Liste concatenate

Lezione 8 di Programmazione 2

Docente: Giuseppe Scollo

Università di Catania, sede di Comiso (RG)
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Studi in Informatica applicata, AA 2007-8

Indice

struttura di una lista concatenata

lista: struttura dati omogenea ad accesso **sequenziale**

lista concatenata: costituita da un insieme di **nodi**, ciascuno dei quali contiene:

- un **elemento** della lista
  - tutti gli elementi della lista sono oggetti dello **stesso tipo**
- un **puntatore**, o **link**, al nodo successivo
  - ciò non vale per il puntatore nell'ultimo nodo della lista

la lista è rappresentata da un puntatore che dà accesso al suo primo nodo

- si accede agli altri nodi seguendo la catena dei link

rappresentazione elementare in C++:

```
struct node { Elem elem; node *next; };
typedef node *nlink;
```

- la definizione della struttura della lista è **ricorsiva**, o autoreferenziante
  - **N.B.** conviene usare un nome diverso da `link` per il tipo del puntatore al nodo, per evitare conflitti con l'uso di tale nome nella libreria standard

allocazione di memoria e inizializzazione

con la definizione elementare della struttura della lista data sopra, l'allocazione di memoria per un nodo fa uso dell'operatore `new`, che restituisce un puntatore al nodo creato:

```
nlink p = new node ;
```

l'inizializzazione del nodo si effettua quindi con l'assegnamento di valori ai suoi membri `(p*).elem` e `(p*).next`; questi si possono designare con una notazione più confortevole:

```
p->elem, p->next
```

si possono combinare allocazione di memoria e inizializzazione di un nuovo nodo se nella definizione si include un'operazione **costruttore**, omonima al tipo della struttura:

```
struct node { Elem elem; node *next;
node(Elem e, node *n) { elem = e; next = n; }; };
typedef node *nlink;
```

questa definizione fornisce anche l'implementazione del costruttore, che inizializza i membri della struttura con i valori dei parametri; ad esempio, per creare un nodo inizializzandone i membri ai valori `a`, `t`, e ottenere in `p` il puntatore al nuovo nodo già inizializzato, basta invocare

```
nlink p = new node(a, t) ;
```

operazioni su liste, rilascio della memoria

prestazioni di operazioni su lista (concatenata) e array (contiguo):

accesso agli elementi, per lettura o scrittura:

più costoso nella lista, per la sequenzialità dell'accesso

modifica della struttura, per inserimento o rimozione di elementi:

più costoso nell'array, per il mantenimento della contiguità di memorizzazione

inserimento di un nodo in una lista concatenata: `t->next = x->next; x->next = t;`

N.B. si inserisce il nodo `t` **dopo** il nodo `x`

rimozione di un nodo da una lista concatenata: `x->next = x->next->next;`

così si rimuove il nodo **dopo** il nodo `x`

e se ne perde il puntatore, o altrimenti: `t = x->next; x->next = t->next;`

il che può servire all'eventuale **rilascio** della memoria allocata al nodo rimosso: `delete t;`

esempio: il problema di Giuseppe Flavio

il puntatore nell'**ultimo nodo** di una lista può essere **nullo**, oppure puntare **allo stesso nodo**, o **al primo nodo:**

in quest'ultimo caso la lista è **circolare**

le liste circolari hanno diverse applicazioni, ne vediamo una in un algoritmo per il **problema di Giuseppe Flavio,** usabile ad es. quale meccanismo di selezione di un leader in un gruppo:

si dispongono gli **N** componenti del gruppo in circolo e, a partire da uno qualsiasi di essi, se ne contano **M**: l'ultimo viene escluso

si reitera il conteggio e la relativa esclusione altre **N-2** volte, ogni volta a partire dal superstite successivo all'ultimo escluso

il "problema" consiste nel prevedere chi sarà il superstite finale, dati **N**, **M** e la posizione di inizio del conteggio selettivo

il problema di Giuseppe Flavio in C++

```
#include <iostream>
#include <cstdlib>
using namespace std;
struct node
{ int elem; node* next;
  node(int e, node* t) { elem = e; next = t; };
};
typedef node* nlink;
int main(int argc, char *argv[])
{ int i, N = atoi(argv[1]), M = atoi(argv[2]);
  nlink t = new node(1, 0); t->next = t;    // creazione lista circolare di 1 nodo
  nlink x = t;
  for (i = 2; i <= N; i++)
    x = (x->next = new node(i, t));         // inserimento i-esimo nodo nella lista
  while (x != x->next)
  {                                         // finché il nodo non punta a sé stesso:
    for (i = 1; i < M; i++) x = x->next;    // scorrimento di M-1 nodi
    x->next = x->next->next;                // rimozione M-simo nodo
  }
  cout << x->elem << endl;                  // output del superstite
}
```

esercizi

1. modificare il programma qui proposto per il problema di Giuseppe Flavio, in modo da produrre in output i valori degli elementi della lista, nell'ordine definito dalla stessa, prima dell'inizio di ogni ciclo che conduce alla rimozione di un nodo
2. quando costruisce la lista, il programma qui proposto per il problema di Giuseppe Flavio esegue tre istruzioni di assegnamento a puntatore per ogni nuovo nodo della lista: una nel costruttore e due nel `main`; ideare un programma equivalente che effettui la suddetta costruzione eseguendo solo due istruzioni di assegnamento a puntatore per ogni nuovo nodo
3. eseguire l'esercizio 1 sul programma ideato nell'esercizio 2 e verificare l'equivalenza del programma così prodotto e di quello prodotto nell'esercizio 1