\*\*\*\*\*\*\*\*\*\*\*基于AList的线性表实现\*\*\*\*\*\*\*\*\*\*\*

1. **List抽象类(list.h)：**

#ifndef LIST

#define LIST

template <typename E> class List { // List ADT 抽象数据类型定义

private:

void operator =(const List&) {} // Protect assignment

List(const List&) {} // Protect copy constructor

public:

List() {} // Default constructor 构造函数

virtual ~List() {} // Base destructor 析构函数

// Clear contents from the list, to make it empty. 清空列表中的内容

virtual void clear() = 0;

// Insert an element at the current location.

// item: The element to be inserted 在当前位置插入元素item

virtual void insert(const E& item) = 0;

// Append an element at the end of the list.

// item: The element to be appended 在表尾添加元素item

virtual void append(const E& item) = 0;

// Remove and return the current element.

// Return: the element that was removed. 删除当前元素，并将其作为返回值

virtual E remove() = 0;

// Set the current position to the start of the list. 将当前位置设置为顺序表起始处

virtual void moveToStart() = 0;

// Set the current position to the end of the list. 将当前位置设置为顺序表末尾

virtual void moveToEnd() = 0;

// Move the current position one step left. No change

// if already at beginning. 将当前位置左移一步，如果当前位置在首位就不变

virtual void prev() = 0;

// Move the current position one step right. No change

// if already at end. 将当前位置右移一步，如果当前位置在末尾就不变

virtual void next() = 0;

// Return: The number of elements in the list. 返回列表当前的元素个数

virtual int length() const = 0;

// Return: The position of the current element. 返回当前元素的位置

virtual int currPos() const = 0;

// Set current position.

// pos: The position to make current. 将当前位置设置为pos

virtual void moveToPos(int pos) = 0;

// Return: The current element. 返回当前元素

virtual const E& getValue() const = 0;

};

#endif

1. **AList物理实现(ALsit.h)：**

#include "list.h"

#include <assert.h>

template <typename E> // Array-based list implementation 基于数组的线性表实现

class AList : public List<E> {

private:

int maxSize; // Maximum size of list 顺序表的容量

int listSize; // Number of list items now 目前的大小

int curr; // Position of current element 当前元素的位置

E\* listArray; // Array holding list elements 列表元素将存放到该数组中

public:

AList(int size=100) { // Constructor 构造函数

maxSize = size;

listSize = curr = 0;

listArray = new E[maxSize];

}

~AList() { delete [] listArray; } // Destructor 析构函数

void clear() { // Reinitialize the list 初始化顺序表

delete [] listArray; // Remove the array 删除原有数组

listSize = curr = 0; // Reset the size 重新设置列表参数

listArray = new E[maxSize]; // Recreate array 新建空数组

}

// Insert "it" at current position 在当前位置插入it

void insert(const E& it) {

assert(listSize < maxSize); //"List capacity exceeded"超出顺序表范围会断言终止程序

for(int i=listSize; i>curr; i--) // Shift elements up 右移元素

listArray[i] = listArray[i-1]; // to make room

listArray[curr] = it;

listSize++; // Increment list size 列表长度加一

}

void append(const E& it) { // Append "it" 在顺序表的末尾追加it

assert(listSize < maxSize); // 断言判断

listArray[listSize++] = it;

}

// Remove and return the current element. 删除并返回当前元素

E remove() {

assert((curr>=0) && (curr < listSize)); // No element 没有元素会断言终止程序

E it = listArray[curr]; // Copy the element 拷贝当前元素

for(int i=curr; i<listSize-1; i++) // Shift them down 左移元素

//listArray[curr]将被覆盖掉

listArray[i] = listArray[i+1];

listSize--; // Decrement size 列表长度减一

return it;

}

void moveToStart() { curr = 0; } // Reset position. 将curr当前位置设置为开头

void moveToEnd() { curr = listSize; } // Set at end . 将curr设置为末尾

void prev() { if (curr != 0) curr--; } // Back up. 将curr前移一位

void next() { if (curr < listSize) curr++; } // Next. 将curr后移一位

// Return list size 返回顺序表长度

int length() const { return listSize; }

// Return current position 返回当前位置

int currPos() const { return curr; }

// Set current list position to "pos" 将当前位置curr设置为pos

void moveToPos(int pos) {

assert ((pos>=0)&&(pos<=listSize)); // Pos out of range

//pos超出顺序表范围会断言终止程序

curr = pos;

}

const E& getValue() const { // Return current element 返回值是当前元素

assert((curr>=0)&&(curr<listSize)); //No current element

//当前位置超出范围时断言终止程序

return listArray[curr];

}

};

1. **主函数测试部分(AListTest.cpp)：**

#include "list.h"

#include "alist.h"

#include<iomanip>

#include<iostream>

using namespace std;

void print(AList<int>& A,int n);

int main(){

int n;

AList<int> L1(100);

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\*顺序表中各个基本操作演示\*"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl<<endl;

cout<<"向顺序表中依次添加元素12、15、9、48、55"<<endl;

//调用顺序表中的append操作

L1.append(12);

L1.append(15);

L1.append(9);

L1.append(48);

L1.append(55);

cout<<"添加元素后顺序表中内容为："<<endl;

n=L1.length(); //获取顺序表长度

print(L1,n);

cout<<"当前顺序表长度为:"<<endl<<L1.length()<<endl<<endl;

cout<<"查看当前curr的值为："<<endl;

cout<<L1.currPos()<<endl<<endl;

cout<<"设置线性表中栅栏curr的位置为2"<<endl;

L1.moveToPos(2);

cout<<"并在当前位置curr插入元素4"<<endl;

//调用顺序表中的插入操作

L1.insert(4);

cout<<"插入元素后顺序表中内容为："<<endl;

n=L1.length(); //获取顺序表长度

print(L1,n);

cout<<"设置线性表中栅栏curr的位置为4"<<endl;

L1.moveToPos(4);

cout<<"并删除当前位置curr处的元素" <<endl;

//调用顺序表的删除操作

L1.remove();

cout<<"删除元素后顺序表中的内容为："<<endl;

n=L1.length(); //获取顺序表长度

print(L1,n);

}

/\*\*

\*function of print

\*打印顺序表中元素

\*\*/

void print(AList<int>& A,int n){

int i;

for(i=0;i<n;i++){

A.moveToPos(i);

cout<<A.getValue()<<" "; //调用getValue操作得到当前元素

}

cout<<endl<<endl;

}

1. **测试结果示例：**



【附录】

版本信息声明：

Dev-C++ 5.11

TDM-GCC 4.9.2 64-bit Release

部分代码来源：

<http://people.cs.vt.edu/~shaffer/Book/>

资料整理人：  
信息科学与工程学院 物联1402班 201408080203 宁静仪

信息科学与工程学院 物联1402班 201408080222 吴彦彦