Tipi di Dato

R.S

April 15, 2025

Tipi di dato composti

Record

Sono una collezione di campi di tipi diversi selezionabili con il proprio nome
In C e in Java si rappresentano come:

- Struct in C:

```c
struct studente {
  char nome[20];
  int matricola; };

  struct studente s;
  s.nome = "Mario"; // errore in C
  strcpy(s.nome, "Mario");
  s.matricola=343536;
```

- Classe in Java:

```java
class Studente {
  public String nome;
  public int matricola;
};

Studente s = new Studente();
s.nome = "Mario";
s.matricola = 343536;
```

Sono memorizzabili e denotabili ma non possono essere definiti senza identificatore (solo in Ada).
Le struct sono esprimibili **soltanto** in Scheme → liste con funzioni di accesso ai campi e funzioni di test del valore del campo:

```
(define (book title authors) (list book title authors))
(define (book-title b) (car (cdr b)))
(define (book-? b) (eq? (car b) book))
```

I campi del record sono memorizzati sequenzialmente e sono allineati;
Si possono disallineare ma l'accesso risulta più complesso → richiede l'assembler.
Si possono riordinare i campi per risparmiare spazio ma bisogna preservare l'ordinamento nel tempo.

Record varianti e Union

- record in cui solo alcuni campi sono attivi in un determinato istante
- alcuni campi sono alternativi tra loro → i campi alternativi possono condividere la stessa locazione di memoria
- possibili varianti → alloco la dimensione massima che posso ottenere
  a seconda della configurazione uso la locazione di memoria per rappresentare tipologie di dato differenti
- rappresentazione:

```
type studente = record
nome : packed array [1..6] of char;
matricola: integer;
case fuoricorso : Boolean of
  true: (ultimoanno: 2020..maxint);
  false:(anno:(primo,secondo,terzo);
        inpari:Boolean;) end;
var s: studente;
s.fuoricorso := true;
s.ultimoanno:= 2021;
```

- i tipi unione possono essere utili per risparmiare spazio o per rappresentsre in modo naturale alcuni tipi come le liste o gli alberi binari (es: alberi con solo un sottoalbero)
  in linguaggi come Rust i tipi unione vengono etichettati e viene definito il codice che gestisce ogni possibile opzione
- con i tipi unione si possono aggirare controlli sul tipo:

```
union Data {
  int i;
  float f;
  char str[20];
} data;
float y;
...
data.str = "abcd";
y = data.f;
```

Array

- insiemi di dati omogenei contraddistinti da un indice (array di caratteri → stringhe)
  possono anche essere multidimensionali, con più indici (array di array)
- posso implementare l'operazione di selezione (A[i]) e in alcuni linguaggi posso anche avere lo slicing per selezionare parti contigue all'interno di un singolo array;
  sono memorizzati in locazioni contigue (singola dimensione) oppure in ordine di riga/colonna (multidimensionali)
- forma:
  - statica → definita a compile time e fissata alla dichiarazione dell'array
  - dinamica → varia durante l'esecuzione e usa il **Dope Vector**
  - Dope Vector → descrittore che contiene:
    - puntatore all'inizio, limite inferiore e occupuazione di ogni dimensione dell'array
    - memorizzato nella parte fissa del record di attivazione
  - accesso al vettore → calcolo dell'indirizzo base a runtime usando le informazioni del Dope Vector