3 Funzioni

Definizione 3.1 (Principio di astrazione)**.** *Ridurre la duplicazione di informazione nei programmi utilizzando* ***funzioni*** *definite dal programmatore e quelle disponibili nelle librerie standard. Facilita la* manutenzione *e* comprensione *del codice.*

```
1   var trovatoA : Bool = false;
2   var trovatoB : Bool = false;
3   i = 1;
4   while (i<=n && !trovatoA) {
5     if (A[i] == k) trovatoA = true;
6     else i = i + 1;
7   }
8   while (i<=n && !trovatoB) {       // Uguale al ciclo precedente se non per l'array
9     if (B[i] == k) trovatoB = true;
10    else i = i + 1;
11  }
12  if (trovatoA && trovatoB) {
13    C
14  }
```

Listing 3: Esempio di codice astraibile

Possiamo semplificarlo tramite la creazione della seguente funzione:

```
1   var trovatoA : Bool = false;
2   var trovatoB : Bool = false;
3   func seqSearch(array:[Int], k:Int) -> Bool {
4     var trovato = false;
5     var i = 1;
6     while (i<=n && !trovato) {
7       if (array[i] == k) {trovato = true}
8       else {i = i+1};
9     }
10    return trovato;
11  }
12  if (trovatoA && trovatoB) {
13    C
14  }
```

Listing 4: Esempio di funzione

3.1 Anatomia di una funzione

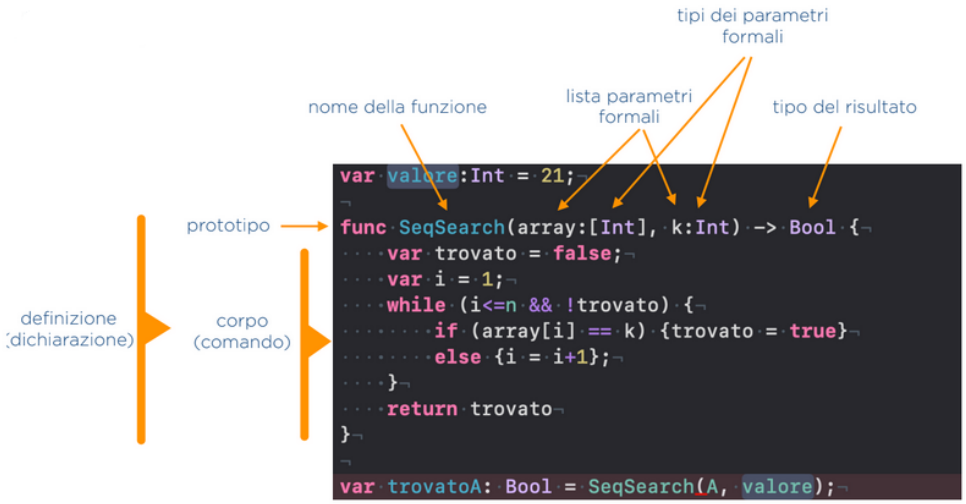

Figure 4

Il **compilatore** dovrà poi eseguire le seguenti verifiche:

- il **numero** di parametri *attuali* deve coincidere con quello dei parametri *formali*
- i **nomi** dei parametri *formali* devono essere tutti distinti
- i **tipi** degli *attuali* e dei *formali* nella stessa posizione devono essere uguali
- non ci devono essere **variabili libere** nel corpo della funzione che non possono essere legate
- deve esserci un **return statement** nel corpo della funzione
- il **tipo** dell'espressione nel *return statement* deve coincidere con quello della dichiarazione

3.1.1 Ambiente statico

Definizione 3.2 (Principio di corrispondenza)**.** *Parti di programma che hanno effetti simili devono avere una sintassi simile. Facilità di apprendimento del linguaggio e di interpretazione dei programmi.*

Definizione 3.3 (Scoping statico)**.** *Le **variabili libere** nel corpo delle funzioni vengono legate a tempo di compilazione costruendo le **chiusure**.*

Definizione 3.4 (Chiusura)**.** *Ciò che viene registrato nell'**ambiente dinamico** al momento dell'elaborazione della **dichiarazione** di funzione, associando al nome della funzione tutto ciò che sta alla sinistra.*

La dichiarazione di funzione genera nell'**ambiente dinamico** un legame tra il **nome della funzione** e una'**astrazione** che contiene tutte le informazioni necessarie ad eseguire la chiamata della funzione.

$$[(nomeFunzione), \lambda(parametriformali).[(\textbf{variabili libere})], C] \tag{1}$$

```
1   var COVID19:Bool = true;
2
3   var mioCosto:Double = 100;
4   var aliquota:Double = 21;
5
6   // Chiusura
7   // [(calcolaIVA, λ(let costo) . {[(aliquota, 12)]; return costo*aliquota/100})]
8
9   func calcolaIVA(let costo:Double) -> Double {
10    return costo*aliquota/100;
11  }
12
13  if (COVID19) {
14    var aliquota:Double = 23;
15    print(calcolaIVA(mioCosto));
16  } else {
17    print(calcolaIVA(mioCosto));
18  }
```

Listing 5: Esempio di scoping statico

3.1.2 Ambiente dinamico

Le **variabili libere** vengono legate a tempo di **esecuzione** quando vengono utilizzate. Nella chiusura della funzione registro solo il suo corpo al momento della dichiarazione.

```
1   var COVID19:Bool = true;
2
3   var mioCosto:Double = 100;
4   var aliquota:Double = 21;
5
6   // Chiusura
7   // [(calcolaIVA, λ(let costo) . {return costo*aliquota/100})]
8
9   func calcolaIVA(let costo:Double) -> Double {
10    return costo*aliquota/100;
```

```
11    }
12
13    if (COVID19) {
14      var aliquota:Double = 23;
15      print(calcolaIVA(mioCosto));
16    } else {
17      print(calcolaIVA(mioCosto));
18    }
```

Listing 6: Esempio di scoping dinamico

IMPORTANTE: la semantica statica ha senso solamente in presenza di uno **scoping statico**, il quale a differenza di quello dinamico ci garantisce la mancanza di errori. Di conseguenza in presenza dello **scoping dinamico** MAIverificare la correttezza sintattica.

3.2 Passaggio dei parametri

3.2.1 Per valore

Viene fatta una copia degli identificatori passati come parametri attuali tra le variabili locali del corpo della funzione. Non modifico il valore esterno di questi identificatori.

3.2.2 Per indirizzo