1 Introduzione

1.1 Puntatori

Gli indirizzi di memoria delle variabili sono interi (rappresentati in esadecimale) che contano i byte a partire dalla posizione 0x0000000. Gli indirizzi di memoria possono essere memorizzati in variabili come ogni altro intero.

Definizione 1.1 (Puntatori)**.** *Definiamo le variabili che memorizzano indirizzi di memoria come* ***puntatori****.*

1.1.1 Operatori sui puntatori

Sono due i principali operatori che possono essere usati con i puntatori.

- **(&) - Operatore indirizzo**. L'operatore di indirizzo è unario[1] e restituisce l'indirizzo di memoria dell'operando (può essere anche un altro puntatore, in questo caso restituisce l'indirizzo in cui è memorizzato il puntatore, cioè l'indirizzo di memoria di una variabile).
- **(*) - Operatore di indirezione o dereferenziazione**. L'operatore di indirezione è unario e restituisce il valore dell'oggetto a cui punta l'operanda.

Note 1.1.1. *Nota che & e * sono uno l'inverso dell'altro, quindi: &*aPtr == *&aPtr.*

Esempio 1.1. Di seguito un esempio di utilizzo di puntatori con anche i vari operatori.

```
1  var a:Character = 'z', b:Character = 'h';
2  ref aPtr:Character = nil                          //0x0
3  aPtr = 0;                                         //0x0
4  aPtr = &a;
5  print(&a, aPtr);                                  //0x7ffeefbff60f, 0x7ffeefbff60f
6  print(*aPtr, a);                                  //z, z
7  print(&aPtr);                                     //0x7ffeefbff60f
8  *aPtr = b;
9  print(*aPtr, a, b);                               //h, h, h
10 print(&b, &a, aPtr);                              // 3 volte 0x7ffeefbff60f
11 ref altro_aPtr:Character = &a;
12 print(altro_aPtr, *altro_aPtr);                   //0x7ffeefbff60f, h
```

Listing 1: Esempio puntatori e operatori sui puntatori

Descrizione esempio: Osservando l'esempio sopra 1.1 possiamo vedere alle righe (11) e (12) che abbiamo una dichiarazione di un nuovo puntatore che punta alla stessa cella di memoria di "aPtr", abbiamo quindi più di un puntatore che punta alla stessa variabile.

Possiamo vedere che le locazioni di memoria sono numeri interi che individuano la posizione della cella di memoria (sono numeri interi scritti in esadecimale, ma sempre numeri interi), è quindi possibili effettuare operazioni aritmetiche sui puntatori, cioè è possibile manipolare tramite operazioni aritmetiche le locazioni.

Esempio 1.2. Se per esempio prendiamo un ambiente ed una memoria così composti:

$$\rho = [(aPtr, l2)] \qquad \sigma = [(l1, 13), (l2, 23)]$$

Abbiamo quini un puntatore "aPtr" che punta alla locazione l1, il quale contiene il numero 13, più una posizione l2, successiva alla l1, che contiene 23.
Se ipotizziamo che il nostro sistemi archivi le informazioni con una base di 32 bit [2] ed andiamo

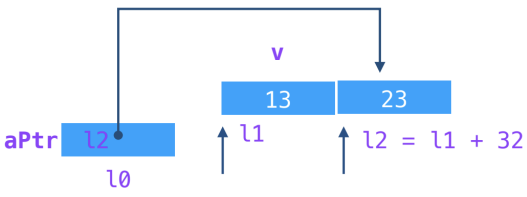

Figure 1: Operazioni algebriche su puntatori

[1] Unario vuol dire che agisce su una sola variabile
[2] Questo vuol dire che ogni vaore è salvato con 32 bit, quindi ogni 32 ci sarà un nuovo valore archiviato

a sommare 32 a "aPtr" succederà che ci sposteremo di 32 posti nella memoria raggiungendo l2, quindi "aPtr" = l2.

1.2 Strutture dati

Definizione 1.2 (Struttura dati)**.** *Una* ***struttura dati*** *è un formato che serve ad organizzare e memorizzare dati in modo da renderli agevolmente disponibili agli algoritmi che li manipolano.*

Alcune caratteristiche delle strutture dati:

- Una struttura dati è detta **omogenea** se contiene dati tutti dello stesso tipo. Altrimenti è **disomogenea**.
- Una struttura dati è **statica** se la sua dimensione non varia durante l'esecuzione del programma. Altrimenti è detta **dinamica**.
- Una struttura dati è **lineare** se i dati sono organizzati come sequenze di valori. Altrimenti è detta **non lineare**.

Una struttura dati è inoltre caratterizzata dalle operazioni elementari disponibili per inserire, reperire e modificare i dati che memorizza.