Corso di Ingegneria Elettronica
Insegnamento di "Tecniche Avanzate di Progettazione Software 1", a.a. 2008-09

Esercitazione n. 7

Sviluppare un sistema software per la gestione di una rete.
Una rete è composta da elementi generici di rete, supponiamo che essi possano essere solo reti e PC.

Un generico elemento di rete (`NetworkItem`) è descritto da un identificatore e un indirizzo `IP` (per esempio, 192.168.1.1); esso avrà inoltre i metodi `Print()` e `Size()`: il primo stampa identificatore, indirizzo IP e, nel caso di una rete, l'elenco di identificatori e IP degli elementi ad essa connessi; il secondo stampa la memoria occupata da tutti gli elementi della rete: la somma dell'occupazione di memoria di ogni PC della rete.

Un PC (`PC`) è inoltre caratterizzato da una certa occupazione di memoria: tale occupazione deve essere gestita con allocazioni dinamiche.

Una rete (`Net`) è descritta invece da una *lista* di puntatori ad elementi generici di rete (`NetItemList`) e da una *lista* di indirizzi (`IPList`); essa dovrà fornire i metodi per aggiungere (`Add()` e `AddCopy()`) e rimuovere (`remove()`) elementi. Alla creazione di un oggetto di tipo `Net`, la lista di `IP` deve essere inizializzata con un elenco di indirizzi.
Il metodo `AddCopy(const NetworkItem* item)` rimuove un `IP` da tale lista di indirizzi, lo assegna all'elemento generico di rete (`item`) ed inserisce una copia di quest'ultimo nella lista `NetItemList`, pertanto e' necessario utilizzare il metodo `Clone()`.
Il metodo `Add(NetworkItem* item)` rimuove un `IP` dalla lista di indirizzi, lo assegna all'elemento generico di rete (`item`) e lo inserisce nella lista `NetItemList`.
Il metodo `remove(const IP ipremove)` ricerca all'interno di una `Net` l'elemento generico di rete caratterizzato dall'ip `ipremove`, rimuove l'elemento dalla lista `NetItemList` e reinserisce l'`IP` dell'elemento rimosso nella lista di `IP` utilizzabili.

Semplificazioni: il metodo `remove` ricerca l'elemento da rimuovere solo tra i `NetworkItem` collegati direttamente alla rete. Inoltre la ricerca dell'elemento da rimuovere viene effettuata solo in base all'indirizzo `IP` e non in base all'identificatore, in quanto questo puo' non essere univoco.

Nota: gli oggetti (`PC` e `Net`) vengono creati dinamicamente e nel caso in cui vengano inseriti in una rete vengono gestiti (ed eventualmente distrutti) dalla rete a cui sono collegati.

Si suggerisce di sviluppare il sistema nel modo seguente:

- Implementare la classe `IP` e verificarne la correttezza.
- Implementare la classe astratta `NetworkItem`.
- Implementare la classe `PC` e verificarne la correttezza.
- Implementare la classe `Net` e verificarne la correttezza.
- Sviluppare un esempio completo del sistema software.

Per la gestione delle liste `IPList` e `NetItemList` si utilizzino gli iteratori e la lista della Libreria Standard.

Si fornisce uno schema delle classi fondamentali che compongono il sistema:

```
class IP{
      int ip[4];
public:
//..};

class NetworkItem{
protected:
      string ItemName;
      IP m_ip;
public:
virtual void Print()const;
virtual int Size()const =0;
virtual NetworkItem* clone() const =0;
//..};


class Net : public NetworkItem{
      list<NetworkItem*> NetItemList;
      list<IP> IPList;
public:
bool AddCopy(const NetworkItem* item);
bool Add(NetworkItem* item);
bool remove(const IP ipremove);
//..};


class PC : public NetworkItem{
      int *mem;
public:
//..};
```

Un possibile test per verificare la correttezza del sistema sviluppato:

```
Net root("root",IP(10,1,3,1));
PC* pc= new PC("pc",IP());
root.Add(pc);
Net* nodo1 = new Net("nodo1",IP());
for(int i=0;i<5;i++){
      PC* pc = new PC("pc1",IP());
      nodo1->Add(pc);}
Net* nodo2 = new Net("nodo2",IP());
for(int i=0;i<5;i++){
      PC* pc = new PC("pc2",IP());
      nodo2->Add(pc);}
nodo1->AddCopy(nodo2);
root.Add(nodo1);
root.AddCopy(nodo2);
root.Add(nodo2);
root.Print("");
cout<<"Net size = "<<root.Size()<<endl;
root.remove(IP(w,x,y,z));//dove w.x.y.z e' l'indirizzo del NetworkItem da rimuovere
root.Print("");
cout<<"Net size = "<<root.Size()<<endl;
```