

1906002132015

Programlama Dilleri Temelleri

BAİBÜ Bilgisayar Müh.

Ders 4

Dr. Öğr. Üyesi İsmail Hakkı Parlak

Kaynaklar: Watt, David A., Programming Language Design Concepts, Wiley

PLC Ders Notları, Şehitoğlu O. T., ODTÜ

Şekiller kaynak kitaptan kopyalanmıştır.

5. Prosedürsel Soyutlama

- Programlamada soyutlama (abstraction), bir program biriminin ne yaptığı ile bu program biriminin nasıl çalıştığı arasında yaptığımız ayrımı ima eder.
- Bu, program birimini kullanan programcılar ile onu uygulayan programcı arasındaki endişelerin ayrılmasını (separation of concerns) sağlar.
- Büyük projelerde hayati öneme sahiptir.

5. Prosedürsel Soyutlama

```
|7f45 4c46 0201 0100 0000 0000 0000 0000
0200 3e00 0100 0000 3004 4000 0000 0000
4000 0000 0000 0000 e019 0000 0000 0000
0000 0000 4000 3800 0900 4000 1f00 1c00
0600 0000 0500 0000 4000 0000 0000 0000
4000 4000 0000 0000 4000 4000 0000 0000
f801 0000 0000 0000 f801 0000 0000 0000
0800 0000 0000 0000 0300 0000 0400 0000
3802 0000 0000 0000 3802 4000 0000 0000
3802 4000 0000 0000 1c00 0000 0000 0000
1c00 0000 0000 0000 0100 0000 0000 0000
0100 0000 0500 0000 0000 0000 0000 0000
0000 4000 0000 0000 0000 4000 0000 0000
fc06 0000 0000 0000 fc06 0000 0000 0000
0000 2000 0000 0000 0100 0000 0600 0000
100e 0000 0000 0000 100e 6000 0000 0000
100e 6000 0000 0000 2802 0000 0000 0000
3002 0000 0000 0000 0000 2000 0000 0000
0200 0000 0600 0000 280e 0000 0000 0000
280e 6000 0000 0000 280e 6000 0000 0000
d001 0000 0000 0000 d001 0000 0000 0000
0800 0000 0000 0000 0400 0000 0400 0000
5402 0000 0000 0000 5402 4000 0000 0000
5402 4000 0000 0000 4400 0000 0000 0000
4400 0000 0000 0000 0400 0000 0000 0000
```

5. Prosedürsel Soyutlama

```
// C# DB bağlantısı
```

```
SqlConnection conn;  
conn = new SqlConnection(connectionString);  
conn.Open();
```

5.1. Fonksiyon ve Prosedürler

- Fonksiyonlar bir değer döndüren ifadeleri barındırmak; prosedürler bir dizi komutu çalıştırmak için kullanılan yapılardır.
- Fonksiyon ve prosedürler genelde 1 programcı tarafından tanımlanır (implementation); birden çok programcı tarafından kullanılır (application).
- Tanımlayıcı istenen sonucun nasıl elde edileceğiyle ilgilenir; uygulayıcı ise p/f gözlemlenebilen davranışı ile...
- Class methodları da bu başlıkta değerlendirilebilir.

5.1. Soyutlama

- Soyutlama (abstraction): bir programı ya da program bloğunu paketleyerek onu tekrar tekrar kullanılabilir hale getirmektir.
- Yani implementasyon ve kullanılışı birbirlerinden ayırmaktır.
- Sindirim sistemi - yemek yemek.
- İdüksiyon akımı - karaoke.
- Potansiyel hataları ve karmaşıklığı azaltır.
- Tekrar kullanılabilirliği (reusability) ve sürdürülebilirliği (maintainability) sağlar.

5.1.1 Soyutlama İlkesi

- Bir fonksiyon, bir ifade (expression) üzerinde soyutlama yapar.
- Bir prosedür, bir komut (command) üzerinde soyutlama yapar.
- Soyutlama ilkesi: Yalnızca sözdizimsel (syntactic) kategorideki yapıların bir tür hesaplama belirtmesi şartıyla, herhangi bir sözdizimsel kategori üzerinde soyutlayan prosedürler/fonksiyonlar tasarlamak mümkündür. Örneğin, değişken erişimi bir değişkeni refere eder. Değişken erişimleri soyutlayan yeni bir prosedür çağrıldığında bir değişken ortaya çıkaracaktır. Bu tür prosedürlere seçici prosedürler adı verilmektedir.

5.1.1 Soyutlama İlkesi

Varlık	Soyutlama
İfade (Expression)	Fonksiyon
Komut (Command)	Prosedür
Seçici (Selector)	Seçici Fonksiyon
Deklarasyon	Genel (Generic)
Komut Bloğu	İteratör

5.1.2 Soyutlama İlkesi (Seçiciler)

```
int sayilar[] = {1, 2, 3};  
int ikinciSayi = sayilar[1];
```

```
struct List {  
    int data;  
    List *next;  
  
    int & operator[] (int el) {  
        int i;  
        List *p = this;  
  
        for (i = 1; i < el; i++)  
            p = p->next;  
  
        return p->data;  
    };  
};  
  
List h;  
...  
  
h[1] = h[2] + 1;
```

5.1.3 Soyutlama İlkesi (Generic T)

```
class node<T> {  
    T data;  
    node<T> next;  
    node(T data) {  
        this.data = data;  
        this.next = null;  
    }  
}  
...  
node<Integer> n = new node<>(5);
```

5.1.4 Soyutlama İlkesi (İteratör)

```
class EvenNumbers:
    last = 0

    def __iter__(self):
        return self

    def __next__(self):
        self.last += 2
        if self.last > 8:
            raise StopIteration
        return self.last

even_numbers = EvenNumbers()

for num in even_numbers:
    print(num)
```

5.2. Parametre ve Argümanlar

- Argüman, bir prosedüre iletilen bir değer veya başka bir varlıktır.
- Gerçek parametre (actual parameter), bir argüman veren bir ifadedir (expression).
- Biçimsel parametre (formal parameter), bir prosedürün bir argümana erişebileceği bir tanımlayıcıdır (identifier).

```
int ilgincTopla (int sayi1, int sayi2) {  
    sayi1 = sayi2;  
    return sayi1 + sayi2;  
}  
int ilgincToplam = ilgincTopla (3, 5);
```

- Gerçek parametreler: 3, 5
- Biçimsel parametreler: sayi1, sayi2

5.2. Parametre ve Argümanlar

- Gerçek (actual) ve biçimsel (formal) parametreler arasındaki ilişkinin kurulma şekline kullanılan dil karar verir.
- Bu ilişkinin kurulması parametre mekanizması (parameter mechanism) terimiyle belirtilir.
- Parametre mekanizmaları iki temel başlık altında incelenebilir:
 - **Kopya parametre mekanizması:** bağımsız değişkenin bir kopyasını içeren biçimsel parametreyi yerel değişkene bağlar.
 - **Referans parametre mekanizması:** biçimsel parametreyi doğrudan argümanın kendisine bağlar.

5.2.1 Kopya Parametre Mekanizmaları

- Copy-in: Fonksiyon çağrıldığında, yerel bir değişken oluşturulur ve argüman değeriyle başlatılır. Prosedürün içinde, bu yerel değişken incelenebilir ve hatta güncellenebilir. Ancak, yerel değişkenin herhangi bir güncellemesinin prosedür dışında bir etkisi yoktur. (**Pass by value**)

5.2.1 Kopya Parametre Mekanizmaları

- Copy-in

```
// C
int arttir (int sayi) {
    sayi = sayi + 1;
    return sayi;
}

int main(void) {
    int a = 5;
    int b = arttir(a);
    printf("%d, %d", a, b);
}
```

5.2.1 Kopya Parametre Mekanizmaları

- Copy-out: Prosedür çağrıldığında, yerel bir değişken oluşturulur ancak bir değer atanmaz. Prosedür döndüğünde (return), o yerel değişkenin son değeri argüman değişkenine atanır.
- Copy-in-out: Prosedür çağrıldığında, yerel bir değişken oluşturulur ve argümanın değeri ile başlatılır. Prosedür döndüğünde, o yerel değişkenin son değeri, argüman değişkenine geri atanır.

5.2.1 Kopya Parametre Mekanizmaları

- Copy-out

// C benzeri kod örneği

```
int x = 3, y = 2;
```

```
fun(int a, int b) {
```

```
    x = a + b;
```

```
    ++a;
```

```
    b = 3 * a;
```

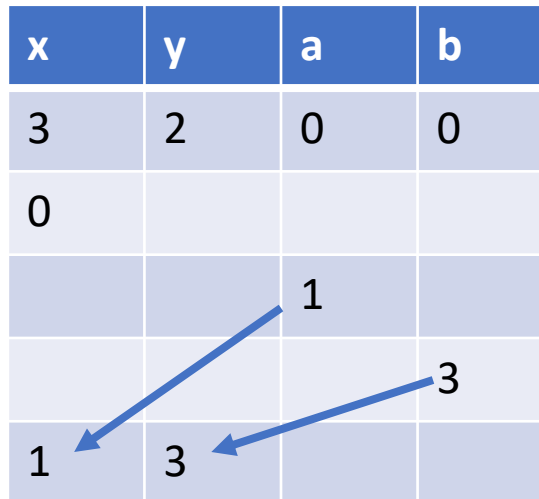
```
}
```

```
main() {
```

```
    fun(x, y);
```

```
    print(x, y); // 1, 3
```

```
}
```



5.2.1 Kopya Parametre Mekanizmaları

- Copy-in-out

// C benzeri kod örneği

```
int x = 3, y = 2;
```

```
fun(int a, int b) {
```

```
    x = a + b;
```

```
    ++a;
```

```
    b = 3 * a;
```

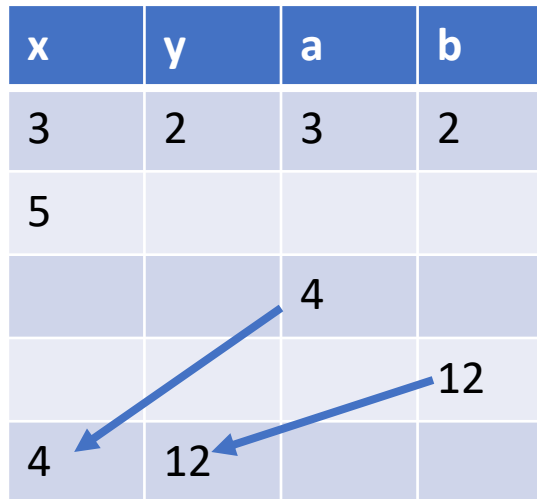
```
}
```

```
main() {
```

```
    fun(x, y);
```

```
    print(x, y); // 4, 12
```

```
}
```



C, C++ ve JAVA yalnızca copy-in parametrelerini destekler; ADA dili, üç kopyalama parametresi mekanizmasını da destekler.

5.2.2 Referans Parametre Mekanizması

- Referans parametre mekanizması, biçimsel parametre *FP*'nin doğrudan argümanın kendisine bağlanmasına izin verir.
 - Sabit (constant) parametre bağlanması: FP, fonksiyonun aktivasyonu sırasında sabit argümana bağlanır. FP incelendiğinde, argümanın değeri incelenmiş olur.
 - Değişken (variable) parametre bağlanması (**pass by reference**): FP incelendiğinde veya güncellendiğinde argüman da incelenmiş veya güncellenmiş olur. C dilinde referans parametre mekanizması doğrudan yoktur. Ancak pointer'lar aracılığıyla bu etki sağlanabilir. C++ pass by reference mekanizmasını destekler.

5.2.2 Referans Parametre Mekanizması

- Pass by reference (C++)

```
int arttir (int &sayi) {  
    sayi++;  
    return sayi;  
}
```

```
int main() {  
    int a = 3;  
    int b = arttir(a) + 1;  
    cout << a << ", " << b << endl; // 4, 5  
}
```