```cpp
#include "Lista.h"
#include <iostream>
#include <assert.h>
using namespace std;
template <typename T>
Lista<T>::Lista()
{
    primero = 0;
    ultimo = 0;
    iterator = primero;
    cantidad = 0;
}

template <typename T>
void Lista<T>::agregarAlPrincipio(T elemento)
{
    nodo* nuevo = new nodo();
    nuevo->elemento = elemento;
    nuevo->sig = primero;
    primero = nuevo;
    if (cantidad == 0 )
        ultimo = primero;
    cantidad++;
}

template <typename T>
void Lista<T>::agregarAlFinal(T elemento)
{
    nodo* nuevo = new nodo();
    nuevo->elemento = elemento;
    nuevo->sig = 0;
    if (primero == 0)
    {
        primero = nuevo;
        ultimo = nuevo;
    }
    else
    {
        ultimo->sig = nuevo;
        ultimo = nuevo;
    }
    cantidad++;
}

template <typename T>
void Lista<T>::agregarEnPos(int pos, T elemento)
{
    int c = 0;
    if (pos == 0)
    {
        agregarAlPrincipio(elemento);
    }
    else
    {
        if (pos == cantidad)
        {
            agregarAlFinal(elemento);
        }
        else
            if (pos < cantidad)
            {
                agregarEnPos(primero, pos, elemento, c);
                cantidad++;
            }
    }
}
```

```
67
68  template<typename T>
69  void Lista<T>::agregarEnPos(nodo* &punt, int pos, T elemento, int c)
70  {
71      if (punt != 0)
72      {
73          if (c == pos)
74          {
75              nodo* nuevo = new nodo();
76              nuevo->elemento=elemento;
77              nuevo->sig=punt;
78              punt=nuevo;
79          };
80          agregarEnPos(punt->sig, pos, elemento, ++c);
81      }
82  }
83
84  template <typename T>
85  int Lista<T>::size() const
86  {
87      return cantidad;
88  }
89
90  template <typename T>
91  Lista<T>::~Lista()
92  {
93      while (primero != 0)
94      {
95          eliminar(0);
96      }
97  }
98
99  template <typename T>
100 bool Lista<T>::esVacia() const
101 {
102     return (primero == 0);
103 }
104
105 template <typename T>
106 bool Lista<T>::esta(T elemento)
107 {
108     return esta(primero, elemento);
109 }
110
111 template <typename T>
112 bool Lista<T>::esta(nodo* punt, T elemento)
113 {
114     if (punt != 0)
115     {
116         if (punt->elemento == elemento)
117             return true;
118         else
119             return esta(punt->sig, elemento);
120     }
121     else
122         return false;
123 }
124
125 template <typename T>
126 const T& Lista<T>::devolver(int pos)
127 {
128     assert(pos < cantidad);
129     int c = 0;
130     return devolver(primero, pos,c);
131 }
132
```

```
133 template <typename T>
134 const T& Lista<T>::devolver(nodo *punt, int pos, int c)
135 {
136     if (punt != 0 )
137     {
138         if (c == pos)
139         {
140             return punt->elemento;
141         }
142         devolver(punt->sig, pos, ++c);
143     }
144 }
145
146 template <typename T>
147 void Lista<T>::inic()
148 {
149     iterator = primero;
150 }
151
152 template <typename T>
153 void Lista<T>::sig()
154 {
155     iterator = iterator->sig;
156 }
157
158 template <typename T>
159 const T& Lista<T>::elemento()
160 {
161     return iterator->elemento;
162 }
163
164 template <typename T>
165 bool Lista<T>::final()
166 {
167     return (iterator == 0);
168 }
169
170 template <typename T>
171 void Lista<T>::eliminar(int pos)
172 {
173     assert(pos < cantidad);
174     nodo* cursor = primero;
175     nodo* aEliminar;
176     int c = 0;
177     bool termino = 0;
178     if (pos > 0)
179     {
180         while (!termino)
181         {
182             if (c == pos-1)
183             {
184                 aEliminar = cursor->sig;
185                 if (ultimo == aEliminar)
186                     ultimo = cursor;
187                 cursor->sig = aEliminar->sig;
188                 delete aEliminar;
189                 aEliminar = 0;
190                 termino = 1;
191             }
192             c++;
193             cursor =cursor->sig;
194         }
195     }
196     if ( pos == 0)
197     {
198         aEliminar = cursor;
```

```
                if (ultimo == aEliminar)
                    ultimo = 0;
                cursor = aEliminar->sig;
                delete aEliminar;
                aEliminar = 0;
                primero = cursor;
    }
    cantidad--;
}


/*****************************************************************************************/

template class Lista<int>;
```