Laboratorio Informatica Classe 4A Serale
Venerdì 18/02/2011

Gruppo

Cognome_____Riontino_____ Nome____Raffaele_____

Cognome_____ Nome_____

TRACCIA 3

ARGOMENTO: OGGETTI E LISTE
TEMPO: 2h

1. Costruire una Classe Lista con tutti i metodi e le proprietà necessarie. Tra i metodi dovranno essere previsti obbligatoriamente i seguenti:
    a. Inserimento in Testa
    b. Cancellazione in Testa
    c. Visualizzazione
    d. Si costruisca un main che utilizzi la classe richiesta.

Soluzione

Proprietà	
int size;	Scopo : memorizzare il numero di elementi inseriti nella lista
	Tipo : Lavoro
struct nodo { l dato; struct nodo * next; };	Scopo : nodo principale della lista
	Tipo : Dato
nodo * testa;	Scopo : puntatore all'indirizzo del primo elemento della lista
	Tipo : Lavoro

Metodi	
Lista ();	Scopo : costruttore della classe
	Inizializza la proprietà size a 0 e crea una lista vuota assegnando il puntatore testa a NULL
int length();	Scopo : ritorna il numero degli elementi presenti nella lista
	Per ritornare il numero degli elementi della lista ritorna il valore della proprietà size
void push_Testa(l);	Scopo : inserisce un elemento in testa

	alla lista
	Crea un nuovo nodo; assegna a dato il valore ricevuto come parametro; inserisce il nodo in testa alla lista.
void push_Coda(l);	Scopo : inserisce un elemento in coda alla lista
	Crea un nuovo nodo; assegna a dato il valore ricevuto come parametro; inserisce il nodo in coda alla lista
bool pop_Testa();	Scopo : eliminare un elemento in testa alla lista
	ritorna true se l'elemento viene eliminato; false se la lista è vuota.
bool pop_Coda();	Scopo : eliminare un elemento in coda alla lista
	Ritorna true se l'elemento viene eliminato; false se la lista è vuota
void visualizzaLista();	Scopo : visualizzare il contenuto della lista
	Scorre tutta la lista per visualizzare tutti gli elementi inseriti (solo se gli elementi inseriti sono dati puri come variabili int , float , char , ….ecc; nel caso di una lista di oggetti non può essere utilizzato);
l elementAt(int);	Scopo : cercare un dato nella lista data la posizione
	Effettua un controllo sulla lista e, se trova la posizione ricevuta come parametro ritorna l'elemento presente, altrimenti ritorna un valore NULL
void modificaAt(int,l)	Scopo : modificare un elemento ad una posizione
	Riceve come parametri la posizione in cui modificare l'elemento e il nuovo valore dell'elemento.In caso di posizione non corretta visualizza un messaggio di errore

Variabili	

Lista <string> lista;	Scopo : istanza di un oggetto di tipo Lista
	Tipo : Dato
char scelta;	Scopo : scegliere le operazioni da effettuare
	Tipo : Lavoro
string dato;	Scopo : contenere il dato da inserire nella lista
	Tipo : Dati

Funzioni	
void titolo();	Scopo : visualizzare una breve descrizione del programma
	Utilizza un system("cls") per ripulire la console e semplici cout per la visualizzazione
char menu();	Scopo : ritornare il carattere corrispondente alla scelta effettuata
	Visualizza una serie di opzioni e utilizza il comando return per ritornare la scelta
void aggiungiTesta(Lista<int>&);	Scopo : permettere di inserire un elemento nella lista
	Riceve il nome della lista per riferimento; permette di inserire il dato da tastiera; utilizza il metodo 'pusch_testa(l)' per l'inserimento del dato in testa alla lista
void eliminaTesta(Lista<int>&);	Scopo : permettere di eliminare un elemento dalla lista
	Riceve il nome della lista per riferimento; Richiama il metodo 'pop_Testa()' per eliminare l'elemento in testa alla lista
void aggiungiCoda(Lista<int>&);	Scopo : permettere di inserire un elemento nella lista
	Riceve il nome della lista per riferimento; richiama il metodo 'push_Coda(l)' per l'inserimento del dato in coda alla lista
void eliminaCoda(Lista<string>&);	Scopo : permettere di eliminare un elemento dalla lista

	Riceve il nome della lista per riferimento; richiama il metodo ‘pop_coda()’ per eliminare l’elemento in coda alla lista
void visualizza(Lista<string>);	Scopo : visualizzare tutti gli elementi della lista
	Riceve il nome della lista per valore; visualizza tutti gli elementi della lista utilizzando il metodo visualizzaLista();
void cercaPosizione(Lista<string>);	Scopo : visualizzare il contenuto di una determinata posizione della lista
	Attraverso il metodo elementAt, scorre la lista come un array e, se trova la posizione cercata, visualizza il suo contenuto,altrimenti ritorna un valore NULL
void modificaDato(Lista<string>&);	Scopo : modificare un dato già presente nella lista
	Richiede di inserire la posizione dell’elemento che si vuole modificare; effettua una ricerca nella lista e se non ci sono errori visualizza il vecchio dato e chiede di inserire il nuovo dato.

CODICE PRODOTTO

```
/*
 Programma : cppTraccia3

 Autore : Riontino Raffaele 4 informatici (corso serale)
        ITIS Molinari - Milano
        20/02/2011

 TRACCIA 3
        ARGOMENTO: OGGETTI E LISTE
        TEMPO: 2h
         Costruire una Classe Lista con tutti i metodi e le proprietà necessarie.
           Tra i metodi dovranno essere previsti obbligatoriamente i seguenti:
        a.      Inserimento in Testa
        b.      Cancellazione in Testa
        c.      Visualizzazione
        d.      Si costruisca un main che utilizzi la classe richiesta.
```

```
*/


#include <iostream>
#include <conio.h>
#include <string>
using namespace std;


template<class l>
class Lista
{
 public : Lista () {this -> size = 0; this -> testa = NULL;} //costruttore
        int length() {return this -> size;}; //ritorno il numero degli elementi inseriti nella lista
        void push_Testa(l); //aggiungo un elemento in testa ricevuto come parametro
        void push_Coda(l); //aggiungo un elemento in coda
        bool pop_Testa(); //rimuovo un elemento in testa
        bool pop_Coda(); //elimino un elemento in coda
        void visualizzaLista(); //visualizzo tutta la lista
        l elementAt(int); //ritorno l'elemento alla posizione cercata
        void modificaAt(int,l); //modifica il dato presente ad una determinata posizione


 protected : int size; //dimensione della lista
           struct nodo { //struttura della lista
                     l dato;
                     struct nodo * next;
                    };
           nodo * testa; // dichiaro un puntatore alla struttura nodo
}; //fine prototipo classe Lista


///prototipi delle funzioni del main

void titolo();

char menu();

void aggiungiTesta(Lista<string>&);

void eliminaTesta(Lista<string>&);

void aggiungiCoda(Lista<string>&);

void eliminaCoda(Lista<string>&);

void visualizza(Lista<string>);

void cercaPosizione(Lista<string>);
```

```
void modificaDato(Lista<string>&);



int main()
{
 Lista <string> lista;
 char scelta;
 do{
   scelta = menu();
   switch (scelta)
         {
           case '1' : aggiungiTesta(lista);
                    break;
           case '2' : eliminaTesta(lista);
                    break;
           case '3' : aggiungiCoda(lista);
                    break;
           case '4' : eliminaCoda(lista);
                    break;
           case '5' : cercaPosizione(lista);
                    break;
           case '6' : modificaDato(lista);
                    break;
           case 'v' : visualizza(lista);
                    break;
         }
  }while (scelta != 'e');
 return 1;
}//fine main


///definizione delle funzioni del main

void titolo()
{
 system("cls");
 cout << "\n\t\tEsercitazione di laboratorio : Oggetto Lista\n\n";
}//fine titolo

char menu()
{
 titolo();
 cout << "\n\n\t\tMenu'\n";
 cout << "\n\t1 - aggiungi in testa\n";
 cout << "\n\t2 - elimina in testa\n";
 cout << "\n\t3 - aggingi in coda\n";
 cout << "\n\t4 - elimina in coda\n";
```

```
 cout << "\n\t5 - cerca per posizione\n";
 cout << "\n\t6 - modifica alla posizione\n";
 cout << "\n\tv - visualizza\n";
 cout << "\n\te - esci\n";
 cout << "\n\tdigita la scelta : ";
 return(getch());
}//fine menu

void aggiungiTesta(Lista<string> &lista)
{
 string dato;
 titolo();
 cout << "\n\tinserimento in testa\n\n";
 cout << "\n\n\tinserisci una stringa : ";
 cin >> dato;
 lista.push_Testa(dato);
 cout << "\n\tdati totali : " << lista.length() << endl;
 getch();
}//fine aggiungiTesta

void aggiungiCoda(Lista<string> &lista)
{
 string dato;
 titolo();
 cout << "\n\tinserimento in coda\n\n";
 cout << "\n\n\tinserisci una stringa : ";
 cin >> dato;
 lista.push_Coda(dato);
 cout << "\n\tdati totali : " << lista.length() << endl;
 getch();
}//fine aggiungiCoda

void eliminaTesta(Lista<string> &lista)
{
 titolo();
 cout << "\n\teliminazione in testa\n\n";
 lista.pop_Testa();
 cout << "\n\tdati totali : " << lista.length();
 getch();
}//fine eliminaTesta

void eliminaCoda(Lista<string> &lista)
{
 titolo();
 cout << "\n\teliminazione in coda\n\n";
 lista.pop_Coda();
 cout << "\n\tdati totali : " << lista.length();
 getch();
}//fine eliminaCoda
```

```
void cercaPosizione(Lista<string> lista)
{
 titolo();
 string dato;
 int posizione;
 cout << "\n\tInserisci la posizione dell'elemento da caricare : ";
 cin >> posizione;
 try{
    if (posizione >= lista.length() || posizione < 0) throw 1;
    dato = lista.elementAt(posizione);
    cout << "\n\tDato alla posizione " << posizione << " : " << dato;
   }
 catch (int e)
   {
    cerr << "\n\tPosizione non corretta !! errore : " << e;
   }
 getch();
}//fine cercaPosizione

void modificaDato(Lista<string>& lista)
{
 titolo();
 cout << "\n\tmodifica di un dato data la posizione\n\n";
 int posizione;
 string dato;
 cout << "\n\tInserisci la posizione dell'elemento da modificare : ";
 cin >> posizione;
 try{
    if (posizione >= lista.length() || posizione < 0) throw 1;
    dato = lista.elementAt(posizione);
    cout << "\n\tDato attuale alla posizione " << posizione << " : " << dato;
    cout << "\n\tinserisci il nuovo dato : ";
    cin >> dato;
    lista.modificaAt(posizione,dato);
   }
 catch (int e)
   {
    cerr << "\n\tPosizione non corretta !! errore : " << e;
   }
 getch();
}//fine modificaDato

void visualizza(Lista<string> lista)
{
 titolo();
 cout << "\n\tvisualizzazione della lista\n\n";
 lista.visualizzaLista();
 getch();
```

```
}//fine visualizza


/////definizione dei metodi della classe

template<class I>
void Lista<I>::push_Testa(I dato)
{
    nodo *nuovo = new nodo ;
    nuovo -> dato = dato;
    nuovo -> next = this -> testa;
    this -> testa = nuovo;
    this -> size ++;
}

template<class I>
void Lista<I>::push_Coda(I dato)
{
   nodo * nuovo = new nodo ;
   nuovo -> dato = dato;
   nuovo -> next = NULL;
   if (!this -> testa)
       this -> testa = nuovo;
   else {
          nodo * scorri = testa;
          while (scorri -> next) scorri = scorri -> next;
          scorri -> next = nuovo;
       }
   this -> size ++;
}

template<class I>
bool Lista<I>::pop_Testa()
{
    if (!this -> testa)
      {
       cout << "\n\tLista vuota!!";
       return false;
      }
    else {
           nodo * temp;
           temp = this -> testa;
           this -> testa = this -> testa -> next;
           delete temp;
           this -> size --;
           cout << "\n\tdato eliminato";
           return true;
        }
}
```

```
template<class l>
bool Lista<l>::pop_Coda()
{
    if (!this -> testa)
      {
       cout << "\n\tLista vuota!!";
       return false;
      }
    if (!this -> testa -> next)
      {
       nodo * temp = this -> testa;
       this -> testa = NULL;
       delete temp;
       this -> size --;
       cout << "\n\tdato eliminato";
       return true;
      }
    else {
          nodo * scorri  = testa;
          while (scorri -> next -> next) scorri = scorri -> next;
          scorri -> next = NULL;
          this -> size --;
          cout << "\n\tddato eliminato";
          return true;
        }
}


template<class l>
void Lista<l>::visualizzaLista()
{
 if (!this -> testa) cout << "\n\tLista vuota";
 else {
      int conto = 0;
      nodo * cursore = this -> testa;
      while (cursore)
           {
            cout << "\n\telemento n. " << conto << "\n\t\t";
            cout << cursore -> dato;
            cursore = cursore -> next;
            conto++;
           }
     }
}


template<class l>
l Lista<l>::elementAt(int pos)
```

```
{
 if (!this -> testa) cout << "\n\tLista vuota";
 else {
      int conto = 0;
      nodo * cursore = this -> testa;
      while (cursore)
           {
            if (conto == pos)
               return cursore -> dato;
            cursore = cursore -> next;
            conto++;
           }
      return NULL; //se non trova l'elemento alla posizione cercata ritorna NULL
     }
}

template<class I>
void Lista<I>::modificaAt(int pos,I dato)
{
 if (!this -> testa) cout << "\n\tLista vuota";
 else {
      int conto = 0;
      nodo * cursore = this -> testa;
      while (cursore)
           {
            if (conto == pos)
               {
                cursore -> dato = dato;
                cout << "\n\tElemento modificato!!\n";
                return;
               }
            cursore = cursore -> next;
            conto++;
           }
      cout << "\n\tImpossibile modificare l'elemento!!\n"; //se non trova l'elemento alla
posizione cercata ritorna NULL
     }
}
```