大家好，我是蓝蓝。

我的B站：蓝蓝希望你上岸呀

我的公主号：蓝蓝考研

我的Github：408bester

今天的这一期视频是关于各大数据结构的定义。这也是我们得分挺关键的一步，也是最基础的一步，希望大家可以通过这份文档，掌握各个数据结构的定义。

1 线性表的结构体定义

1.1 顺序表结构体定义

这里包含静态和动态两种方式，大家要注意哈，通常的情况下，静态的时候用数组的方式，动态都会使用到开辟释放空间。

对于内存空间的申请，一定要写会，尽量去理解，实在理解不了，给我**每天写一遍，写个一周。**

> 静态顺序表

```
#define Max 50 //定义线性表最大长度
//顺序表定义
typedef struct
{
   int data[maxSize];
   int length;//实际数据的个数 是用来判断是否达到最大空间
}Sqlist;
//考试的时候可以用下面的这种定义 直接一个数组
int A[maxSize];
int n;
```

> 动态顺序表

```
typedef struct {
    ElmeType* data;
    int size;//用来记录表中数据的实际个数方便比较
    int capacity;//设置用来记录L的最大空间 若是达到最大空间则需要扩容
}SeqList;
```

关于申请空间，一定一定要会哈。

```
L.data=(ElemType*)malloc(sizeof(Elemtype)*InitSize)
```

1.2 链表结构体定义

每一个结点不仅要放数据域，还要有一个指针域。

> 单链表

```
typedef struct LNode{
    ElemType data;
    struct LNode *next;
}LNode,*LinkList;//这两个是等价的
```

> 双链表

```
typedef struct DNode{
    ElemType data;
    struct LNode *prior, *next;
}DNode,*DLinkList;//这两个是等价的
```

2 栈和队列的结构体定义

2.1 栈结构体定义

> 顺序栈

```
//顺序栈定义
typedef struct
{
    int data[maxSize];
    int top;//栈顶指针
}SqStack;
//假设元素为int型，可以直接使用下面方法进行定义并初始化
int stack[maxStack];
int top=-1;
```

> 动态栈

```
typedef struct{
    ElemType* data;
    int top;
    //int size;可以不需要size  因为top 就是指向最后的位置
    int capacity;//这个时候是需要容量的 方便扩容
}SqStack;
```

> 链栈

```
// 链栈的存储结构
typedef struct StackNode
{
    int data;
    struct StackNode *next;
}StackNode,*LinkStack;
```

2.2 队列结构体定义

> 顺序队列[公众号：蓝蓝考研]

```
//顺序队列定义
typedef struct
{
    int data[maxSize];
    int front;
    int rear;
}SqQueue;
```

> 链队

```
//链队定义
//1.队节点定义
typedef struct QNode
{
    int data;//数据域
    struct QNode *next;//指针域
}QNode;
//2.类型定义
typedef struct
{
    QNode *front;//队头指针
    QNode *rear;//队尾指针
}LiQueue;
```

3 串结构体定义

3.1 定长顺序存储结构体定义

串的定长顺序存储好像没有什么结构上的要求要求 直接使用静态线性表的结构体定义即可

```
#define MAXLEN 255 //预定义最大串长为255

typedef struct{
    char ch[MAXLEN];//每个分量存储一个字符
    int length;//串的实际长度
}SString;
```

3.2 堆分配结构体定义

```
typedef struct{
    char *ch;//按照串长分配存储区 ch指向串的基地址
    int length;//串的长度
}HString;
```

3.3 块存储结构体定义

```
//块链存储
typedef struct Chunk{
    char ch[3];//这里我定义的是放三个值
    struct Chunk *next;
}Chunk;
```

//定义头尾指针的作用是 方便进行连接操作
//连接的时候别忘了处理第一个串尾的无效字符

```
typedef struct{
    Chunk *head,*tail;//串的头尾指针
    int curlen; //串的当前长度 (链表的节点数)
}LString;
```

4 树和二叉树

正如我们所熟知的 存储方式一般有两种，顺序存储或者链式存储，若是使用顺序存储，则一般二叉树为了能反映二叉树中结点之间的逻辑关系，只能添加并不存在的空结点构造树像完全二叉树一样， 每一个结点与完全二叉树上的结点对应，再存储到一维数组的相应分量中。这样有可能造成空间的极大浪费，所以这里我们使用链式存储 ，为了方便各种操作，链式存储中也分为了几种方式，这里就不多赘述了，用到一个写一个

4.1 链式存储结构体定义

```
//二叉树的存储结构，一个数据域，2个指针域
 typedef struct BiTNode
{
     char data;
     struct BiTNode *lchild,*rchild;
  }BiTNode,*BiTree;
```

4.2 线索二叉树结构体定义

```
typedef struct BiThrNode{
    struct BiThrNode* Lchild;
    int Ltag;
    Elemtype data;
    int Rtag;
    struct BiThrNode* Rchild;
}BiThrNode;
```

4.3 哈夫曼树结构体定义

```
typedef struct HNode
{
    char data; //数据,非叶节点为NULL
    double weight;//权重
    int parent;//双亲，-1表示没有双亲，即根节点
    int lchild;//左孩子，数组下标,-1表示无左孩子，即叶节点
    int rchild;//右孩子
}Hnode;
```

4.4 树之双亲表示法结构体定义

```
typedef struct Snode{
    char data;
    int parent;
} PTNode;

typedef struct{
    PTNode tnode[MAX_SIZE];  // 存放树中所有结点
    int n;  // 结点数
}
```

4.5 孩子兄弟表示法结构体定义

```
#define tree_size 100//宏定义树中结点的最大数量
#define TElemType int//宏定义树结构中数据类型
typedef struct PTNode{
    TElemType data;//树中结点的数据类型
    int parent;//结点的父结点在数组中的位置下标
}PTNode;
typedef struct {
    PTNode nodes[tree_size];//存放树中所有结点
    int r,n;//根的位置下标和结点数
}PTree;
```

5 图

5.1 邻接矩阵法结构体定义

```
//图的邻接矩阵定义
typedef struct
{
    int no;//顶点编号
    char info;
}VertexType;
typedef struct
{
    int edges[maxSize][maxSize];
    int n,e;//顶点个数和边个数
    VertexType vex[maxSize];//存放节点信息
}MGraph;
//上面的定义如果没记住，也要记住里面的元素，如n，e等含义
```

5.2 邻接表法结构体定义

```

//结点的定义
typedef char VertexType;
typedef int EdgeType;
#define MaxVex 100
typedef struct EdgeNode //边表结点
{
    int adjvex; //邻接点域，存储邻接顶点对应的下标
    EdgeType weight; //用于存储权值，对于非网图可以不需要
    struct EdgeNode *next; //链域，指向下一个邻接点
}EdgeNode;
typedef struct VertexNode  //顶点表结点
{
    VertexType data;  //顶点域，存储顶点信息
    EdgeNode *firstedge; //边表头指针
}VertexNode,AdjList[MaxVex];

typedef struct
{
    AdjList adjList;
    int numVertexes,numEdges;
}//图中当前顶点数和边数
```

小结

恭喜大家看到了这里，希望大家务必能掌握上面的内容，如果有错误请及时联系我，互帮互助，一起成长，一战成硕，尽全力的去享受这个过程。

如果上述有任何问题，一定记得联系我哈，我的wx(lanlankaoyanshan)在下面。