Algoritmalar ve Programlama I – BLM 103

Hafta 9: İşaretçiler

Fenerbahçe Üniversitesi

9. Hafta İçeriği

- İşaretçi Tanımlaması
- İşaretçi İşlemleri
- NULL İşaretçiler

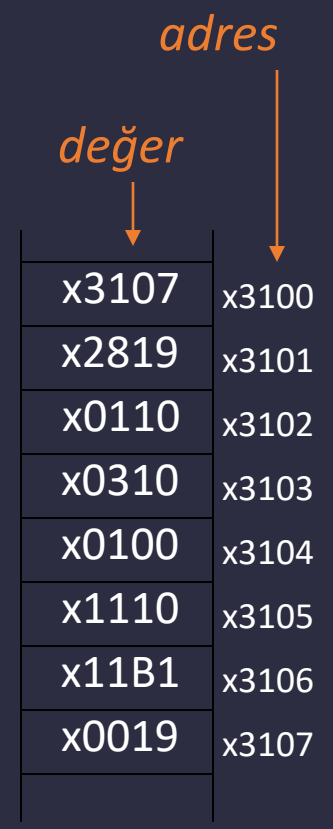
İşaretçi (Pointer) ve Diziler (Arrays)

- İşaretçi
  - Bellekteki bir değişkenin adresini tutar
  - Bellekte değiştirilmek veya okunmak istenen bir adrese erişimi sağlar.

- Diziler
  - Bellekte ard arda dizilmiş alanlardır.
  - Örneğin kullanıcıdan 100 sayı alınıp saklanmak isteniyor. Bunun için diziler kullanılabilir.

Adres ve Değer

- Tüm değişkenlerin bellekte bir adres karşılığı vardır.
- Özellikle gömülü sistemler alanında işaretçiler çok önemlidir.



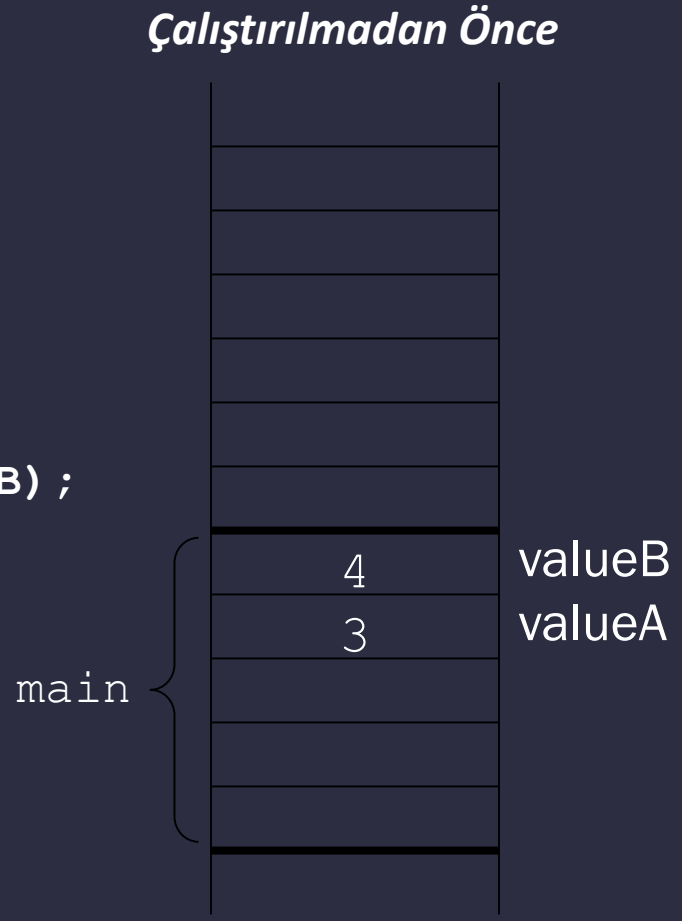
Adres Kullanımının Gerekliliği

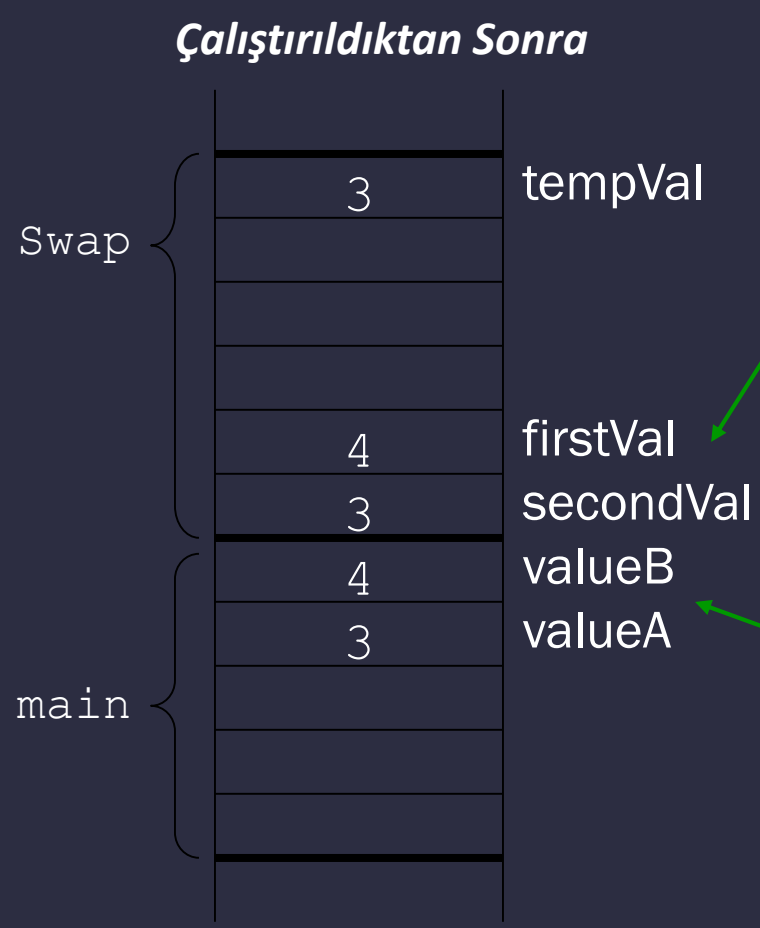
- Aşağıdaki fonksiyon kendisine verilen değişkenlerin değerlerini birbirleri ile değiştirmektedir.

- ```
  void Swap(int firstVal, int secondVal)
  {
    int tempVal = firstVal;
    firstVal = secondVal;
    secondVal = tempVal;
  }
  ```
  ```

Swap Fonksiyonunu Çalıştırılması

```
main(){
…
Swap(valueA,valueB);
}
```





C Dilinde İşaretçiler (Pointer)

- Tanımlama
- ```
  int *p;   /* p tamsayı bir işaretçidir */
  ```
- Bir veri tipi tanımlamasının sonuna * işareti konulduğunda işaretçi olarak tanımlanmış olur: `int*, double*, char* gibi…`

- Operasyonlar
- `*p` -- p işaretçisinin gösterdiği yerin, içindeki değeri döndürür
- int z = 5;

  `&z` -- z değişkeninin adresini döndürür.
  ```

Örnek

- `int i;`
- `int *ptr;`

i ismi ile değişken tanımlanmış ve ptr ismi ile bir işaretçi tanımlanmıştır.

- `i = 4;`

i değişkenine 4 atanmaktadır.

- `ptr = &i;`

i değişkenin, &i ile adresi alınıp, ptr işaretçisine atanmıştır.
Bu aşamada ptr işaretçisinin gösterdiği adres ile i'nin değerinin tutulduğu adres aynıdır.

- `*ptr = *ptr + 1;`

*ptr denerek, ptr'nin gösterdiği adresin içindeki değer alınmıştır (Biraz önce i'nin adresine atandığı için, i'nin değeri alınmaktadır. Sonuçta i'nin değeri 1 artmış olacaktır.

Argüman olarak İşaretçiler

- Bir fonksiyonun işaretçi olarak argümanı olduğunda, kendisine gönderilen değişkenin değerini değiştirebilir.

- ```
  void NewSwap(int *firstVal, int *secondVal)
  {
     int tempVal = *firstVal;
     *firstVal = *secondVal;
     *secondVal = tempVal;
  }
  ```

Argümanlar tamsayı işaretçiler
Bu fonksiyon çağrılırken, adres verilecektir.
Fonksiyon çalıştıktan sonra, fonksiyona verilen adresin içindeki değerler değiştirilmiş olacaktır.
  ```

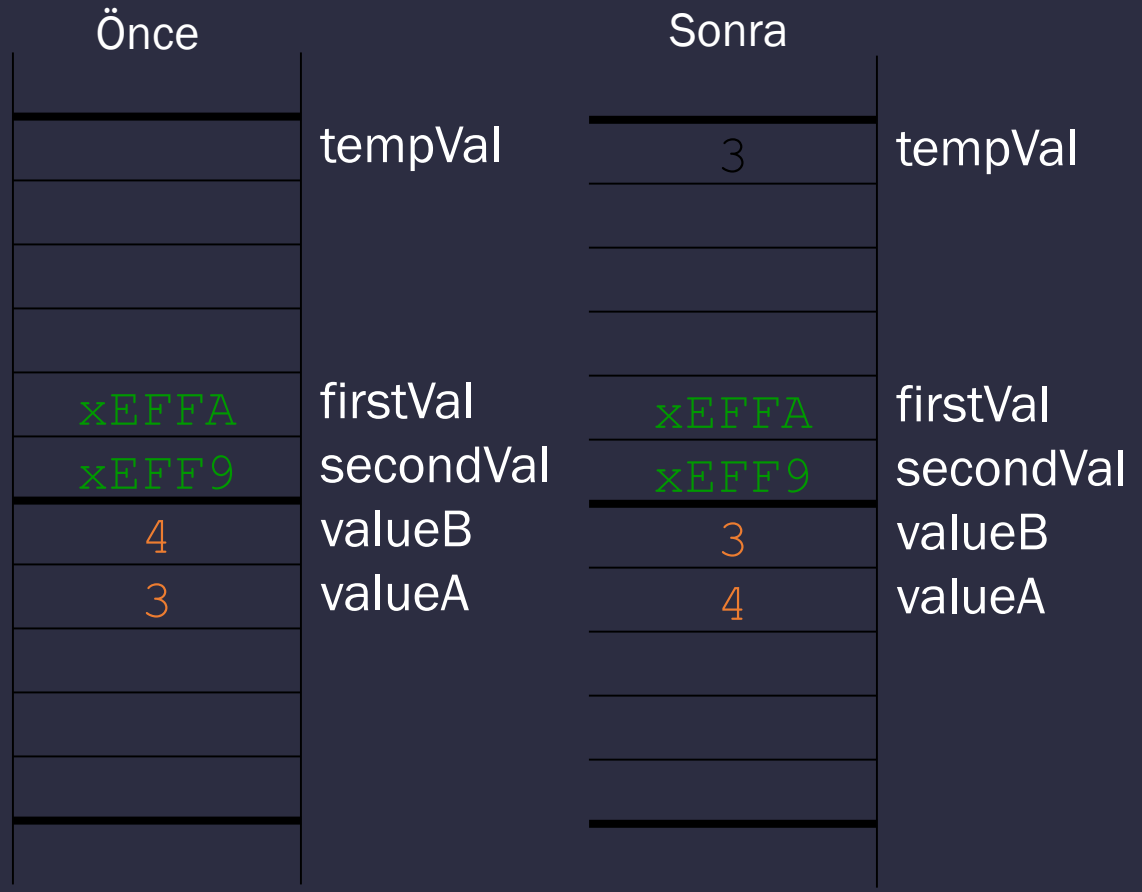
Argüman olarak İşaretçiler

- main() fonksiyonunda tanımlı olan valueA ve valueB değerlerinin içeriklerini birbirleri ile değiştirmek istiyoruz.
- NewSwap fonksiyonuna valueA ve valueB'nin adresleri gönderilmektedir:

```
NewSwap(&valueA, &valueB);
```

C dilinde İşaretçiler

- NewSwap fonksiyonunda yapılan işlemler
-
```
int tempVal = *firstVal;
*firstVal = *secondVal;
*secondVal = tempVal;
```



Null İşaretçisi

- Bir işaretçinin hiçbir yeri göstermesinin istendiği durumda NULL değeri atanır.
- Diğer bir ifade ile, işaretçi tanımlandı ancak henüz bir yeri göstermiyor.

```
int *p;
p = NULL;   /* p null işaretçisidir */
```

- `NULL` daha önceden tanımlanmış olan bir sabit sayıdır.
  - Genellikle, NULL = 0'dır. Çünkü bir çok programlama dilinde 0 erişilmesi yasaklanmış bir adrestir.

Sonucu Almak için Argüman Gönderimi

- Bir fonksiyonun bir değer hesaplanıp geri döndürmesi için return kullanılabilir.

- Diğer bir yaklaşım ise, fonksiyona argüman olarak adres verilip, fonksiyonun o adresin içerisine işlem sonucunu yazması istenebilir.

- Ayrıca bir fonksiyondan birden çok değer döndürülmesi gereken durumlarda, fonksiyona birden çok adres verilerek, içleri doldurulması istenebilir.

Sonucu Almak için Argüman Gönderimi

- Scanf'te kullanılan & işaretinin nedeni, dataIn değişkeninin içerisine doldurabilmesini sağlamak içindir.
-
```
scanf("%d ", &dataIn);
```

Kullanıcıdan tamsayı okuyup, dataIn değişkenine yazmaktadır.

İşaretçi Operasyonları için Syntax

- İşaretçi tanımlaması:

```
type* var;
```

- Değişkenin adresini almak için:

```
&var
```

- Diğer gösterimler

```
**var
```
var işaretçisinin gösterdiği yerdeki değer alınıyor (*var), bu değer, bellekte adres olarak karşılık gelen yerdeki değer alınıyor.

- `*3` 3 nolu bellek yerindeki içeriği alır.

İşaretçi Operasyonları için Syntax

- int* var;
- int abc = 5;
- 5 nolu adresin içinde ise 3 olsun.
- var = &abc;

- **var'ın değeri için;
- Önce *var hesaplanır, (*var) abc'nin içindeki değer, yani 5
  *(*var) = *(5) olur
  Yani *5 te 5 nolu adresin içindeki 3 değeri olur.

İşaretçi Örneği

- İki tamsayı alıp bölümlerinden kalan ve bölümü döndüren bir fonksiyon tanımlanmıştır.

```
int IntDivide(int x, int y, int *quoPtr, int *remPtr);

main()
{
    int dividend, divisor;
    int quotient, remainer;
    int error;
    …
    error = IntDivide(dividend, divisor, &quotient, &remainder);
    …
}
```

IntDivide için C kodu

```
int IntDivide(int x, int y, int *quoPtr, int *remPtr)
{
    if (y != 0) {
        *quoPtr = x / y;
        *remPtr = x % y;
        return 0;
    }
    else
        return -1;
}
```