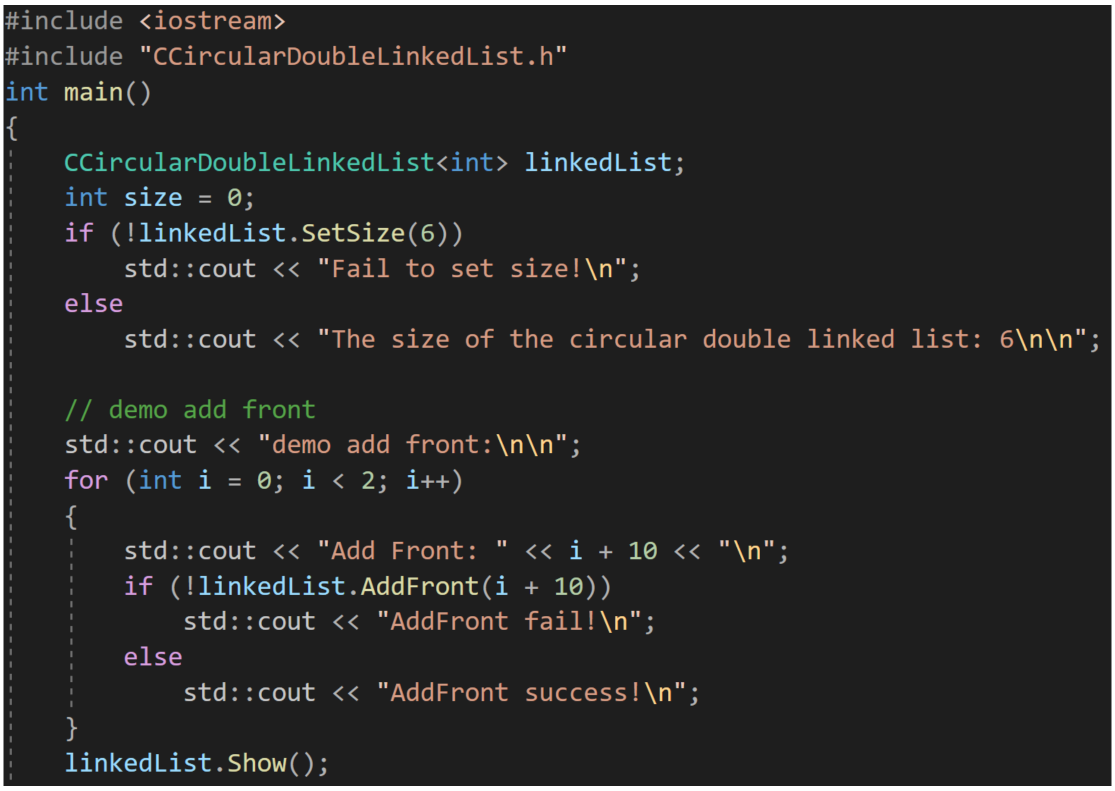
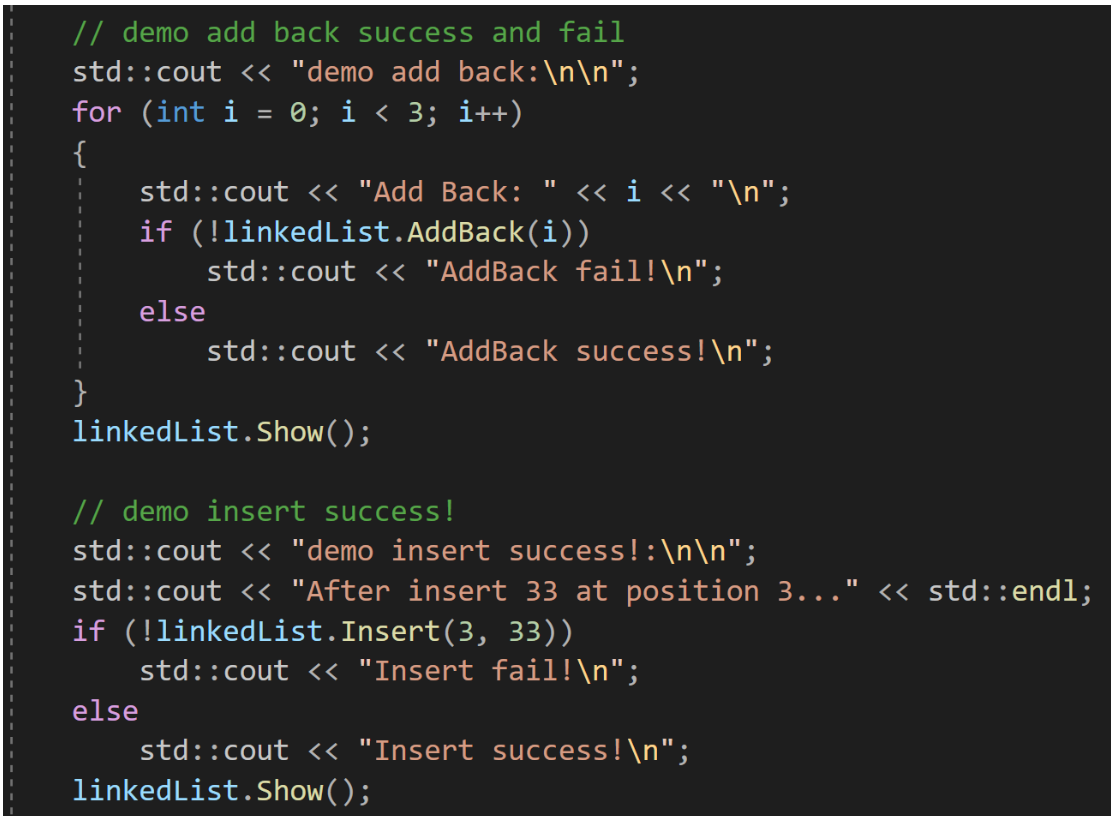
HW4:雙向環狀鏈結串列

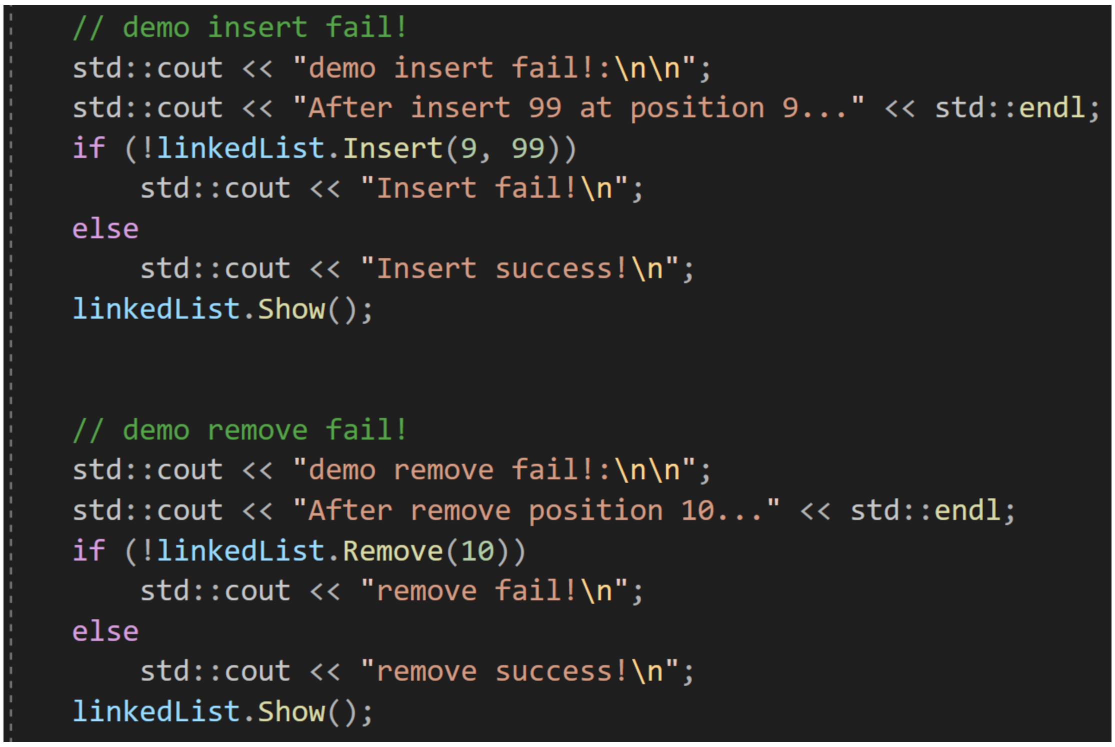
HW4CPP.cpp



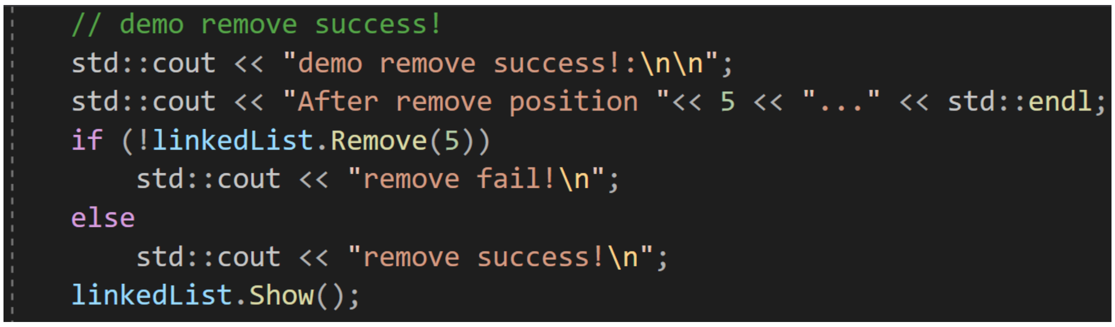
```cpp
#include <iostream>
#include "CCircularDoubleLinkedList.h"
int main()
{
    CCircularDoubleLinkedList<int> linkedList;
    int size = 0;
    if (!linkedList.SetSize(6))
        std::cout << "Fail to set size!\n";
    else
        std::cout << "The size of the circular double linked list: 6\n\n";

    // demo add front
    std::cout << "demo add front:\n\n";
    for (int i = 0; i < 2; i++)
    {
        std::cout << "Add Front: " << i + 10 << "\n";
        if (!linkedList.AddFront(i + 10))
            std::cout << "AddFront fail!\n";
        else
            std::cout << "AddFront success!\n";
    }
    linkedList.Show();
```



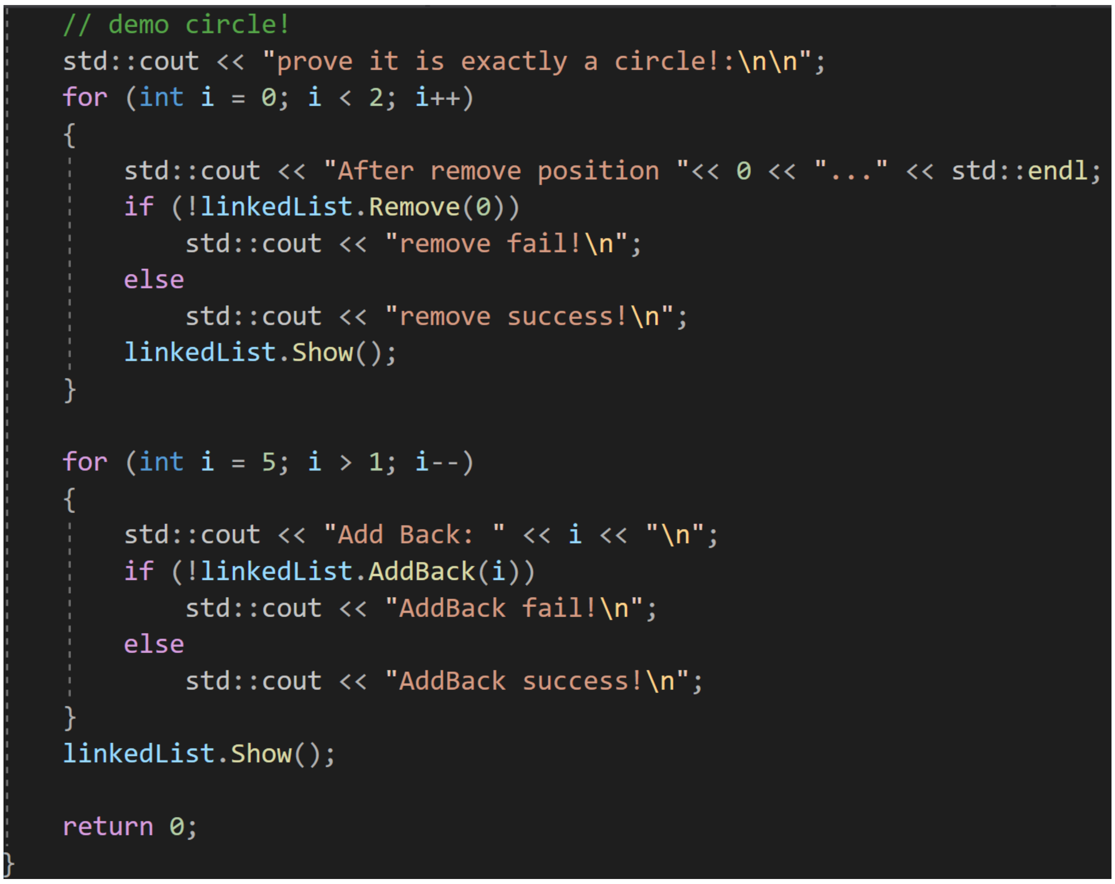
```cpp
    // demo add back success and fail
    std::cout << "demo add back:\n\n";
    for (int i = 0; i < 3; i++)
    {
        std::cout << "Add Back: " << i << "\n";
        if (!linkedList.AddBack(i))
            std::cout << "AddBack fail!\n";
        else
            std::cout << "AddBack success!\n";
    }
    linkedList.Show();

    // demo insert success!
    std::cout << "demo insert success!:\n\n";
    std::cout << "After insert 33 at position 3..." << std::endl;
    if (!linkedList.Insert(3, 33))
        std::cout << "Insert fail!\n";
    else
        std::cout << "Insert success!\n";
    linkedList.Show();
```



```cpp
    // demo insert fail!
    std::cout << "demo insert fail!:\n\n";
    std::cout << "After insert 99 at position 9..." << std::endl;
    if (!linkedList.Insert(9, 99))
        std::cout << "Insert fail!\n";
    else
        std::cout << "Insert success!\n";
    linkedList.Show();


    // demo remove fail!
    std::cout << "demo remove fail!:\n\n";
    std::cout << "After remove position 10..." << std::endl;
    if (!linkedList.Remove(10))
        std::cout << "remove fail!\n";
    else
        std::cout << "remove success!\n";
    linkedList.Show();
```

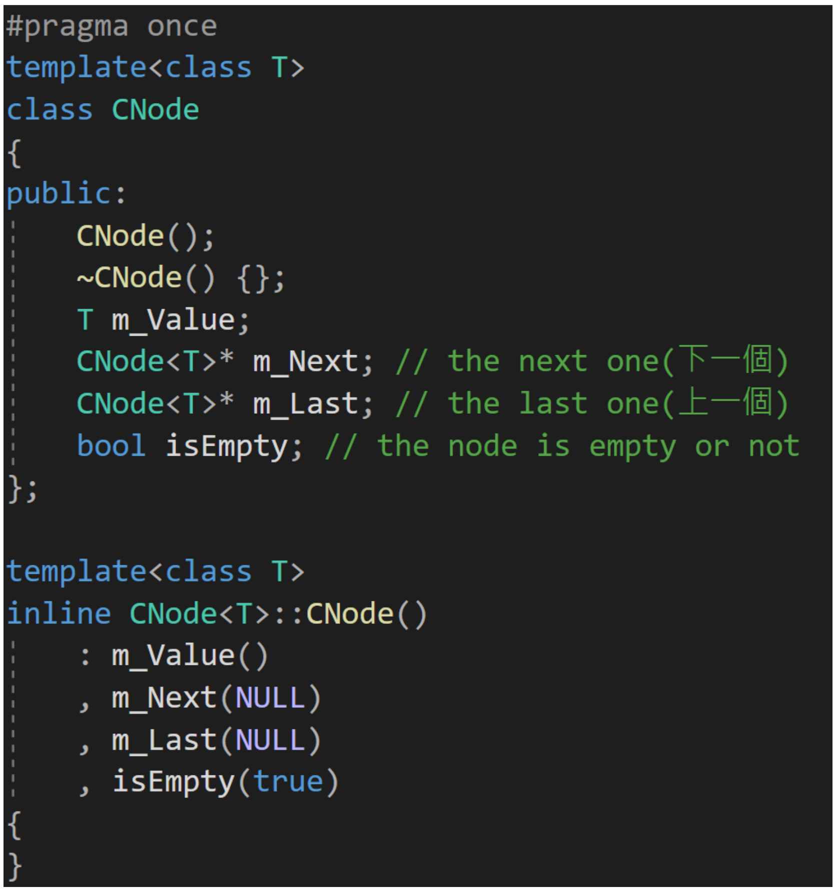


```cpp
    // demo remove success!
    std::cout << "demo remove success!:\n\n";
    std::cout << "After remove position "<< 5 << "..." << std::endl;
    if (!linkedList.Remove(5))
        std::cout << "remove fail!\n";
    else
        std::cout << "remove success!\n";
    linkedList.Show();
```



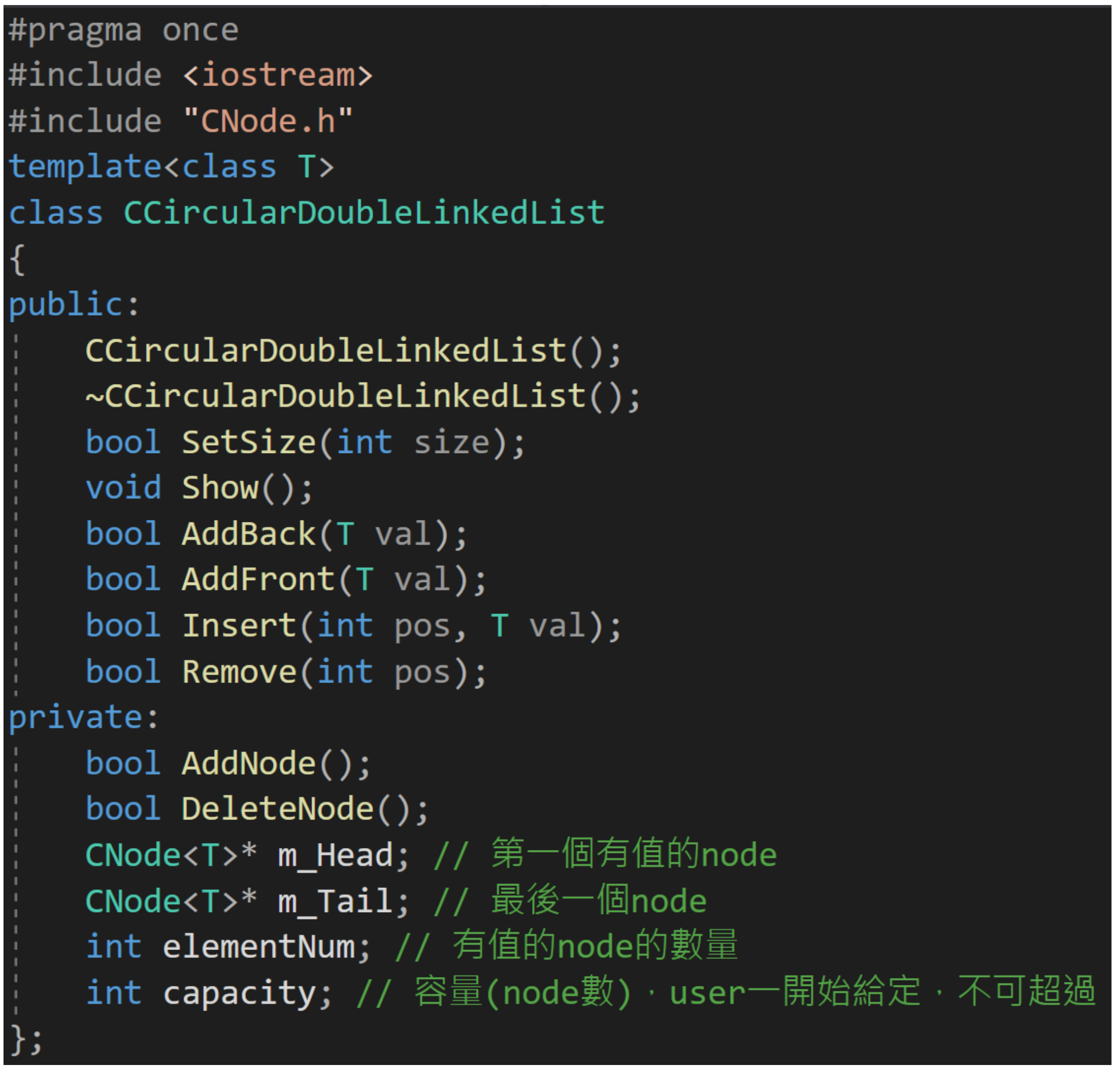
```cpp
    // demo circle!
    std::cout << "prove it is exactly a circle!:\n\n";
    for (int i = 0; i < 2; i++)
    {
        std::cout << "After remove position "<< 0 << "..." << std::endl;
        if (!linkedList.Remove(0))
            std::cout << "remove fail!\n";
        else
            std::cout << "remove success!\n";
        linkedList.Show();
    }

    for (int i = 5; i > 1; i--)
    {
        std::cout << "Add Back: " << i << "\n";
        if (!linkedList.AddBack(i))
            std::cout << "AddBack fail!\n";
        else
            std::cout << "AddBack success!\n";
    }
    linkedList.Show();

    return 0;
}
```

CNode.h

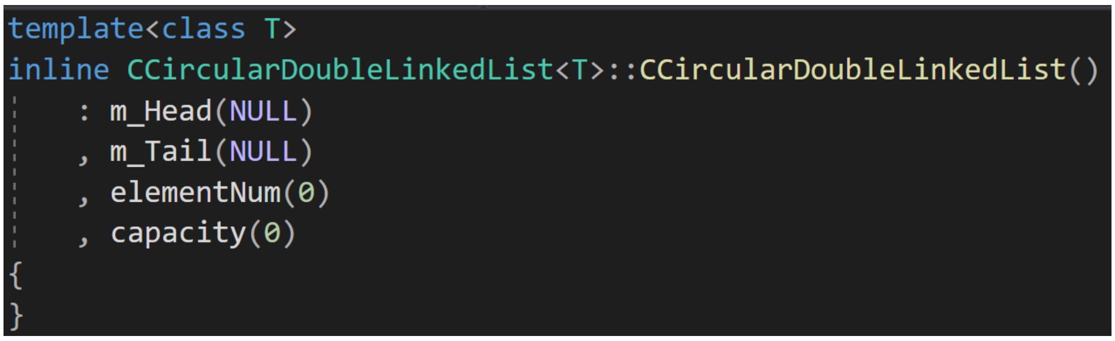


```cpp
#pragma once
template<class T>
class CNode
{
public:
    CNode();
    ~CNode() {};
    T m_Value;
    CNode<T>* m_Next; // the next one(下一個)
    CNode<T>* m_Last; // the last one(上一個)
    bool isEmpty; // the node is empty or not
};

template<class T>
inline CNode<T>::CNode()
    : m_Value()
    , m_Next(NULL)
    , m_Last(NULL)
    , isEmpty(true)
{
}
```

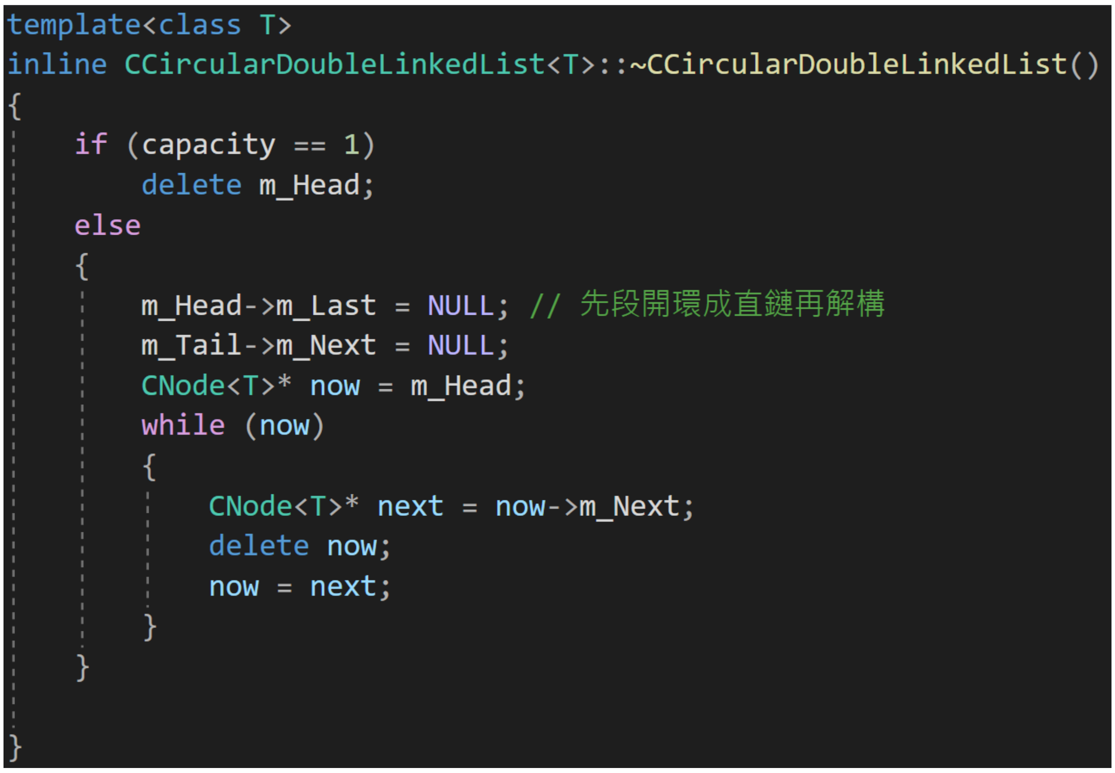
CCircularDoubleLinkedList.h



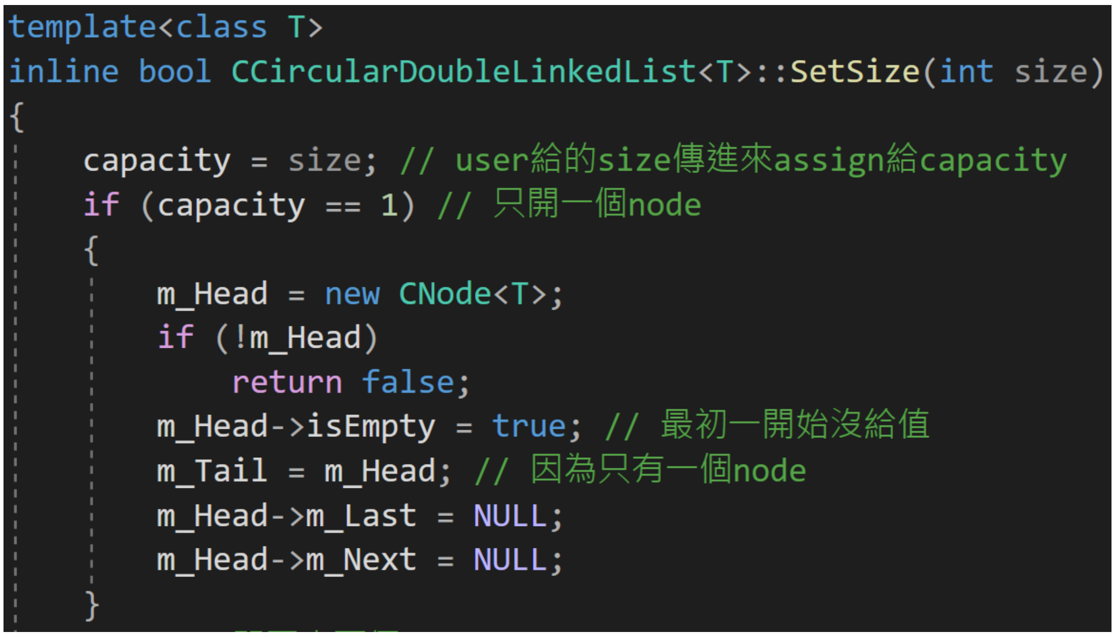
```cpp
#pragma once
#include <iostream>
#include "CNode.h"
template<class T>
class CCircularDoubleLinkedList
{
public:
    CCircularDoubleLinkedList();
    ~CCircularDoubleLinkedList();
    bool SetSize(int size);
    void Show();
    bool AddBack(T val);
    bool AddFront(T val);
    bool Insert(int pos, T val);
    bool Remove(int pos);
private:
    bool AddNode();
    bool DeleteNode();
    CNode<T>* m_Head; // 第一個有值的node
    CNode<T>* m_Tail; // 最後一個node
    int elementNum; // 有值的node的數量
    int capacity; // 容量(node數)，user一開始給定，不可超過
};
```



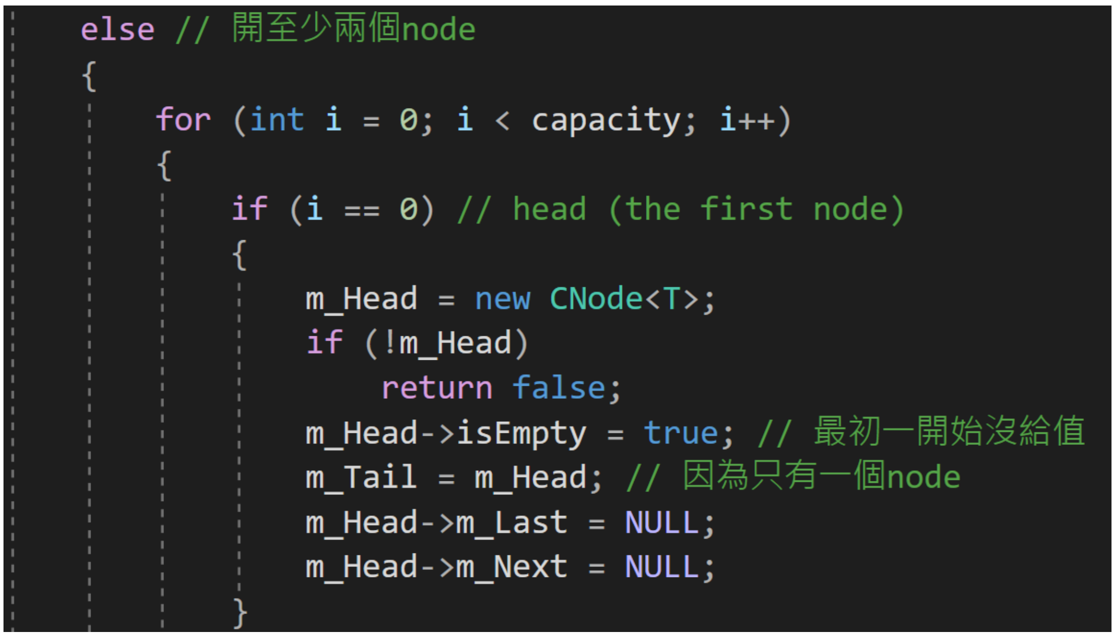
```cpp
template<class T>
inline CCircularDoubleLinkedList<T>::CCircularDoubleLinkedList()
    : m_Head(NULL)
    , m_Tail(NULL)
    , elementNum(0)
    , capacity(0)
{
}
```



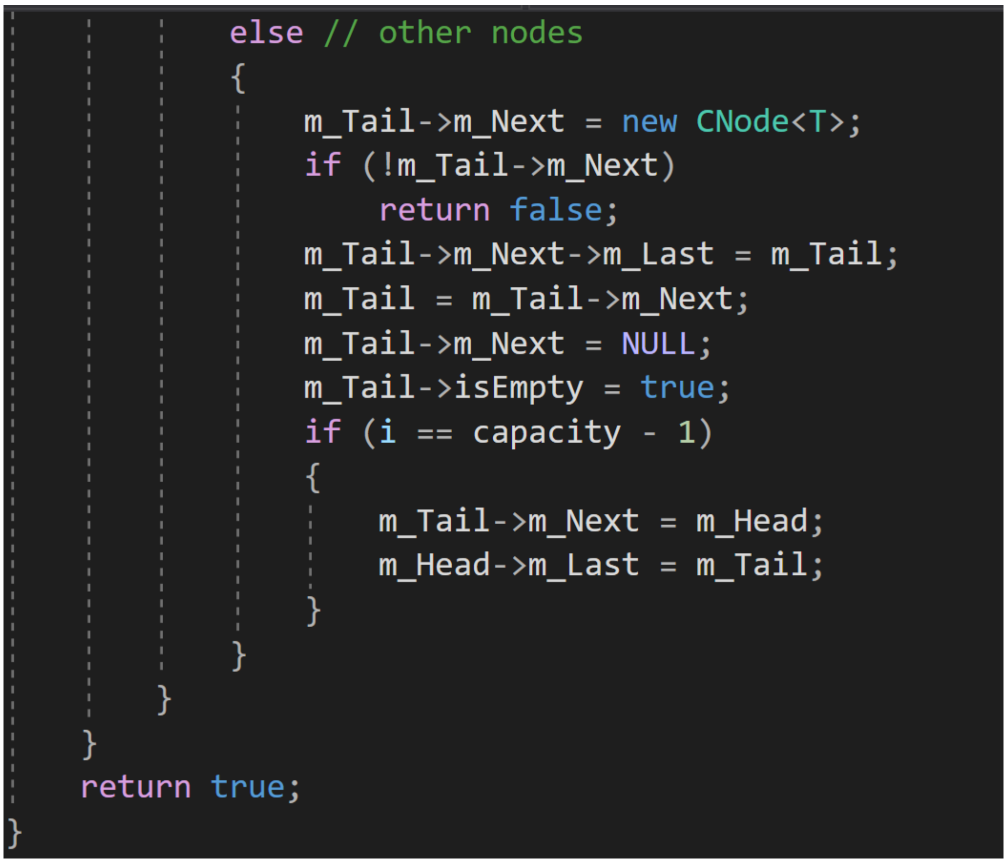
```cpp
template<class T>
inline CCircularDoubleLinkedList<T>::~CCircularDoubleLinkedList()
{
    if (capacity == 1)
        delete m_Head;
    else
    {
        m_Head->m_Last = NULL; // 先段開環成直鏈再解構
        m_Tail->m_Next = NULL;
        CNode<T>* now = m_Head;
        while (now)
        {
            CNode<T>* next = now->m_Next;
            delete now;
            now = next;
        }
    }

}
```



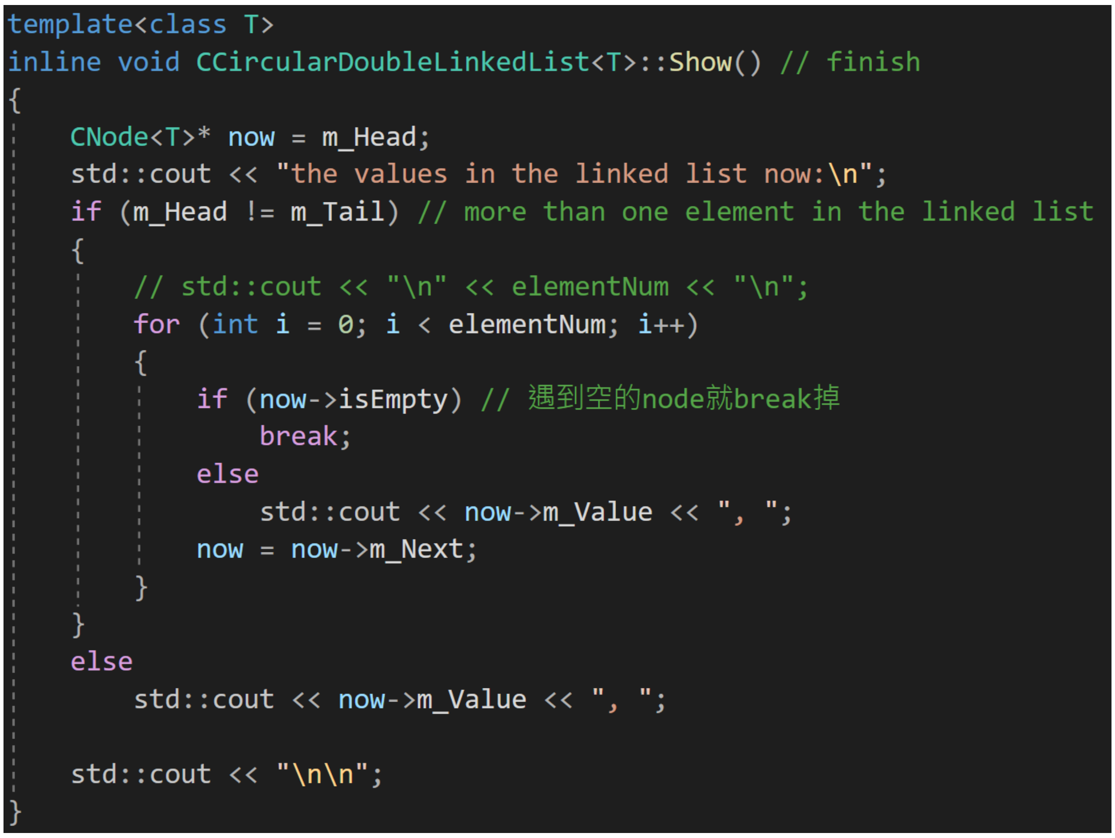
```cpp
template<class T>
inline bool CCircularDoubleLinkedList<T>::SetSize(int size)
{
    capacity = size; // user給的size傳進來assign給capacity
    if (capacity == 1) // 只開一個node
    {
        m_Head = new CNode<T>;
        if (!m_Head)
            return false;
        m_Head->isEmpty = true; // 最初一開始沒給值
        m_Tail = m_Head; // 因為只有一個node
        m_Head->m_Last = NULL;
        m_Head->m_Next = NULL;
    }
```



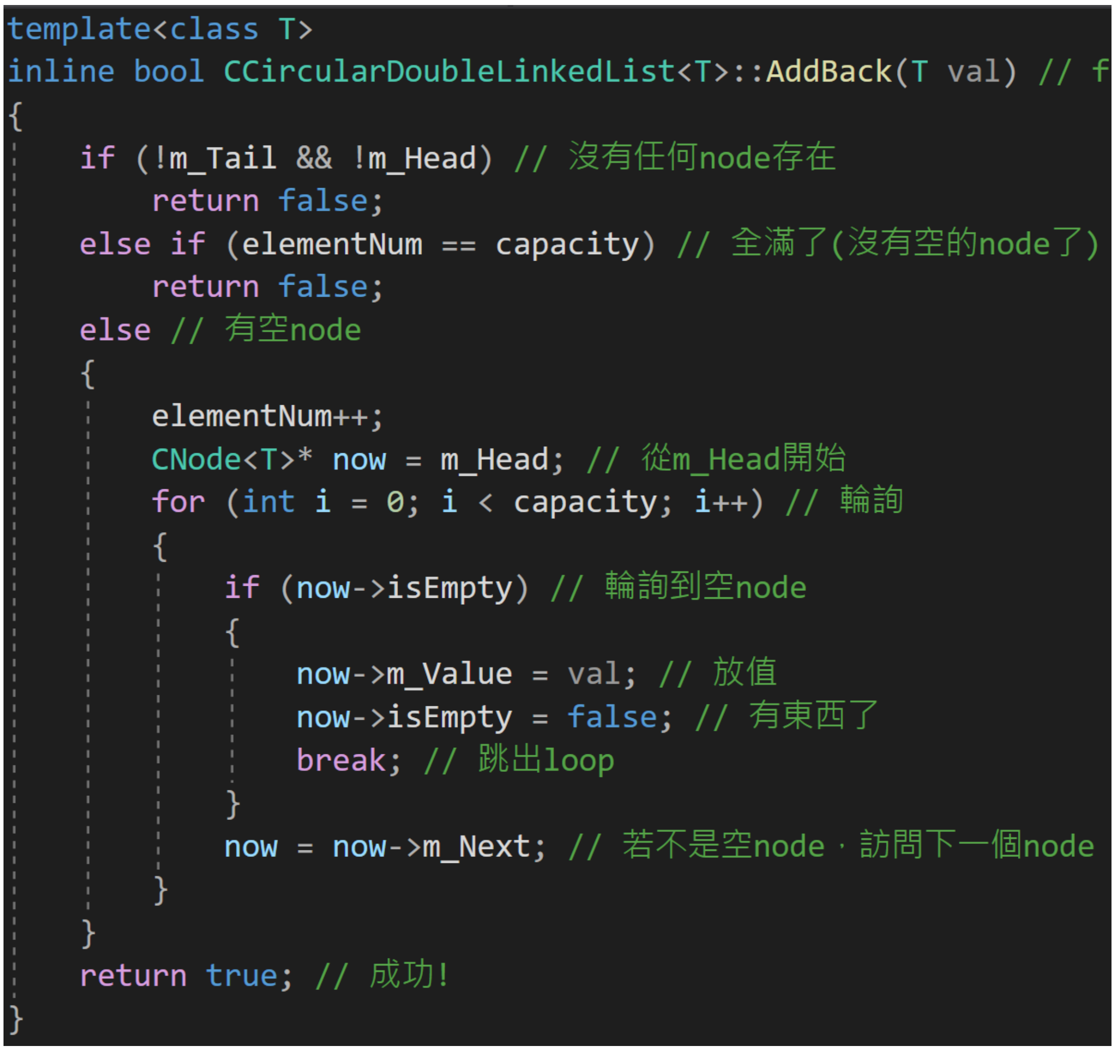
```cpp
    else // 開至少兩個node
    {
        for (int i = 0; i < capacity; i++)
        {
            if (i == 0) // head (the first node)
            {
                m_Head = new CNode<T>;
                if (!m_Head)
                    return false;
                m_Head->isEmpty = true; // 最初一開始沒給值
                m_Tail = m_Head; // 因為只有一個node
                m_Head->m_Last = NULL;
                m_Head->m_Next = NULL;
            }
```



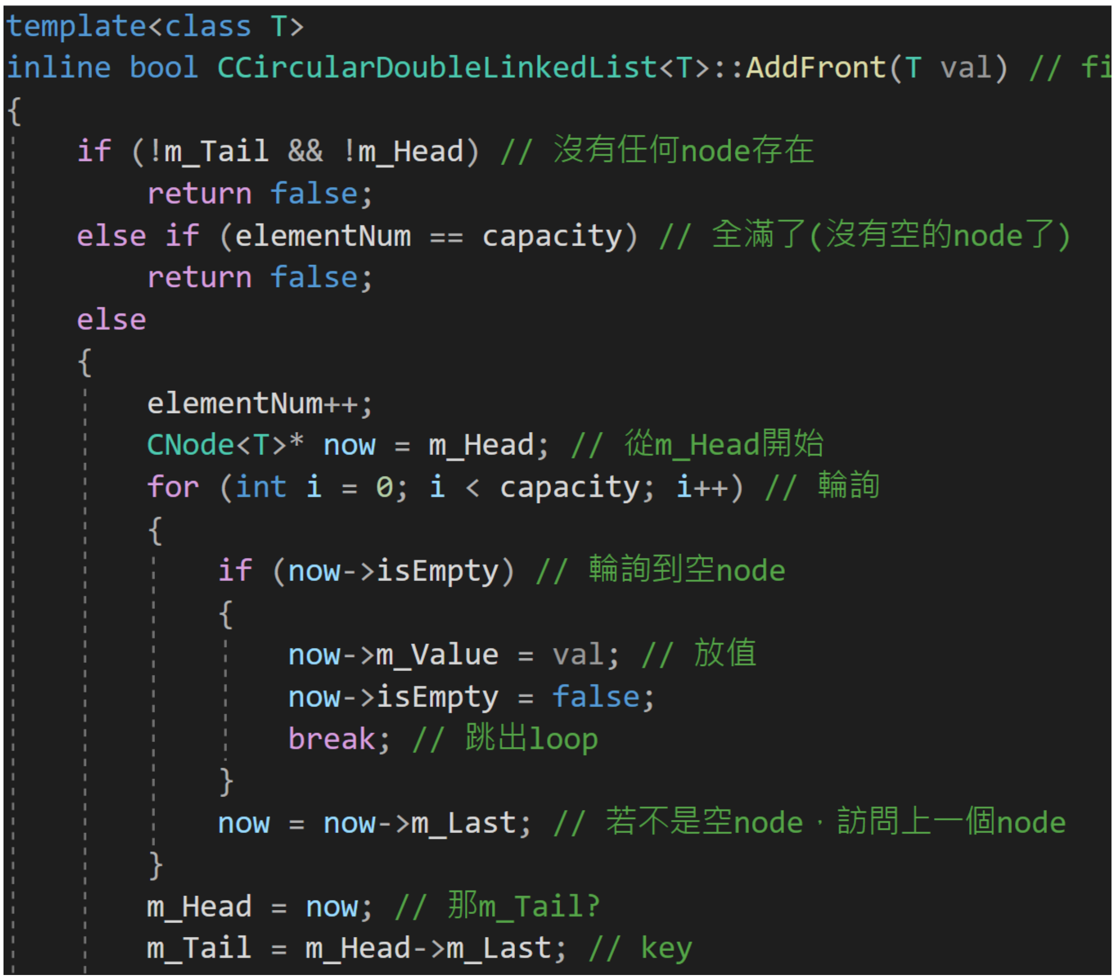
```cpp
            else // other nodes
            {
                m_Tail->m_Next = new CNode<T>;
                if (!m_Tail->m_Next)
                    return false;
                m_Tail->m_Next->m_Last = m_Tail;
                m_Tail = m_Tail->m_Next;
                m_Tail->m_Next = NULL;
                m_Tail->isEmpty = true;
                if (i == capacity - 1)
                {
                    m_Tail->m_Next = m_Head;
                    m_Head->m_Last = m_Tail;
                }
            }
        }
    }
    return true;
}
```



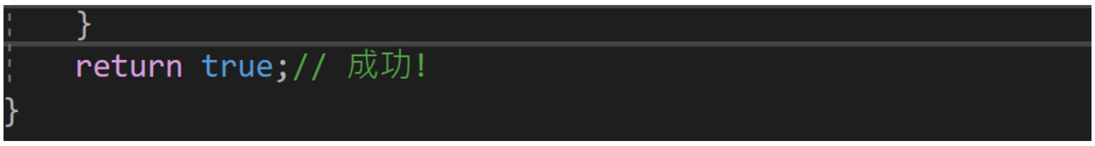
```cpp
template<class T>
inline void CCircularDoubleLinkedList<T>::Show() // finish
{
    CNode<T>* now = m_Head;
    std::cout << "the values in the linked list now:\n";
    if (m_Head != m_Tail) // more than one element in the linked list
    {
        // std::cout << "\n" << elementNum << "\n";
        for (int i = 0; i < elementNum; i++)
        {
            if (now->isEmpty) // 遇到空的node就break掉
                break;
            else
                std::cout << now->m_Value << ", ";
            now = now->m_Next;
        }
    }
    else
        std::cout << now->m_Value << ", ";

    std::cout << "\n\n";
}
```



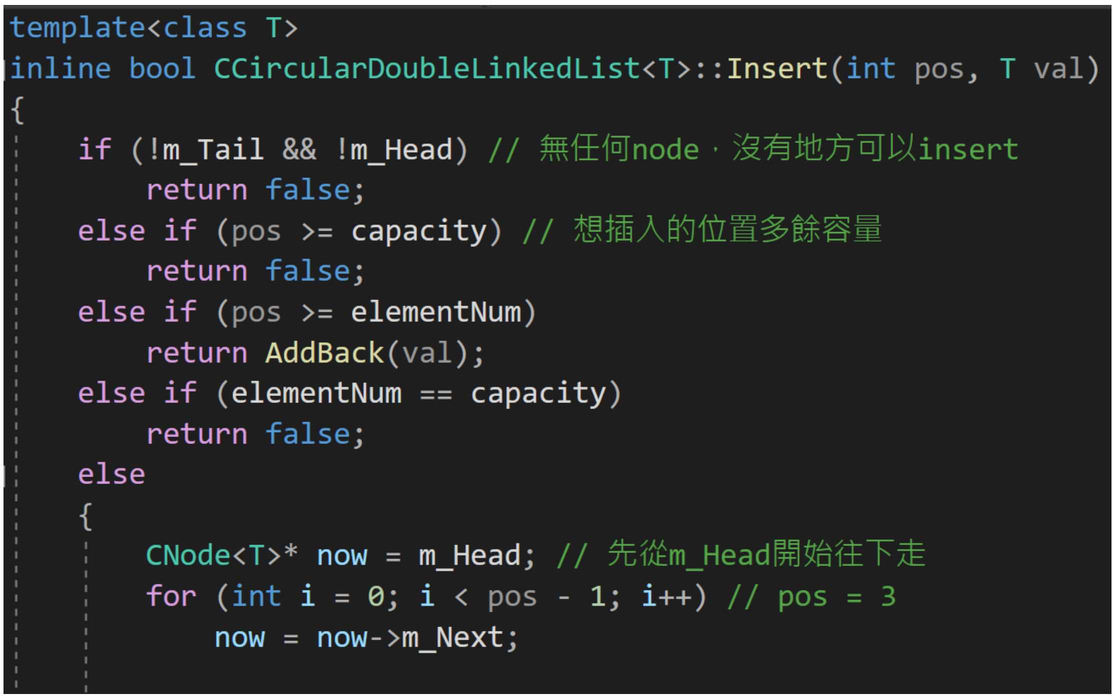
```
template<class T>
inline bool CCircularDoubleLinkedList<T>::AddBack(T val) // f
{
    if (!m_Tail && !m_Head) // 沒有任何node存在
        return false;
    else if (elementNum == capacity) // 全滿了(沒有空的node了)
        return false;
    else // 有空node
    {
        elementNum++;
        CNode<T>* now = m_Head; // 從m_Head開始
        for (int i = 0; i < capacity; i++) // 輪詢
        {
            if (now->isEmpty) // 輪詢到空node
            {
                now->m_Value = val; // 放值
                now->isEmpty = false; // 有東西了
                break; // 跳出loop
            }
            now = now->m_Next; // 若不是空node，訪問下一個node
        }
    }
    return true; // 成功!
}
```



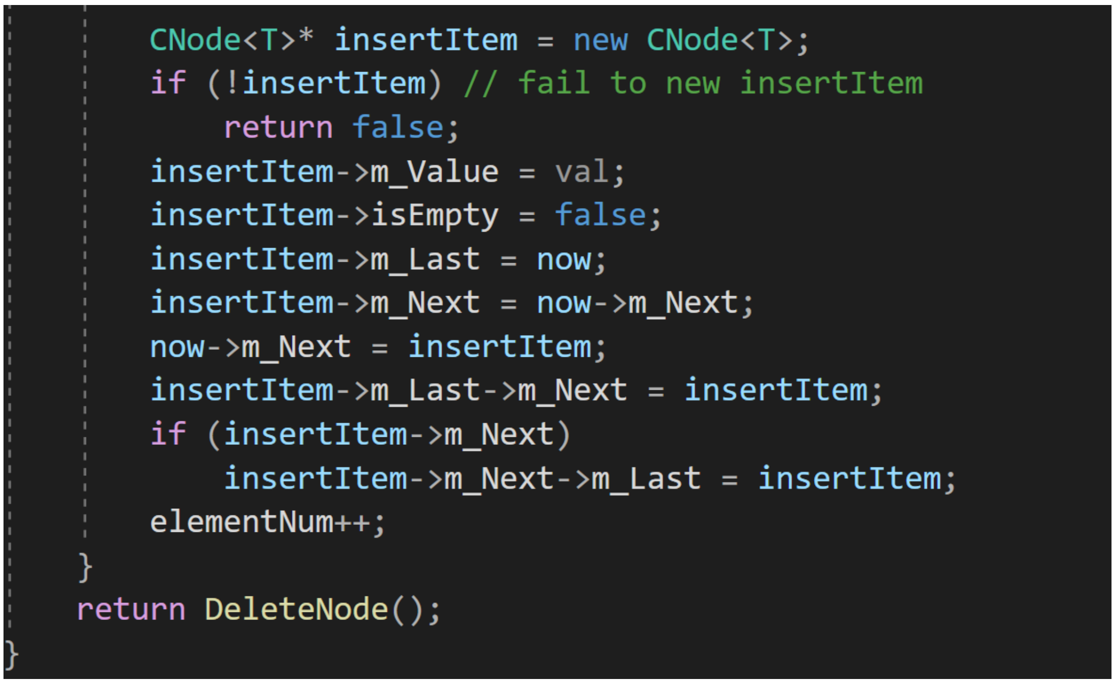
```cpp
template<class T>
inline bool CCircularDoubleLinkedList<T>::AddFront(T val) // fi
{
    if (!m_Tail && !m_Head) // 沒有任何node存在
        return false;
    else if (elementNum == capacity) // 全滿了(沒有空的node了)
        return false;
    else
    {
        elementNum++;
        CNode<T>* now = m_Head; // 從m_Head開始
        for (int i = 0; i < capacity; i++) // 輪詢
        {
            if (now->isEmpty) // 輪詢到空node
            {
                now->m_Value = val; // 放值
                now->isEmpty = false;
                break; // 跳出loop
            }
            now = now->m_Last; // 若不是空node，訪問上一個node
        }
        m_Head = now; // 那m_Tail?
        m_Tail = m_Head->m_Last; // key
```



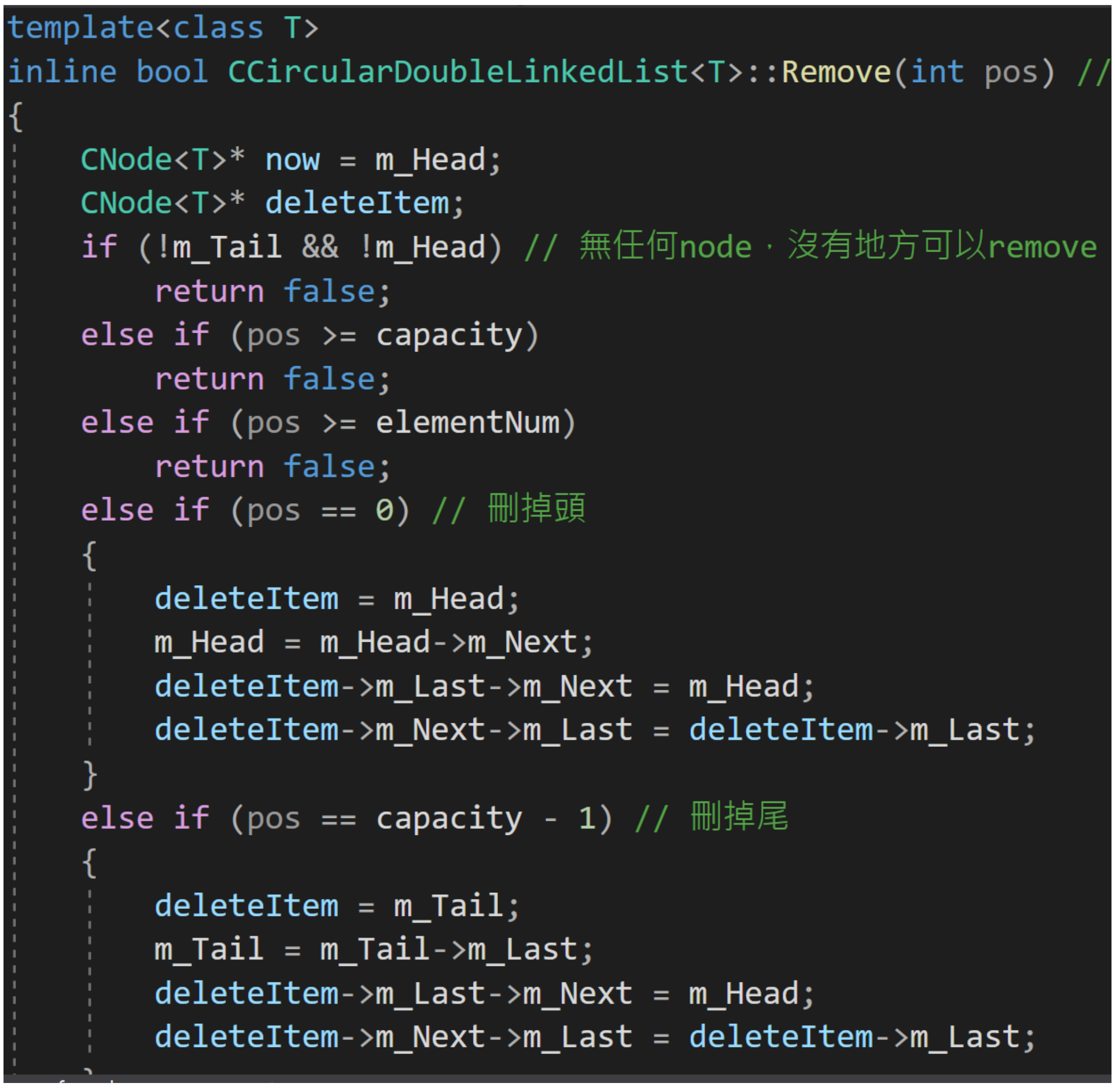
```cpp
    }
    return true;// 成功!
}
```



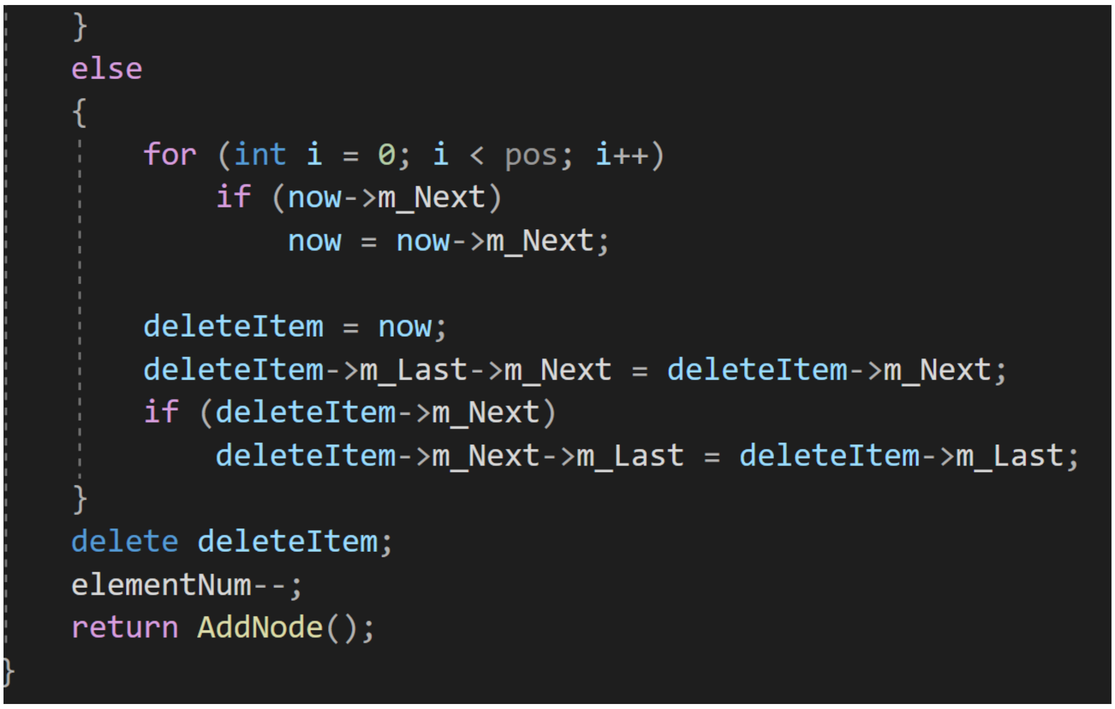
```cpp
template<class T>
inline bool CCircularDoubleLinkedList<T>::Insert(int pos, T val)
{
    if (!m_Tail && !m_Head) // 無任何node，沒有地方可以insert
        return false;
    else if (pos >= capacity) // 想插入的位置多餘容量
        return false;
    else if (pos >= elementNum)
        return AddBack(val);
    else if (elementNum == capacity)
        return false;
    else
    {
        CNode<T>* now = m_Head; // 先從m_Head開始往下走
        for (int i = 0; i < pos - 1; i++) // pos = 3
            now = now->m_Next;
```



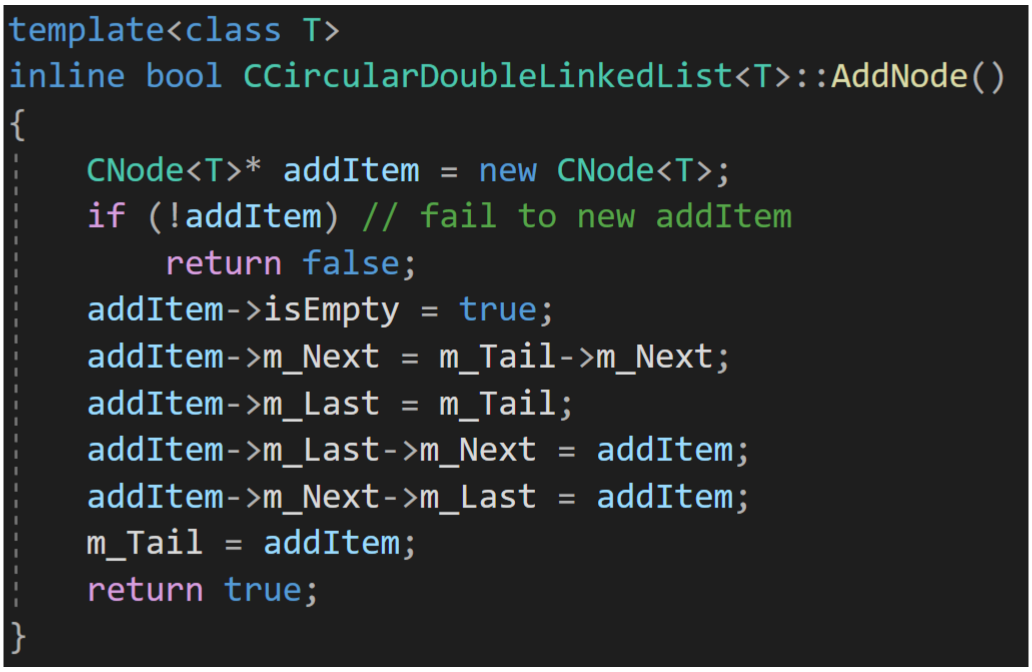
```
        CNode<T>* insertItem = new CNode<T>;
        if (!insertItem) // fail to new insertItem
            return false;
        insertItem->m_Value = val;
        insertItem->isEmpty = false;
        insertItem->m_Last = now;
        insertItem->m_Next = now->m_Next;
        now->m_Next = insertItem;
        insertItem->m_Last->m_Next = insertItem;
        if (insertItem->m_Next)
            insertItem->m_Next->m_Last = insertItem;
        elementNum++;
    }
    return DeleteNode();
}
```



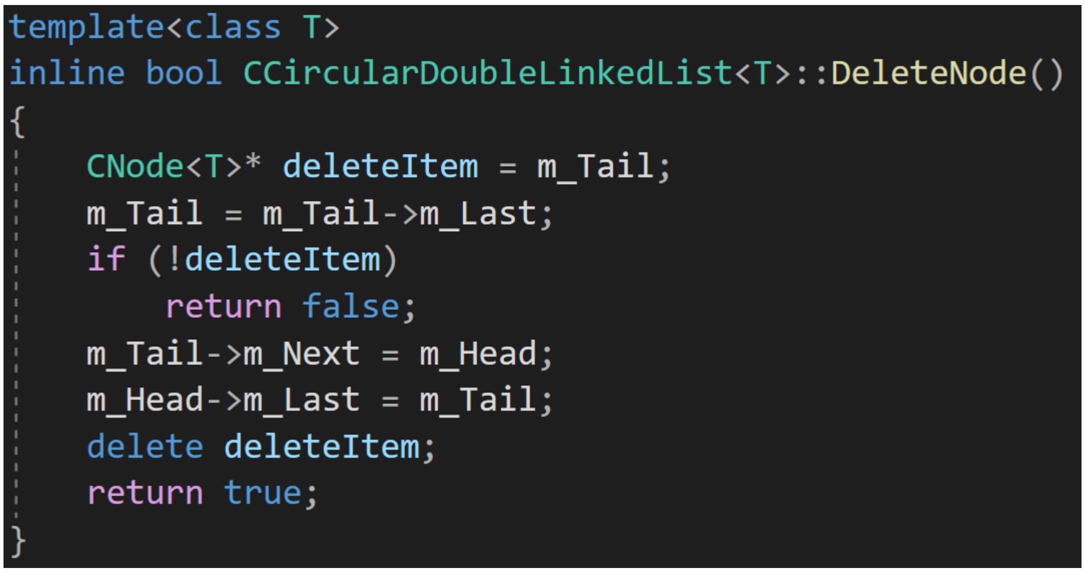
```
template<class T>
inline bool CCircularDoubleLinkedList<T>::Remove(int pos) //
{
    CNode<T>* now = m_Head;
    CNode<T>* deleteItem;
    if (!m_Tail && !m_Head) // 無任何node，沒有地方可以remove
        return false;
    else if (pos >= capacity)
        return false;
    else if (pos >= elementNum)
        return false;
    else if (pos == 0) // 刪掉頭
    {
        deleteItem = m_Head;
        m_Head = m_Head->m_Next;
        deleteItem->m_Last->m_Next = m_Head;
        deleteItem->m_Next->m_Last = deleteItem->m_Last;
    }
    else if (pos == capacity - 1) // 刪掉尾
    {
        deleteItem = m_Tail;
        m_Tail = m_Tail->m_Last;
        deleteItem->m_Last->m_Next = m_Head;
        deleteItem->m_Next->m_Last = deleteItem->m_Last;
```



```cpp
    }
    else
    {
        for (int i = 0; i < pos; i++)
            if (now->m_Next)
                now = now->m_Next;

        deleteItem = now;
        deleteItem->m_Last->m_Next = deleteItem->m_Next;
        if (deleteItem->m_Next)
            deleteItem->m_Next->m_Last = deleteItem->m_Last;
    }
    delete deleteItem;
    elementNum--;
    return AddNode();
}
```

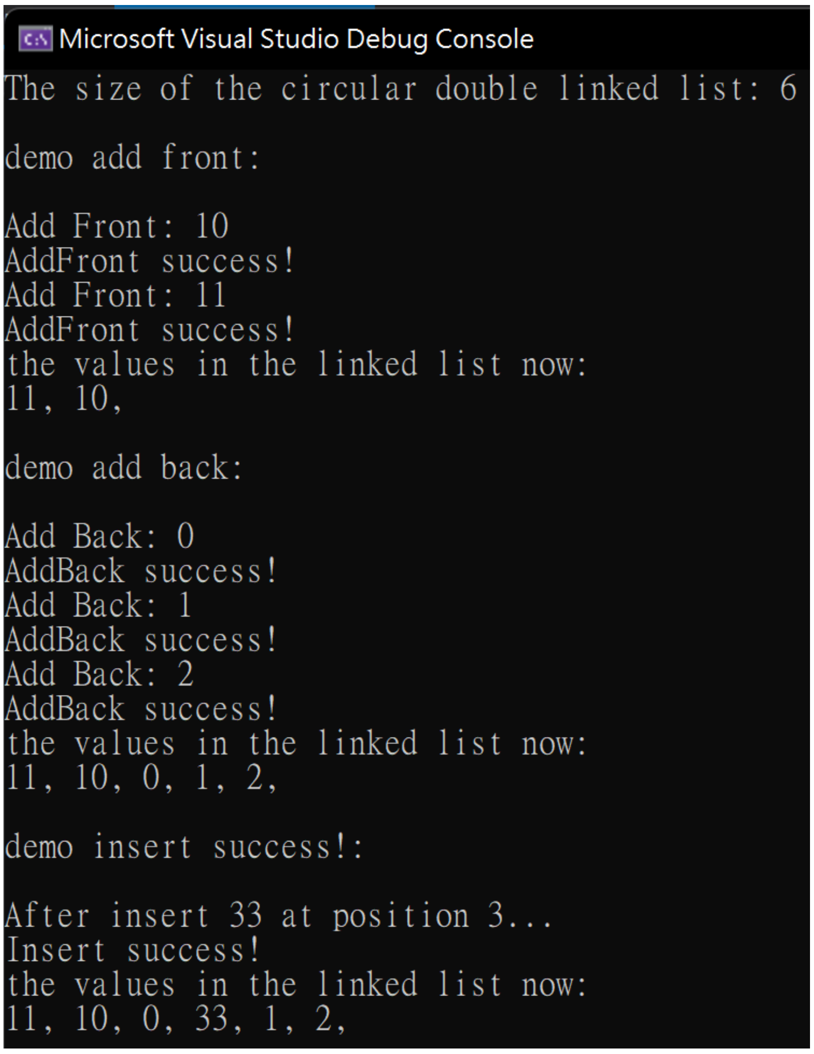


```cpp
template<class T>
inline bool CCircularDoubleLinkedList<T>::AddNode()
{
    CNode<T>* addItem = new CNode<T>;
    if (!addItem) // fail to new addItem
        return false;
    addItem->isEmpty = true;
    addItem->m_Next = m_Tail->m_Next;
    addItem->m_Last = m_Tail;
    addItem->m_Last->m_Next = addItem;
    addItem->m_Next->m_Last = addItem;
    m_Tail = addItem;
    return true;
}
```

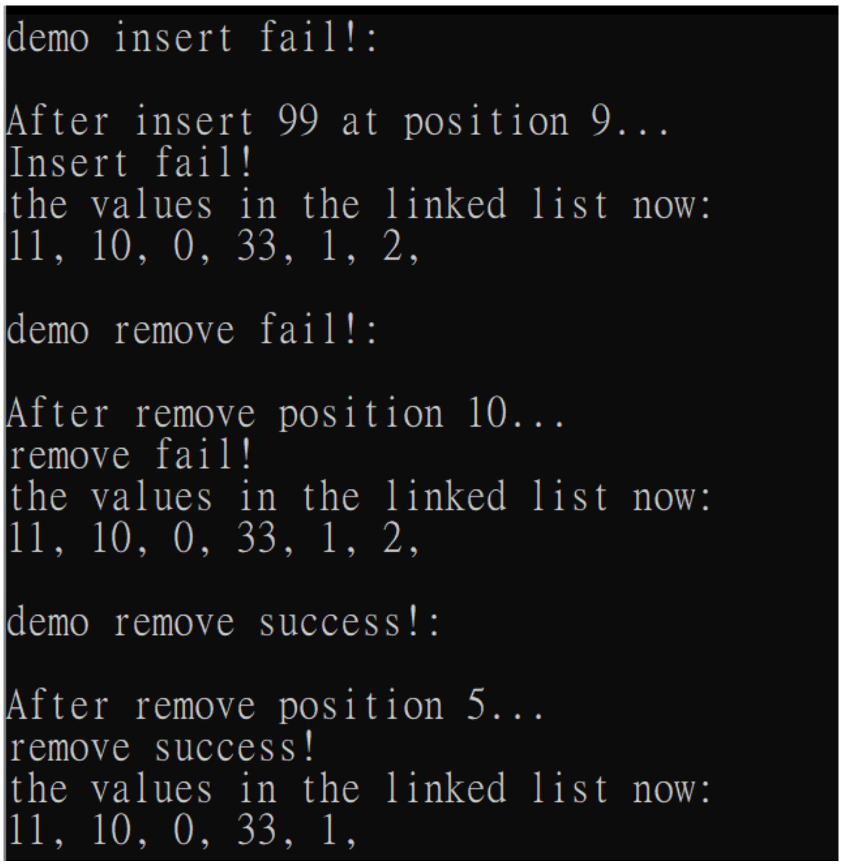


```cpp
template<class T>
inline bool CCircularDoubleLinkedList<T>::DeleteNode()
{
    CNode<T>* deleteItem = m_Tail;
    m_Tail = m_Tail->m_Last;
    if (!deleteItem)
        return false;
    m_Tail->m_Next = m_Head;
    m_Head->m_Last = m_Tail;
    delete deleteItem;
    return true;
}
```

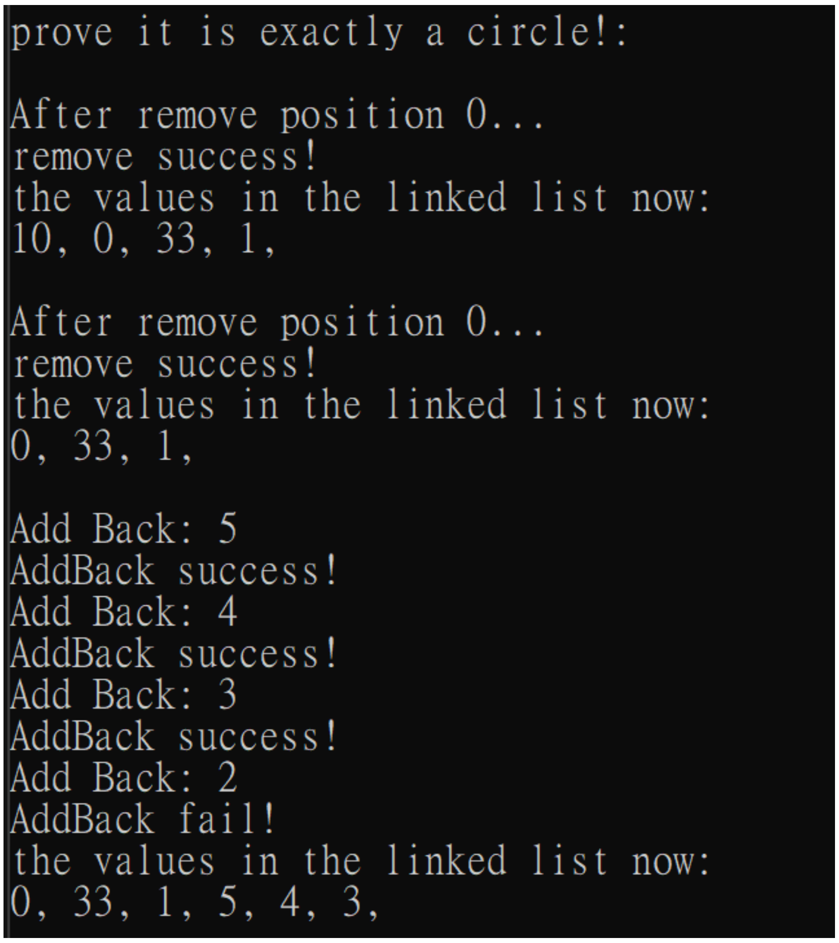
Result:



```
The size of the circular double linked list: 6

demo add front:

Add Front: 10
AddFront success!
Add Front: 11
AddFront success!
the values in the linked list now:
11, 10,

demo add back:

Add Back: 0
AddBack success!
Add Back: 1
AddBack success!
Add Back: 2
AddBack success!
the values in the linked list now:
11, 10, 0, 1, 2,

demo insert success!:

After insert 33 at position 3...
Insert success!
the values in the linked list now:
11, 10, 0, 33, 1, 2,
```

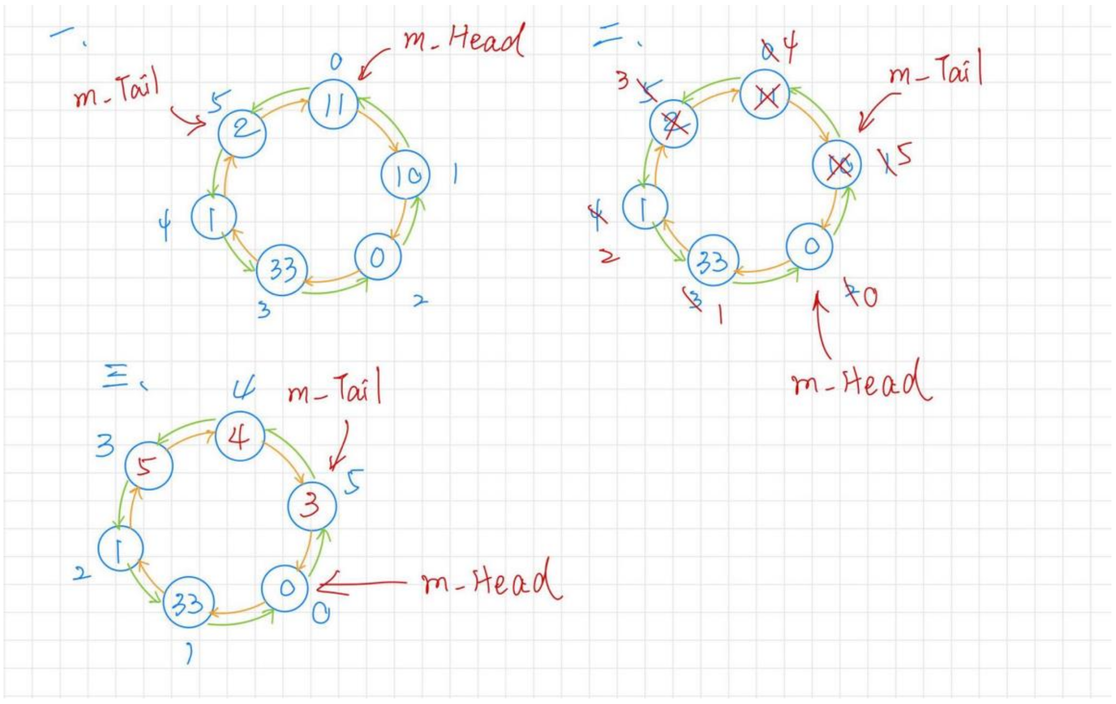


```
demo insert fail!:

After insert 99 at position 9...
Insert fail!
the values in the linked list now:
11, 10, 0, 33, 1, 2,

demo remove fail!:

After remove position 10...
remove fail!
the values in the linked list now:
11, 10, 0, 33, 1, 2,

demo remove success!:

After remove position 5...
remove success!
the values in the linked list now:
11, 10, 0, 33, 1,
```



```
prove it is exactly a circle!:

After remove position 0...
remove success!
the values in the linked list now:
10, 0, 33, 1,

After remove position 0...
remove success!
the values in the linked list now:
0, 33, 1,

Add Back: 5
AddBack success!
Add Back: 4
AddBack success!
Add Back: 3
AddBack success!
Add Back: 2
AddBack fail!
the values in the linked list now:
0, 33, 1, 5, 4, 3,
```

圖像:


一、
m_Head
m_Tail
11
10
0
33
1
2
二、
m_Tail
m_Head
33
1
0
三、
m_Tail
m_Head
4
5
3
1
33
0