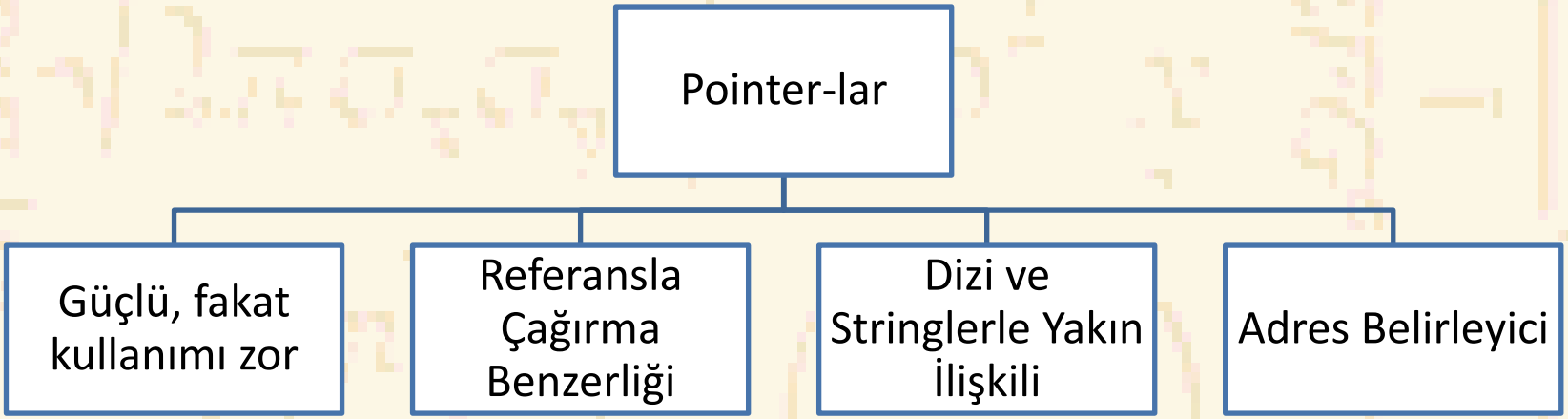
Bölüm 7 – Pointer (İşaretleyici)

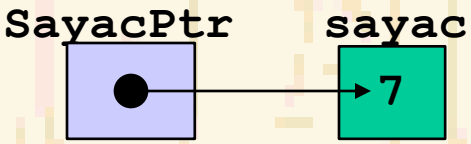
İçerik

7.1 Giriş


Pointer-lar
Güçlü, fakat kullanımı zor
Referansla Çağırma Benzerliği
Dizi ve Stringlerle Yakın İlişkili
Adres Belirleyici

- Pointer Değişkenleri
  - Değer olarak değişkenlerin bellek adreslerini gösterir
  - Normal değişkenler özel bir değer içerir (doğrudan referans)

  - Pointer-lar özel bir değere sahip değişkenlerin adreslerini gösterirler(dolaylı referans)
  - Dolaylılık – bir pointer değeri referans verir



7.2 Pointer Değişken Tanımlaması

- Pointer tanımlanması
  - ***** pointer değişkeni ile kullanılır

```
int *degiskenPtr;
```

  - Bir tamsayıya (**int**) bir pointer gösterir (**int *** tipinde pointer)
  - Çoklu pointer-lar için her bir değişken tanımlamasının önünde bir ***** gerektirir

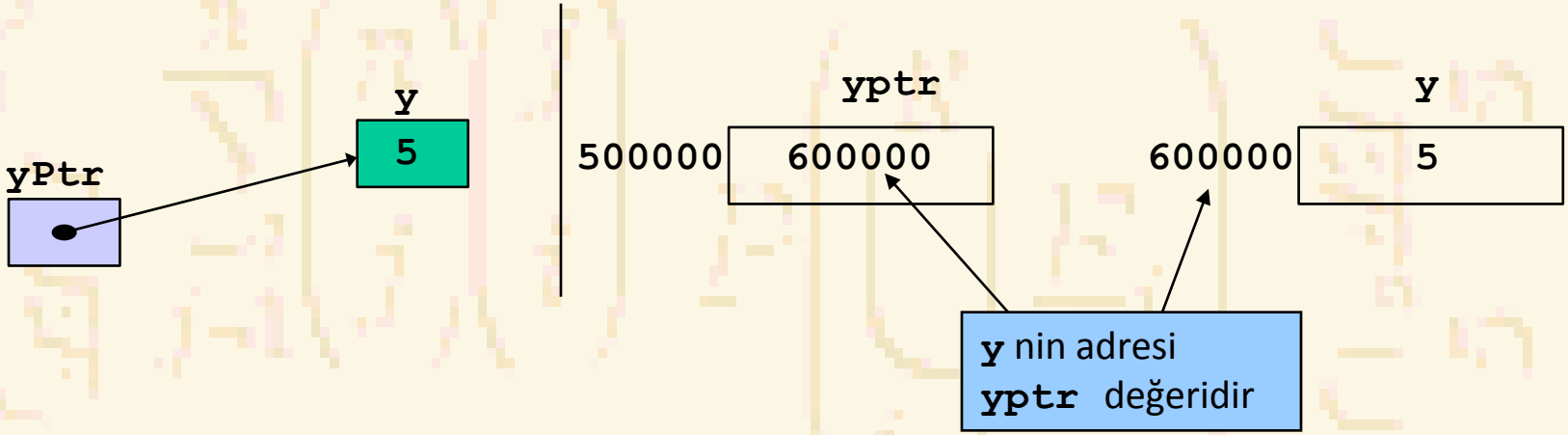
```
int * degiskenPtr1, * degiskenPtr2;
```

  - Her tür veri tipine pointer tanımlanabilir
  - Pointer-lara **0**, **NULL**, veya bir adres atanabilir
    - **0** veya **NULL** – boşluğu(hiçbir şeyi) işaret eder (**NULL** tercih edilir)

7.3 Pointer Operatörleri

- **&** (adres operatörü)
  - Operandın adresini gönderir

```
int y = 5;
int *yPtr;
yPtr = &y;      // yPtr y nin adresini alır
yPtr y yi "işaret eder"
```



7.3 Pointer Operatörleri

- **`*`** (dolaylı/ters referans operatörü)
  - Operandın işaret ettiği değere bir başka ad (lakap) gönderir
  - **`*yptr, y`** değerini gönderir(çünkü **`yptr y`** yi işaret eder)
  - **`*`** atama için kullanılabilir
    - Bir nesnenin lakabını temsil eder

```
*yptr = 7;  // y değerini 7 alır
```

  - Pointer-a (**`* in`** operandına) dolaylı referans verir sol değer olmalıdır (sabit olamaz)
- **`*`** ve **`&`** birbirinin tersidir
  - Birbirlerini yok ederler

7.3 Pointer Operatörleri

```
/* Örnek 7.4
   & ve * operatörlerini kullanma */
#include <stdio.h>

int main()
{
   int a;        /* a, bir tamsayı */
   int *aPtr;    /* aPtr bir tamsayıya bir pointer */

   a = 7;
   aPtr = &a;    /* aPtr, a nın adresini alır*/

   printf( "a nın adresi:%p"
           "\n aPtr nin değeri= %p", &a, aPtr );

   printf( "\n a nın değeri= %d"
           "\n*aPtr nin değeri=%d", a, *aPtr );

   printf( "\nDikkat edilirse * ve & birbirlerinin"
           "tersidir.\n&*aPtr = %p"
           "*&aPtr = %p\n", &*aPtr, *&aPtr );

   return 0;
}
```

a nın adresi **aPtr** ın değeridir

***** , operandın işaret ettiği değerin diğer adıdır. **aPtr a** yı işaret eder böylece ***aPtr a** nın değerini gönderir

***** ve **&** birbirlerinin tersleridir

```
a nın adresi= 0012FF88
aPtr nin değeri=0012FF88
a nın değeri= 7
*aPtr nin değeri=7
Dikkat edilirse * ve & birbirlerinin tersidir.
&*aPtr = 0012FF88
*&aPtr = 0012FF88
```

7.4 Fonksiyonu Referans ile Çağırma

- Pointer kullanarak referans ile çağırma
  - **&** operatörü kullanarak argümentin adresini aktar
  - Bellekteki esas değerin değiştirilmesine izin verir (taşınma)
  - Diziler **&** ile aktarılamaz çünkü dizi adı halihazırda bir pointer-dır
- ***** operatörü
  - Fonksiyonun içindeki bir değişken için bir lakap olarak kullanılır

```
void cıft( int *sayi )
 {
  *sayi = 2 * ( *sayi );
 }
```

  - ***sayi** aktarılan değişkenin lakabı olarak kullanılıyor

7.4 Fonksiyonu Referans ile Çağırma

```
/* Örnek 7.7
   Pointer kullanarak referansla çağırma ile
 sayının kübünü bulma */

#include <stdio.h>

void Ref_ilekup( int * );

int main()
{
   int sayi = 5;

   printf( "Sayının orjinal değeri= %d", sayi );
 Ref_ilekup( &sayi );
   printf( "\nSayının yeni değeri= %d\n", sayi );

   return 0;
}

void Ref_ilekup( int *nPtr )
{
   *nPtr = *nPtr * *nPtr * *nPtr;
}
```

Fonksiyon bir pointer-ı tamsayı olarak alır (**int ***).

sayi nın adresinin nasıl değiştiğine dikkat - **Ref_ilekup** pointer (degiskenin adresini) bekliyor

Ref_ilekup içinde, ***nPtr** kullanıldı (***nPtr**, **sayi** dır).

```
Sayının orjinal değeri= 5
Sayının yeni değeri= 125
```

- **`const`** belirteci
  - Değişken değiştirilemez
  - **`Const`**, değişken değişimine gereksinim yoksa kullanılır
  - **`const`** değişkenini değiştirmeye kalkılırsa hata mesajı gelir
- **`const`** pointer-lar
  - Sabit bir bellek yerini işaret eder
  - Tanımlandığında ilk atama yapılmalıdır
  - **`int *const birPtr = &x;`**
    - Tipi: **`int *const`** –bir **`int`** (tamsayıya) sabit bir pointer
  - **`const int *birPtr = &x;`**
    - Bir **`const int`** (sabit tamsayıya) bir normal pointer
  - **`const int *const Ptr = &x;`**
    - **`const int`**-e bir **`const`** pointer
    - **`x`** değişebilir, fakat **`*Ptr`** değişmez

7.5 Pointer ile const Belirtecini Kullanma

```
1  /* Örnek 7.13
2     Bir sabit pointer- değiştirmeye kalkmak
3     Veri sabit değil */
4
5  #include <stdio.h>
6
7  int main()
8  {
9     int x, y;
10
11    int * const ptr = &x; /* ptr bir tamsayıya bir sabit
12                             pointerdır. Tamsayı ptr ile
13                             değiştirilebilir, fakat ptr herzaman
14                             aynı bellek yerini gösterir. */
15    *ptr = 7;
16    ptr = &y;
17
18    return 0;
19 }
```

***ptr** değişebilir– **x** bir sabit değildir

ptr değiştirmek hata verir– **ptr** bir sabit pointerdır.

```
Örnek 7.13.c:
Error E2024 Örnek 7.13.c 16: Cannot modify a const object in
function main
*** 1 error in Compile ***
```

7.6 Referans ile Çağırmayla Balon Sıralaması

- Balon sıralaması-pointer ile
  - İki elemanın yerini değiştir
  - **`degistir`** fonksiyonu dizi elemanlarının adresini almalı(**`&`** kullanarak)
    - Dizi elemanları default olarak değer(eleman) ile çağıra sahiptir
  - Pointerlar ve **`*`** operatörü kullanarak, **`degistir`** dizi elemanlarını değiştirebilir
- Önkod

Diziyi gir

Orjinal sırada diziyi yaz

Balon fonksiyonunu çağır

Sıralı diziyi yaz

Balon-u tanımla

7.6 Referans ile Çağırmayla Balon Sıralaması

- **`sizeof`**
  - Byte cinsinden operandın boyutunu gönderir
  - Diziler için: 1 elemanın boyutu çarpı eleman sayısı
  - Eğer **`sizeof( int ), 4`** byte ise, bu durumda

```
int dizi[ 10 ];
printf( "%d", sizeof( dizi) );
```

    - **`40`** olacaktır
- **`sizeof`**
  - Değişken adları ile
  - Tip adları ile
  - Sabit değerler ile

  kullanılabilir

```c
/* Örnek 7.15
   Bu program verileri bir diziye aktarır, artan sırada
   sıralar, ve sonucu yazar. */
#include <stdio.h>
#define SIZE 10
void balon( int *, const int );

int main()
{

   int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int i;

   printf( "Orjinal sıradaki veriler:\n" );

   for ( i = 0; i < SIZE; i++ )
      printf( "%4d", a[ i ] );

   balon( a, SIZE );           /* diziyi sırala */
   printf( "\nArtan sırada dizi elemanları:\n" );

   for ( i = 0; i < SIZE; i++ )
      printf( "%4d", a[ i ] );

   printf( "\n" );

   return 0;
}

void balon( int *dizi, const int boyut )
{
   void degistir( int *, int * );
```

balon dizinin adresini (pointerları) alır.
Dizi adı bir pointerdır.

7.6 Referans ile Çağırmayla Balon Sıralaması

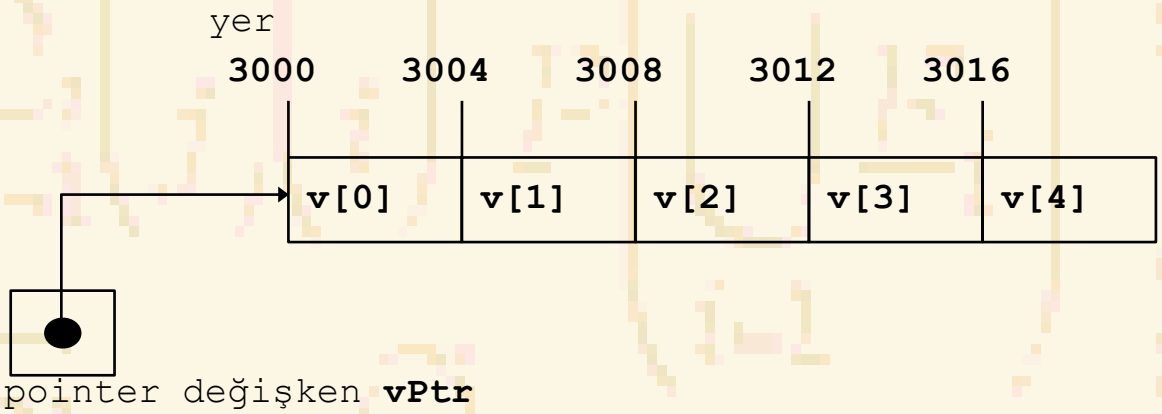
```
33     int gec, j;
34     for ( gec = 0; gec < boyut - 1; gec++ )
35
36        for ( j = 0; j <gec - 1; j++ )
37
38           if ( dizi[ j ] > dizi[ j + 1 ] )
39              degistir( &dizi[ j ], &dizi[ j + 1 ] );
40  }
41
42  void degistir( int *eleman1Ptr, int *eleman2Ptr )
43  {
44     int gec = *eleman1Ptr;
45     *eleman1Ptr = *eleman2Ptr;
46     *eleman2Ptr = gec;
47  }
```

```
Orjinal sıradaki veriler:
   2    6    4    8   10   12   89   68   45   37
Artan sırada dizi elemanları
   2    4    6    8   10   12   37   45   68   89
```

7.7 Pointer İfadeleri ve Pointer Aritmetiği

- Pointerlar üzerinde aritmetik işlemler yapılabilir
  - Pointerı artır/azalt (**++** veya **--**)
  - Bir pointıra tamsayı ekle (**+** veya **+=** , **-** veya **-=**)
  - Pointerlar birbirlerinden çıkarılabilir
  - Bir dizi üzerinde uygulanmadıkça operasyonlar anlamsız

- 4 byte **int-li** bir makinede 5 elemanlı **int** dizisi
  - **vPtr** ilk eleman **v[0]** ı işaret eder
    - Adresi **3000** ise (**vPtr = 3000**)
  - **vPtr += 2; vPtr** yi **3008** e götürür
    - **vPtr,** şimdi **v[2]** yi işaret eder (2 adres ileri gitti), makine 4 byte **int** olduğundan **3008** adresini işaretler



7.7 Pointer İfadeleri ve Pointer Aritmetiği

Pointerları Çıkarma

- Birinden diğerine eleman sayısını gönderir. Eğer
  - **vPtr2 = v[2];**
  - **vPtr = v[0]; ise**
- **vPtr2 - vPtr** değeri 2 dir

Pointer karşılaştırma (<, == , >)

- Hangi pointerın daha büyük indisli eleman olduğunu işaret eder
- Ayrıca pointerın **0** ı işaretleyip işaretlemdiğini gösterir

Aynı tip pointerlar birbirlerine aktarılabilir

- Eğer aynı tip değilse, değiştirme operatörü kullanılmalıdır
- **void** (tip **void** *) e olan pointer hariç:
  - Genel pointer olup, herhangi bir tipi temsil eder
  - Bir pointeri **void** pointera değiştirmek gerekli değildir
  - **void** pointerlar tekrar referans edilemezler

7.8 Pointerlar ve Diziler Arasındaki İlişki

- Diziler ve pointerlar yakından ilişkilidir
  - Dizi adı bir sabit pointer gibidir
  - Pointerlar dizi indis işlemleri yapabilirler
- `b[ 5 ]` dizisini ve `bPtr` pointerını tanımla
  - Birbirlerine eşitlemek için:

```
bPtr = b;
```

    - Dizi adı (`b`) aslında `b[ 5 ]` dizisinin ilk elemanının adresidir

```
bPtr = &b[ 0 ]
```

    - `bPtr` yi `b` nin ilk elemanının adresi olarak alır

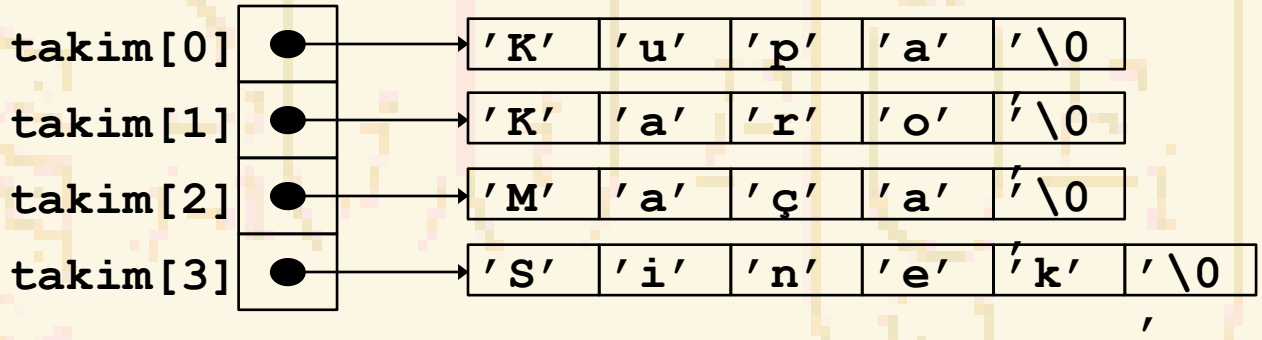
  - `b[ 3 ]` elemanına
    - `*( bPtr + 3 )` ile ulaşılabilir
      - (Pointer/ dolaylı ulaşım) gösterimi
    - veya `bptr[ 3 ]` ile ulaşılabilir
      - pointer/indis gösterimi
      - `bPtr[ 3 ], b[ 3 ]` ile aynıdır
    - Dizi üzerinde pointer aritmetiği ile de ulaşılabilir

```
*( b + 3 )
```

- Dizi, pointer içerebilir
- Örneğin: Bir string dizisi

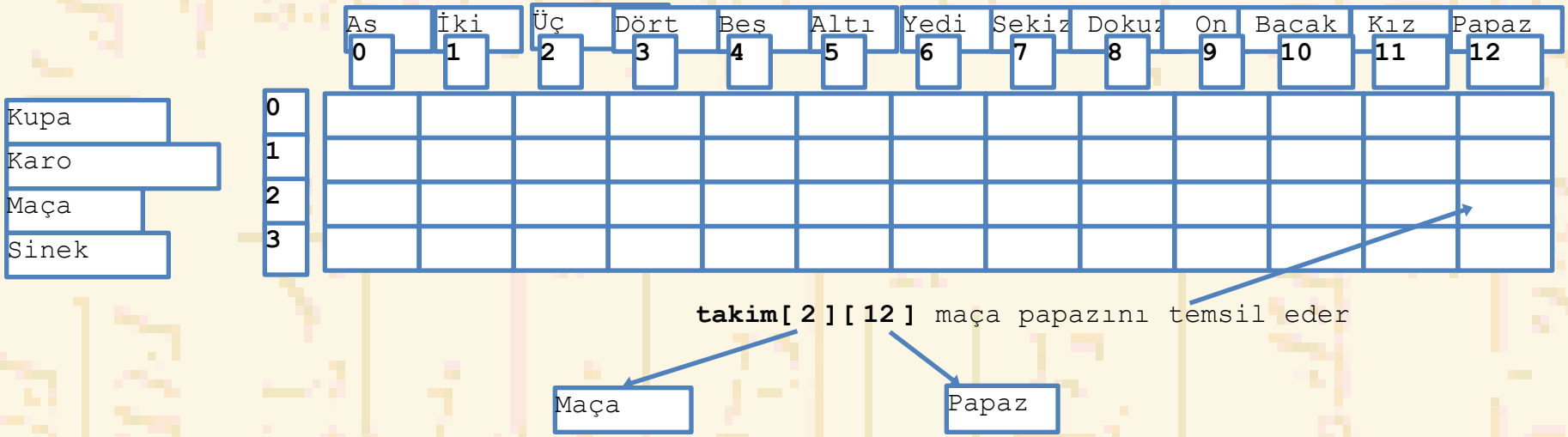
```
char *takim[ 4 ] = { "Kupa", "Karo",
   "Maça", "Sinek" };
```

  - Stringler, ilk karakterlere pointerdır
  - **char *** – **takim ın** ilk elemanı bir **char** e pointerdır
  - Stringler aslında **takim** dizisine yüklenmemiştir, sadece stringe pointerlar yüklenmiştir



  - **takim** dizisinin boyutu sabittir, fakat stringlerin boyutu değişik olabilir

- Kart karma programı
  - Stringler için pointer dizisi kullan
  - İki indisli dizi (matris) kullan (takım, yüz)



  - 1-52 sayıları diziye girer
    - Kartların çekiliş sırasını temsil eder

7.10 Uygulama: Kart Karma ve Dağıtma Simulasyonu

- Önkod
  - Üst düzey:

52 kartı kar ve çek

- İlk düzenleme:

Takım dizisini oluştur
Yüz dizisini oluştur
Kart dizisini oluştur
Kartı kar
52 kartı dağıt

- İkinci düzenleme
  - *kart karma* işlemi

52 kartın her biri için
Rasgele sırada yerleştir

  - *52 kart çek* işlemi

52 kartın her biri için
Rasgele bir kart çek ve göster

- Üçüncü düzenleme
  - *kart karma* işlemi

Rasgele deste diz
Önceden seçilen ise
rasgele yeniden diz

  - *52 kart çek* işlemi

Her bir deste için
Rasgele bir kart çek ve göster

```c
/* Örnek 7.24
   Kart karma çekme programı */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void kar( int [][ 13 ] );
void cek( const int [][ 13 ], const char *[], const char *[] );

int main()
{
   const char *takim[ 4 ] =
      { "Kupa", "Karo", "Maça", "Sinek" };
   const char *yuz[ 13 ] =
      { "As", "İki", "Üç", "Dört",
        "Beş", "Altı", "Yedi", "Sekiz",
        "Dokuz", "On", "Bacak", "Kız", "Papaz" };
   int deste[ 4 ][ 13 ] = { 0 };

   srand( time( 0 ) );

   kar( deste );
   cek( deste, yuz, takim );

   return 0;
}

void kar( int vdeste[][ 13 ] )
{
   int sat, kol, kart;

   for ( kart = 1; kart <= 52; kart++ ) {
```

7.10 Uygulama: Kart Karma ve Dağıtma Simulasyonu

```
33          do {
34             sat = rand() % 4;
35             kol = rand() % 13;
36          } while( vdeste[ sat ][ kol ] != 0 );
37
38          vdeste[ sat ][ kol ] = kart;
39       }
40    }
41
42    void cek( const int vdeste[][ 13 ], const char *vyuz[],
43              const char *vtakim[] )
44    {
45       int kart, sat, kol;
46
47       for ( kart = 1; kart <= 52; kart++ )
48
49          for ( sat = 0; sat <= 3; sat++ )
50
51             for ( kol = 0; kol <= 12; kol++ )
52
53                if ( vdeste[ sat][ kol ] == kart )
54                   printf( "%5s - %-8s%c",
55                   vyuz[ kol ], vtakim[ sat ],
56                   kart % 2 == 0 ? '\n' : '\t' );
```

1-52 sayıları rasgele **deste** dizisine atanır.

kart numarası için **deste yi** tarar ve **yuz** ve **takim**. ı yazar

```
  Altı - Kupa

...(gibi 52 kart sıralanır)
```

7.11 Fonksiyonlara Pointer

- Fonksiyona Pointer
  - Fonksiyonun adresini içerir
  - Dizi adının ilk elemanın adresi olmasına benzerdir
  - Fonksiyon adı fonksiyonu tanımlayan kodun başlama adresidir
- Fonksiyon pointerları
  - Fonksiyonlara aktarılabilir
  - Dizilere yüklenebilir
  - Diğer fonksiyon pointerlarına atanabilir

7.11 Fonksiyonlara Pointer

- Örnek: balon-sıralaması
  - **`Balon`** fonksiyonu bir fonksiyon pointerı alır
    - **`balon`** bu yardımcı fonksiyonu çağırır
    - ki o da artan yada azalan sıraya karar verir
  - Fonksiyon pointerı için **`balonsirala`** daki argüment:

```
bool ( *karsilastir )( int, int )
```

    Balonsirala ya bir fonksiyona pointer beklemesini ve iki tamsayı almasını ve bir **`bool`** göndermesini söyler
  - Eğer parantez kullanılmaz ise:

```
bool *karsilastir( int, int )
```

    - İki tamsayı alan ve bir pointera bir **`bool`** gönderen bir fonksiyon tanımlar

```
1  /* Örnek 7.26
2     Fonksiyon pointerı kullanarak cok amaclı sıralama */
3  #include <stdio.h>
4  #define SIZE 10
5  void balon( int [], const int, int (*)( int, int ) );
6  int artan( int, int );
7  int azalan( int, int );
8
9  int main()
10 {
11
12    int sıra,
13        savac,
14        a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
15
16    printf( "Artan sıralama icin 1 gir,\n"
17            "Azalan sıralama icin 2 gir: " );
18    scanf( "%d",  &sıra );
19    printf( "\nOrıinal sıradaki veriler:\n" );
20
21    for ( savac = 0; savac < SIZE; savac++ )
22       printf( "%5d", a[ savac ] );
23
24    if ( savac == 1 ) {
25       balon( a, SIZE, artan );
26       printf( "\n Artan sırada veriler:\n" );
27    }
28    else {
29       balon( a, SIZE, azalan );
30       printf( "\nAzalan sırada veriler:\n" );
31    }
32
```

Fonksiyon pointer parametresine dikkat

```
33    for ( savac = 0; savac < SIZE; savac++ )
34       printf( "%5d", a[ savac ] );
35
36    printf( "\n" );
37
38    return 0;
39 }
40
41 void balon( int is[], const int bovut,
42             int (*karsilastir)( int, int ) )
43 {
44    int aec, sav;
45
46    void degistir( int *, int * );
47
48    for ( aec = 1; aec < bovut; aec++ )
49
50       for ( sav = 0; sav < bovut - 1; sav++ )
51
52          if ( (*karsilastir)( is[ sav ], is[ sav + 1 ] ) )
53             degistir( &is[ sav ], &is[ sav + 1 ] );
54 }
55
56 void degistir( int *eleman1Ptr, int *eleman2Ptr )
57 {
58    int temp;
59
60    temp = *eleman1Ptr;
61    *eleman1Ptr = *eleman2Ptr;
62    *eleman2Ptr = temp;
63 }
64
```

artan and **azalan** return **doğru** veya **yanlış** gönderir. Eğer fonksiyon **doğru** gönderirse **balon, degistir** i çağırır

Fonksiyon pointerın nasıl çağrıldığına dikkat. * gerekli değil, fakat **karsilastir** ın bir fonksiyon değil fonksiyon pointerı olduğunu vurgular

```
65 int artan( int a, int b )
66 {
67    return b < a;    /* b, a dan küçük ise değiştir */
68 }
69
70 int azalan( int a, int b )
71 {
72    return b > a;    /* b, a dan büyük ise değiştir */
73  }
```

```
Artan sıralama için 1 gir,
Azalan sıralama için 2 gir: 1

Orjinal siradaki veriler:
   2    6    4    8   10   12   89   68   45   37
Artan sırada veriler:
   2    4    6    8   10   12   37   45   68   89
```

```
Artan sıralama için 1 gir,
Azalan sıralama için 2 gir: 2

Orjinal siradaki veriler:
   2    6    4    8   10   12   89   68   45   37

Azalan sırada veriler:
  89   68   45   37   12   10    8    6    4    2
```