

Inversione di una lista

1. Introduzione e requisiti del problema
2. Specifica
3. Progetto della soluzione
4. Codifica

1. Introduzione e requisiti del problema

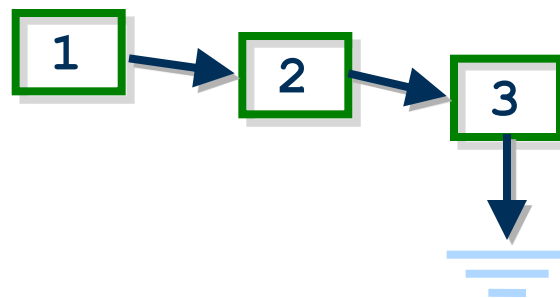
Requisiti del problema

Si realizzi una funzione C++ che riceva in ingresso una lista di interi e restituisca una lista contenente gli stessi numeri, ma in ordine inverso rispetto alla lista di ingresso.

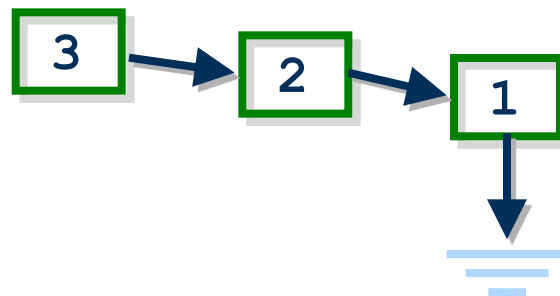
In questo esercizio si studierà come invertire una lista dinamica di interi. Tale esercizio ci consentirà di studiare le strutture dinamiche e il loro utilizzo.

La specifica chiede di scrivere una funzione per l'inversione di una lista di interi.

Ad esempio data la seguente lista:



Si vuole ottenere la seguente lista:



La specifica non indica come deve avvenire l'inversione.

Esistono diverse soluzioni che verranno esaminate in questo esercizio.

Casi di test


caso 1: inserimento di una lista

Input: Quanti valori si vogliono inserire:3
Inserire il termine 0:1
Inserire il termine 1:2
Inserire il termine 2:3

Output: Stampa della lista
3
2
1

caso 2:


Input: Quanti valori si vogliono inserire:0
Output: Lista vuota




```
Definizione dati
struct Nodo {
    int valore;
    Nodo *nextPtr;
};
```

Definizione dati


```
struct Nodo {  
    int valore;  
    Nodo *nextPtr;  
};
```



```
Definizione dati  
struct Nodo {  
    int valore;  
    Nodo *nextPtr;  
};
```



```
Definizione dati  
struct Nodo {  
    int valore;  
    Nodo *nextPtr;  
};
```

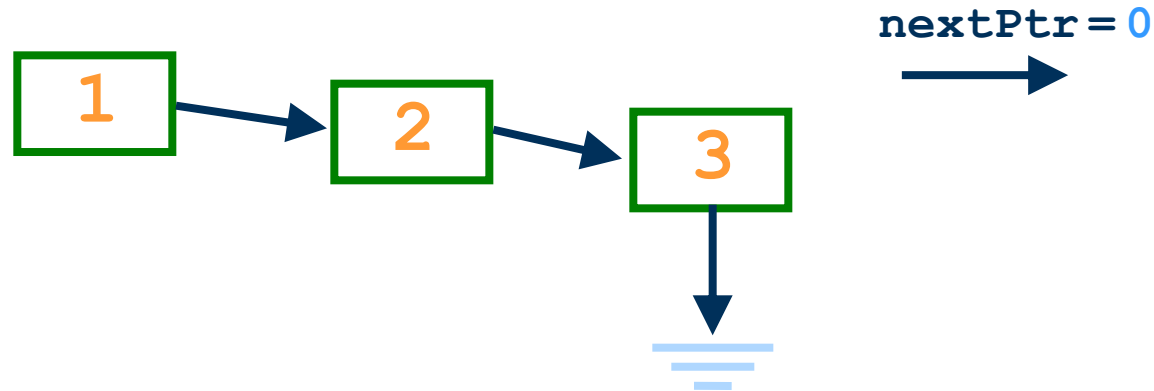



```
Definizione dati  
struct Nodo {  
    int valore;  
    Nodo *nextPtr;  
};
```

```
Definizione dati  
struct Nodo {  
    int valore;  
    Nodo *nextPtr;  
    Nodo *prevPtr;  
};
```

Soluzione semplice, ma obbliga a modificare la struttura dati tipica per le liste dinamiche.

3. Progetto della soluzione



Diverse soluzioni possibili.

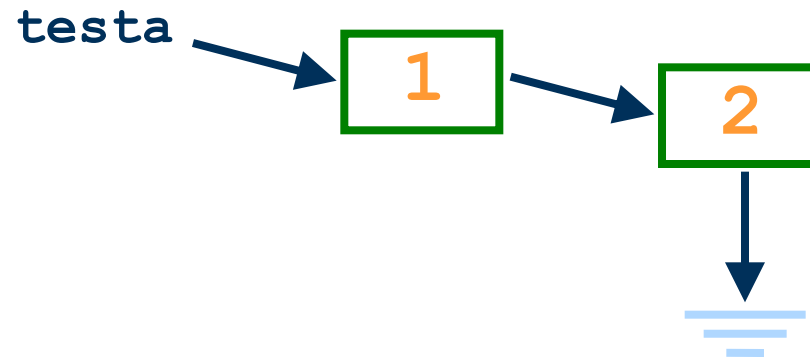
In questa esercitazione ne verranno mostrate due.

Lo studente è invitato a pensarne altre.

Prima *soluzione*

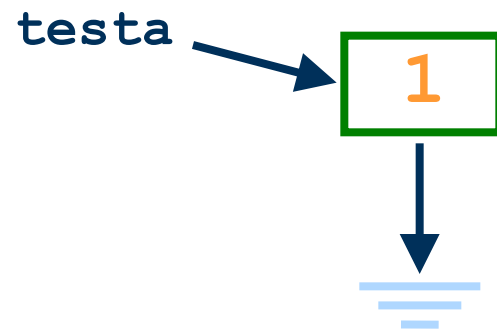


Prima *soluzione*

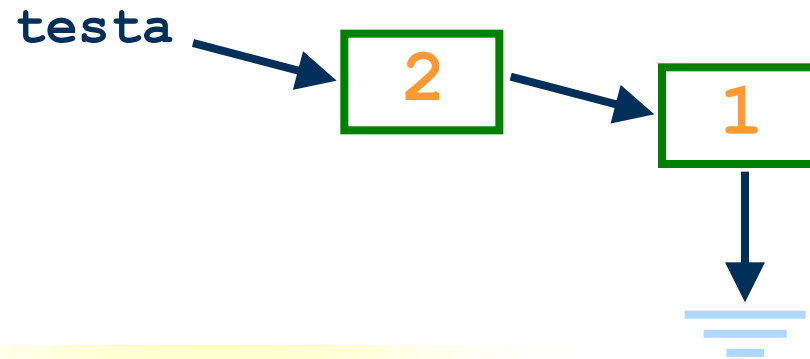


La prima soluzione dell'esercizio utilizza la modifica dell'algoritmo di inserimento dei dati.

Prima *soluzione*



Prima *soluzione*



Input: Valore 1
Valore 2
Output: Valore 2
Valore 1

E' possibile usare una lista come input dei dati.
La lista finale sarà l'inversa di quella d'ingresso.

Funzione invertiLista

```
{  
    for (finché la lista in ingresso non è vuota)  
        inserisciInTesta (listaInversa, valore corrente  
            lista in ingresso)  
}
```

Funzione inserisciInTesta

```
{  
    Crea nuovo elemento Nodo  
    Assegna al nuovo elemento il valore numerico  
    Assegna al puntatore del nuovo elemento  
        l'indirizzo di testa  
    Assegna a testa l'indirizzo del nuovo elemento  
}
```


Funzione invertiLista

```
{  
    for (finché la lista in ingresso non è vuota)  
        inserisciInTesta (listaInversa, valore corrente  
            lista in ingresso)  
}
```

Funzione inserisciInTesta

```
{  
    Crea nuovo elemento Nodo  
    Assegna al nuovo elemento il valore numerico  
    Assegna al puntatore del nuovo elemento  
        l'indirizzo di testa  
    Assegna a testa l'indirizzo del nuovo elemento  
}
```

Funzione invertiLista

- Un parametro in ingresso (la lista da invertire)
- Un parametro in uscita (la lista invertita)

Funzione inserisciInTesta

```
{  
    Crea nuovo elemento Nodo  
    Assegna al nuovo elemento il valore numerico  
    Assegna al puntatore del nuovo elemento  
        l'indirizzo di testa  
    Assegna a testa l'indirizzo del nuovo elemento  
}
```

Funzione invertiLista

```
void invertiLista(Nodo *testa, Nodo *&testaInv)
```

Funzione inserisciInTesta

```
{  
    Crea nuovo elemento Nodo  
    Assegna al nuovo elemento il valore numerico  
    Assegna al puntatore del nuovo elemento  
        l'indirizzo di testa  
    Assegna a testa l'indirizzo del nuovo elemento  
}
```

Funzione invertiLista

```
{  
    for (finchè la lista in ingresso non è vuota)  
        inserisciInTesta (listaInversa, valorecorrente  
            lista in ingresso)  
}
```

Funzione inserisciInTesta

```
{  
    Crea nuovo elemento Nodo  
    Assegna al nuovo elemento il valore numerico  
    Assegna al puntatore del nuovo elemento  
        l'indirizzo di testa  
    Assegna a testa l'indirizzo del nuovo elemento  
}
```

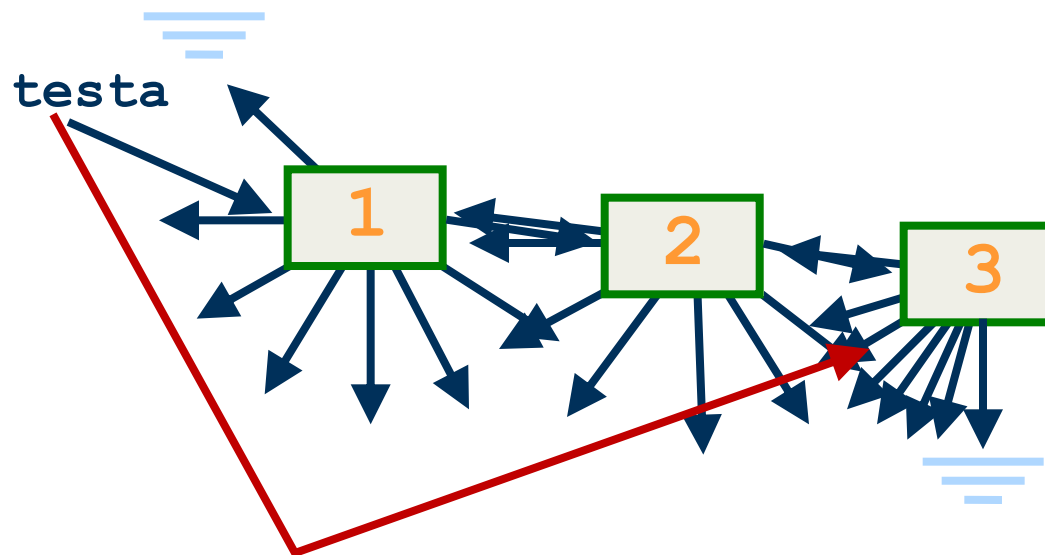
Funzione invertiLista

```
{  
    for (finchè la lista in ingresso non è vuota)  
        inserisciInTesta (listaInversa, valorecorrente  
            lista in ingresso)  
}
```

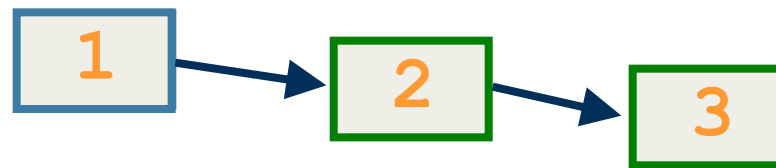
Funzione inserisciInTesta

```
void inserisciInTesta(Nodo *&testa, int valore)
```

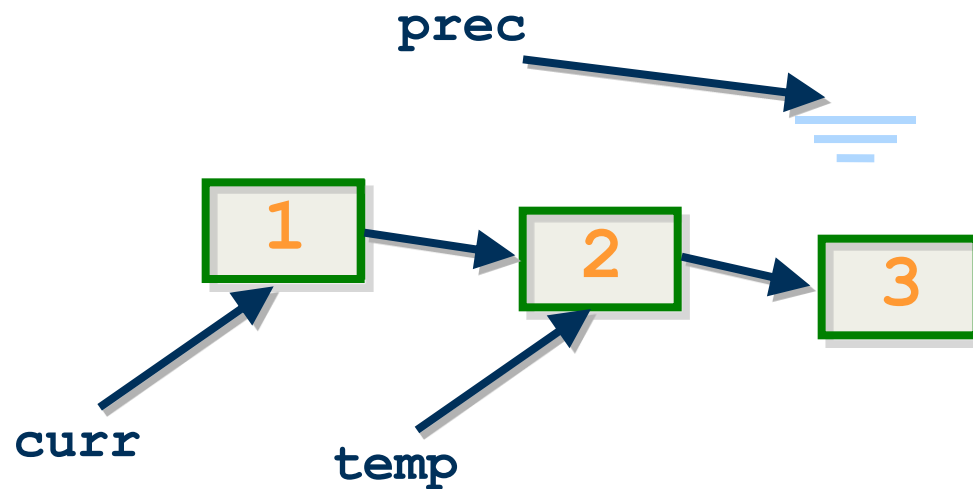
Seconda *soluzione*



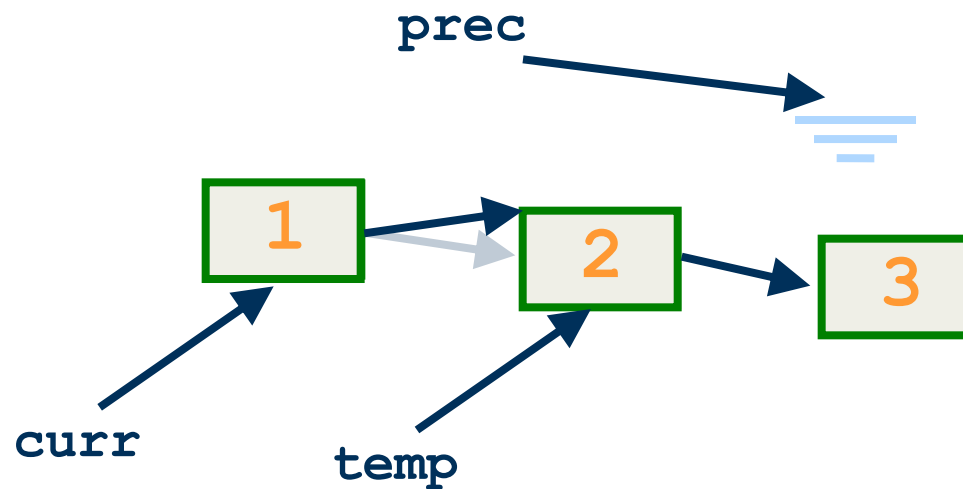
Seconda *soluzione*



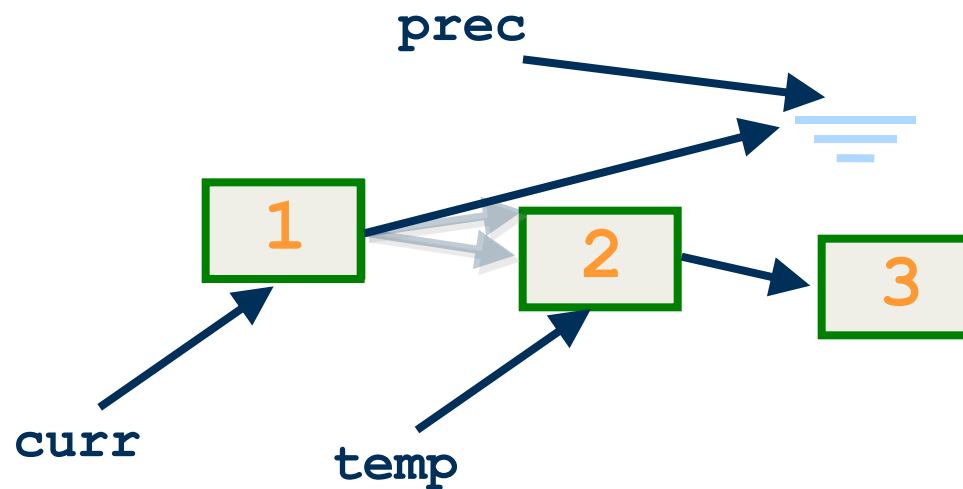
Seconda *soluzione*



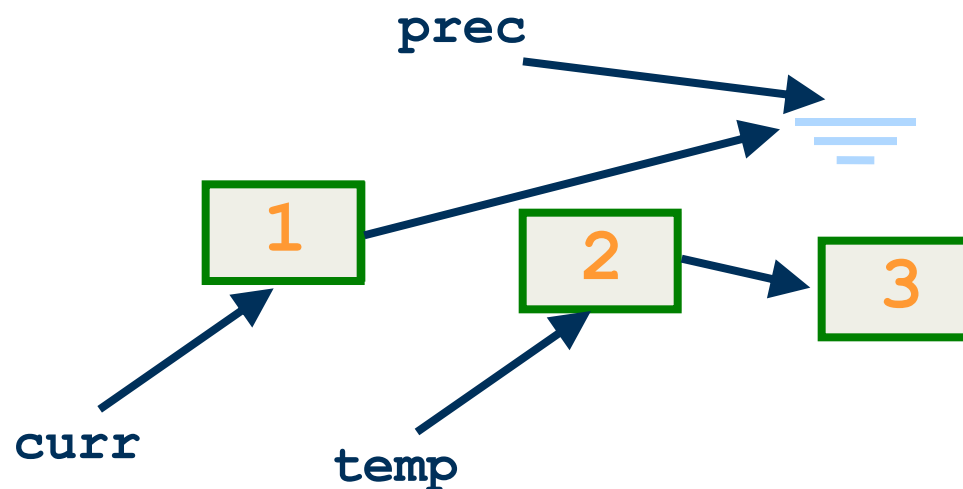
Seconda *soluzione*



Seconda *soluzione*

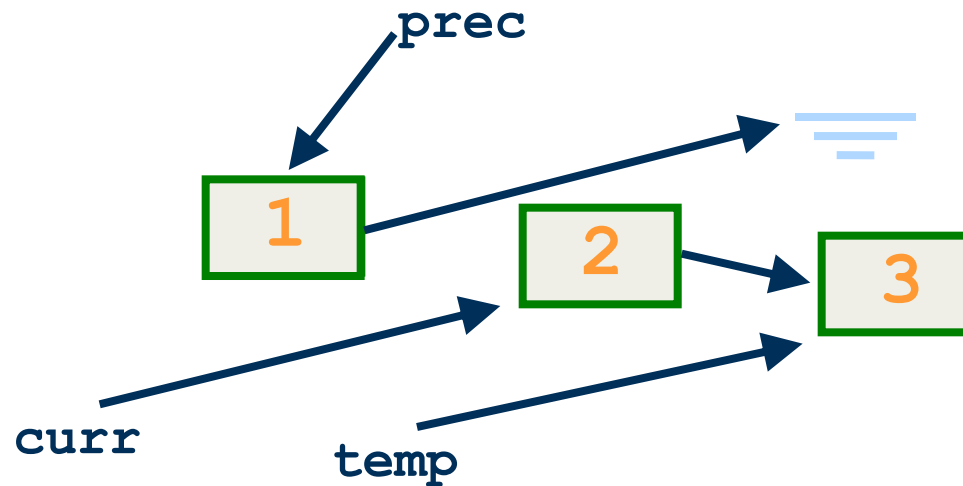


Seconda *soluzione*



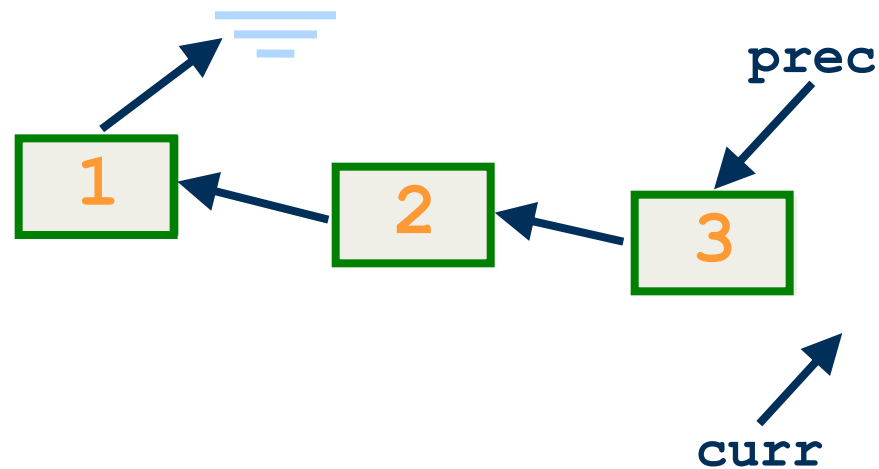
```
curr->nextPtr=prec;
```

Seconda *soluzione*



```
curr->nextPtr=prec;
```

Seconda *soluzione*



```
TestaInv=prec;
```

L'algoritmo distrugge la lista di ingresso.