动态顺序表的实现（第一次）

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

typedef int SLDataType;

typedef struct SeqList

{

SLDataType\* a;

int size;

int capacity;

}SeqList;

void SeqListInit(SeqList\* psl)

{

psl->a=NULL;

psl->size=psl->capacity=0;

}

void SeqListDestory(SeqList\* psl)

{

free(psl->a);//别忘了free

psl->a=NULL;

psl->size=psl->capacity=0;

}

void SeqListPrint(SeqList\* psl)

{

int i;

if(psl->size==0)

{

printf("NULL");

}

else

{

for(i=0;i<psl->size;i++)

{

printf("%d ",psl->a[i]);

}

printf("\n");

}

}

void CheckCapacity(SeqList\*\* ppsl)//这个函数错误很多很多

{

(\*ppsl)->capacity=(\*ppsl)->capacity==0?4:(\*ppsl)->capacity\*2;

//只是一个表达式不行，要赋值capacity才会变；

//光改capacity不行，要真正扩容必须开辟新的空间

SLDataType\* tmp=(SLDataType\*)realloc((\*ppsl)->a,(\*ppsl)->capacity\*sizeof(SLDataType));

//不管怎样，还是加一个验证扩容是否失败好一点

if(tmp==NULL)

{

printf("realloc fail\n");

exit(-1);

}

(\*ppsl)->a=tmp;//指向这个开辟的内存

}

void SeqListPushBack(SeqList\* psl,SLDataType x)

{

if(psl->size==psl->capacity)

{

CheckCapacity(&psl);

}

psl->a[psl->size]=x;//注意size是结构体里面的，所以size要有箭头

psl->size++;

}

void SeqListPopBack(SeqList\* psl)//这个函数基本没问题

{

assert(psl->size>0);

psl->size--;

}

void SeqListPushFront(SeqList\* psl,SLDataType x)

{

if(psl->capacity==psl->size)

CheckCapacity(&psl);

int end=psl->size;

for(end=psl->size;end>0;end--)//用while也行

{

psl->a[end]=psl->a[end-1];

}

psl->a[end]=x;

psl->size++;//长度要增加不能忘；

}

void SeqListPopFront(SeqList\* psl)

{

assert(psl->size>0);

SLDataType begin;

for(begin=0;begin<psl->size-1;begin++)

{

psl->a[begin]=psl->a[begin+1];

}

psl->size--;

}

int SeqListFind(SeqList\* psl,SLDataType x)

{

int pos;

for(pos=0;pos<psl->size;pos++)//不能忘了没找到的情况，没找到返回-1

{

if(psl->a[pos]==x)

return pos;

}

return -1;

}

void SeqListInsert(SeqList\* psl,int pos,SLDataType x)

{

if(psl->size==psl->capacity)

CheckCapacity(&psl);

int end=psl->size;

for(end=psl->size;end>pos;end--)

psl->a[end]=psl->a[end-1];

psl->a[pos]=x;

psl->size++;//插入之后不能忘了size++

}

void SeqListErase(SeqList\* psl,int pos)

{

assert(psl->size>pos&&pos>=0);//光pos小于size还不够，pos还不能小于0

for( ;pos<psl->size-1;pos++)

psl->a[pos]=psl->a[pos+1];

psl->size--;//总是不记得size

}

void TestSeqList1()//SeqListPushBack,SeqListPopBack,SeqListInit,SeqCheckCapacity,SeqListPrint测试无误；

{

SeqList sl;

SeqListInit(&sl);

SeqListPushBack(&sl,1);

SeqListPushBack(&sl,2);

SeqListPushBack(&sl,3);

SeqListPushBack(&sl,4);

SeqListPushBack(&sl,5);

SeqListPushBack(&sl,6);

SeqListPrint(&sl);

SeqListPopBack(&sl);

SeqListPopBack(&sl);

SeqListPrint(&sl);

SeqListPopBack(&sl);

SeqListPopBack(&sl);

SeqListPrint(&sl);

}

void TestSeqList2()//SeqListPushFront,SeqListPopFront测试无误；

{

SeqList sl;

SeqListInit(&sl);

SeqListPushFront(&sl,1);

SeqListPushFront(&sl,2);

SeqListPushFront(&sl,3);

SeqListPushFront(&sl,4);

SeqListPushFront(&sl,5);

SeqListPushFront(&sl,6);

SeqListPrint(&sl);

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPrint(&sl);

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPrint(&sl);

}

void TestSeqList3()//SeqListFind测试无误

{

SeqList sl;

int pos1,pos2;

SeqListInit(&sl);

SeqListPushFront(&sl,1);

SeqListPushFront(&sl,2);

SeqListPushFront(&sl,3);

SeqListPushFront(&sl,4);

SeqListPushFront(&sl,5);

SeqListPushFront(&sl,6);

SeqListPrint(&sl);

pos1=SeqListFind(&sl,3);

pos2=SeqListFind(&sl,2);

printf("%d %d %d\n",pos1,pos2,SeqListFind(&sl,7));

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPrint(&sl);

SeqListPopFront(&sl);

SeqListPopFront(&sl);

SeqListPrint(&sl);

}

void TestSeqList4()//SeqListInsert,SeqListErase,SeqListDestory测试无误；

{

SeqList sl;

SeqListInit(&sl);

SeqListPushFront(&sl,1);

SeqListPushFront(&sl,2);

SeqListPushFront(&sl,3);

SeqListPushFront(&sl,4);

SeqListPushFront(&sl,5);

SeqListPushFront(&sl,6);

SeqListInsert(&sl,3,40);

SeqListInsert(&sl,2,50);

SeqListInsert(&sl,2,50);

SeqListPrint(&sl);

SeqListErase(&sl,5);

SeqListPrint(&sl);

SeqListErase(&sl,5);

SeqListErase(&sl,5);

SeqListErase(&sl,5);

// SeqListErase(&sl,5);

SeqListPrint(&sl);

SeqListDestory(&sl);

}

int main()

{

// TestSeqList1();

// TestSeqList2();

TestSeqList3();

// TestSeqList4();

}

动态顺序表的实现（第二次）

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

typedef int SLDatatype;

typedef struct SeqList

{

SLDatatype\* a;

int size;

int capacity;

}SeqList;

void SeqListInit(SeqList\* psl)

{

psl->a=NULL;

psl->size=psl->capacity=0;

}

void SeqListDestroy(SeqList\* psl)//这里开始没写对

{

free(psl->a);

psl->a=NULL;

psl->capacity=psl->size=0;

}

void CheckCapacity(SeqList\*\* ppsl)

{

int newcapacity;

newcapacity=(\*ppsl)->capacity==0?4:(((\*ppsl)->capacity)\*2);

(\*ppsl)->a=(SLDatatype\*)realloc((\*ppsl)->a,newcapacity\*sizeof(SLDatatype));

(\*ppsl)->capacity=newcapacity;

}

void SeqListPrint(SeqList sl)

{

int i=0;

if(sl.size==0)

{

printf("NULL\n");

}

else

{

while(i<sl.size)

{

printf("%d ",sl.a[i]);

i++;

}

printf("\n");

}

}

void SeqListPushBack(SeqList\* psl,SLDatatype x)

{

if(psl->size==psl->capacity)

{

CheckCapacity(&psl);

}

psl->a[psl->size]=x;

psl->size++;

}

void SeqListPopBack(SeqList\* psl)

{

assert(psl->size!=0);

psl->size--;

}

void SeqListPushHead(SeqList\* psl,SLDatatype x)

{

if(psl->capacity==psl->size)

{

CheckCapacity(&psl);

}

int i=psl->size;

while(i)

{

psl->a[i]=psl->a[i-1];

i--;

}

psl->a[0]=x;

psl->size++;

}

void SeqListPopHead(SeqList\* psl)

{

assert(psl->size>0);

int i=0;

while(i<psl->size-1)

{

psl->a[i]=psl->a[i+1];

i++;

}

psl->size--;

}

int SeqListFind(SeqList sl,SLDatatype x)

{

assert(sl.size>0);

int i=sl.size;

for(i=0;i<sl.size;i++)

{

if(sl.a[i]==x)

return i;

}

return -1;

}

void SeqListInsert(SeqList\* psl,int pos,SLDatatype x)

{

assert(pos<psl->size&&pos>=0);

if(psl->size==psl->capacity)

CheckCapacity(&psl);

int i=psl->size;

while(i>pos)

{

psl->a[i]=psl->a[i-1];

i--;

}

psl->a[pos]=x;

psl->size++;

}

void SeqListErase(SeqList\* psl,int pos)

{

assert(pos<psl->size&&pos>=0);

assert(psl->size>0);

int i=pos;

for(i=pos;i<psl->size-1;i++)

{

psl->a[i]=psl->a[i+1];

}

psl->size--;

}

void SeqListTest1()

{

SeqList sl;

SeqListInit(&sl);

SeqListPushBack(&sl,1);

SeqListPushBack(&sl,2);

SeqListPrint(sl);

SeqListPushBack(&sl,3);

SeqListPushBack(&sl,4);

SeqListPrint(sl);

SeqListPopBack(&sl);

SeqListPopBack(&sl);

SeqListPrint(sl);

SeqListPopBack(&sl);

SeqListPopBack(&sl);

SeqListPrint(sl);

SeqListDestroy(&sl);

}

void SeqListTest2()

{

SeqList sl;

SeqListInit(&sl);

SeqListPushHead(&sl,1);

SeqListPushHead(&sl,2);

SeqListPrint(sl);

SeqListPushHead(&sl,3);

SeqListPushHead(&sl,4);

SeqListPrint(sl);

SeqListPopHead(&sl);

SeqListPopHead(&sl);

SeqListPrint(sl);

SeqListPopHead(&sl);

SeqListPopHead(&sl);

SeqListPrint(sl);

}

void SeqListTest3()

{

SeqList sl;

SeqListInit(&sl);

SeqListPushHead(&sl,1);

SeqListPushHead(&sl,2);

SeqListPrint(sl);

SeqListPushHead(&sl,3);

SeqListPushHead(&sl,4);

SeqListInsert(&sl,2,30);

SeqListInsert(&sl,2,40);

int pos1=SeqListFind(sl,3);

int pos2=SeqListFind(sl,30);

printf("pos1:%d\tpos2:%d\n",pos1,pos2);

SeqListPrint(sl);

SeqListErase(&sl,3);

SeqListErase(&sl,4);

SeqListPrint(sl);

SeqListDestroy(&sl);

}

int main()

{

// SeqListTest1();//少了括号也是没谁了

// SeqListTest2();//居然少了初始化函数

SeqListTest3();

}

总结：第一次是几天前敲的代码，第二次是今天敲的代码，敲代码的过程中出现的每个问题都有在后面批注，两次实现动态顺序表第二次明显比第一次熟练许多，很多第一次没有注意到的问题都注意到了，但是自己还是经常犯各种各样的小错误，归根到底还是敲得代码太少了，多练、走出舒适圈是硬道理。