari\n");
a++;
}
while(a < 5);
}
```

while & do-while döngüsü

while	do while
Döngüye koymak için kontrol eder	Döngüden çıkarmak için kontrol eder.
Döngü çalışabilir veya çalışmaz	Döngü en az bir kere çalışır

Döngü Kontrol Ifadeleri

- Döngü içerisinde bazı durumların oluşması sonucu döngüden çıkmak veya döngüyü bir sonraki değeri için çalıştırmak gerekebilir. C dilinde bu işlemleri gerçekleştirmek üzere iki komut mevcuttur;
  - ✓ ***break***
  - ✓ ***continue***
- ***Break*** komutu, programın icrasını döngü dışına taşır ve programın icrası döngünün bittiği yerden devam eder. ***Break*** komutunu içeren en içteki blok terk edilir. Genellikle bazı koşulların oluşması sonucunda döngünün diğer deyimlerinin icra edilmemesi gereken veya edilmesinin gereksiz olduğu durumlarda kullanılır.
- ***Switch-case, for, while*** veya ***do-while*** komutları ile oluşan bloklar içinde yer alabilir. Genellikle, ***switch-case*** yapısında her bir ***case*** satırının sonunda ***break*** kullanılır. Aksi takdirde, herhangi bir satırda ***break*** komutuna rastlanıncaya kadar sonraki ***case*** değerleri ile karşılaştırma yapılmadan o satırlara ait olan tüm komutlar icra edilir.

Döngü Kontrol Ifadeleri

- ***Continue*** komutu, döngü bloğuna ait diğer komutların atlanmasını ve programın icrasının döngünün bir sonraki çevriminden devam etmesini sağlar. Genellikle bazı koşulların oluşması sonucunda döngünün diğer deyimlerinin icra edilmemesi gereken veya edilmesinin gereksiz olduğu ve döngünün tekrar icrasının gerçekleştiği (kontrol ifadesine bağlı olarak) durumlarda kullanılır.
- ***For, while*** veya ***do-while*** komutları ile oluşan bloklar içinde yer alabilir.

Döngü Kontrol İfadeleri

if-break

```
#include <stdio.h>
int main()
{
for(int i = 0; i < 5; i++)
{
 if(i == 2){
break;
}
printf("Kod Bloklari\n");
}
printf("Dongu sonlandi ve program dongunun sonundan devam ediyor.");
}
```

while-break

```
#include <stdio.h>
int main ( ){
int sayi=0;
while ( sayi < 20 ){
printf("%d \n",sayi);
if ( sayi == 10 ){
break ;
}
sayi ++;
}
return 0 ;
}
```

Döngü Kontrol İfadeleri

if-continue

```
#include <stdio.h>
main()
{
for(int i = 0; i < 5; i++)
{
if(i == 2){
continue;
}
printf("KOD BLOKLARI\n");
}
}
```

if-continue

```
#include <stdio.h>
main()
{
int toplam = 0;
for(int i = 1; i < 10; i++)
{
if(i == 4){
continue;
}
toplam += i;
}
printf("Toplam = %d",toplam);
}
```

İç içe Döngüler

Döngüleri iç içe kullanabiliriz.

Çoğu programda bu durum gerekmektedir.

Örnek: Çarpım tablosunu yazdıralım.

```
#include <stdio.h>
int main()
{
int i, j;
for (i=1; i <= 10; i++){
printf("i: %d: ",i);
for (j=1; j <= 10; j++){
printf("%5d" , i*j);
} /* end-for-içerdeki */
printf("\n");
} /* end-for-dışardaki */
}
```

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

EEM165 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 6

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

Diziler

- Bilgisayarlar yardımıyla yapılan işlemlerde, çok sayıda veri girilmesi ve girilen verilerin işlenerek belirli bir sistematiğe göre sıralanması gerekebilir.Belirli bir düzende olan verilerin işlenmesi hem daha kolay hem de daha pratiktir.
- Bu nedenle bilgisayar programlarında çoklu verileri işlemek için "dizi" olarak adlandırılan sıralı veri alanları kullanılır.
- –dizi elemanların bellekte (program çalıştığı sürece) sürekli biçimde bulunması
- –dizi elemanların aynı türden değişkenler olması
- Tek isimle adlandırılan bu veri alanları belleğe ardışık olarak yerleşirler.

Diziler

- Aynı amaç için birden fazla aynı tip değişkene ihtiyacımız olur.
- Örneğin,100 kişilik bir sınıfın "Programlama Dilleri" dersinden aldığı yılsonu notlarını tek tek değişkenlere aktarmak yerine (100 tane değişken adı gerekli), bunları ‘Notlar’ isimli bir dizide tutulabilir.
- Bu şekilde birçok değişken adı ve alanı kullanımına gerek kalmaz.
- Bilgiler tek bir isimle belirli bir yapı altında tutulur ve hızlı bir şekilde işlenirler.

Diziler

- Birden fazla aynı tip değişkeni bir arada tutan veri yapısıdır.
- En basit tipi bir boyutlu olanıdır. Bir boyutlu dizinin elemanları bir satırda bir biri ardına dizilmiş şekilde kabul edilir.
- Basit Kodlama:

```
float kutle[5]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
int maliyet[3] = { 25, 72, 94 };
double a[4] = { 10.0, 5.2, 7.5, 0.0};
```

Diziler

- Dizinin n. elemanı c[n-1]ile gösterilir.
- c[0]+ c[1]+ c[2]+…….c[n-1]
- Dizi elemanları normal değişkenler gibidir.
- c[0] = 3;
- printf(“%d”, c[0]);
- İndis numarası üzerinde işlemler gerçekleştirilebilir.
- a= 2, b=3 ise c[a+b]+= 8; // c[5] eleman değerine 8 ekler.
- 4. Dizinin ilk üç elemanının değerleri toplamını yazdırmak içinprintf(“%d”, c[0]+c[1]+c[2]);

Dizilere Değer Atama

- Dizilere tanımlama sırasında ilk değer atanabilir.

```
int A[10]={8, 4, 10, 2, 5, 6, 7, 8, 9, 4};
```

- Eğer ilk değerler dizinin eleman sayısından az ise kalan elemanların değeri 0 olur.

```
int A[10]={1, 2, 3, 4};/* A[10] dizisinin ilk değerleri{1, 2, 3, 4, 0, 0, 0, 0, 0, 0}*/
```

- Eğer ilk değerlerle bir dizi tanımlıyorsak, dizinin boyutunu boş bırakabiliriz.

```
int A[]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
/* A dizisinin10 elemanı var. A[0]..A[9] */
```

- Bir dizinin uzunluğu belirtilmeden de başlangıç değeri atamak mümkündür.

```
int a[] = { 100, 200, 300, 400 };
float v[] = { 9.8, 11.0, 7.5, 0.0, 12.5};
```

Diziler

- Dizi elemanlarının başlangıç değerleri otomatik olarak sıfır olmaz. Bunun için en azından ilk eleman değeri sıfır yapılmalıdır.
- int n[5] = {0}; // tüm elemanların değeri 0 olur
- Eğer gereğinden fazla başlangıç değeri varsa hata oluşur.
- int n[5] = {1, 2, 3, 4, 5, 6}; //altı adet başlangıç değeri

Dizi Kullanımı

- Dizinin elemanlarına ulaşılırken genelde döngüler kullanılır, ve döngünün her iterasyonunda dizinin bir elemanı üzerinde çalışılır.
- En sık kullanılan döngü *for* döngüsüdür. Çünkü döngü ifadesinde açıkça hem ilk değer atamaları hem de indeks değişkeni kullanılabilmektedir:

```
notlar[0] = 0;
notlar[1] = 0;
notlar[2] = 0;
notlar[3] = 0;
notlar[4] = 0;
```

→

```
int i;
for(i = 0; i < MAX_OGR_SAYISI; i++)
        notlar[i] = 0;
```

ÖRNEKLER

❖ ***Örnek1:***

a[10] = {25, 18, 20, 0, 29, 5, 4, 8,19,13} dizisinin 2. ve 8. elemanlarının toplamını ekrana yazdıran programı C dilinde yazınız.

```
#include<stdio.h>
#include<conio.h>
int main(){
int a[10] = {25, 18, 20, 0, 29, 5, 4, 8,19,13};
a[7]= a[7] + a[1];
printf("%d",a[7]);
getch();return 0;}
```

ÖRNEKLER

❖ **Örnek 2:**

Klavyeden girilen pozitif bir N tamsayısına kadar olan sayıları bir diziye yazan ve diziden okuyarak ekrana listeleyen C programını yazınız.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int i,N;
        printf("Bir N degeri giriniz:");
        scanf("%d",&N);
        int dizi [N];
        for (i=1;i<=N;i++){
                dizi[i-1]=i;
        }
        for(i=1;i<=N;i++){
                printf("%d",dizi[i]);
        }
        return 0;
}
```

ÖRNEKLER

❖ ***Örnek 3:***

Klavyeden girilen pozitif bir N tamsayısına kadar olan sayıları bir diziye yazan ve diziden okuyarak sayıların ***karesini*** *ekrana listeleyen program kodunu yazınız.*

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int i,N;
        printf("Bir N degeri giriniz:");
        scanf("%d",&N);
        int dizi[N];
        for (i=1;i<=N;i++){
                dizi[i]=i;
        }
        for (i=1;i<=N;i++){

        printf("%d=%d\n",i,dizi[i]*dizi[i]);
        }
        return 0;
}
```

ÖRNEKLER

❖ **Örnek 4:**

Kullanıcı tarafından girilen birbirinden farklı 10 adet pozitif tam sayı arasından en büyük olanı bulmaya yarayan C programını yazınız.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int i,dizi[10],enbuyuk;
        for(i=0;i<10;i++){
                printf("%d.sayiyi giriniz:",i+1);
                scanf("%d",&dizi[i]);
        }
        enbuyuk=dizi[0];
        for(i=1;i<10;i++){
                if (dizi[i]>enbuyuk){

        enbuyuk=dizi[i];
                }
        }
        printf("En buyuk deger=%d\n",enbuyuk);
        return 0;
}
```

ÖRNEKLER

❖ **Örnek 5:**

Kullanıcı tarafından girilen birbirinden farklı 10 adet pozitif tam sayı içerisinden

istenilen bir sayının dizinin kaıçıncı sırasında olduğunu bulmaya yarayan C programını yazınız

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

int main(){
        int i, dizi[10],aranan;
        for(i=0;i<10;i++){
                printf("%d.sayiyi giriniz:",i+1);
                scanf("%d", &dizi[i]);
        }
        printf("\n Aranan degeri giriniz:");
        scanf("%d",&aranan);
        for(i=0;i<10;i++){
                if(dizi[i]==aranan){
        printf("Aranan deger dizinin %d.sirasinda bulundu.",i+1);
                }
        }
        return 0;
}
```

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr.Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

EEM165 BİLGİSAYAR PROGRAMLAMA I

Ders Notu 7

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

ÖRNEKLER (KOŞUL İFADELERİ-DÖNGÜLER)

❖ ***Örnek 1 :*** *Klavyeden girilen pozitif bir tam sayının faktoriyelini hesaplayan C programını yazınız.*

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int sayi,i;
 int sonuc=1;
        printf("Bir sayi giriniz:");
        scanf("%d",&sayi);
        for(i=1;i<=sayi;i++){
                sonuc=sonuc*i;
        }
        printf("%d!=%ld",sayi,sonuc);
        return 0;
}
```

ÖRNEKLER (KOŞUL İFADELERİ-DÖNGÜLER)

❖ ***Örnek 2 :*** *Klavyeden girilen alt ve üst sınır değerleri için, bu sınır aralıklarında kalan çift sayıları yazdıran programın kodunu C dilinde yazınız.*

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int i, alt,ust;
        printf("Alt siniri giriniz:");
        scanf("%d",&alt);
        printf("Ust siniri giriniz:");
        scanf("%d",&ust);
        for (i=alt;i<=ust;i++){
                if(i% 2==0){

        printf("%d\t",i);
                }
        }
        return 0;
}
```

ÖRNEKLER (KOŞUL İFADELERİ-DÖNGÜLER)

- ***Örnek 3 :*** *Klavyeden taban ve üs değerleri girilen bir sayinin kuvvetini hesaplayan C programını yazınız.*
- *(pow ( )komutu kullanmayınız)*

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int taban,us,i;
        long int sonuc=1;
        printf("Taban degerini giriniz:");
        scanf("%d",&taban);
        printf("Us degerini giriniz:");
        scanf("%d",&us);
        for(i=1;i<=us;i++){
        sonuc=sonuc*taban;
        }
        printf("%d^%d=%ld\n",taban,us,sonuc);
        return 0;
}
```

ÖRNEKLER

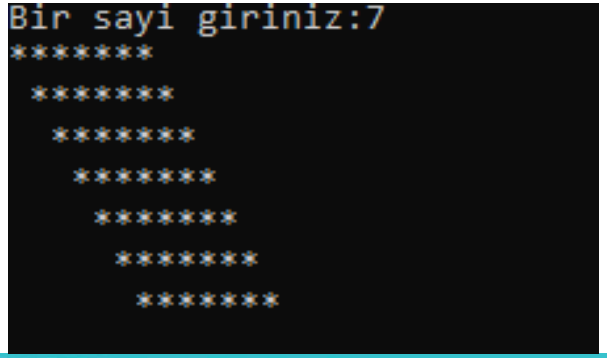
❖ ***Örnek 4 :*** *Kullanıcı tarafından girilen satır adeti kadar Floyd Üçgeninin elde edilmesini sağlayan C programını döngü kullanarak yazınız*

```
Floyd ucgeni icin satir sayisini giriniz:5
1
23
456
78910
1112131415
```

```
#include <stdio.h>
#include <stdlib.h>
int main(){
        int satir,i,j,sayac=1;
        printf("Floyd ucgeni icin satir sayisini giriniz:");
        scanf("%d",&satir);
        for(i=1;i<=satir;i++){
                for(j=1;j<=i;j++){
                    printf("%d",sayac);
                    sayac++;
                }
                printf("\n");
        }
        return 0;
}
```

ÖRNEKLER

❖ ***Örnek 5 :*** *Aşağıda ekran görüntüsü verilen şekildeki gibi klavyeden girilen bir sayı adeti kadar ekrana * karakteri bastıran C programını yazınız.*




```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int sayi,i,j,k;
        printf("Bir sayi giriniz:");
        scanf("%d",&sayi);
        for(i=0;i<sayi;i++){
                for(j=0;j<i;j++){
                printf(" ");
                }
                for(k=0;k<sayi;k++){
                printf("*");
                }
                printf("\n");
                }
        return 0;
        }
```

ÖRNEKLER

- ***Örnek 6 :*** *Aşağıda verilen matematiksel işlemin sonucunun* ***do-while*** *döngü yapısını kullanılarak elde edilmesini ve elde edilen toplam sonucunun ekrana gösterilmesini sağlayan C programını yazınız.*
- *N ve M değerleri kullanıcı tarafından girilecektir.*

$$\sum_{\text{İ}=1}^{N}\sum_{J=1}^{M}(i^2+2ij+j^2)$$

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int i,j,N,M,toplam=0;
        printf("N sayisini giriniz:"); scanf("%d",&N);
        printf("M sayisini giriniz:"); scanf("%d",&M);
        i=1;
        do{
                j=1;
                do{
toplam=toplam+((i*i)+(2*i*j)+(j*j));
                        j++; }
                while(j<=M);
                i++;
        }
        while(i<=N);
        printf("Sonuc=%d",toplam);
        return 0;
                }
```

DÜZCE ÜNİVERSİTESİ | MÜHENDİSLİK FAKÜLTESİ

TEŞEKKÜRLER…

Dr. Öğr. Üyesi Enes KAYMAZ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ