C语言实现动态顺序表的实现代码

主要介绍了C语言实现动态顺序表的实现代码的相关资料,动态顺序表在内存中开辟一块空间，可以随我们数据数量的增多来扩容，需要的朋友可以参考下

C语言实现动态顺序表的实现代码

顺序表是在计算机内存中以数组的形式保存的线性表，是指用一组地址连续的存储单元依次存储数据元素的线性结构。线性表采用顺序存储的方式存储就称之为顺序表。顺序表是将表中的结点依次存放在计算机内存中一组地址连续的存储单元中。

静态实现：结构体内部只需两个成员，其中一个为固定大小(MAX)的数组，用来存放我们的数据。数组大小我们可以通过在头文件中改变MAX的值来改变。

动态实现：在内存中开辟一块空间，可以随我们数据数量的增多来扩容。

来看看动态的顺序表实现：

1.seqlist.h

```
#define _CRT_SECURE_NO_WARNINGS 1

#ifndef __SEQLIST_H__
#define __SEQLIST_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

typedef int DataType;

#define DEFAULT_SZ 3
#define INC_SZ 2

typedef struct SeqList
{
 DataType *data;
 int sz;
 int capacity;
}SeqList,*pSeqList;

void InitSeqList(pSeqList pList);
void PushBack(pSeqList pList,DataType d);
void PopBack(pSeqList pList);
void PushFront(pSeqList pList, DataType d);
void PopFront(pSeqList pList);
int Find(pSeqList pList, DataType d);
void Remove(pSeqList pList, DataType d);
void RemoveAll(pSeqList pList, DataType d);
void BubbleSort(pSeqList pList);
int BinarySearch(pSeqList pList, DataType d);
void PrintfSeqList(pSeqList pList);
void Insert(pSeqList pList, int pos, DataType d);
void ReverseList(pSeqList pList);
void DestroySeqList(pSeqList pList);




#endif//__SEQLIST_H__
```

2.seqlist.c

```
#define _CRT_SECURE_NO_WARNINGS 1

#include "seqlist.h"

void InitSeqList(pSeqList pList)
{
 pList->sz = 0;
 pList->data = (DataType*)malloc(sizeof(DataType)*DEFAULT_SZ);
 if (pList->data == NULL)
 {
  perror("malloc");
  return;
 }
 memset(pList->data, 0, sizeof(DataType)*DEFAULT_SZ);
```

```
}
void CheckCapacity(pSeqList pList)
{
 assert(pList);
 if (pList->sz == pList->capacity)
 {
  DataType*ret = (DataType*)realloc(pList->data, sizeof(DataType)*(pList->capacity + INC_SZ));
  if (ret == NULL)
  {
   perror("realloc");
  }
  pList->data = ret;
  pList->capacity += INC_SZ;

 }
}
void PushBack(pSeqList pList, DataType d)
{
 assert(pList);
 if (pList->sz == pList->capacity)
 {
  CheckCapacity(pList);
 }
 pList->data[pList->sz] = d;
 pList->sz++;
}
void PopBack(pSeqList pList)
{
 int i = 0;
 assert(pList);
 if (pList->sz == 0)
 {
  printf("顺序表为空:<");
  return;
 }
 pList->sz--;
}
void PushFront(pSeqList pList, DataType d)
{
 int i = 0;
 assert(pList);
 if (pList->sz == pList->capacity)
 {
  CheckCapacity(pList);
 }
 for (i = pList->sz; i >= 1; i--)
 {
  pList->data[i] = pList->data[i - 1];
 }
 pList->data[0] = d;
 pList->sz++;
}
void PopFront(pSeqList pList)
{
 int i = 0;
 assert(pList);
 for (i = 0; i < pList->sz; i++)
 {
  pList->data[i] = pList->data[i + 1];
 }
 pList->sz--;
}
int Find(pSeqList pList, DataType d)
{
 int i = 0;
 assert(pList);
 while (i < pList->sz)
 {
  if (pList->data[i] == d)
  {
   return i;
  }
  else
  {
   i++;
  }
 }
 return -1;
}
void Remove(pSeqList pList, DataType d)
{
```

```
 int i = 0;
 int pos = 0;
 assert(pList);
 while (pList->data[pos=Find(pList,d)]==d)
 {
  for (i = pos; i < pList->sz-1; i++)
  {
   pList->data[i] = pList->data[i + 1];
  }
  pList->sz--;
 }
}
void RemoveAll(pSeqList pList, DataType d)
{
 int i = 0;
 int pos = 0;
 assert(pList);
 while ((pos = Find(pList, d)) != -1)
 {
  for (i = pos; i < pList->sz - 1; i++)
  {
   pList->data[i] = pList->data[i + 1];
  }
  pList->sz--;
 }
}
void BubbleSort(pSeqList pList)
{
 int i = 0;
 assert(pList);
 for (i = 0; i < pList->sz - 1; i++)
 {
  int j = 0;
  for (j = 0; j < pList->sz - i - 1; j++)
  {
   if (pList->data[j]>pList->data[j + 1])
   {
    DataType tmp = pList->data[j];
    pList->data[j] = pList->data[j + 1];
    pList->data[j + 1] = tmp;
   }
  }
 }
}
int BinarySearch(pSeqList pList, DataType d)
{
 int left = 0;
 int right = pList->sz - 1;
 assert(pList);
 while (left <= right)
 {
  int mid = left - ((left - right) >> 1);
  if (d > pList->data[mid])
  {
   left = mid + 1;
  }
  else if (d < pList->data[mid])
  {
   right = mid - 1;
  }
  else
   return mid;
 }
 return -1;
}
void PrintfSeqList(pSeqList pList)
{
 int i = 0;
 for (i = 0; i < pList->sz; i++)
 {
  printf("%d ", pList->data[i]);
 }
}
void Insert(pSeqList pList, int pos, DataType d)
{
 int i = 0;
 if (pList->sz == pList->capacity)
 {
  CheckCapacity(pList);
 }
 for (i = pList->sz - 1; i >= pos; i--)
```

```
 {
  pList->data[i + 1] = pList->data[i];
 }
 pList->data[pos] = d;
 pList->sz++;
}
void ReverseList(pSeqList pList)
{
 int left = 0;
 int right = pList->sz - 1;
 assert(pList);
 while (left < right)
 {
  DataType tmp = pList->data[left];
  pList->data[left] = pList->data[right];
  pList->data[right] = tmp;
  left++;
  right--;
 }
}
void DestroySeqList(pSeqList pList)
{
 free(pList->data);
 pList->data = NULL;
}
```

3.test.c

```
#define _CRT_SECURE_NO_WARNINGS 1
#include "seqlist.h"

//void Test()
//{
// SeqList *List;
// InitSeqList(&List);
// PushBack(&List, 1);
// PushBack(&List, 2);
// PushBack(&List, 3);
// PushBack(&List, 4);
// PushBack(&List, 5);
// PopBack(&List);
// printf("%d ", Find(&List, 2));
// PrintfSeqList(&List);
//}

void Test2()
{
 SeqList List;
 InitSeqList(&List);
 PushBack(&List, 1);
 PushBack(&List, 2);
 PushBack(&List, 3);
 PushBack(&List, 4);
 PushBack(&List, 5);
 PushFront(&List, 5);
 PushFront(&List, 2);
 PushFront(&List, 3);
 //PopFront(&List);
 RemoveAll(&List, 5);
 //BubbleSort(&List);
 //BinarySearch(&List, 3);
 PrintfSeqList(&List);
}
int main()
{
 Test2();
 system("pause\n");
 return 0;
}
```

静态顺序表的实现：//www.jb51.net/article/120875.htm

以上就是动态实现顺序表的实例，如有疑问请留言或者到本站社区交流讨论，感谢阅读，希望能帮助到大家，谢谢大家对本站的支持！