Tipi *struct* e liste in linguaggio C

Tipi di dato semplici

Abbiamo utilizzato esclusivamente tipi di dato *elementari:*

- tipi numerici: `int, float, ...`
- array e puntatori

Tipi di dato semplici

Abbiamo utilizzato esclusivamente tipi di dato *elementari:*

- tipi numerici: `int, float, ...`
- array e puntatori

Nella pratica spesso vorremmo rappresentare oggetti più complessi, dotati di struttura:

Tipi di dato composti...

Esempio: punti nel piano cartesiano

Un punto è una coppia di coordinate intere (x, y)

- Rappresentazione mediante tipi elementari:

```
int x, y;
```

- *Svantaggi*:
  - Difficoltà di comprensione: variabili apparentemente scollegate
  - Difficile dichiarare, allocare o passare come parametri oggetti complessi
  - Molto complicata la gestione dei puntatori

Le *struct*

Sintassi per la definizione di tipi composti:

```
struct Punto{

  int  x;
  int  y;

};
```

L'effetto è quello di introdurre un nuovo tipo `Punto`, composto da due interi `x` e `y`

Utilizzo

```
struct Punto{
  int  x, y;
};


main() {

 struct Punto p1;      // dichiarazione
 struct Punto p[100];

 p1.x = 100;           // accesso ai campi con '.'
 p1.y = p1.x;
 p[10].y=0;

}
```

Scorciatoia con *typedef*

```
typedef struct __Punto{
  int  x, y;
} Punto ;


main() {

 Punto p1;                // dichiarazione
 Punto p[100];

 p1.x = 100;             // accesso ai campi con '.'
 p1.y = p1.x;
 p[10].y=0;

}
```

Puntatori a struct

```
typedef struct __Punto{
   int  x, y;
} Punto ;


main() {

 Punto *p1;        // dichiarazione
 ....

 p1->x = 100;     // accesso ai campi con '->'
 p1->y = p1->x;

}
```

struct e tipi ricorsivi

E' ammissibile che uno dei campi di una struct sia un puntatore allo stesso tipo di struct :

```
struct Punto{

 int  x;
 int  y;
 struct Punto* ptr; //funziona solo coi puntatori!

};
```

Ciò è utile nell'implementazione di tipi di dato ricorsivi: come liste, alberi, etc...

Liste in linguaggio C

Liste

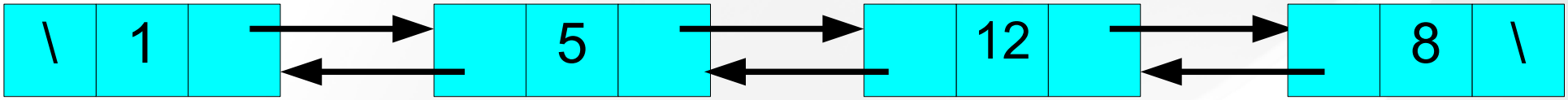
Una lista è una struttura dati costituita da una collezione di nodi collegati da puntatori:

In una lista **singolarmente linkata**, ciascun nodo è costituito da:

- Il valore di una chiave
- Un puntatore al nodo successivo nella lista (o NULL)

Liste

Una lista è una struttura dati costituita da una collezione di nodi collegati da puntatori:



In una lista **doppiamente linkata**, ciascun nodo è costituito da:

- Il valore di una chiave
- Un puntatore al nodo successivo nella lista (o NULL)
- Un puntatore al nodo precedente nella lista (o NULL)

Liste singolarmente linkate in C

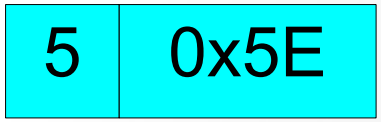
```
// definizione del tipo Nodo

struct Nodo{
  int  key;             // chiave intera
  struct Nodo* next; // puntatore al prox. nodo
};


// Una lista è rappresentata da un puntatore
// al suo nodo di testa (o NULL se vuota)

typedef Nodo* Lista;
```

Liste singolarmente linkate in C

```
// Creazione di una lista di due nodi

struct Nodo* n1= malloc(sizeof(Nodo));
struct Nodo* n2= malloc(sizeof(Nodo));
Lista l;

n1->key= 5; n1->next = n2;
n2->key= 14; n2->next = NULL;

l = n1;
```

Rappresentazione in memoria

n1: 0x2F



n2: 0x5E

I : 0x2F

Nota

Lettura consigliata per tutti:

http://it.wikipedia.org/wiki/Indentazione