

# Bilgisayar Programlama

## BM101

---

Prof. Dr. İskender Öksüz  
Araştırma Görevlisi Kiraz Candan Herdem

Ders kitabı: **C Programlama Dili** SİSTEM YAYINCILIK Brian W.  
Kernighan/ Dennis M. Ritchie

# İnternet kaynakları

---

- <http://web.inonu.edu.tr/~mkarakaplan/ckitabi.pdf>
- [http://www.acm.uiuc.edu/webmonkeys/book/c\\_guide/index.html](http://www.acm.uiuc.edu/webmonkeys/book/c_guide/index.html)
- [http://www.physics.drexel.edu/courses/CompPhys/General/C\\_basics/c\\_tutorial.html](http://www.physics.drexel.edu/courses/CompPhys/General/C_basics/c_tutorial.html)
- <http://www.eskimo.com/~scs/cclass/>
- [http://publications.gbdirect.co.uk/c\\_book/](http://publications.gbdirect.co.uk/c_book/)

# 1 Bilgisayar programlama:

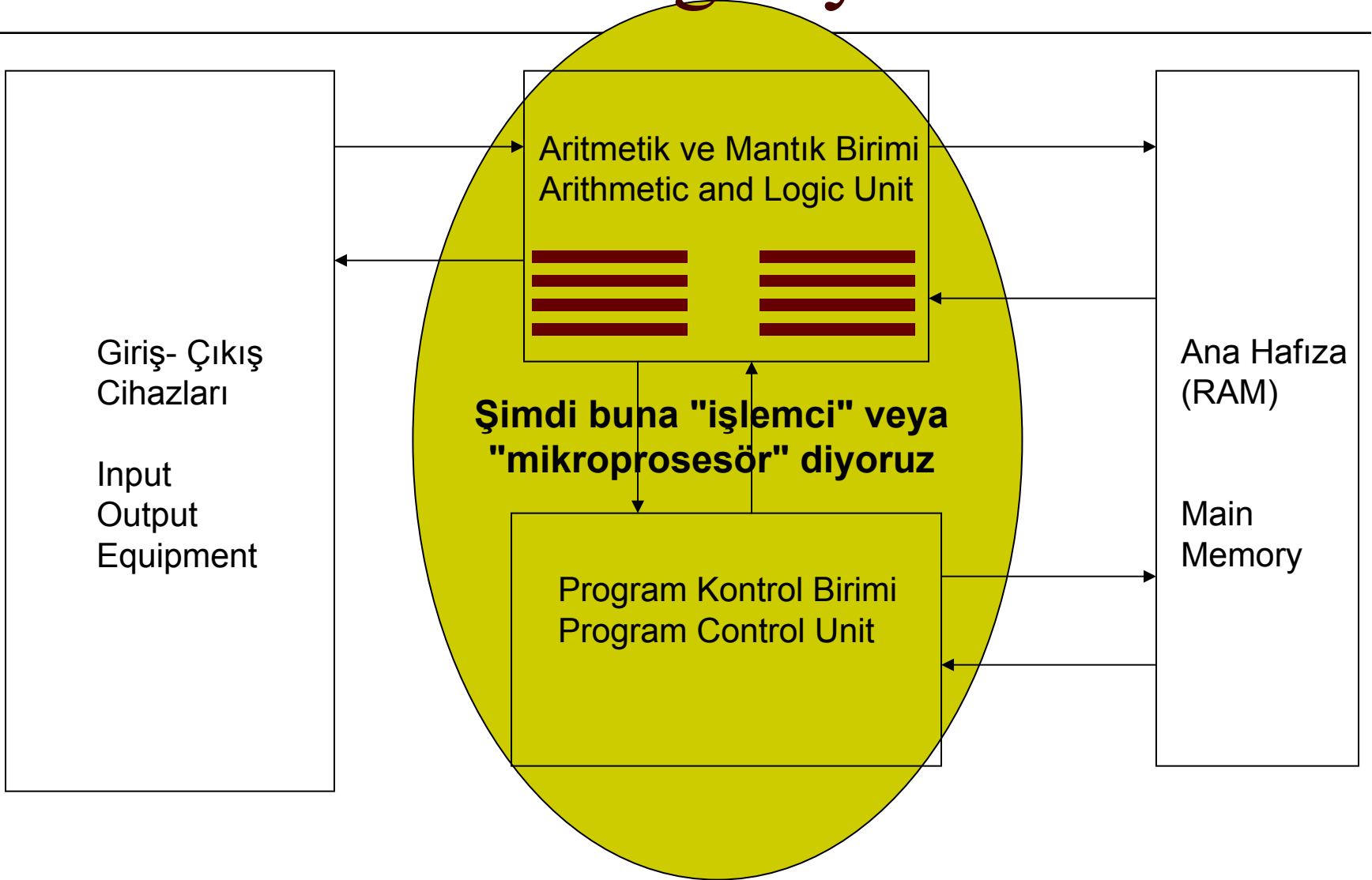
---

Programlama dilleri, makine dili, yüksek düzey diller

Derleyici, "source", "object", "link" kavramları.

Algoritma, bir programın hayat hikâyesi

# Von Neumann bilgisayarı



# Makine dili

---

$$X = A * B + C$$

- 1) Hafızanın A adresindeki değeri al, ALU'da bir "register"e yükle;
- 2) Hafızanın B adresindeki değeri al, yukardaki değerle çarp ve sonucu aynı "register"de tut.
- 3) Hafızanın C adresindeki değeri registerdeki değere ekle.
- 4) Registerdeki değeri hafızanın X adresine koy.

# Makine dili

---

$$X = A * B + C$$

	Opcode	Operand		
1)	<b>0001000000000000000000001000000000</b>	(A'daki değeri reg. yükle)		
	Hex: 08	00	04	00
2)	<b>0010010000000000000000001000000001</b>	(B'deki değerle çarp)		
	Hex: 24	00	04	01
3)	<b>0010001100000000000000001000000010</b>	(C'deki değeri reg. ile topla)		
	Hex: 23	00	04	02
4)	<b>0001000100000000000000001000000011</b>	(Registerdeki değeri X'e taşı)		
	Hex: 11	00	04	03

İki tabanındaki sayılar : 0, 1

Ondalık sayılar : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Onaltılık (hex) sayılar : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

# Biraz daha hex

---

Ondalık → Hex

1 =

10 =

36 =

147 =

255 =

Hex → Ondalık

A5 =

DD =

# Makine dili - Assembler

---

$$X = A * B + C$$

08	00	04	00
24	00	04	01
23	00	04	02
11	00	04	03





# Yüksek seviye dilleri?

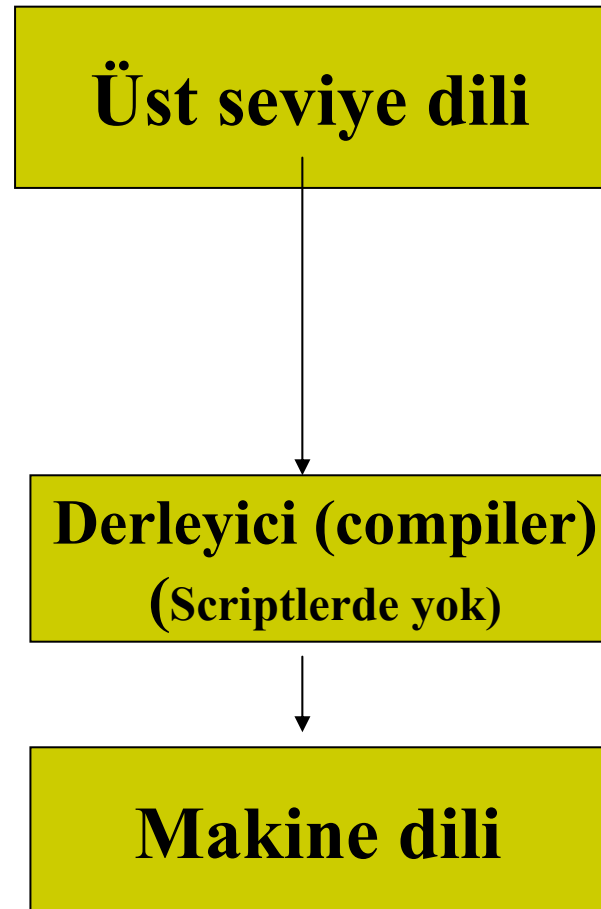
---

$$X = A * B + C$$

# Üst seviye dili

---

- FORTRAN
- C
- Pascal
- Basic
- C++
- Java
- C#



- 
- <http://www.digibarn.com/collections/posters/tongues/tongues.jpg>

# Derleme (Compilation)

Kaynak kod  
(Source code)  
Source module

**Üst seviye dili**

$X = A * B + C$

Ara dil (Java, C#)

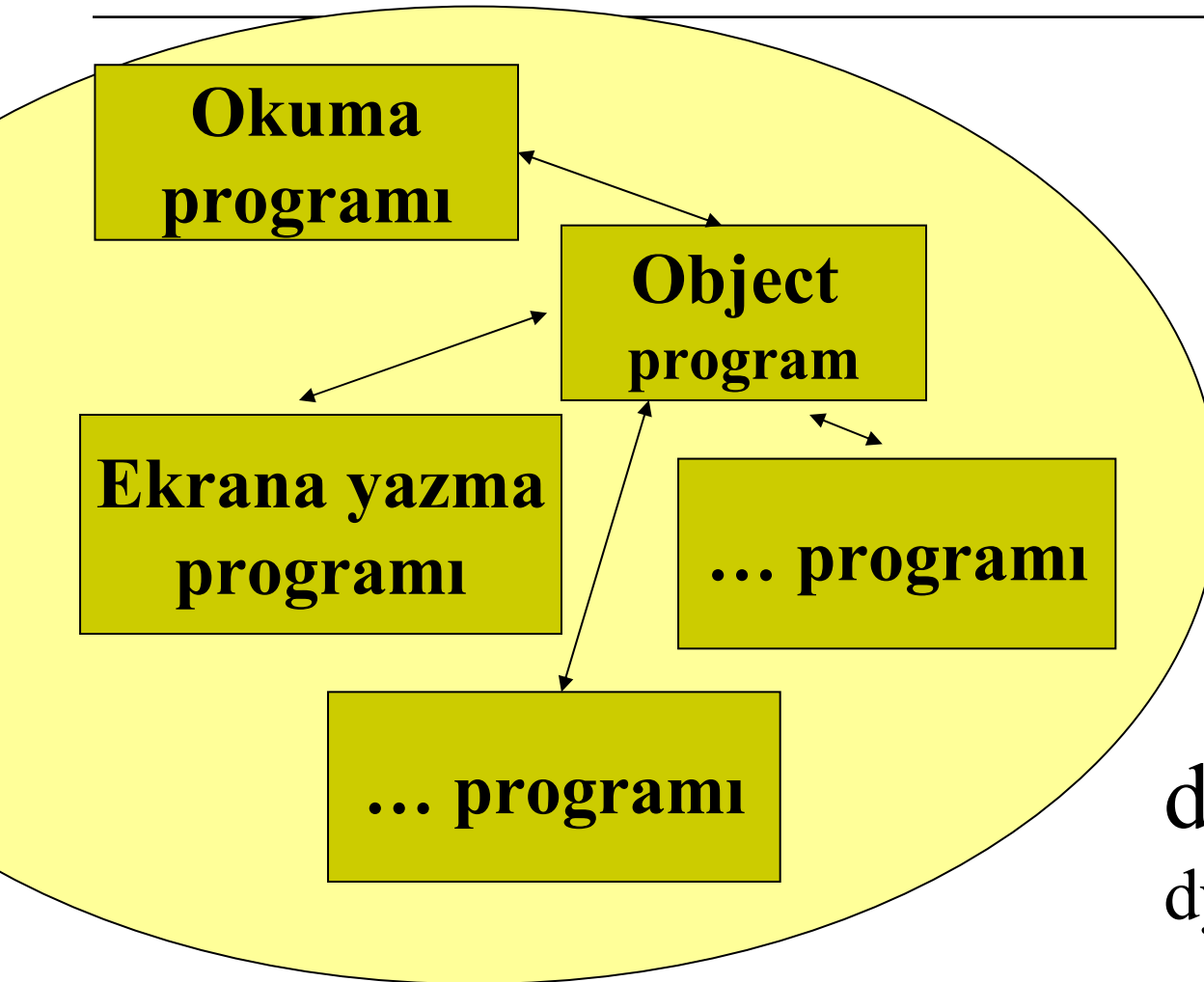
**Derleyici (compiler)**  
(Scriptlerde yok)

Object code  
Object module

**Makine dili**

08	00	04	00
24	00	04	01
23	00	04	02
11	00	04	03

# Bitti mi?



Bağlama- link

İcra edilebilir  
program:  
Executable

Çarp\_Topla.exe

dll :  
dynamic link library

# Merhaba dünya! (Konsol)

---

```
#include <stdio.h>

void main() {
    printf("\nHello world\n");
}
```

# Biraz daha ustalaşalım

```
#include <stdio.h>
void main() {
    int a, b, c, x;
    printf("\nBir sayi yazin\n");
    x = scanf("%d", &a);
    printf("\nBir sayi daha...\n");
    scanf("%d", &b);
    printf("\nVallahi bu son...\n");
    scanf("%d", &c);
    x = a * b + c;
    printf("\nSonuc %d\n", x);
}
```

# Hatalar:

---

- Derleme hataları (Compiler error)
- Bağlama hataları (Falan modülü bulamadım)
- İcra hataları (Run time error)



# 2 Sabitler, deęişkenler, operatörler

---

- Sabit, bir deęerdir:

A 129 3.14 gibi

- Deęişken, sabitin depo edildięi hafıza adresinin ismidir.

- bizimHarf kacKisi piSayisi gibi

Farklı cins sabitler temelde farklı özelliklere sahip olduklarından hafızada tuttukları yer ve onlara yapılan muamele de farklıdır.

# Sabitler

---

Sabitler temelde farklı özelliklere sahip olduklarından hafızada tuttukları yer ve her birine yapılan muamele de farklıdır:

Hafızada nasıl saklarsınız? Nasıl yerleştirirsiniz?

129 gibi bir tam sayı?

İki veya dört byte (16 bit- 32 bit) halinde saklanır  
içinde 00 81 değeri vardır.

A harfini nasıl saklarsınız? Kodu = 65

Bir byte yeter. İçine 65 koyarsınız. veya 41 hex: 0X41

3.14159?


İki tane tam sayı halinde saklanır.


içinde 314 ve -2 değerleri vardır:  $314 \cdot 10^{-2}$  - 4 byte, 8 byte, 16 byte

# Tanımlayıcılar (değişkenler)

---

Değerleri (sabitleri) hafızaya nasıl koyacağımızı biliyoruz ama bunları kullanmak için adreslerini bilmemiz, yani ad vermemiz lâzım:

129            kacKisi

A            bizimHarf

3.14159            piSayisi

O halde kaç tip sabit varsa, o kadar cins de tanımlayıcı (değişken) veri tipi olmalı.

# Veri tipleri

---

## Orjinal C

- int
- char
- float
- double
- void

## ANSI C

- signed char
- unsigned char
- int
- unsigned int
- signed int
- short int
- unsigned short int
- signed short int
- long int
- unsigned long int
- signed long int
- float
- double
- long double

# C# veri tipleri

Tip	Bit?	Değerler
bool	8	true – false
char	16	'\u0000' - '\uFFFF'
byte	8	0 - 255
sbyte	8	-128 - +127
short	16	-32.768 - +32.767
ushort	16	0 – 65.535
int	32	-2.147.483.648

Tip	Bit?	Değerler
uint	32	0 – 4.294.967.295
ulong	64	$2^{64} - 1$
decimal	128	$1,0 \cdot 10^{-28} - 7,9 \cdot 10^{28}$
float	32	$\pm 1,5 \cdot 10^{-45} - \pm 3,4 \cdot 10^{38}$
double	64	$\pm 5, \cdot 10^{-324} - \pm 1,7, \cdot 10^{308}$
string		
object		



# signed char

---

127

0 1 1 1 1 1 1 1

-127

1 1 1 1 1 1 1 1



# signed short int

---

32 767

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

-32 767

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

65 534?

# Değişkenlere isim verelim

---

kacKisi, bizimHarf, piSayisi

int kacKisi;

char bizimHarf;

float piSayisi;

double piSayisi;



# Değişkenler: İsim verme kuralları

---

Harfler, rakamlar, \_

Rakamla başlanmaz

\_ ile başlamamanız tavsiye edilir

İlk 31 karakter anlamlıdır...

senden\_bilirim\_yok\_bana\_bir\_faide\_ey\_gul  
senden\_bilirim\_yok\_bana\_bir\_faide\_ey\_diploma

aynı değişkendir.


Küçük ve büyük harf fark eder: neHaber  NeHaber farklıdır.

## Tavsiyeler:

camelCasing

intKacKisi

Kısaltma kullanmamaya gayret edin...



---

Bizim "değişken" dediğimize  
aslında  
"tanımlayıcı" (identifier)  
deniyor.

# Tanımlayıcıların ne cins olduğunu derleyiciye nasıl bildiririz?

Aslında bunu yaptık:

```
int kacKisi;
```

```
char bizimHarf;
```

```
float piSayisi;
```

```
double piSayisi;
```

Bunlara tip tarif ifadeleri ("type definition statements") deniyor.

Tip tarif ifadeleri en tepede bulunur. (Neyin "en tepesinde" bulunduğunu sonra göreceğiz.)

Tanımlayıcılara tarif sırasında ilk değerleri verilebilir- verilmeye de bilir.

---

- `int kacKisi = 52;`
- `char bizimHarf;`

Birden fazla tanımlayıcı bir tarifte verilebilir:

```
int kacKisi, toplam, aAlanlar= 8, hocaSayisi;
```

# Atama komutu:

---

□ `kacKisi = 234;`

234 sayısını `kacKisi` adlı tanımlayıcının içine sokuyor.

Aslında "`kacKisi`"i RAM'de içinde int tutacak şekilde ayrılmış bir adresin adıdır.

Bu komut, 234 sayısını o adrese sok demektir.

□ Bazı dillerde atama sola yönelmiş okla gösterilir:  
`kacKisi <= 234` gibi... Bazılarında `kacKisi := 234`

# Atama komutu (genel gramer)

---

- *tanımlayıcı = ifade;*
- $kacKisi = gelenler + salondakiler;$
- $X = A * B + C;$

*ifade*: tanımlayıcılar ve operatörler.

$*$ ,  $+$ , v. s. operatörlerdir.

int

int birinciSayi;

birinciSayi = 1256;

tanımlayıcı

sabit

atama

char

char bizimHarf;

bizimHarf = 'A'

tanımlayıcı

sabit

atama

Atama ifadesi: Assignment statement

---

=

**identifier = expression**



# Atama ifadesi

---

Tip ilanları (type declarations), aslında hafızada (RAM'da) yer ayırır.

**double** xKoordinati, yKoordinati;

**int** sayi, terim;

xKoordinati	?	
ykoordinati	?	
sayi	?	
terim	?	

# Atama ifadesi

---

xKoordinati = 5.23

yKoordinati = 5.0

sayi = 17

terim = sayi / 3 + 2

xKoordinati = 2.0 \* yKoordinati

xKoordinati	<b>10.46</b>
yKoordinati	<b>5.0</b>
sayi	<b>17</b>
terim	<b>7</b>

# Atama ifadesi

---

- $A = B$  ne demek?
- $A = B$  icra edildiği anda
- $A$  ile  $B$ 'yi yer değiştirmek istersek?
- $C = A$
- $A = B$
- $B = C$  icra edilir

A	-18.567
B	-18.567

A	-18.567
B	122.45
C	122.45

# Atama ifadesi

---

- $A = A + B$ ; ne demek?

Böyle bir ifade acemi programcuyu şaşırtabilir.

- $A = A + B$ ; icra edildiği anda...
- Eşdeğer:  
 $A += B$ ;

A	103.883
B	-18.567

# Operatörler

Aritmetik operatörleri		Mantık operatörleri	
Operatör	C, C++, C# ifadesi	Operatör	C, C++, C# ifadesi
+	$f + 7$	==	$x == y$
-	$p - c$	!=	$x != y$
*	$b * m$	>	$x > y$
/	$x / y$	<	$x < y$
%	$x \% S$	>=	$x >= y$
		<=	$x <= y$

# Operatörler devam...

---

Operatör	C# ifadesi
&&	bool a && bool b
	bool a    bool b
++	a++ veya ++a
--	a-- veya --a
<< >> &	bitlerle oynama



Not:

---

= ile diğer operatörler  
birleştirilebilir:

$a = a + b$  yerine  $a += b$

$a = a / b$  yerine  $a /= b$

# Diğer önemli işaretler

- ( ve ) işlemlerde öncelik değiştirmek için.
- { ve } program bloklarını ayırmak için
- //  
x = y + z      /\* y ile z'yi topluyorum \*/
- /\* ve \*/
- /\*  
y ile  
z'yi topluyorum  
\*/

Boşluk!

En önemli işaret: a = x + y



# Operatör öncelikleri (operator precedence)

---

- Önce parantez içleri
- Sonra soldan sağa  $*$  ve  $/$  ve  $\%$  işlemleri
- Sonra soldan sağa  $+$  ve  $-$  işlemleri

# • İşlem Öncelikleri (Örnek)

$$y = 2 * 5 * 5 + 3 * 5 + 7;$$

1. Adım  $2 * 5 = 10$  (En soldaki çarpma)

$$y = 10 * 5 + 3 * 5 + 7;$$

2. Adım  $10 * 5 = 50$  (En soldaki çarpma)

$$y = 50 + 3 * 5 + 7;$$

3. Adım  $3 * 5 = 15$  (Toplamadan önceki çarpma)

$$y = 50 + 15 + 7;$$

4. Adım  $50 + 15 = 65$  (En soldaki toplama)

$$y = 65 + 7;$$

5. Adım  $65 + 7 = 72$  (Son toplama)

6. Adım  $y = 72;$  (Son işlem 72 değeri y değişkenine aktarılır)

Biraz daha ustalaşalım:

$ax^2 + bx + c = 0$  denkleminin çözümü

İki kök var. Ama önce diskriminantı hesaplamamız lâzım: -

$$\Delta = b^2 - 4ac$$

$$x_1 = (-b + \sqrt{\Delta}) / (2a)$$

$$x_2 = (-b - \sqrt{\Delta}) / (2a)$$

$\Delta > 0$  ise her şey yolunda....

$\Delta = 0$  ise  $x_1 = x_2$  ...tadından yenmez.


$\Delta < 0$  ise battık: ☹

$$x_1 = (-b + i \sqrt{-\Delta}) / (2a) \dots$$

$$x_{1R} = -b/(2a), x_{1I} = \sqrt{-\Delta} / (2a)$$

$$x_2 = (-b - i \sqrt{-\Delta}) / (2a), x_{2R} = x_{1R}, x_{2I} = -x_{1I}$$

```
#include <stdio.h>
#include <math.h>
void main() {
    float a, b, c, delta, x1, x2;
    printf("a, b, c degerlerini girin\n");
    scanf("%f %f %f", &a, &b, &c);
    delta = b * b - 4.0 * a * c;
    delta = sqrt(delta);
    x1 = (-b + delta) / (2.0 * a);
    x2 = (-b - delta) / (2.0 * a);
    printf("\nx1 = %f , x2 = %f\n", x1, x2);
}
```



Her türlü  $a$ ,  $b$ ,  $c$  ile  
çalışacak bir ikinci  
derece programı  
yazmak!...

# 3 Akış kontrolü

---

if

```
float a, b, c, delta, x1R, x1I=0.0, x2R, x2I=0.0;
```

```
delta = b * b - 4.0 * a * c;
```

```
if(delta <= 0) goto hapiyuttuk;
```

```
/* Aynen daha önceki gibi */
```

```
goto yazma;
```

```
hapiyuttuk:
```

```
if(delta < 0) goto sanal;
```

```
x1R = -b/ (2.0 * a); x2R = x1R;
```

```
goto yazma;
```

```
sanal:
```

```
delta = sqrt(-delta);
```

```
x1R = -b/ (2.0 * a); x2R = x1R;
```

```
x1I = sqrt(delta) / (2.0 * a); x2I = -x1I;
```

```
yazma: printf("\nx1R = %f x1I = %f x2R = %f\n x2I = %f\n"
```

```
x1R, x1I, x2R, x2I);
```

# Structured programming ~ 1970

---

- goto **yassak!**
- komut etiketleri **yassak!**
- program bloklarını belli edecek şekilde içerden başlayınız.



```
if(delta <= 0) {  
    /* Aynen daha önceki gibi */  
    printf("\nx1R = %f x1I = %f x2R = -----);  
}
```

```
else if(delta == 0){  
    x1R = -b/ (2.0 * a); x2R = x1R;  
    printf("\nx1R = %f x1I = %f x2R = -----);  
}
```

```
else{  
    delta = sqrt(-delta);  
    x1R = -b/ (2.0 * a); x2R = x1R;  
    x1I = sqrt(delta) / (2.0 * a); x2I = -x1I;  
    printf("\nx1R = %f x1I = %f x2R = -----);  
}
```

# if çeşitleri

---

- *if( mantık\_ifadesi ) ifade;*
- *if( mantık\_ifadesi ) {*  
*ifade*  
*ifade*  
*.....*  
*ifade*  
*}*

# else

```
if (a > b){  
    printf (“a, b’den büyük”);  
}  
else{  
    printf (“a, b’den büyük değil”);  
}
```

```
if (a > b){  
    printf( “a, b’den büyük”);  
}  
else if(a == b){  
    printf(“a, b’ye eşit”);  
}  
else{  
    printf (“a, b’den küçük”);  
}
```

# En sık kullanılan karmaşık if yapısı

---

```
if(.....){
    ....
    ....
}
else if(.....){
    ....
}
else if(.....){
    ....
}
else if(.....){
    ....
}

.....
else if(.....){
    ....
}
else{
    ....
}
}
```

# Not verme programı

---

100-90      A

89- 80      B

79- 70      C

69- 60      D

59- 0        F

100-90 A; 89- 80 B;79- 70 C; 69- 60 D; 59- 0 F

---

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

```
void main() {
```

```
    double not;
```

```
    scanf("%lf", &not);
```

```
    if(not > 89) printf("\nNotunuz A - tebrikler!\n");
```

```
    else if (not > 79.0) printf ("\nNotunuz B- idare eder!\n");
```

```
    else if (not > 69.0) printf ("\nNotunuz C- vasat olmak icin mi dunyaya geldin?\n");
```

```
    else if (not > 59.0) printf ("\nNotunuz D - vaaaah vaaaah!\n");
```

```
        else printf( "\nNotunuz F- seneye goruselim...\n");
```

```
}
```

# Bunun tersi: Harf verince notun nerede olduğunu söylesin:

---

```
#include <stdio.h>
void main() {
    char harf;
    scanf("%c", &harf);
    if(harf == 'a' || harf == 'A') printf("\Not 90'in ustunde\n");
    else if (harf == 'b' || harf == 'B') printf ("\Not 80- 89 arasinda\n");
    else if (harf == 'c' || harf == 'C') printf ("\Not 70- 79 arasinda\n");
    else if (harf == 'd' || harf == 'D') printf ("\Not 60- 69 arasinda\n");
    else if (harf == 'f' || harf == 'F') printf ("\Not 60'in altinda\n");
    else printf( "\nBoyle bir not yok.\n");
}
```

# Aynı işi başka türlü yapalım:

```
char harf;
scanf("%c", &harf);
switch(harf){
    case 'a':
        printf("\nNot 90'in ustunde\n");
        break;
    case 'B':
        printf ("\nNot 80- 89 arasinda\n");
        break;
    case 'C':
        printf ("\nNot 70- 79 arasinda\n");
        break;
    case 'D':
        printf ("\nNot 60- 69 arasinda\n");
        break;
    case 'F':
        printf ("\nNot 60'in altinda\n");
        break;
    default:
        printf( "\nBoyle bir not yok.\n");
}
```



# Aynı işi...

```
char harf;
scanf("%c", &harf);
switch(harf){
    case 'a':
    case 'A':
        printf("\Not 90'in ustunde\n");
        break;
    case 'b':
    case 'B':
        printf ("\Not 80- 89 arasinda\n");
        break;
    case 'c':
    case 'C':
        printf ("\Not 70- 79 arasinda\n");
        break;
```

```
    case 'd':
    case 'D':
        printf ("\Not 60- 69 arasinda\n");
        break;
    case 'f':
    case 'F':
        printf ("\Not 60'in altinda\n");
        break;
    default:
        printf( "\nBoyle bir not yok.\n");
}
```

# Bir başka if-- if'siz if

---

`delta > 0 ? delta = sqrt(delta) : delta = sqrt(-delta);`

eşdeğer if

```
if (delta > 0) delta = sqrt (delta);  
else delta = sqrt ( - delta);
```

# Genel gramer

---

*mantık ifadesi ? doğruysa \_bu\_ atama; yanlışsa \_bu\_ atama;*

# Algoritma: İkinci derece denklem çözümü

---

$$x_{1I} = 0; x_{2I} = 0$$

a, b, c değerlerini oku

$$\Delta = b^2 - 4ac$$

$$\Delta > 0 ?$$

$$\Delta = \sqrt{\Delta}$$

$$x_{1R} = (-b + \Delta) / 2a$$

$$x_{2R} = (-b - \Delta) / 2a$$

$\Delta > 0$  şıkkının sonu

# Algoritma... devam

---

$$\Delta = 0 ?$$

$$x_{1R} = -b / 2a$$

$$x_{2R} = x_{1R}$$

$\Delta = 0$  şikkinin sonu

$$\Delta < 0$$

$$\Delta = \sqrt{-\Delta}$$

$$x_{1R} = -b / 2a; x_{1I} = \Delta / 2a$$

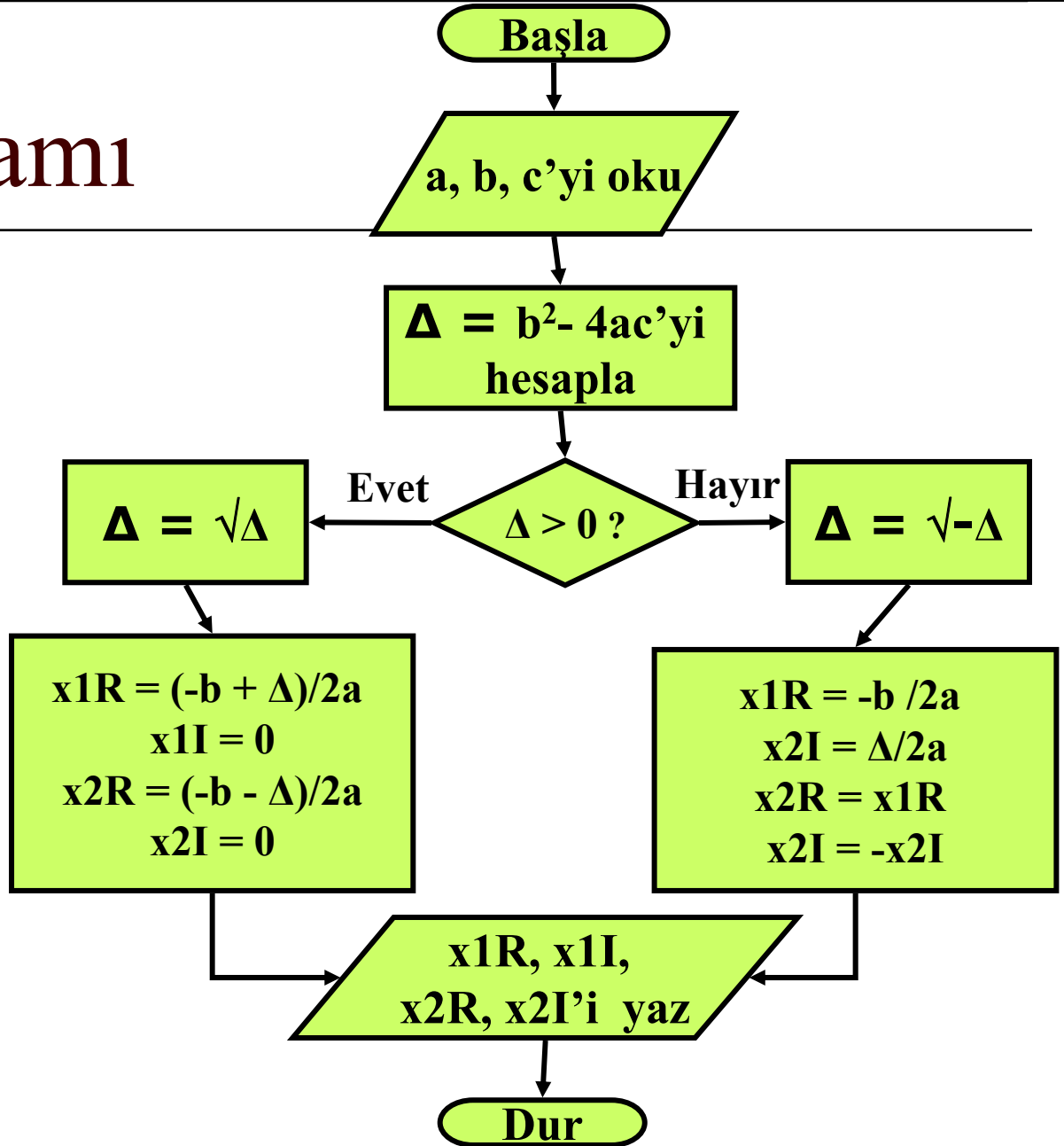
$$x_{2R} = x_{1R}; x_{2I} = -\Delta / 2a$$

$\Delta < 0$  şikkinin sonu

$x_{1R}, x_{1I}, x_{2R}, x_{2I}$  değerlerini yazdır

# Akış diyagramı

İkinci  
derece  
denklem  
akış  
diyagramı



# Ara: Algoritma ~ El Harezmi

---

Ebû Cafer Muhammed bin Musâ el- Harezmi

(Ölümü: 847)

İlk cebir kitabının yazarı:

**"Kitabı muhtasar fi hisabil- cebr ve'l mukabele"**

İlk "algoritma" fikri de onun; o yüzden adını taşıyor... Lâtinceye çevirirken H ile G'yi karıştırmışlar: H:  $\text{خ}$  G:  $\text{غ}$

# Akış diyagramının parçaları:



Algoritmanın  
başı veya sonu



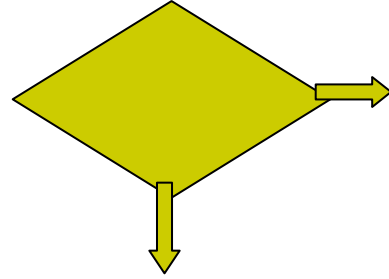
Veri girişi  
veya çıkışı



İşlem

7

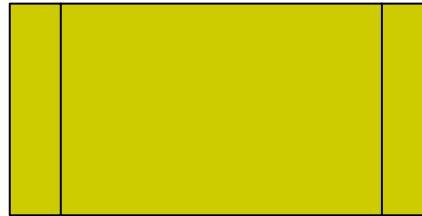
Akış başka bir yerde  
devam ediyor- oraya git



Akış bir karara göre  
ikiye (veya daha fazla  
yola) ayrılıyor: Yol  
çatallanıyor.



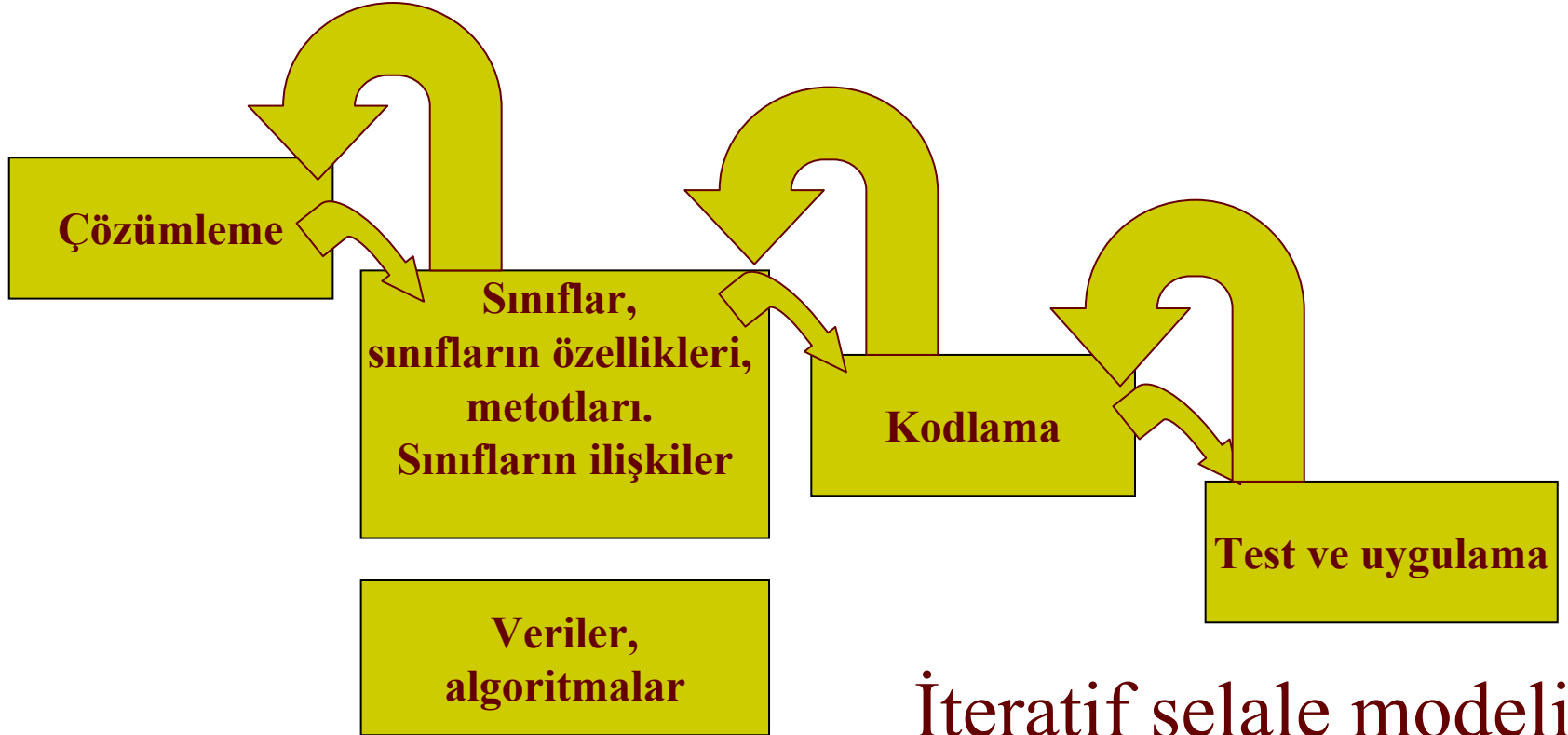
Döngü



Faydalanılan  
başka bir  
metot



# Yazılım geliştirme aşamaları



Şelale modeli

İteratif şelale modeli

Bakım (maintenance)

# Programlama paradigmaları

---

- Batch
- Prosedürel- yapılı (structured)
- Etkileşimli - olayların sürdüğü (event driven)
- Nesneye yönelik  
(Object oriented- OOP)



## 4. Akış kontrolü

---

# döngüler

# Not verme programını bütün sınıf için çalışır hale getirelim (namussuz metot)

---

```
oku: scanf("%lf", &not);
```

```
if(not < 0) exit;
```

```
    if(not > 89) printf("\nNotunuz A - tebrikler!\n");
```

```
    else if (not > 79.0) printf ("\nNotunuz B- idare eder!\n");
```

```
    else if (not > 69.0) printf ("\nNotunuz C- vasat olmak icin mi dunyaya geldin?\n");
```

```
    else if (not > 59.0) printf ("\nNotunuz D - vaaaah vaaaah!\n");
```

```
    else printf( "\nNotunuz F- seneye goruselim...\n");
```

```
goto oku;
```

# Not verme programını bütün sınıf için çalışır hale getirelim (namuslu metot)

---

```
int ogrenciSayisi, i;
printf(" Ogrenci sayisini giriniz\n ");
scanf("%d", &ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++) {
    scanf("%lf", &not);
    if(not > 89) printf("\nNotunuz A - tebrikler!\n");
    else if (not > 79.0) printf ("\nNotunuz B- idare eder!\n");
    else if (not > 69.0) printf ("\nNotunuz C- vasat olmak icin mi dunyaya geldin?\n");
    else if (not > 59.0) printf ("\nNotunuz D - vaaaah vaaaah!\n");
    else printf( "\nNotunuz F- seneye goruselim...\n");
}
```

# Bir de ortalama alalım

---

```
double ortalama = 0.0;
int ogrenciSayisi;
printf(" Ogresci sayisini giriniz\n ");
scanf("%d", &ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++){
    scanf("%lf", &not);
    ortalama += not;
    if(not > 89) printf("\nNotunuz A - tebrikler!\n");
    else if (not > 79.0) printf ("\nNotunuz B- idare eder!\n");
    else if (not > 69.0) printf ("\nNotunuz C- vasat olmak icin mi dunyaya geldin?\n");
    else if (not > 59.0) printf ("\nNotunuz D - vaaaah vaaaah!\n");
    else printf( "\nNotunuz F- seneye goruselim...\n");
}
ortalama /=ogrenciSayisi;
printf(" \nSinif ortalamasi: %lf ", ortalama);
```

# for döngüsü: Genel gramer

---

```
for(tanımlayıcı = baslangic_değeri; devam_şartı; değişim) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (i = 0; i < ogrenciSayisi; i++) {  
    scanf("%lf", &not);  
    ortalama += not;  
}
```

# for döngüsü: Misaller

---

```
for(tanımlayıcı = baslangic_değeri; devam_şartı; değişim) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (cift = 0; cift < ustLimit; cift += 2) {  
    .....  
}
```



# for döngüsü: Misaller

---

```
for(tanımlayıcı = baslangic_değeri; devam_şartı; değişim) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (geriSayim = 10; i >=0; geriSayim--){  
    printf(“\n%d\n”, geriSayim);  
}
```

# for döngüsü: Misaller

---

```
for(tanımlayıcı = baslangic_değeri; devam_şartı; değişim) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (sayi = baslangic; sayi < ustLimit; sayi +=  
    3.12159) {  
    .....  
}
```

# for döngüsü: Misaller

---

```
for(tanımlayıcı = baslangic_değeri; devam_şartı; değişim) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (;;;){  
    .....  
}  
/* Bu bir sonsuz döngüdür*/
```

# while döngüsü: Genel gramer

```
while(devam_sartı) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (i = 0; i < ogrenciSayisi; i++){  
    scanf("%lf", &not);  
    ortalama += not;  
}
```

```
i = 0;  
while (i < ogrenciSayisi) {  
    scanf("%lf", &not);  
    ortalama += not;  
    i++;  
}
```

# while döngüsü: Misaller

```
while(devam_şartı) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
for (cift = 0; cift < ustLimit; cift += 2) {  
    .....  
}
```

```
cift = 0;  
while ( cift < ustlimit) {  
    .....  
    cift = cift + 2;  
}
```

# Bir başka while

---

```
while(devam_şartı) {  
    döngüde yerine getirilecek ifadeler  
}
```

```
do {  
    döngüde yerine getirilecek ifadeler  
} while(devam_şartı);
```

# break

---

break: Programı içinde bulunduğu { } blokundan çıkarır.

```
for( ; ; ){  
    oku: scanf("%lf", &not);  
    if(not < 0) break;  
        if(not > 89) printf("\Notunuz A - tebrikler!\n");  
        .....  
}
```

# continue

---

```
for (i = 0; i < ogrenciSayisi; i++) {  
    scanf("%lf", &not);  
    if (not < 25) continue;  
    ortalama += not;  
}
```



# Gerçek bir uygulama: Sinüs fonksiyonu

---

Taylor serisi ile sinüs:

$$f(x) = f(0) + f'(0) / 1! + f''(0) / 2! + \dots$$

$$f^n(0) / n! + \dots$$

$$\begin{aligned} \sin(x) = & \sin(0) + \cos(0) x - \sin(0) x^2/2! - \\ & \cos(0) x^3/3! + \sin(0) x^4/4! \dots \end{aligned}$$

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \dots$$

# Sinüs fonksiyonu: Hazırlık

---

- 1) Negatif açıların sinüsü pozitifin sinüsünün negatfidir:  $\text{Sin}(-a) = -\text{Sin}(a)$
- 2)  $\text{Sin}(a) = \text{Sin}(\text{mod}(a, 180))$  fakat % çalışmaz
- 3) Giriş 0 - 90 arasına sıkıştırmalıyız:  
 $a > 90$  için  $\text{Sin}(a) = \text{Sin}(180 - a)$
- 4) Derece okuyup radyanla hesaplamalıyız.

# Sinüs fonksiyonu: Hazırlık

Negatif açılarının sinüsü pozitifin sinüsünün negatifidir:  $\sin(-a) = -\sin(a)$

$\sin(a) = \sin(\text{mod}(a, 180))$  fakat C'de sadece tam sayı modülüsü var.

Girişi 0 - 90 arasına sıkıştırmalıyız:

$a > 90$  için  $\sin(a) = \sin(180 - a)$

Derece okuyup radyanla hesaplamalıyız.

```
double aci, isaretDuzeltme = 1.0, onceki, terim, toplam;
```

```
scanf("%lf ", &aci);
```

```
aci = aci - 180.0 * floor(ac / 180);
```

```
if (aci < 0.0) {
```

```
    isaretDuzeltme = -1.0;
```

```
    aci = - aci;
```

```
}
```

```
if(ac / 90.0) aci = 180.0 - aci;
```

```
aci = (aci / 180.0) * 3.14159265358979323846;
```

# Sinüs fonksiyonu

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \dots$$

---

Her terimi şöyle elde edebiliriz:

$$\text{terim}_i = - \text{onceki} * aci * aci / (i * (i-1))$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$\text{terim}_i = - \text{onceki} * \text{aci} * \text{aci} / (i * (i-1))$

---

onceki = aci;

toplamlam = aci;

```
for(i = 3; i < 100; i += 2) {
```

```
    terim = - onceki * aci * aci / ((double)(i*(i-1)));
```

```
    toplamlam += terim;
```

```
    onceki = terim;
```

```
}
```

```
printf("\nBizim sinus: %20.15lf", isaretDuzeltme*toplamlam);
```

```
printf(" \nC'nin sinusunu: %20.15lf ", Math.Sin(aci));
```

# Bir başka hesap döngüsü

---

```
for(i = 3; i < 100; i += 2){  
    terim = - onceki * aci * aci /(((double)(i*(i-1))));  
    toplam += terim;  
    onceki = terim;  
}
```

```
i = 3;  
while(i < 100 && fabs(terim)/toplam > 1.0E-9){  
    terim = - onceki * aci * aci /(((double)(i*(i-1))));  
    toplam += terim;  
    onceki = terim;  
    i += 2;  
}
```

# Son bir döngü

```
i = 3;
while(i < 100 && fabs(terim)/toplam > 1.0D-9) {
    terim = - onceki * aci * aci /(((double)(i*(i-1))));
    toplam += terim;
    onceki = terim;
    i += 2;
}
```

```
i = 3;
do {
    terim = - onceki * aci * aci /(((double)(i*(i-1))));
    toplam += terim;
    onceki = terim;
    i += 2;
} while(i < 100 && fabs(terim)/toplam > 1.0E-9);
```

# Ara: Not okuma ve ortalama alma

```
int ogrenciSayisi;  
double not, toplam = 0.0, ortalama;  
scanf("%d", &ogrenciSayisi);
```

```
for (i = 0; i < ogrenciSayisi; i++) {  
    scanf("%lf", &not);  
    toplam += not;  
}
```

```
ortalama = toplam / ogrenciSayisi;  
printf("%lf", ortalama);
```

**Problem:** Her öğrencinin ortalamadan farkını da hesapla ve bastır.



# 5. Diziler

```
int i, ogrenciSayisi;
float toplam = 0.0, ortalama;
float not[100], fark[100];
scanf("%d", &ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++) {
    scanf("%f", &not[i]);
}
for (i = 0; i < ogrenciSayisi; i++) {
    toplam += not[i];
}
ortalama = toplam / ogrenciSayisi;
for (i = 0; i < ogrenciSayisi; i++) {
    fark[i] = not[i] - ortalama;
    printf("\nnot = %f fark = %f", not[i], fark[i]);
}
```

# Sıraya koyma: Sort- “Bubble sort”

---

Algoritma:

- 1) Dizinin ilk elemanını al
- 2) Yukarı doğru ilerle ve her elemanla karşılaştır
- 3) Karşılaştırdığın elindekienden küçük ise onunla değiştir yukarı tırmanmaya devam et
- 4) Başa dön. Her dönüşte bir sonraki elemandan başla.

# Bubble sort

---

21 18 76 98 31 17

18 21 76 98 31 17

17 21 76 98 31 18

Birinci tur bitti

17 21 76 98 31 18

17 18 76 98 31 21

İkinci tur bitti

17 18 76 98 31 21

17 18 31 98 76 21

17 18 21 98 76 31

Üçüncü tur bitti

# Bubble sort: Program

```
int i, j, k, degistirme, sayiSayisi, aski, sayilar[20];
scanf("%d", &sayiSayisi);
for(i = 0; i < sayiSayisi; i++)scanf("%d", &sayilar[i]);
k = 0;
for(i = 0; i < sayiSayisi; i++){
    for(j = i; j < sayiSayisi; j++){
        if(sayilar[j] > sayilar[i]) continue;
        aski = sayilar[j];
        sayilar[j] = sayilar[i];
        sayilar[i] = aski;
    }
    if(degistirme == 0) break;
    degistirme = 0;
}
for(i = 0; i < sayiSayisi; i++)printf("\n%d", sayilar[i]);
```

## 6. İşaretçiler- göstergeler- pointers

---

“Değişkenler” veya “tanımlayıcılar” aslında hafıza adresleridir...dedik.

O “adresi” nasıl öğrenebiliriz?

`int ogrenciSayisi`: İçinde tam sayı bulunan bir adres.

`&ogrenciSayisi`: O tam sayının adresi

`scanf(“%d”, &ogrenciSayisi)` dediğimizde, `scanf` fonksiyonuna o adresi veriyoruz...

# Felsefe arası...

---

Hafızada bir yere 266 sayısını yazacağız:

Gerçekte olan: (yalancı bir makine diliyle)

Komut(yaz)	266 sayısı	Adres	
A4 45	01 0A	88 CB	Makine dili
MOV	01 0A	SAYI	Assembler

Bu “SAYI” dediğimiz nesne nedir?

# Yüksek dillerde nasıl göstermeli?

---

Komut(yaz)	266 sayısı	Adres	
A4 45	01 0A	88 CB	Makine dili
MOV	01 0A	SAYI	Assembler

$$\text{SAYI} = 266$$

SAYI, 266 değerini mi temsil etsin?

266 değerinin bulunduğu adresi mi?

Birinci tercih edilmiş...



Komut(yaz)	266 sayısı	Adres	
A4 45	01 0A 88 CB		Makine dili
MOV	01 0A	SAYI	Assembler

$SAYI = 266$

SAYI, 266 değerini mi temsil etsin?

266 değerinin bulunduğu adresi mi?

Birinci tercih edilmiş...

$SAYI = 266$

Ya adres?

$\&SAYI$

Diyelim ki adres, 1 536 444...

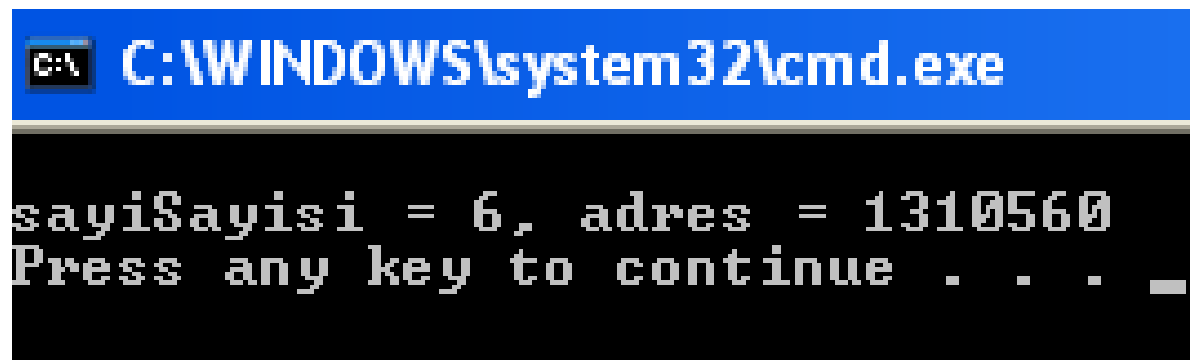


Simetriyi severiz...

<u>Tanımlayıcı</u>	<u>ne demek?</u>	<u>ifade</u>	<u>adres?</u>
int Sayi	266	Sayi	&Sayi
	İçinde ne var?	266	1 536 444
<u>Tanımlayıcı</u>	<u>ne demek?</u>	<u>ifade</u>	<u>sayi?</u>
int* pSayi	1 536 444	pSayi	*pSayi
	İçinde ne var?	1 536 444	266


---

```
int sayiSayisi;  
sayiSayisi = 6;  
printf("\nsayiSayisi = %d, adres = %d\n", sayiSayisi,  
&sayiSayisi);
```



The screenshot shows a Windows command prompt window with a blue title bar that reads "C:\WINDOWS\system32\cmd.exe". The command prompt has a black background with white text. The output of the program is displayed as follows:

```
sayiSayisi = 6, adres = 1310560  
Press any key to continue . . . _
```



```
int sayi;  
scanf("%d", &sayi); /*Bunu biliyorsunuz */  
printf("\n sayi = %d, adres = %d\n", sayi, &sayi);
```

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

```
-----
```

```
int *sayi;  
sayi = malloc(sizeof(int));  
scanf("%d", sayiSayisi); /*Ya bunu? */  
printf("\n sayi = %d, adres = %d\n", *sayi, sayi);
```

```
int i, ogrenciSayisi;
float toplam = 0.0, ortalama, *not, *fark;
scanf("%d", &ogrenciSayisi);
not = malloc(sizeof(float) * ogrenciSayisi);
fark = malloc(sizeof(float) * ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++)scanf("%f", not+i);
for (i = 0; i < ogrenciSayisi; i++)toplam += *(not + i);
ortalama = toplam / ogrenciSayisi;
for (i = 0; i < ogrenciSayisi; i++){
    *(fark + i) = *(not + i) - ortalama;
    printf("\nnot = %f fark = %f", *(not + i), *(fark + i));

free(not);
free(fark);
```

# Pointer aritmetiği

---

```
int dizi[50]----- int *dizi; dizi = malloc(4*50);  
                                     veya sizeof(int) * 50;
```

```
int sayilar[50];
```

```
int *psayilar;
```

```
psayilar = sayilar;
```

```
sayilar[0]----- *psayilar    sayilar = psayilar
```

```
sayilar[18] ----- *(psayilar + 18)
```

## 7. Çok boyutlu diziler

Öğrenci numarası	Notu
1234567	98
7653443	77
6567132	88
3456294	55
4359874	69

# Öğrenci numara ve notunu okuyup yazdıralım:

```
int i, ogrenciSayisi, notlar [5] [2];
scanf("%d", &ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++){
    scanf("%d %d", &notlar[i][0], &notlar[i][1]);
}
for (i = 0; i < ogrenciSayisi; i++){
    printf("\nnumarasi = %d  notu = %d", notlar[i][0], notlar[i][1]);
}
```

# Sonuç:

---

```
C:\WINDOWS\system32\cmd.exe
5
345678 54
654873 98
123456 76
543987 89
235122 92

numarasi = 345678    notu = 54
numarasi = 654873    notu = 98
numarasi = 123456    notu = 76
numarasi = 543987    notu = 89
numarasi = 235122    notu = 92Press an
```



# Peki bunun pointerlisi nasıl?

---

```
int notlar[5][2];
```

yerine

```
int *notlar
```

```
notlar = malloc(sizeof(int) * 5 * 2);
```

```
notlar[3][1] = 98;
```

```
*(*(notlar + 3) + 1) = 98;
```

veya

```
int *notlar ---- deyip hesapla yer bulunu:
```

```
*(notlar + 3 * 2 + 1) = 98;
```

```
int i, ogrenciSayisi, *notlar;
scanf("%d", &ogrenciSayisi);
notlar = malloc(sizeof(int) * 2 * ogrenciSayisi);
for (i = 0; i < ogrenciSayisi; i++){
    scanf("%d %d", (notlar + 2 * i), (notlar + 2 * i) + 1);
}
for (i = 0; i < ogrenciSayisi; i++){
    printf("\nnumarasi = %d notu = %d", *(notlar + 2 * i),
*(notlar + 2 * i + 1));
}
```

## 8. Dizgiler- string

---

Karakter dizilerine dizgi deniyor...

Strings are char arrays

a, k, m... karakterdir. Bunlar **char** veri tipinde depolanır.

“C dilini seviyorum” bir “string” veya “dizgi”dir.

string veya dizgi, C’de bir char dizisidir.

C		d	i	l	i	n	i		s	e	v	i	y	o	r	u	m	\0
---	--	---	---	---	---	---	---	--	---	---	---	---	---	---	---	---	---	----

En çok 20 harfli bir satırı okuyalım, istediğimiz bir harfi bulalım ve hem satırı, hem harfi yazalım:

```
char satir[21] = "Bir satir yaziniz: ", harf;
int hangiHarf;
printf(satir);
scanf(" \n%[a -z]", satir);
printf(" %s", satir);
printf("\nHangi harfi istiyorsunuz?");
scanf(" %d", &hangiHarf);
harf = satir[hangiHarf];
printf("\nYazdiginiz: %s, secilen harf: %dci, bu
%c\n\n", satir, hangiHarf, harf);
```

C:\WINDOWS\system32\cmd.exe

Bir satir yaziniz: iste size bir satir  
iste size bir satir

Hangi harfi istiyorsunuz?11

Yazdiginiz: iste size bir satir, secilen harf: 11ci, bu i

Press any key to continue . . . \_

# Dizgi fonksiyonları:

```
#include <string.h>
```

---

gets(s)

strcpy(s1, s2)

strcat(s1, s2)

strlen(s)

strcmp(s1, s2)

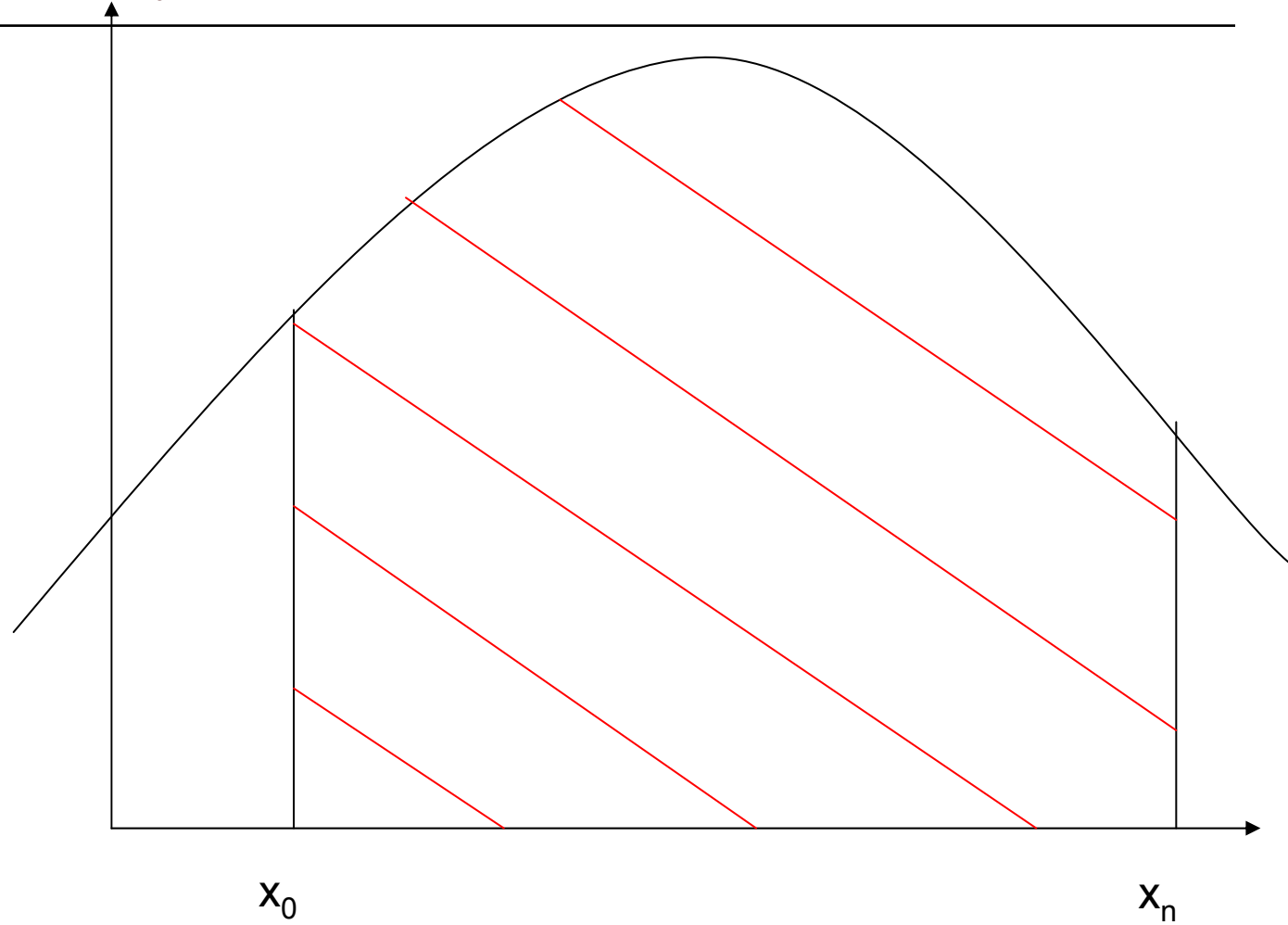
strchr(s1, ch)

strstr(s1, s2)

# Entegral almada yamuk metodu

Şekilde gösterilen fonksiyonun  $x_0$ 'dan  $x_n$ 'e kadar entegralini almak istiyoruz.

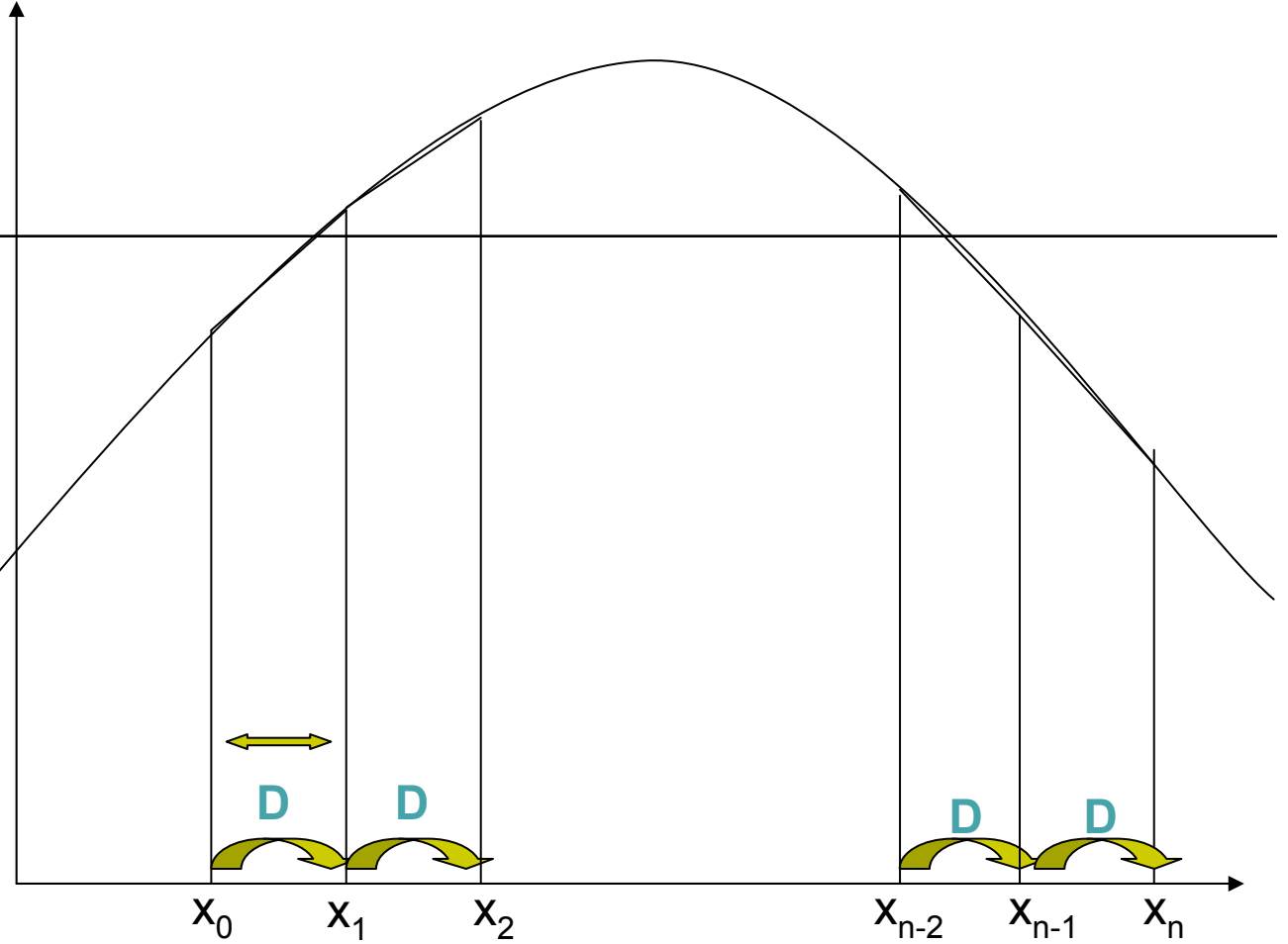
Bu, taralı alanın bulunması demektir.



X eksenini boyunca bir birine eşit  $n$  adım atalım. Her adımın uzunluğu  $D$  olsun.

Sonra alanı  $n$  tane yamuğa bölelim.

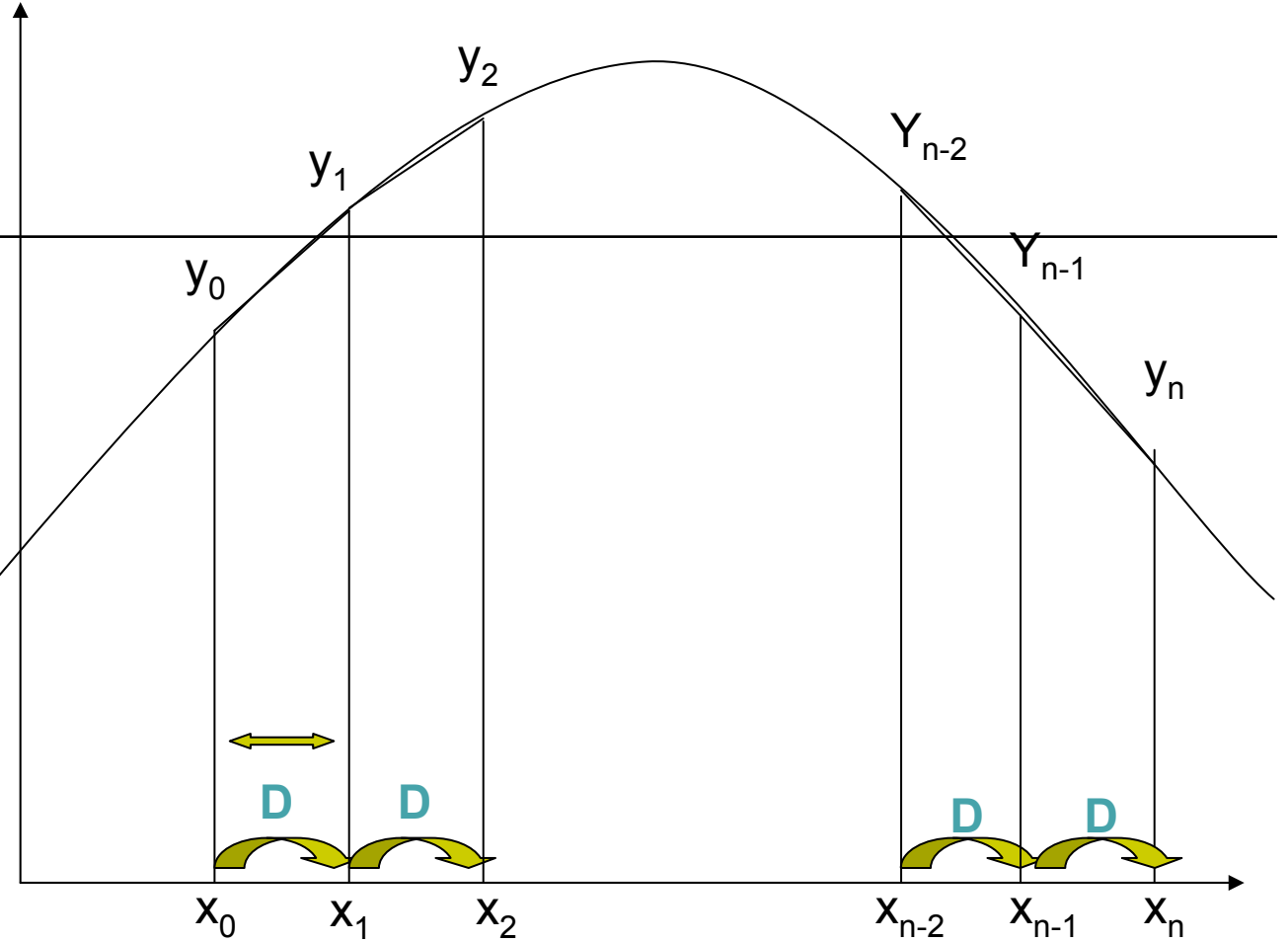
Yamukların yüksekliği  $D$  oluyor.





Her  $X_i$ 'e karşılık gelen fonksiyon değeri  $Y_i$  ise birinci yamuğun alanı  $A_1 = 0.5(Y_0 + Y_1)D$ , ikincinin alanı  $A_2 = 0.5(Y_1 + Y_2)D$ , gibi hesaplanır.

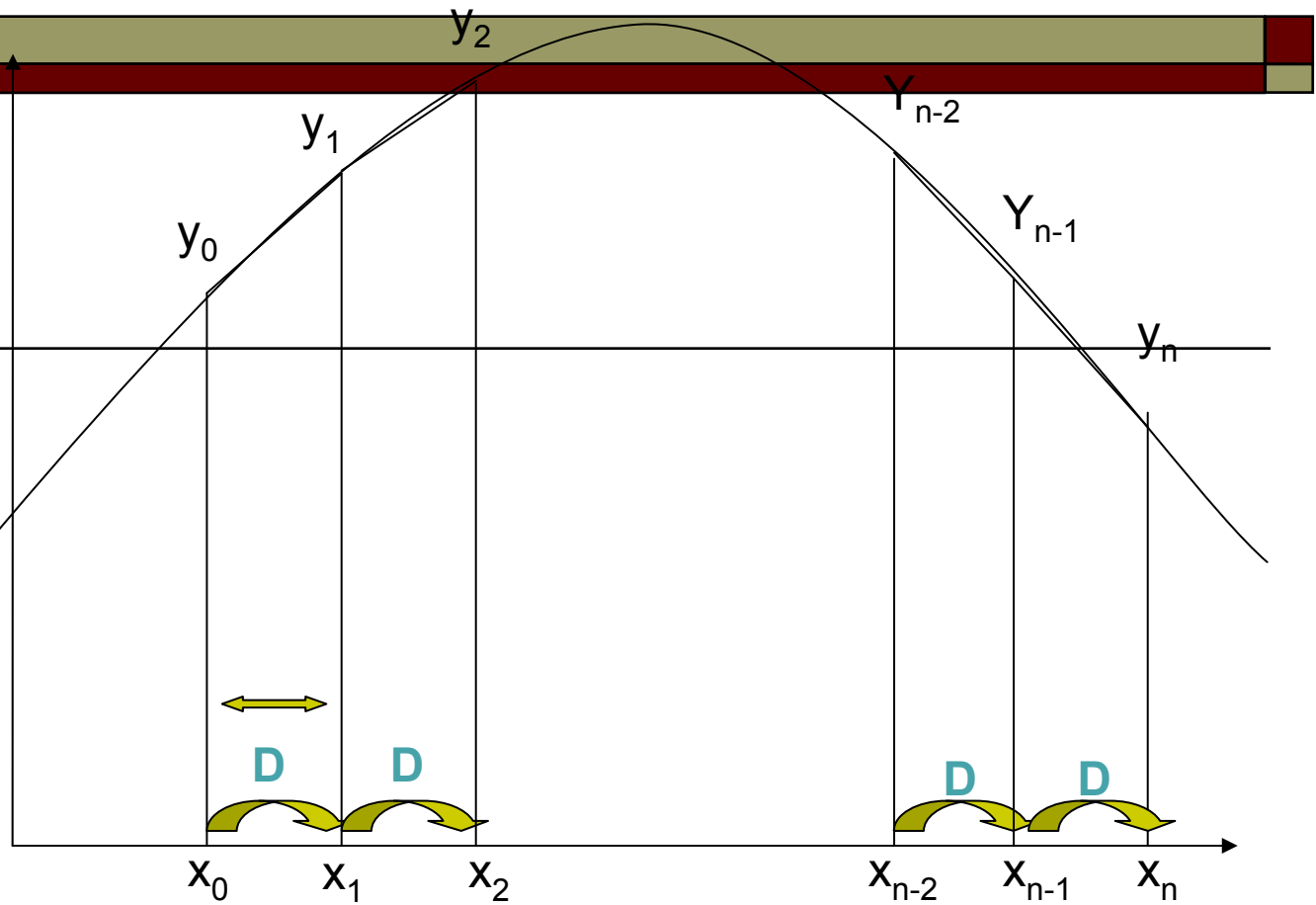
(Yamuğun alanı, iki tabanının uzunluğunun toplamıyla yüksekliğinin çarpımının yarısına eşittir.)



$D$   $A_i = 0.5(Y_{i-1} + Y_i)D$

Bütün yamukların alanları toplamı başta hesaplamak istediğimiz integralin alanına yaklaşık bir değer verir.

D sıfıra, yani n sonsuza yaklaşırken yamuk alanlarının toplamı da integralin değerine yaklaşır.



Bütün yamukların alanları toplamı ~ Entegral =  $0.5(Y_0 + Y_1) D + 0.5(Y_1 + Y_2) D + \dots + 0.5(Y_{n-2} + Y_{n-1}) D + 0.5(Y_{n-1} + Y_n) D$  dir.  $Y_0$  ve  $Y_n$  dışındaki bütün  $Y$ 'lerin ikişer kez toplamaya katıldığına dikkat edin.

Düzenlersek, Entegral =  $D \sum Y_i - 0.5 D (Y_0 + Y_n)$  olur. Toplam  $i = 0$ 'dan  $i = n$ 'e kadar gitmektedir. Negatif terim, birinci ve sonuncu  $Y$ 'lerin toplamaya katkısını düzeltmek için konulmuştur.

Adım uzunluğu  $D = (x_n - x_0) / n$  denkleminde hesaplanabilir.

# Soru

Programınızda  $x_0$ ,  $x_n$  ve  $n$ , kullanıcı tarafından klavyeden girilecektir

- Fonksiyon  $F(x) = x^2$ ,  $F(x) = \sin(x)$  veya başka herhangi bir fonksiyon olabilir.
- Program Entegral =  $D \sum Y_i - 0.5 D (Y_0 + Y_n)$  ve  $D = (Y_n - Y_0) / n$  formüllerini kullanarak entegrali hesaplayacaktır.
- Programınızı  $n=10$  ve  $n=20$  için hesaplanan değerler arasındaki farka bakarak deneyiniz.

Başarılar!



# Dosyalar

---

# Dosyalar- okuma ve yazma

---

Dosya?- file

Stream?- dere?- katar?

Donanım:

Disk, klavye, ekran, manyetik bant, yazıcı...  
ve bunlarla haberleşme sırasında kullanılan RAM tamponu.

Dosya (iki anlamı var): Bunları temsil eden bir mantık yapısı...  
veya bunların bir yerine kayıtlı veri grubu.

Katar: Dosyaya ulaşmak için kurulan mantık bağlantısı- “logical device”

Dosya işaretçisi (file pointer): Dosya ve katarın özelliklerinin bulunduğu bir yapıya işaretçi...

# Dosya fonksiyonları

---

fopen()

fclose()

putc()

getc()

fseek()

fprintf()

fscanf()

feof()

ferror()

rewind()

remove()

flush()



# Dosya fonksiyonları

---

fopen()

FILE \*fopen(const char \**filename*, const char \**mode*)

FILE \*laflar;

laflar = fopen("benimDosyam", "r")

```
#include <stdio.h>
void main()
{
    FILE *laflar;
    char harf;
    laflar = fopen("lafdosyasi.txt", "w");
    do{
        harf = getchar();
        putc(harf, laflar);
    }while(harf != 'x');
    fclose(laflar);
    laflar = fopen("lafdosyasi.txt", "r");
    do{
        harf = getc(laflar);
        putchar(harf);
    }while(harf != 'x');
    fclose(laflar);
}
```



```
C:\WINDOWS\system32\cmd.exe
lag basini duman almıs
gumus dere durmaz akar

gunes ufuktan simdi dogar
yuruyelim arkadaslar

x
lag basini duman almıs
gumus dere durmaz akar

gunes ufuktan simdi dogar
yuruyelim arkadaslar

xPress any key to continue . . .
```

```
lafdosyasi.txt - Notepad
File Edit Format View Help
lag basini duman almıs
gumus dere durmaz akar

gunes ufuktan simdi dogar
yuruyelim arkadaslar

x
```

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

```
void main()
```

```
{
```

```
    FILE *laflar;
```

```
    char harf;
```

```
    laflar = fopen("lafdosyasi.txt", "w+");
```

```
    do{
```

```
        putc(harf = getchar(), laflar);
```

```
    }while(harf != 'x');
```

```
    rewind(laflar)
```

```
    do{
```

```
        harf = getc(laflar);
```

```
        putchar(harf);
```

```
    }while(!feof(laflar));
```

```
    fclose(laflar);
```

```
}
```

# Dosya hangi parametrelerle açılabilir?

---

r      rb

w      wb

a      ab

r+     r+b

w+     w+b

a+     a+b

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

```
void main()
```

```
{  
    FILE *laflar;  
    char harf;  
    laflar = fopen("lafdosyasi.txt", "w+");  
    if(ferror(laflar)) exit();  
    do {  
        putc(harf= getchar(), laflar);  
    } while(harf != 'x');  
    fclose(laflar);  
    laflar = fopen("lafdosyasi.txt", "r");  
    while(!feof(laflar)) {  
        putchar(getc(laflar));  
    }  
    fclose(laflar);  
}
```

```
FILE *laflar;
char harf;
laflar = fopen("lafdosyasi.txt", "w");
do {
    putc(harf = getchar(), laflar);
} while(harf != 'x');
rewind(laflar);
do {
    putchar(harf = getc(laflar));
    if(ferror(laflar)) {
        printf("Bir hadise var getc ile dosya arasinda!");
        fclose(laflar);
        exit();
    }
} while(!feof(laflar));
fclose(laflar);
```

C:\WINDOWS\system32\cmd.exe

Her zamanki gibi dosyaya  
bir seyler yazmaya calisiyoruz

ve

iks harfi ile sonlandiriyoruz

x

Bir hadise var getc ile dosya arasinda!Press any key to continue . . .

# fseek

---

`int fseek(FILE *fp, long numbytes, int origin)`

*origin*, `stdio.h`'taki üç sabitten biri olabilir:

`SEEK_SET` dosyanın başı

`SEEK_CUR` şu anda bulunulan yer

`SEEK_END` dosyanın sonu

Problem: Az önceki programımız, sadece sondan 10karakter yazsın.

```
#include <stdio.h>
void main()
{
    FILE *laflar;
    char harf;
    laflar = fopen("lafdosyasi.txt", "w+");
    do{
        putc(harf = getchar(), laflar);
    }while(harf != 'x');
    rewind(laflar);
    fseek(laflar, -50, SEEK_END);
    do{
        putchar(harf = getc(laflar));
    }while(!feof(laflar));
    fclose(laflar);
}
```



C:\WINDOWS\system32\cmd.exe

```
Bir hadise var can ile canan arasında  
Kaldım yine bir ateş-i hicran arasında  
Bir tîr-i kaza var yine müjgan arasında  
Kasd etmek için canıma hicran arasındax  
arasında  
Kasd etmek için canıma hicran arasındax Press any key to continue . . .
```

# Sonradan eklenen iki fonksiyon:

---

```
size_t fread(void *buffer, size_t numbytes,  
             size_t count, FILE *pf)
```

```
örnek: fread(depo, 1, 1456, laflar);
```

```
size_t fwrite(void *buffer, size_t numbytes,  
             size_t count, FILE *pf)
```

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    FILE *laflar;
    int i = 0;
    char *depoYeri, harf;
    laflar = fopen("lafdosyasi.txt", "r");
    depoYeri = malloc(500);
    fread(depoYeri, 1, 500, laflar);
    while(*(depoYeri + i) != 'x'){
        putchar(*(depoYeri + i));
        i++;
    }
    putchar('\n');
    fclose(laflar);
}
```

C:\WINDOWS\system32\cmd.exe

```
Bir hadise var can ile canan arasında  
Kaldım yine bir ateş-i hicran arasında  
Bir tîr-i kaza var yine müjgan arasında  
Kasd etmek için canıma hicran arasında  
Press any key to continue . . .
```

# Dosyalar son:

---

freopen kullanılarak stdin ve stdout yeniden tarif edilebilir.

```
FILE *freopen(const char *filename,  
              const char *mode, FILE *stream)
```

```
FILE *laflar, *yeniEkran;
```

```
int i = 0;
```

```
char *depoYeri;
```

```
laflar = fopen("lafdosyasi.txt", "r");
```

```
depoYeri = malloc(500);
```

```
fread(depoYeri, 1, 500, laflar);
```

```
yeniEkran = freopen("Ekran.txt", "w", stdout);
```

```
while(*(depoYeri + i) != 'x'){
```

```
    putchar(*(depoYeri + i)); i++;
```

```
}
```

```
putchar('\n');
```

```
fclose(laflar);
```



# Fonksiyonlar

---

# Yazma

```
type fonksiyon_ismi(type parametre1, type  
    parametre2, ... type parametreN)
```

```
{
```

```
    ifadeler
```

```
}
```

```
double Sinus(double aci){
```

```
    /* sinüsün hesabı buraya girecek */
```

```
}
```



# Şifreleme fonksiyonu

---

```
char* sifrele(char* satir, int anahtar)
{
    int boy = strlen(giren);
    for(int i=0; i < boy; i++){
        if( satir[i] + anahtar > 122) satir[i] = satir[i] + anahtar -26;
        else satir[sayac] = satir[sayac] + anahtar;
    }
    return satir;
}
```

# Kullanma

---

```
char yazi[50];
```

```
int oteleme;
```

```
printf("Oteleme sayisini yaziniz\n");
```

```
scanf("%d", &oteleme); getchar();
```

```
printf("\nBir ifade yaziniz\n");
```

```
gets(yazi);
```

```
yazi = sifrele( yazi, oteleme);
```

```
printf("\%s", yazi);
```

```
char* sifrele(char* satir, int anahtar)
{
    ...
    return satir;
}
```

# Parametre geçirme- parameter passing

---

## By value:

```
double Sinus(double aci)
```

```
{
```

```
    /* burada açı indirgeniyor */
```

```
    /* sonra radyana çevriliyor */
```

```
    -----
```

```
    return seri;
```

```
{
```

**Çağırılan programdaki değişken olduğu gibi kalır**

Bir önceki misalde “by value” değildi

---

```
char* sifrele(char* satir, int anahtar)
```

```
{
```

```
    ...
```

```
    return satir;
```

```
}
```

**İşaretçi geçirdiğimiz için çağıran programdaki değer de değişti.**

**Passing arguments “by reference”- “by ref”**

# Bir açının Sin için normalleşmesi

```
double Naci(double aci) {
int isaret = 1;
if (aci < 0) {
    aci = -aci;
    isaret = - isaret;
}
while(aci > 360) aci -= 360;
```

- A) Açı negatifse, isaret = - isaret ve aci = - aci yapın.
- B) Sonra aci > 360 ise aci'nin içindeki bütün 360'ları çıkarın.
- C) aci > 180 ise aci = aci - 180 ve isaret = - isaret yapın.
- D) aci > 90 ise aci = 180 - aci yapın.

```
if (aci > 180) {
    aci = -aci;
    isaret = - isaret;
}
if (aci > 90) aci = 180 - aci;
return isaret * aci;
}
```

# Kullanılış:

---

```
void main(){
    double aci, normalAci;
    printf("Bir aci giriniz \n");
    scanf("%lf", &aci);
    normalAci = Naci(aci);
    printf("\nBu aci su ac
idir: %lf", normalAci);
    printf("\nEski aç ı su idi : %lf", aci);
}
```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
double Naci(double aci);
void main()
{
    double aci, normalAci;
    printf("Bir aci giriniz \n");
    scanf(" %lf", &aci);
    normalAci = Naci(aci);
    printf("\nBu aci su acidir:
    %lf", normalAci);
    printf("\nEski aci su idi :
    %lf\n", aci);
}

```

```

double Naci(double aci){
    int isaret = 1;
    if (aci < 0){
        aci = -aci;
        isaret = - isaret;
    }
    while(aci > 360) aci -= 360;
    if (aci > 180){
        aci = -aci;
        isaret = - isaret;
    }
    if (aci > 90) aci = 180 - aci;
    return isaret * aci;
}

```

```

C:\WINDOWS\system32\cmd.exe
Bir aci giriniz
376.897

Bu aci su acidir: 16.897000
Eski aci su idi : 376.897000
Press any key to continue . . . _

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
double Naci(double* aci);
void main()
{
    double aci, normalAci;
    printf("Bir aci giriniz \n");
    scanf(" %lf", &aci);
    normalAci = Naci(&aci);
    printf("\nBu aci su acidir:
    %lf", normalAci);
    printf("\nEski aci su idi :
    %lf\n", aci);
}

```

```

double Naci(double *aci){
    int isaret = 1;
    if (aci < 0){
        *aci = -*aci;
        isaret = - isaret;
    }
    while(*aci > 360) *aci -= 360;
    if (aci > 180){
        *aci = -*aci;
        isaret = - isaret;
    }
    if (*aci > 90) *aci = 180 - *aci;
    return isaret * *aci;
}

```

```

C:\WINDOWS\system32\cmd.exe
Bir aci giriniz
397.23

Bu aci su acidir: 37.230000
Eski aci su idi : 37.230000
Press any key to continue . . .

```

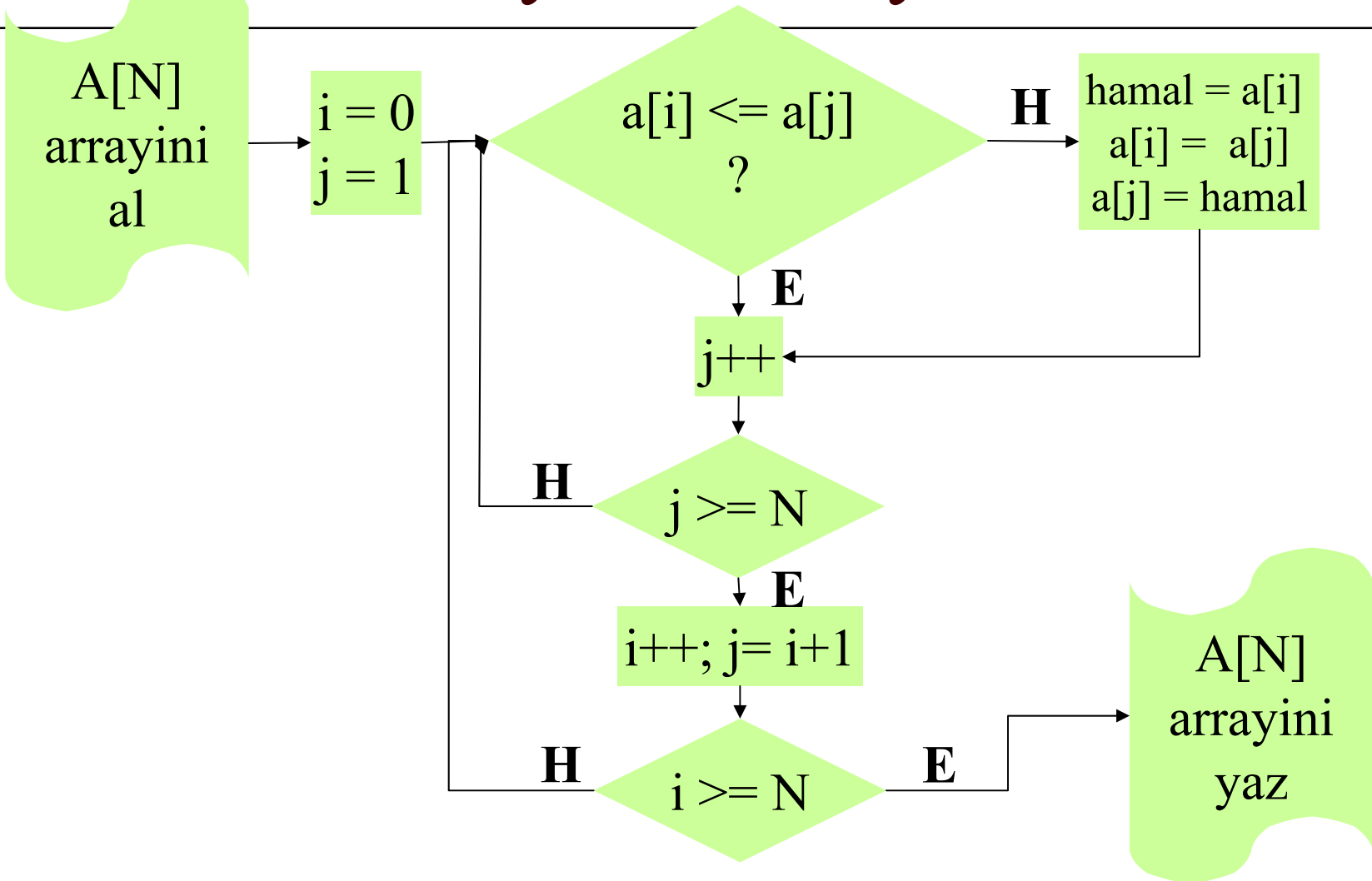


# Kendini çağırın fonksiyonlar: Recursive

```
unsigned long Fact(unsigned long sayi){
    unsigned long sonuc = 1;
    if (sayi < 2) return sonuc;
    do{
        sonuc *= sayi;
        sayi--;
    }
    while(sayi > 1);
    return sonuc;
}
```

```
unsigned long Fact(unsigned long sayi){
    if(sayi < 2)return 1;
    return sayi * Fact(sayi-1);;
}
```

# Bubble sort'u yeniden ziyaret edelim



```
double* BSort(double* dizi, int N){
    int i = 0, j = 1;
    double hamal;
    while(i < N){
        while(j < N){
            if(dizi[i] > dizi[j]){
                hamal = dizi[i];
                dizi[i] = dizi[j];
                dizi[j] = hamal;
            }
            j++;
        }
        i++;
        j = i + 1;
    }
    return dizi;
}
```

```

#include <stdio.h>
#include <stdlib.h>
double* BSort(double* dizi, int N);
void main()
{
    double* a;
    int sayi, i;
    printf("Kac sayi yazacaksiniz? \n");
    scanf(" %d", &sayi);
    a = malloc(sizeof(double)*sayi);
    for(i = 0; i < sayi; i++)scanf("\n%lf", a + i);
    BSort(a, sayi);
    printf("\n\n");
    for(i = 0; i < sayi; i++)printf("\n%lf", *(a + i));
}

```

```

C:\WINDOWS\system32\cmd.exe
Kac sayi yazacaksiniz?
4
4.22
3.12
-8.65
1.009

-8.650000
1.009000
3.120000
4.220000Press any key to continue . . .

```

## recursive çözüm:

```
double* BSort(double* dizi, int i, int N)
{
    int j; double hamal;
    if(i == N) return dizi;
    j = i + 1;
    while(j < N) {
        if(dizi[i] > dizi[j]) {
            hamal = dizi[i];
            dizi[i] = dizi[j];
            dizi[j] = hamal;
        }
        j++;
    }
    BSort(dizi, i+1, N);
    return dizi;
}
```

```
double* BSort(double* dizi, int N)
{
    int i = 0, j = 1;
    double hamal;
    while(i < N) {
        while(j < N) {
            if(dizi[i] > dizi[j]) {
                hamal = dizi[i];
                dizi[i] = dizi[j];
                dizi[j] = hamal;
            }
            j++;
        }
        i++;
        j = i + 1;
    }
    return dizi;
}
```

# Değişkenlerin erişim- kapsama alanı: scope

---

Fonksiyonlardaki değişkenler, fonksiyon dışında görünmez. “Yerel değişken = local variable”

Fonksiyondan çıkıldığında yerel değişkenler yok olur.

Yok olmasını istemiyorsanız static deyin:

**static** double sayi;  
gibi.

main de bir fonksiyondur:

```
void main();  
int main();  
int main(int argc, char* argv);
```

# Değişkenlerin erişim- kapsama alanı: scope

---

*/\* buradaki değişkenler her yerde geçerlidir “global”\*/*

```
void main()
```

```
{
```

```
    /* buradaki değişkenler sadece {} içinde geçerlidir */
```

```
}
```

```
int fonksiyon(int p1, int p2){
```

```
    /* buradaki değişkenler sadece {} içinde geçerlidir */
```

```
}
```

# Birkaç ek değişken özelliği:

---

- static
- const
- register
- volatile

değişkenler nerede? >> stack ve heap  
değişkenlerin gizlenmesi



# Laboratuvar:

---

- Sin hesaplamayı fonksiyon olarak yazın ve deneyin.
- Sin hesaplamadaki terimi recursive bir fonksiyon şeklinde yazın.
- Entegral programını fonksiyon şeklinde yazın.
- Yazdığınız entegral programına entegrali alınacak fonksiyon parametre olarak verilsin. (fonksiyon gösteren işaretçi: pointer to function)



# struct

---

# struct

```
struct type_olarak_isim {  
    type_tanimlayici;  
    type_tanimlayici;  
    ....  
} [tanimlayici1, tanimlayici2...];
```

```
struct kayıt {  
    int telno;  
    char isim;  
    char adres;  
} telefonkayit;
```

# kullanılışı

---

```
struct kayıt{  
    int telno;  
    char isim[50];  
    char adres[50];  
} telefonkayit;
```

```
struct kayıt{  
    int telno;  
    char isim[50];  
    char adres[50];  
};  
struct kayıt telefonkayit
```

```
telefonkayit.telno = 2345555;  
telefonkayit.isim = "Can Candan";  
telefonkayit.adres = "Billur Sokak, 12/3, Maltepe,  
Ankara"  
printf("%d", telefonkayit.telno);
```

# Gerçek bir problem

---

- ❑ Excel'de bir telefon rehberi var
- ❑ Bu rehberi C ile manipüle edilecek bir dosya haline getirelim
- ❑ Dosyada mükerrer kayıt bulunup bulunmadığını kontrol edelim
- ❑ Mükerrer kayıt bırakmayalım (tekrar eden kayıtları silelim)
- ❑ Dosyayı temizlenmiş şekli ile yeniden yazalım.

```
FILE *rehber;
int j, telnumara;
char satir[120], numara[12], isim[52],adres[51];
rehber = fopen("Telefonlar.prn", "r");
while(fgets(satir, 120, rehber)){
    for(j=0; j<7; j++)numara[j] = satir[j];
    numara[7] = '\0';
    telnumara = atoi(numara);
    for(j=0; j< 51; j++) isim[j] = satir[7 + j];
    isim[47] = '\0';
    for(j=0; j< 50; j++) adres[j] = satir[58 + j];
    *(strchr(adres, '\n')-1)= '\0';
    printf(" %d",telnumara);
    printf(" %s",isim);
    printf(" %s\n",adres);
}
fclose(rehber);
```

C:\WINDOWS\system32\cmd.exe

4409515 Gurcu Gurcan Yulek Ankara	Gaziosmanpasa Cankaya
4411667 I Ertan Yulek ra	Buyukesat Cankaya Anka
4393029 Murat Ali Yulek Ankara	Gaziosmanpasa Cankaya
4417609 Oguz Yurekli Ankara	Gaziosmanpasa Cankaya
4467056 Hakan Yuzbasioglu Ankara	Gaziosmanpasa Cankaya
4393592 Mehmet Yologlu ra	Buyukesat Cankaya Anka
4373259 Zekai Yaroglu Ankara	Gaziosmanpasa Cankaya
4411090 Aylin Yeloglu Ankara	Gaziosmanpasa Cankaya
4406225 Can Yesilada ra	Buyukesat Cankaya Anka
4371941 Uygur Yoruk Ankara	Gaziosmanpasa Cankaya
4376590 Kutbettin Amil Yucesumbul Ankara	Gaziosmanpasa Cankaya
4393032 Zekiye Zor	Cankaya Cankaya Ankara

Press any key to continue . . .

```
struct kayit{
    int telnumara;
    char isim[52];
    char adres[51];
}madde;
FILE *rehber; int j, telnumara; char satir[120], numara[12];
rehber = fopen("Telefonlar.prn", "r");
while(fgets(satir, 120, rehber)){
    for(j=0; j<7; j++)numara[j] = satir[j];
    numara[7] = '\0';
    madde.telnumara = atoi(numara);
    for(j=0; j< 51; j++) madde.isim[j] = satir[7 + j];
    madde.isim[47] = '\0';
    for(j=0; j< 50; j++) madde.adres[j] = satir[58 + j];
    *(strchr(madde.adres, '\n')-1)= '\0';
    printf(" %d", madde.telnumara);
    printf(" %s", madde.isim);
    printf(" %s\n", madde.adres);
}
fclose(rehber);
```



# pointer to structure

---

```
struct kayıt {
    int telnumara;
    char isim[52];
    char adres[51];
}madde;
struct kayıt *maddeP;
int birSey, oburSey;


.....
birSey = madde.telnumara;
oburSey = maddeP -> telnumara;
if(birSey == oburSey) doğrudur
```

# struct- bu kadar zahmete değer mi?

---

```
struct kayıt {  
    int telnumara;  
    char isim[52];  
    char adres[51];  
} madde[1600]}
```

Şimdi değer!



---

```
struct kayıt {
    int telnumara;
    char isim[52];
    char adres[51];
}madde[1600];
FILE *rehber, *rehbertemiz;
int i, j, sayi, ayniflag;
char satir[120], numara[12];
rehber = fopen("Telefonlar.prn", "r");
```

```
i = 0;
while(fgets(satir, 120, rehber)){
    for(j=0; j<7; j++)numara[j] = satir[j];
    numara[7] = '\0';
    sayi = atoi(numara);
    ayniflag = 0;
    for(j = 0; j <i; j++){
        if(sayi == madde[j].telnumara){
            ayniflag = 1;
            break;
        }
    }
    if(ayniflag != 0)continue;
```

```
if(ayniflag != 0)continue;
madde[i].telnumara = sayi;
for(j=0; j< 51; j++) madde[i].isim[j] = satir[7 + j];

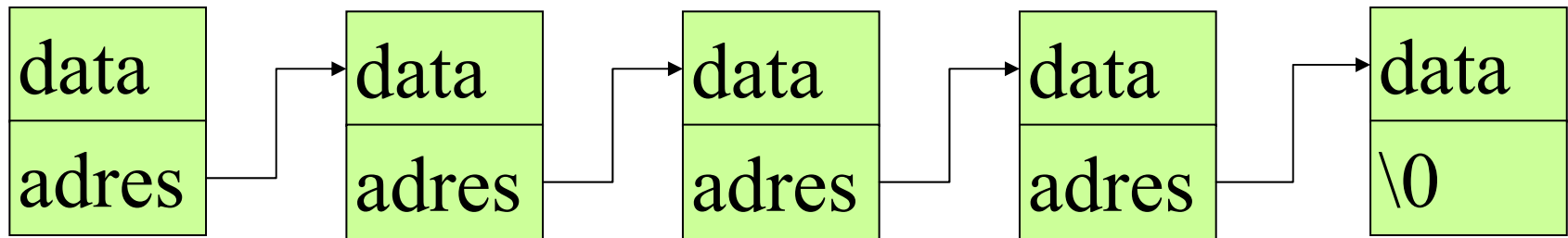

---


madde[i].isim[47] = '\0';
for(j=0; j< 50; j++) madde[i].adres[j] = satir[58 + j];
*(strchr(madde[i].adres, '\n')-1)= '\0';
i++;
}
fclose(rehber);
rehbertemiz =fopen("TemizRehber.prn", "w");
for(j = 0; j <= i; j++)
    fprintf(rehbertemiz, "%d %s %s\n",
madde[j].telnumara, madde[j].isim, madde[j].adres);
fclose(rehbertemiz);
```

4272033	Meral Akca	Gaziosmanpasa Cankaya Ankara
4271762	Sabahattin Akcay	Kavaklidere Cankaya Ankara
4364347	Hanife Akkaya	Gazi Osman Pasa Cankaya Ankara
4477055	Muhammad Akram Alvi	Kavaklidere Cankaya Ankara
4270694	Suleyman Orhan Andac	Gaziosmanpasa Cankaya Ankara
4274130	Sezgin Akan	Gaziosmanpasa Cankaya Ankara
4272611	Kubilay Acarbay	Kavaklidere Cankaya Ankara
4271721	Nuran Acarbay	Kavaklidere Cankaya Ankara
4463371	Ali Riza Balta	Gaziosmanpasa Cankaya Ankara
4679745	Birsen Baskurt	Gaziosmanpasa Cankaya Ankara
4276316	Cemal Baskurt	Gaziosmanpasa Cankaya Ankara
4276556	Tuncer Bulutay	Gazi Osman Pasa Cankaya Ankara
4286807	Ali Levent Bekensir	Gazi Osman Pasa Cankaya Ankara
4272565	Ali Bulent Birhekimoglu	Gaziosmanpasa Cankaya Ankara
4671218	Cemal Bayri	Kavaklidere Cankaya Ankara
4277040	Arslan Bibioglu	Kavaklidere Cankaya Ankara
4360064	Celalettin Egesuren	Gaziosmanpasa Cankaya Ankara
4275129	M Ufuk Ergun	Gaziosmanpasa Cankaya Ankara
4272610	Recep Orhan Erisen	Gaziosmanpasa Cankaya Ankara
4681679	Adnan Erturk	Kavaklidere Cankaya Ankara
4361922	Tavanc Ervavuz	Gaziosmanpasa Cankaya Ankara

# Linked lists

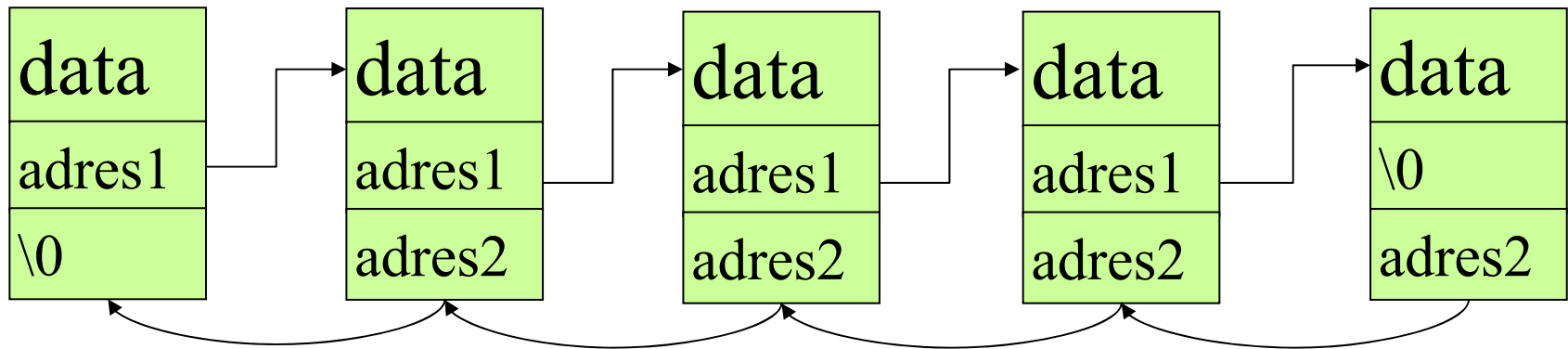
---



```
struct liste {  
    int falan;  
    char *filan;  
    struct liste* adres;  
} listeYav;
```

```
struct liste {  
    int falan;  
    char *filan;  
    struct liste* onceki;  
    struct liste* sonraki;  
} listeYav;
```

# Doubly linked list



```
struct liste {  
    int falan;  
    char *filan;  
    struct liste* onceki;  
    struct liste* sonraki;  
} listeYav;
```



# bit alanları (bit fields)

Serial port:

Bit	1 ise anlamı
0	change in ccts line
1	change in data-set-ready
2	trailing edge dedected
3	change in receive line
4	clear to send
5	data set ready
6	telephone ringing
7	received signal

```
struct status{
    unsigned dcts: 1;
    unsigned ddsr: 1;
    unsigned ted: 1;
    unsinged drl: 1;
    unsigned cts: 1;
    unsigned dsr: 1;
    unsigned zil: 1;
    unsigned sig: 1;
} seriPort;

if (seriPort.dcts ==1) {
    xxxx
}
```