

315 programlama Dilleri

Yrd. Doç. Dr. Ahmet Arif AYDIN

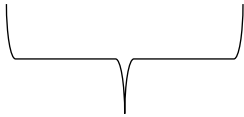
Bu Haftanın Konusu

Data Types-2 (Veri Tipleri)

Associative Arrays (ilişkisel diziler)

Verinin anahtar (key) ve sakladığı değer (value) ikililerinden (pair) oluşan dizilerdir.

{ key 1 -> value 1 , key 2 → value 2 , , key n → value n }



Anahtar değer ikilisi

Associative Arrays (ilişkisel diziler)

- İlişkisel dizilerde indexler de verilerle beraber kaydedilmektedir
- Indexler (anahtar) sıralı değildir.

{ key 1 -> value 1 , key 2 → value 2 , , key n → value n }

İlişkisel olmayan dizilerde indexler düzenli olarak arttığı için tekrar kaydedilmezler.

Associative Arrays (ilişkisel diziler)

- İlişkisel diziler bazı dillerde direkt olarak tanımlıdır.
 - Perl
 - Ruby
 - Python
 - Lua (Oyun ve gömülü sistem programlama)
- Java, C++, ve C#, dillerinde ise standard sınıf kütüphaneleri yardımıyla kullanılmaktadır.

Associative Arrays (ilişkisel diziler)

RUBY dilinde bulunan ilişkisel dizi HASH olarak adlandırılır.

```
Aylar = Hash.new
```

```
asci_hash = Hash [ "a" => 97, "b" => 98 , ... , "z" => 122]
```

```
renkler = { "kırmızı" => 0xf00, "yeşil" => 0x0f0 }
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}    PYTHON
```

Associative Arrays (ilişkisel diziler)

```
ascii_hash = Hash [ "a" => 97, "b" => 98 , ... , "z" => 122]
```

```
ascii_hash .each do |key, value|  
  print key, " is ", value, "\n"  
end
```

Associative Arrays (ilişkisel diziler)

RUBY

```
def create_hash(a)
  h=Hash.new

  for i in 0..(a.size-1)
    for j in 0..(a[i].size-1)
      word=a[i][j]
      if h.keys.include? word
        h[word]+=1
      else
        h[word]=1
      end
    end
  end

  print h
end
```

iki boyutlu bir dizinin içinde
bulunan kelimelerin
Hash 'ini oluşturur

Associative Arrays (ilişkisel diziler)

```
{  
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",  
  "id": 850006245121695744,  
  "id_str": "850006245121695744",  
  "text": "1/ Today we're sharing our vision for the future of  
the Twitter API platform! nhttps://t.co/XweGngmxlP",  
  "user": {},  
  "entities": {}  
}
```

<https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object>

Associative Arrays (ilişkisel diziler)

```
{"created_at":"Thu Oct 24 01:14:25 +0000 2013", "id":393183667205328896,"id_str":"393183667205328896",  
"text":"the marinated tofu tacos at Casa Alvarez make me want to live until I'm 100. \n#bomber #boulder","source":"\u003ca  
href=\"http://twitter.com/download/iphone\" rel=\"nofollow\" \u003eTwitter for  
iPhone\u003c/a\u003e","truncated":false,"in_reply_to_status_id":null,  
"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":74547  
732,"id_str":"74547732","name":"betsy","screen_name":"beewrks","location":"boulder, co","url":null,"description":"veggie crunchin' lady  
lover who wants to play outside, find good music, laugh with friends, and drink good  
spro.\n#sharkfacegang","protected":false,"followers_count":133,"friends_count":514, "listed_count":8,"created_at":"Tue Sep 15 20:33:29  
+0000 2009","favourites_count":96,"utc_offset":-21600,"time_zone":"Mountain Time  
(US&Canada)","geo_enabled":false,"verified":false,"statuses_count":3042,"lang":"en","contributors_enabled":false,"is_translator":false,"prof  
ile_background_color":"352726","profile_background_image_url":"http://a0.twimg.com/profile_background_images/38515621/trees.jp  
g","profile_background_image_url_https":"https://si0.twimg.com/profile_background_images/38515621/trees.jpg","profile_background  
_tile":false,"profile_image_url":"http://pbs.twimg.com/profile_images/1555215571/feet_normal.JPG","profile_image_url_https":"https://  
pbs.twimg.com/profile_images/1555215571/feet_normal.JPG","profile_link_color":"D02B55","profile_sidebar_border_color":"829D5E"  
,"profile_sidebar_fill_color":"99CC33","profile_text_color":"3E4415","profile_use_background_image":true,"default_profile":false,"default_pr  
ofile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":nul  
l,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[{"text":"bomber","indices":[78,85]},{"text":"boulder","indices":[86,94]}],"symb  
ols":[],"urls":[],"user_mentions":[]},"favorited":false,"retweeted":false,"filter_level":"medium","lang":"en"}
```

Record Types (Kayıt Tipleri)

Aynı tipde veya uzunlukta olmayan verileri (heterojen) tanımlamak için kullanılır.

Bir çalışanın ismi , çalışanın numarası , adres bilgilerini içeren bir tip tanımlanabilir.

Record Types (Kayıt Tipleri)

ADA

```
type Employee_Record_Type is record
```

```
    Employee_Name: Employee_Name_Type;
```

```
    Hourly_Rate: Float;
```

```
end record;
```



```
type Employee_Name_Type is record
```

```
    First : String (1..20);
```

```
    Middle : String (1..10);
```

```
    Last : String (1..20);
```

```
end record;
```



```
Employee_Record: Employee_Record_Type;
```

Record Types (Kayıt Tipleri)

C, *C++* ve *C#* da struct yapısını kullanarak record tipi oluşturulabilir.

.

Record Types (Kayıt Tipleri)

C, C++ ve C# da struct yapısını kullanarak record tipi oluşturulabilir.

Python ve Ruby de kayıt tipi nasıl oluşturulabilir?

?

.

.

Record Types (Kayıt Tipleri)

<http://jcsites.juniata.edu/faculty/rhodes/cs2/ch05a.htm>

```
struct NameType {  
    char first[15];  
    char middleInit;  
    char last[15];  
};
```

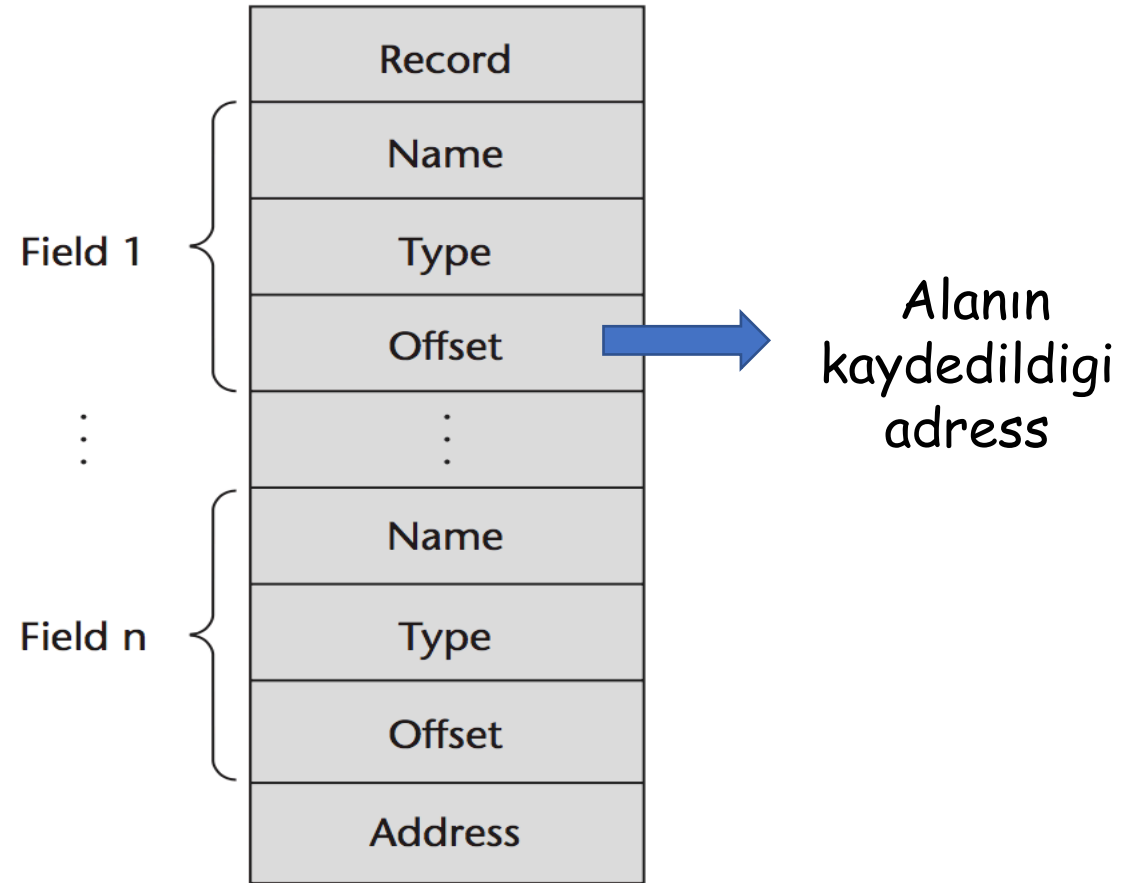
```
struct StudentType {  
    NameType name;  
    int    idNum;  
    float  credits;  
    float  gpa;  
};
```

```
StudentType student1, student2;  
student1.name.last;  
student2.name.first[0];
```

<https://www.programiz.com/cpp-programming/structure>

Record Types (Kayıt Tipleri)

Her bir alanın (field) hafıza da kaplayacağı yer kayıtın tipine göre değiştiğinden her bir alanın hafızadaki yerini tanımak için offset kullanılır



Derleme zamanı kayıt tanımı

Tuple type

- Kayıt (record) tipine benzemektedir.
- Elemanlar isimlendirilmemiştir

Tuple type (demet)

- Python tuple tipi içerir.
- Oluşturulduktan sonra içeriği değiştirilemez (immutable)
- Yazmaya karşı korumak istediğiniz veriler için kullanılabilir.
- Index ile elemanlara erişilir.

```
myTuple = (3 , 5.8 , 'elma')
```

Union Types

Program çalıştığında farklı zamanlarda farklı veri tipi tutabilen tipdir.

```
union flexType {  
    int intEl;  
    float floatEl;  
};
```

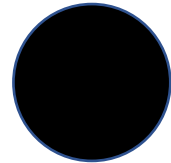
```
union flexType el1;  
float x;
```

C ve C++ **union** construct ile tanımlanır

Union Types

ADA

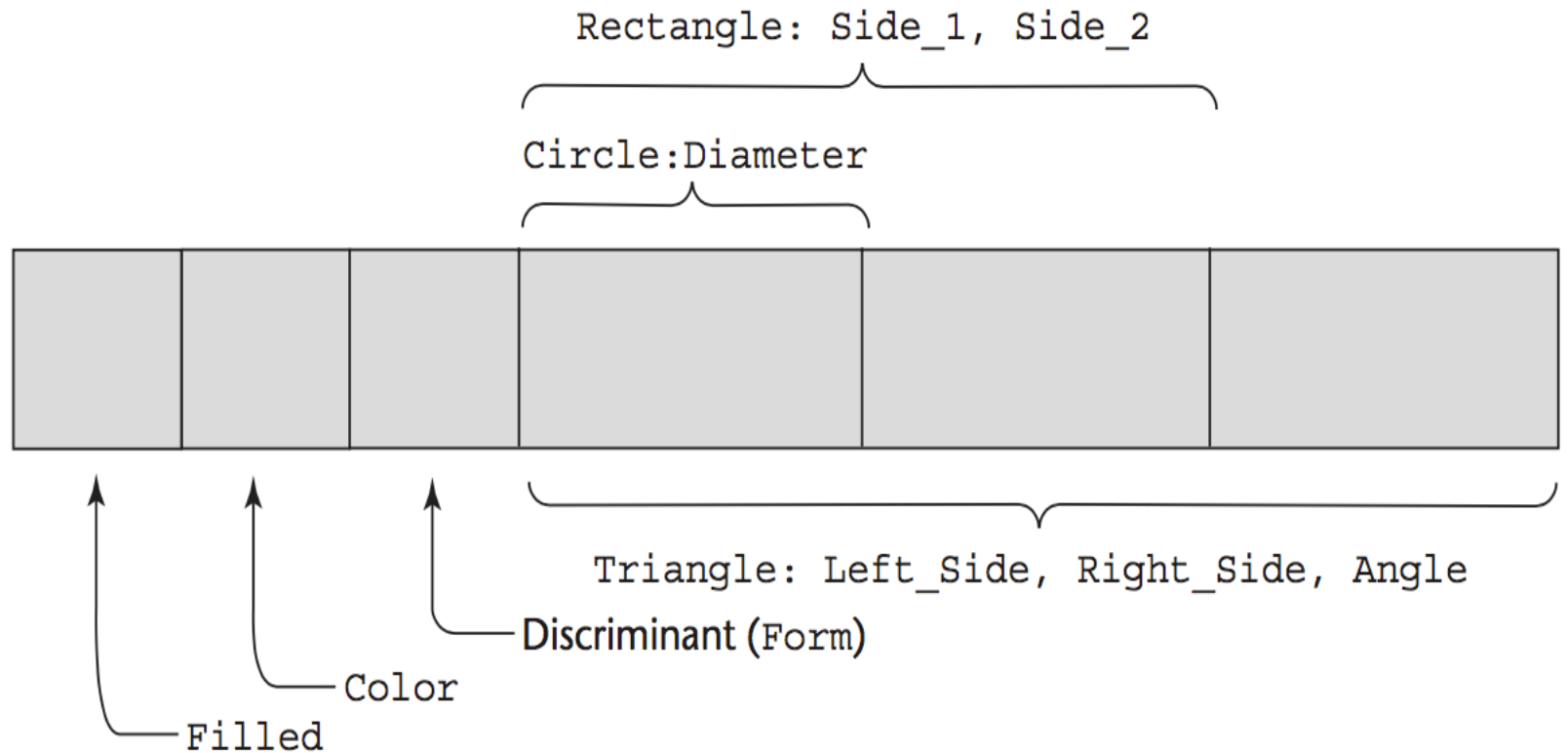
```
type Shape is (Circle, Triangle, Rectangle);  
type Colors is (Red, Green, Blue);  
type Figure (Form : Shape) is  
  record  
    Filled : Boolean;  
    Color : Colors;  
    case Form is  
      when Circle =>  
        Diameter : Float;  
      when Triangle =>  
        Left_Side : Integer;  
        Right_Side : Integer;  
        Angle : Float;  
      when Rectangle =>  
        Side_1 : Integer;  
        Side_2 : Integer;  
    end case;  
  end record;
```



Union Types

ADA

```
type Shape is (Circle, Triangle, Rectangle);
type Colors is (Red, Green, Blue);
type Figure (Form : Shape) is
  record
    Filled : Boolean;
    Color : Colors;
    case Form is
      when Circle =>
        Diameter : Float;
      when Triangle =>
        Left_Side : Integer;
        Right_Side : Integer;
        Angle : Float;
      when Rectangle =>
        Side_1 : Integer;
        Side_2 : Integer;
    end case;
  end record;
```



Cok güvenli bir yapı degildir. cunku tip kontrolu yapilmaz

Pointer Types (İşaretçi Tipi)

Pointerlar iki farklı kullanım için tasarlanmışlardır:

- Dolaylı (indirect) adresleme olanakı sağlarlar (Assembly dilinde)
- Dinamik bellek yönetimine imkan verir
 - Yığın (heap) üzerinde adres tanımlanır.

Pointer Types (İşaretçi Tipi)

Bir değişkenin hafızadaki yerine & (reference) sembolüyle erişilir.

```
int var1 = 3;
```

```
cout << &var1 << endl;
```

Pointer Types (İşaretçi Tipi)

Bir değişkenin hafızadaki yerine & (reference) sembolüyle erişilir.

```
int var1 = 3;
```

```
cout << &var1 << endl;
```

```
0x7fff5fbff8ac
```


Pointer Types (İşaretçi Tipi)

C++ pointers

Hafızaya erişip hafızada var olan değeri işlemek için kullanılır

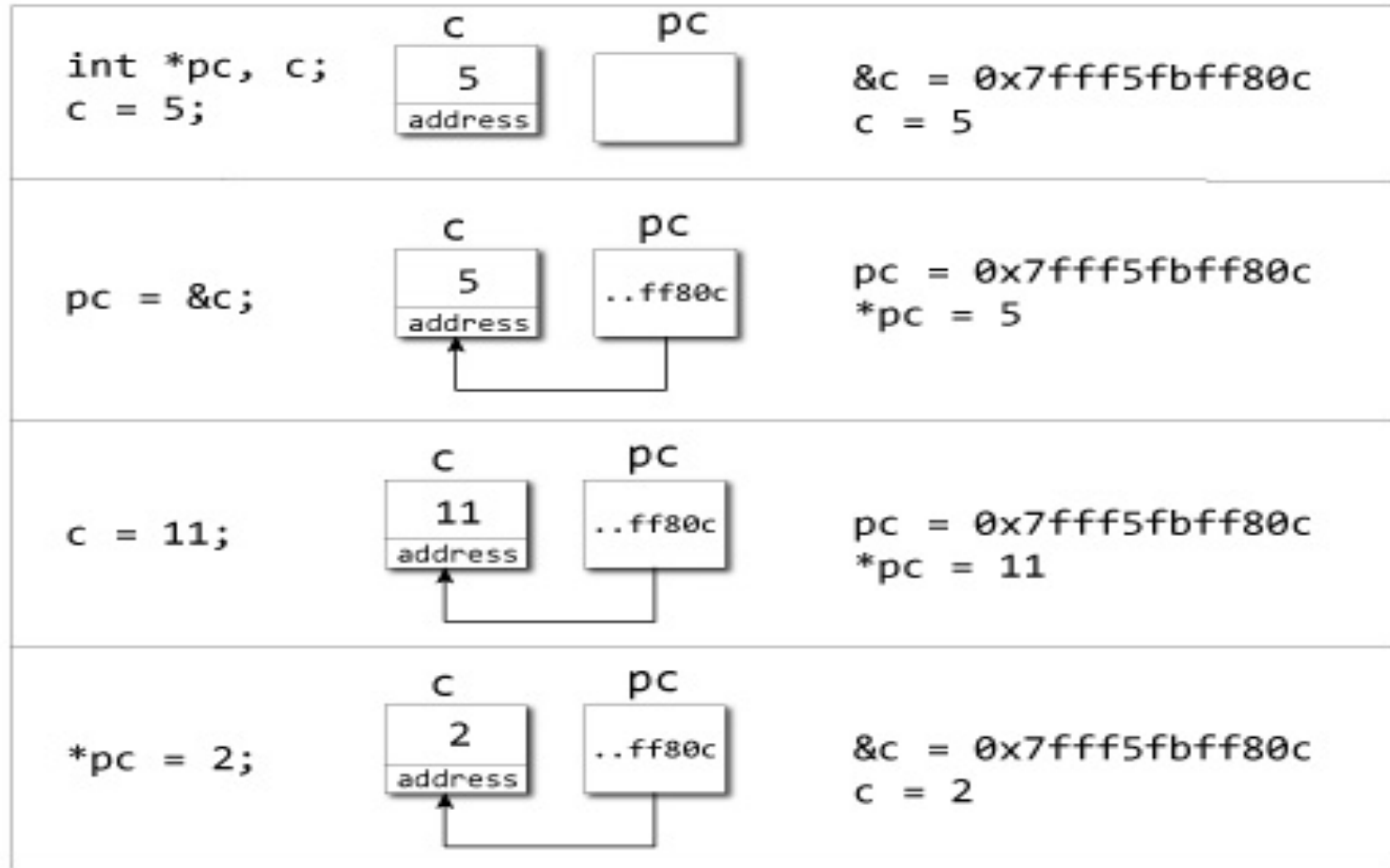
Pointer tanımlamak için

```
int *p;
```

```
int* p;
```

Pointer Types (İşaretçi Tipi)

C++ pointer



Pointer Types (İşaretçi Tipi)

Dangling (bağlı olmayan) pointer

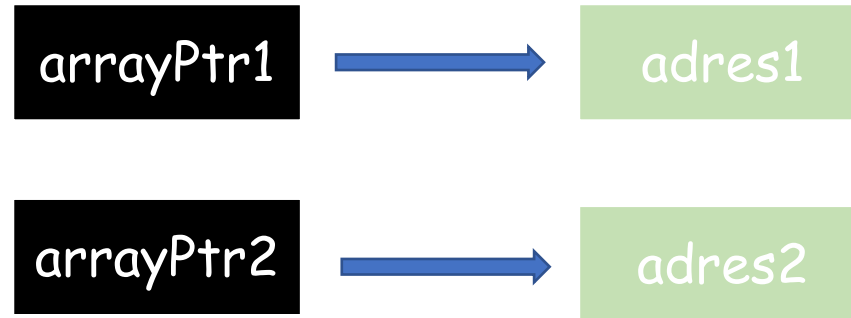
Pointerin işaret ettiği hafıza biriminin bırakılması (deallocation) ile ortaya çıkar.

Pointer Types (İşaretçi Tipi)

Dangling (bağlı olmayan) pointer

Pointerin işaret ettiği hafıza biriminin bırakılması (deallocation) ile ortaya çıkar.

```
int * arrayPtr1 = new int[100];  
int * arrayPtr2 = new int[100];
```

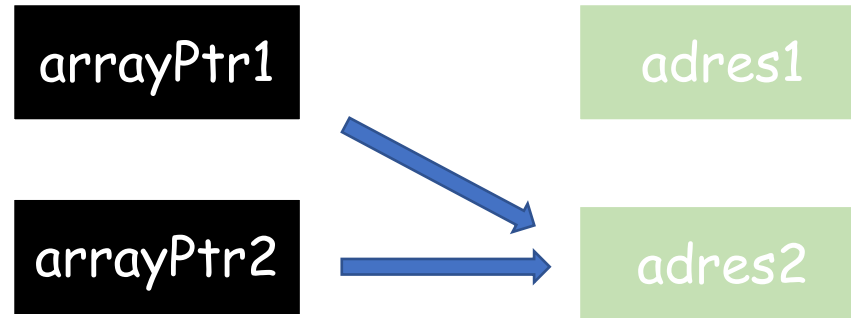


Pointer Types (İşaretçi Tipi)

Dangling (bağlı olmayan) pointer

Pointerin işaret ettiği hafıza biriminin bırakılması (deallocation) ile ortaya çıkar.

```
int * arrayPtr1= new int[100];  
int * arrayPtr2 = new int[100];  
arrayPtr1 = arrayPtr2;
```

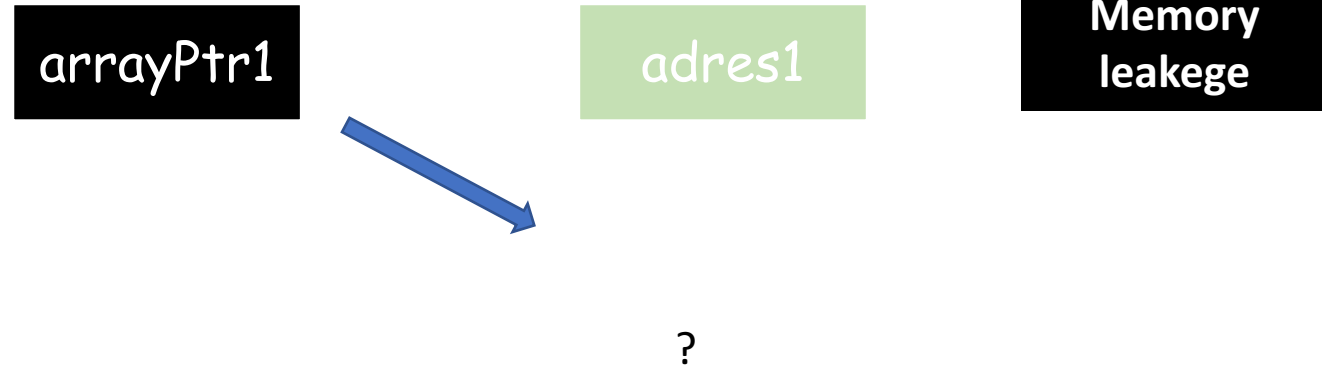


Pointer Types (İşaretçi Tipi)

Dangling (bağlı olmayan) pointer

Pointerin işaret ettiği hafıza biriminin bırakılması (deallocation) ile ortaya çıkar.

```
int * arrayPtr1 = new int[100];  
int * arrayPtr2 = new int[100];  
arrayPtr1 = arrayPtr2;  
delete [] arrayPtr2;
```



Pointer Types (İşaretçi Tipi)

Pointer problemlerinin çözümü için

1- Tombstones (Lomet, 1975)

- Fazladan bir hücre tanımlanan pointerın hafızayı kullanıp kullanmadığını belirlemek için kullanılır.
- Kullanılan hafıza alanı belleğe geri eklenince bu hücredeki değer 'nil' olur.
- Zaman ve kapladığı alandan dolayı maliyetli

Pointer Types (İşaretçi Tipi)

Pointer problemlerinin çözümü için

2- **locks-and-keys** (Fischer and LeBlanc, 1977, 1980)

- Pointer oluşturulurken (anahtar , adres) ikilisi oluşturulur
- Bir nesne oluşturulduğunda bu anahtar hem nesneye hemde pointer eklenir
- Nesne silindiğinde anahtar değeri değiştirilir

Pointer Types (İşaretçi Tipi)

C++

```
int list [10];
```

```
int *ptr;
```

```
ptr = list; ( list[0] in adresini ptr ye atadı )
```

```
*(ptr + 1) ----- list[1]
```

```
*(ptr + index) ---- list[index].
```

```
ptr[index] ----- list[index].
```

Reference Types (Referans Tipleri)

- Pointerlara benzemektedir
- Referans tipi hafızadaki değere referans oluşturur.
- Hafızadaki adres bilinmeden referansı oluşturulan veriye erişim sağlanır.
- & (ampersands) ile tanımlanırlar.

```
int result = 0;  
int &ref_result = result;  
...  
ref_result = 100;
```

Type Checking

Bir operatorun işlediği değerlerin operatöre uygun olup olmadığının kontrolü işlemidir.

- C de integer değer bekleyen bir fonksiyona float deger gönderince
 - A type error
- Javascript dinamik tip kontrolu bulunmaktadır.

Strong Typing

- 1970 li yıllarda yapısal programlama (structured programming) ile ortaya çıkmıştır.
- Bir programlama dilinde tip hataları sürekli tespit ediliyorsa o dile strongly typed denir.
- Bu işlem derleme veya çalışma zamanında bütün tip tanımlamalarının gerçekleştirilmesini gerektirir.
- Java, C# ve Ada bu sınıftandır.
- C ve C++ dahil değildir (union type dan dolayı)

Type Equivalence

- Bir biriyle uyuşabilen tipler arasında çalışma zamanında işlemlerin yapılabilmesine imkan sağlar (Compatibility)
 - Önceden tanımlı olan sayısal değerler arasındaki işlemler
 - Integer * float