顺序表

const MAXSIZE = 20

type ElemType int

//SqList 顺序表（SequenceList）

type SqList struct {

data [MAXSIZE]ElemType

length int

}

//InitList 初始化顺序表

func InitList() \*SqList {

if MAXSIZE == 0 {

return nil

} else {

return &SqList{data: [MAXSIZE]ElemType{}, length: 0}

}

}

//CreateList 顺序表赋值

func (l \*SqList) CreateList(data []ElemType) {

for i := 0; i < len(data); i++ {

l.data[i] = data[i]

l.length++

}

}

//RangeList 遍历顺序表

func (l \*SqList) RangeList() {

for i := 0; i < l.length; i++ {

fmt.Print(l.data[i], " ")

}

fmt.Println()

}

//ListEmpty 判断顺序表是否为空

func (l \*SqList) ListEmpty() bool {

if l.length != 0 {

return false

} else {

return true

}

}

//ClearList 清空顺序表

func (l \*SqList) ClearList() {

l.data = [MAXSIZE]ElemType{}

l.length = 0

}

//ListInsert 指定位置插入元素

func (l \*SqList) ListInsert(i int, e ElemType) {

if l.length == MAXSIZE { //表满

fmt.Println("顺序表已满，插入数据失败")

return

}

if i > l.length+1 || i < 1 { //i不在表内范围

fmt.Println("请选择合适的数据插入位置，插入数据失败")

return

}

if i <= l.length { //插入位置不在表尾,将数据向后移

for j := l.length - 1; j >= i-1; j-- {

l.data[j+1] = l.data[j]

}

}

l.data[i-1] = e //插入元素

l.length++

}

//ListDelete 指定位置删除元素

func (l \*SqList) ListDelete(i int) {

if l.length == 0 { //表空

fmt.Println("表空，删除元素失败")

return

}

if i > l.length+1 || i < 1 { //i不在表内范围

fmt.Println("删除位置不在表范围内，删除元素失败")

return

}

if i <= l.length { //删除位置不在表尾，将数据前移

for j := i; j < l.length; j++ {

l.data[j-1] = l.data[j]

}

}

l.length--

}

//GetElem 获取位置元素

func (l \*SqList) GetElem(i int) ElemType {

return l.data[i-1]

}

//LocateElem 查找元素下标

func (l \*SqList) LocateElem(e ElemType) int {

for i, v := range l.data {

if v == e {

return i

}

}

return 0

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.InitList()

2.CreateList()

3.RangeList()

4.ListEmpty()

5.ClearList()

6.ListInsert()

7.ListDelete()

8.GetElem()

9.LocateElem()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*