Tipi *struct* e liste in linguaggio C

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;
 struct Punto p[100];

 p1.x = p2.y = 100;
 p[10].y= 0;
}
```

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

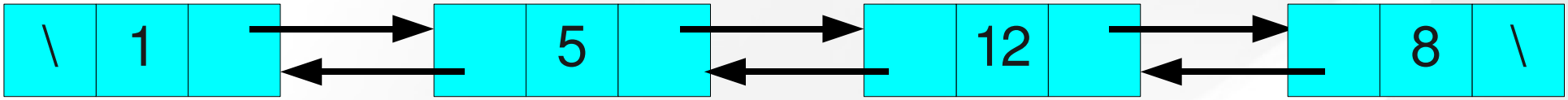
Una lista è una struttura dati costituita da una collezione di nodi collegati da puntatori:

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

- Il valore di una chiave
- Un puntatore al nodo successivo nella lista

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
- Un puntatore al nodo precedente nella lista

Liste di interi 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
// alla suo nodo di testa

typedef Nodo* Lista;
```

Liste di interi in C

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

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

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

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

l = n1;
```

Tabelle Hash

Tabelle Hash

Soluzione standard per il problema del *dizionario dinamico:*

- **Insert**(S,x) : aggiungi chiave x ad S
- **Search**(S,x): determina se x appartiene ad S

Tabelle Hash

Soluzione standard per il problema del *dizionario dinamico:*

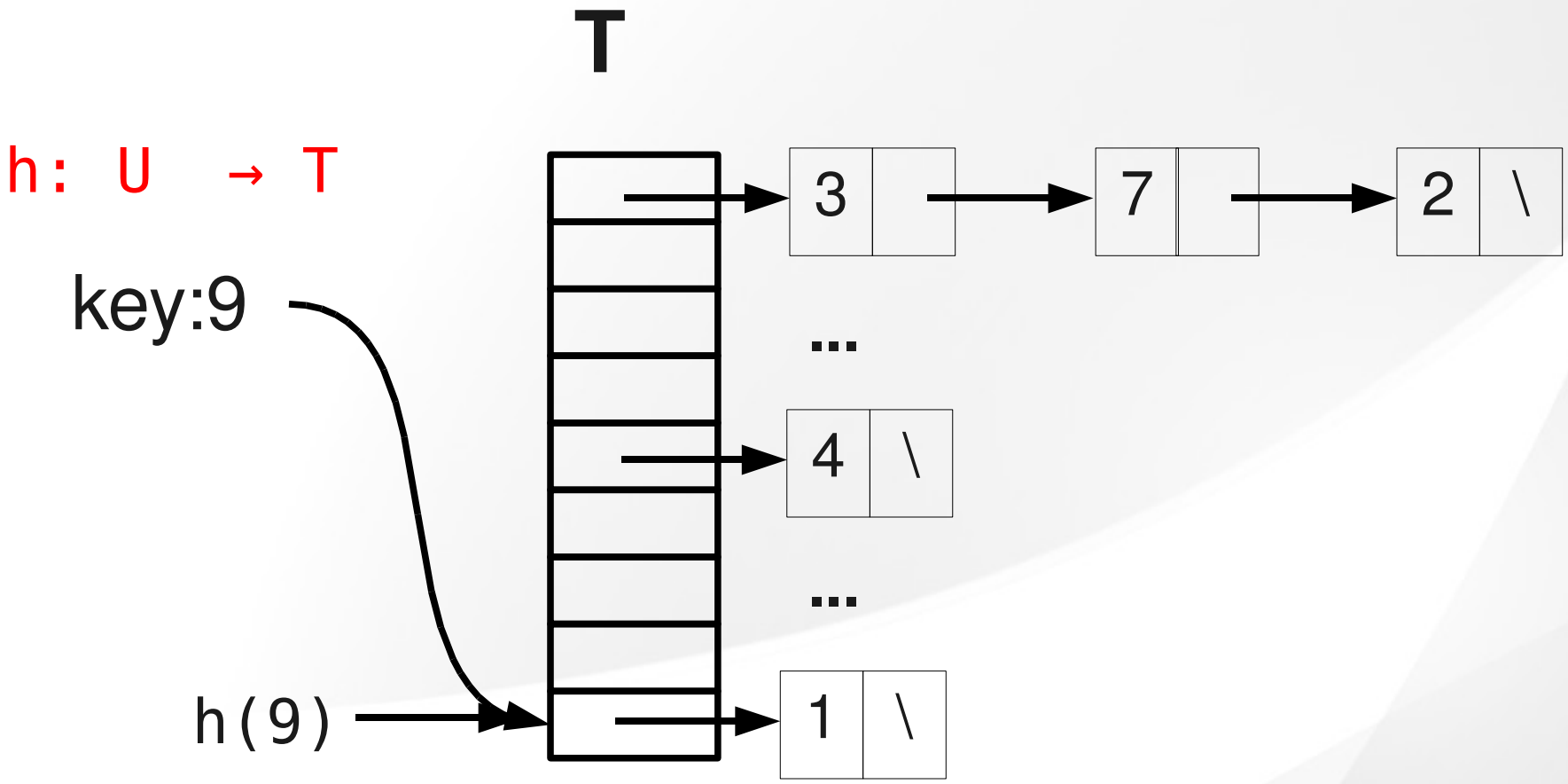
- **`Insert`**`(S,x)` : aggiungi chiave x ad S

- **`Search`**`(S,x)`: determina se x appartiene ad S

Gestione dei conflitti tramite:

- Liste di trabocco
- ...

Inserzione/Ricerca



Inserzione/Ricerca