

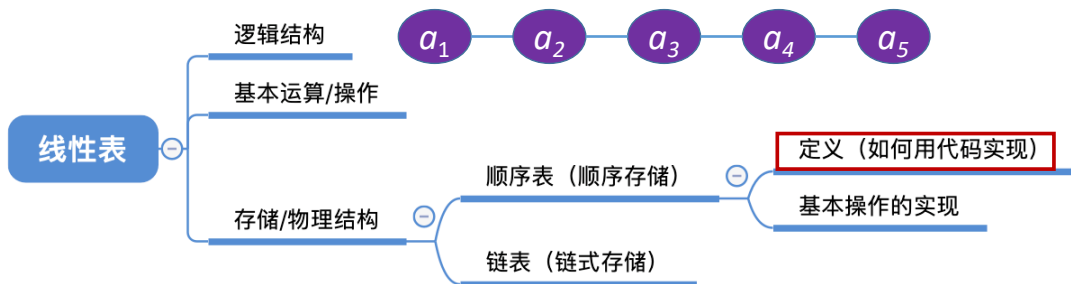
本节内容

顺序表 定义

王道考研/CSKAOYAN.COM

1

知识总览



王道考研/CSKAOYAN.COM

2

顺序表的定义

线性表 L 逻辑结构

每个数据元素所占空间一样大

线性表是具有相同数据类型的 n ($n \geq 0$) 个数据元素的有限序列

顺序表——用顺序存储的方式实现线性表顺序存储。把逻辑上相邻的元素存储在物理位置上也相邻的存储单元中，元素之间的关系由存储单元的邻接关系来体现。

```
typedef struct {
    int num; //号数
    int people; //人数
} Customer;
```

内存

设线性表第一个元素的存放位置是 LOC(L)
LOC 是 location 的缩写

LOC(L)

LOC(L)+数据元素的大小

LOC(L)+2*数据元素的大小

...

ElemType 就是你的顺序表中存放的数据元素类型

如何知道一个数据元素大小?
C语言 `sizeof(ElemType)`
Eg:
sizeof(int) = 4B
sizeof(Customer) = 8B

王道考研/CSKAOYAN.COM

3

顺序表的实现——静态分配

```
#define MaxSize 10
typedef struct{
    ElemType data[MaxSize];
    int length;
}SqList;
```

Sq: sequence —— 顺序, 序列

//定义最大长度

//用静态的“数组”存放数据元素

//顺序表的当前长度

//顺序表的类型定义（静态分配方式）

给各个数据元素分配连续的存储空间，大小为 $\text{MaxSize} * \text{sizeof}(\text{ElemType})$

内存

王道考研/CSKAOYAN.COM

4

```

#include <stdio.h>
#define MaxSize 10 //定义最大长度
typedef struct{
    int data[MaxSize]; //用静态的“数组”存放数据元素
    int length; //顺序表的当前长度
}SqList; //顺序表的类型定义

//基本操作——初始化一个顺序表
void InitList(SqList &L){
    ② for(int i=0; i<MaxSize; i++){
        L.data[i]=0; //将所有数据元素设置为默认初始值
    }
    ③ L.length=0; //顺序表初始长度为0
}

int main() {
    ① SqList L; //声明一个顺序表
    InitList(L); //初始化顺序表
    //....未完待续, 后续操作
    return 0;
}

```

内存

王道考研/CSKAOYAN.COM

本例中数据元素的类型 (ElemType) 是 int

5

```

/*不初始化数据元素, 内存不刷0*/
#include <stdio.h>
#define MaxSize 10 //定义最大长度
typedef struct{
    int data[MaxSize]; //用静态的“数组”存放数据元素
    int length; //顺序表的当前长度
}SqList; //顺序表的类型定义

//基本操作——初始化一个顺序表
void InitList(SqList &L){
    ③ L.length=0; //顺序表初始长度为0
}

int main() {
    ① SqList L; //声明一个顺序表
    InitList(L); //初始化顺序表
    //尝试“违规”打印整个 data 数组
    for(int i=0; i<MaxSize; i++){
        printf("data[%d]=%d\n", i, L.data[i]);
    }
    return 0;
}

```

内存

王道考研/CSKAOYAN.COM

没有设置数据元素的默认值

内存中会有遗留的“脏数据”

这种访问方式也不够好，更好的做法是使用基本操作来访问各个数据元素

思考：这一步是否可省略？

6

顺序表的实现——静态分配

```
#define MaxSize 10           //定义最大长度
typedef struct{
    ElemType data[MaxSize];  //用静态的“数组”存放数据元素
    int length;              //顺序表的当前长度
}SqList;                    //顺序表的类型定义
```



给各个数据元素分配连续的存储空间，大小为
MaxSize*sizeof(ElemType)



Q: 如果“数组”存满了怎么办?

A: 可以放弃治疗，顺序表的表长刚开始确定后就无法更改（存储空间是静态的）

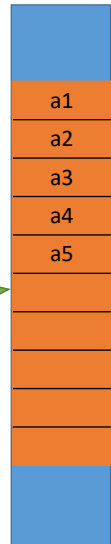
思考：如果刚开始就声明一个很大的内存空间呢？存在什么问题？



同学，浪费是不行的

王道考研/CSKAOYAN.COM

内存



7

顺序表的实现——动态分配

```
#define InitSize 10          //顺序表的初始长度
typedef struct{
    ElemType *data;           //指示动态分配数组的指针
    int MaxSize;              //顺序表的最大容量
    int length;               //顺序表的当前长度
} SeqList;                  //顺序表的类型定义（动态分配方式）
```

Key: 动态申请和释放内存空间

C —— malloc、free 函数

L.data = (ElemType *) malloc(sizeof(ElemType) * InitSize);

C++ —— new、delete 关键字

malloc 函数返回一个指针，
需要强制转型为你定义的
数据元素类型指针

malloc 函数的参数，指明要
分配多大的连续内存空间

内存



王道考研/CSKAOYAN.COM

8

malloc、free
函数的头文件

#include <stdlib.h>

#define InitSize 10 //默认的最大长度

typedef struct{
int *data; //指示动态分配数组的指针
int MaxSize; //顺序表的最大容量
int length; //顺序表的当前长度
}SeqList;

int main() {
SeqList L; //声明一个顺序表
InitList(L); //初始化顺序表
//...往顺序表中随便插入几个元素...
IncreaseSize(L, 5);
return 0;
}

注：realloc 函数也可实现，但建议初学者使用 malloc 和 free 更能理解背后过程

顺序表的实现——动态分配

void InitList(SeqList &L){
//用 malloc 函数申请一片连续的存储空间
L.data=(int *)malloc(InitSize*sizeof(int));
L.length=0;
L.MaxSize=InitSize;
}

//增加动态数组的长度
void IncreaseSize(SeqList &L, int len){
int *p=L.data;
L.data=(int *)malloc((L.MaxSize+len)*sizeof(int));
for(int i=0; i<L.length; i++){
L.data[i]=p[i];
}
L.MaxSize=L.MaxSize+len;
free(p);
}

时间开销大

内存

MaxSize
length
*data
data[0]
data[1]
.....
data[9]
*p
data[0]
data[1]
.....
data[9]
.....
data[14]

王道考研/CSKAOYAN.COM

9

顺序表的实现

顺序表的特点：
①随机访问，即可以在 $O(1)$ 时间内找到第 i 个元素。
②存储密度高，每个节点只存储数据元素
③拓展容量不方便（即便采用动态分配的方式实现，拓展长度的时间复杂度也比较高）
④插入、删除操作不方便，需要移动大量元素

代码实现：data[i-1];
静态分配、动态分配都一样

内存

LOC (L)
LOC (L)+sizeof(ElemType)
LOC (L)+2*sizeof(ElemType)
.....
a₁
a₂
a₃
a₄
a₅

顺序存储

链式存储

存放指针

王道考研/CSKAOYAN.COM

10

王道考 研,cskaoyan.com

5

