```
// ListaProdotti_2.CPP
// Programma che consente di creare una lista di PRODOTTI (nome e prezzo
//
// Operazioni: inser. in testa, in coda, ricerca di un elemento,
//              canc. in testa e in coda, visualizzaz. elementi della li
//              modifica i dati di un elemento
//
//   aggiunta: canc. elemento dopo averlo ricercato per nome prodotto
//
// Puntatore al primo elemento della lista: variabile LOCALE nel main
//-------------------------------------------------------------------------

#include <iostream>
#include <iomanip>

using namespace std;

// parte "Informazione" di ogni elemento della lista
struct prodotto{
       string Nome;
       float Prezzo;
       };

// Elemento della lista composto da   |Informazione|Puntatore|
struct elemento{
       prodotto Prod;
       elemento *succ;
       };

// Prototipi
void menu();

elemento *CreaElemento();
void AcquisisciDatiElemento(elemento *);

elemento* InizializzaLista(elemento *);
elemento* SvuotaLista(elemento *);
bool ListaVuota(elemento *);

elemento* InserisciInTesta(elemento *);
elemento* InserisciInCoda(elemento*);
void VisualizzaLista(elemento *);
elemento* EliminaInTesta(elemento *);
elemento* EliminaInCoda(elemento *);

elemento *CercaElemento (string x,elemento *);
// >>> ricerca per cancellazione (OVERLOADING di funzione)
elemento *CercaElemento (string x,elemento *, elemento **);

void VisualizzaDatiElemento(elemento *);

// >>> Elimina elemento (punt elemento, punt prec, testa della lista)
elemento *EliminaElemento(elemento *, elemento *, elemento * );
```

```
//====================== Programma principale
int main()
{
   int scelta;
   elemento *p, *prec;
   string x;

   // puntatore alla testa della lista (var. locale)
   elemento *Testa;
   // inizializza la lista (puntatore alla testa della lista vuota)
   Testa=InizializzaLista(Testa);

 do
 {
  // presenta il menu di scelta
  menu();
  cout<<" Inserisci la tua scelta:  ";
  cin>>scelta;

  switch(scelta)
  {
   case 1: Testa=InserisciInTesta(Testa);  // Crea un nuovo elemento
           break;                          // e lo inserisce in testa alla lista
   case 2: Testa=EliminaInTesta(Testa);    // Cancella il primo elemento
           break;                          // e libera la memoria
   case 3: Testa=InserisciInCoda(Testa);   // Crea un nuovo elemento
           break;                          // e lo inserisce in coda alla lista
   case 4: Testa=EliminaInCoda(Testa);     // Cancella l'ultimo elemento
           break;                          // e libera la memoria
   case 5: VisualizzaLista(Testa);         // visualizza tutti gli elementi
         break;                            // della lista
   case 6: Testa=SvuotaLista(Testa);       // Cancella tutti gli elementi
           break;                          // della lista deallocando la memoria
   case 7: cout <<"\nInserisci il nome del prodotto da cercare : ";
           cin >>x;
           p = CercaElemento (x,Testa);
           if (p!=NULL) VisualizzaDatiElemento(p);
           else cout<<x<<"  NON TROVATO\n\n";
           break;
   case 8: cout <<"\nInserisci Nome prodotto per modifica dati : ";
           cin >>x;
           p = CercaElemento (x,Testa);
           if (p!=NULL) AcquisisciDatiElemento(p);
           else cout<<x<<"   NON TROVATO\n\n";
           break;
   case 9: cout <<"\nInserisci il nome del prodotto da eliminare : ";
           cin >>x;
           p = CercaElemento (x,Testa, &prec);
           if (p!=NULL)
                Testa=EliminaElemento(p, prec, Testa);
           else cout<<x<<"  NON TROVATO\n\n";
           break;

   case 0: Testa=SvuotaLista(Testa);
           break;
```

```
  default: cout <<"Scelta errata\n  ";
  }
 }while(scelta!=0);
}  // ========================================================================

// FUNZIONI

// --- presenta il menu
void menu()
{
 cout<<"                      MENU'  di GESTIONE LISTA DI PRODOTTI";
 cout<<'\n';
 cout<<"                  1 - Inserisci elemento in testa\n";
 cout<<"                  2 - Cancella elemento in testa\n";
 cout<<"                  3 - Inserisci elemento in coda\n";
 cout<<"                  4 - Cancella elemento in coda\n";
 cout<<"                  5 - Visualizza elementi della lista\n";
 cout<<"                  6 - Svuota tutta la lista\n";
 cout<<"                  7 - Ricerca elemento per nome prodotto\n";
 cout<<"                  8 - Modifica dati di un prodotto\n";
 cout<<"                  9 - Cancella elemento con nome prodotto specificato\n";
 cout<<"                  0 - FINE\n";
 cout<<'\n';
}

// --- Imposta il puntatore di testa a NULL
elemento* InizializzaLista(elemento *testa)
{ testa=NULL;   // lista vuota
  return testa;
}


// --- Controlla se la lista è vuota
bool ListaVuota(elemento *testa)
{ if (testa==NULL)
     return true;
 else
     return false;
}


// --- Svuota lista deallocando la memoria
elemento* SvuotaLista(elemento *testa)
{elemento *temp;             // puntatore d'appoggio
 // dealloca tutti gli elementi
 while (testa != NULL)
    {temp = testa;
     testa = testa->succ;
     delete temp;               // dealloca l'elemento
     }
 cout <<"\n---- LISTA CANCELLATA\n\n";
 return testa;
 }
```

```
// --- Crea un nuovo elemento e restituisce il puntatore
elemento *CreaElemento()
  {elemento  *p;
   p=new (elemento);        // crea il nuovo elemento
   return p;                // restituisce il puntatore al nuovo elemento creato
   }


// --- Acquisisci da tastiera i dati dell'elemento della lista puntato d
void AcquisisciDatiElemento(elemento *p)
{ cout <<"Inserisci Nome del prodotto : ";
  cin >>p->Prod.Nome;
  cout <<"Inserisci il prezzo : ";
  cin >>p->Prod.Prezzo;
  return;
}


// --- Inserimento di un nuovo elemento in testa
elemento* InserisciInTesta(elemento *testa)
  {
   elemento *nuovo=CreaElemento(); // crea un nuovo elemento
   AcquisisciDatiElemento(nuovo);  // acquisisce i valori del campo Informazione

   nuovo->succ = testa;
   testa = nuovo;
   return testa;
  }


// --- Inserimento di un nuovo elemento in coda
elemento* InserisciInCoda(elemento* testa)
{
 elemento *p, *nuovo;
 nuovo=CreaElemento();     // crea un nuovo elemento
 AcquisisciDatiElemento(nuovo);   // acquisisce i valori del campo Informazione

 nuovo->succ = NULL;       // assegna la marca di fine lista

 if (ListaVuota(testa))    // Se la lista è vuota diventa il primo elemento
    testa = nuovo;         // e l'ultimo della lista
 else                      // altrimenti scorre la lista
    {p = testa;
     while (p->succ!=NULL)
          p = p->succ;
     p->succ=nuovo;
    }
 return testa;
}


// --- Visualizza i dati dell'elemento della lista puntato da p
void VisualizzaDatiElemento(elemento *p)
  {cout <<left<<setw(15)<<p->Prod.Nome<<right<<setw(10)<<p->Prod.Prezzo<<endl;
```

```
   return;
   }


// --- Visualizza le informazioni contenute nella lista
void VisualizzaLista (elemento *testa)
{if (ListaVuota(testa))
      cout<<"--- LISTA  VUOTA\n";
 else
     {elemento *p=testa;
      cout<<"---- Elenco prodotti ----\n";
      while (p!=NULL)
            {
             VisualizzaDatiElemento( p );
             p=p->succ;
             }
      cout<<"---- Fine Elenco --------\n\n";
      }
}


// --- Elimina elemento in testa
elemento* EliminaInTesta(elemento *testa)
{if (ListaVuota(testa))
      cout<<"--- LISTA  VUOTA\n";
 else
     {elemento *temp=testa;     // libera il primo elemento
      testa=temp->succ;
      cout <<"Prodotto "<<temp->Prod.Nome<<" cancellato\n\n";
      delete temp;
      }
 return testa;
}


// --- Elimina elemento in coda
elemento* EliminaInCoda(elemento *testa)
{if (ListaVuota(testa))              // lista vuota
      cout<<"--- LISTA  VUOTA\n";
 else
     {elemento *prec, *p=testa;      // prec puntatore al precedente
                                     // p puntatore all'elem. attuale
      while (p->succ!=NULL)          // Scorre fino a fine lista
        {prec = p;
         p=p->succ;
        }

      if (p == testa)               // lista di un solo elemento
         testa = NULL;              // allora cambia la testa della lista
      else
         prec->succ=NULL;          // il precedente diventa l'ultimo della lista

      cout <<"Elemento "<<p->Prod.Nome<<" cancellato\n\n";
      delete p;                     // libera l'area di memoria dell'elemento
```

```
     }
return testa;
}


// --- Ricerca  per nome del prodotto
// Restituisce il puntatore all'elemento (NULL = non trovato)
elemento *CercaElemento (string x,elemento *testa)
{ if (ListaVuota(testa)) return NULL;      // Lista vuota
  else
     {elemento *p = testa;
      bool trovato=false;
      while (p !=NULL && !trovato)
        {if (p->Prod.Nome == x)
             trovato=true;
         else
             p=p->succ;
        }
      return p;
      }
 }




// aggiunta >>>--- Ricerca  per nome del prodotto
// Restituisce il puntatore all'elemento (NULL = non trovato)
// prec  puntatore al precedente (passato per indirizzo)
//             (prec=NULL  => elemento non trovato o è il primo della lis
elemento *CercaElemento (string x,elemento *testa, elemento **prec )
{
  *prec = NULL;
  if (ListaVuota(testa)) return NULL;      // Lista vuota
  else
     {
      elemento *p = testa;
      bool trovato=false;
      while (p !=NULL && !trovato)
        {if (p->Prod.Nome == x)
             trovato=true;
         else
             {
               *prec=p;
               p=p->succ;
             }

        }
      return p;
      }
 }
```

```
// aggiunta >>>--- Elimina l'elemento
// p    punta all'elemento da eliminare
// prec punta all'elemento precedente
elemento *EliminaElemento(elemento *p, elemento *prec, elemento *testa)
{
  if (p==testa)
     {
      elemento *temp = testa;      // libera il primo elemento
      testa=temp->succ;
      cout <<"Prodotto "<<temp->Prod.Nome<<" cancellato\n\n";
      delete temp;
     }

  else
      {prec->succ = p->succ; // modifico il puntatore dell'elemento precedente
                             // per puntare all'elemento successivo a quello
                             // che viene cancellato
       cout <<"Prodotto "<<p->Prod.Nome<<" cancellato\n\n";
       delete p;
      }

  return testa;
}
```