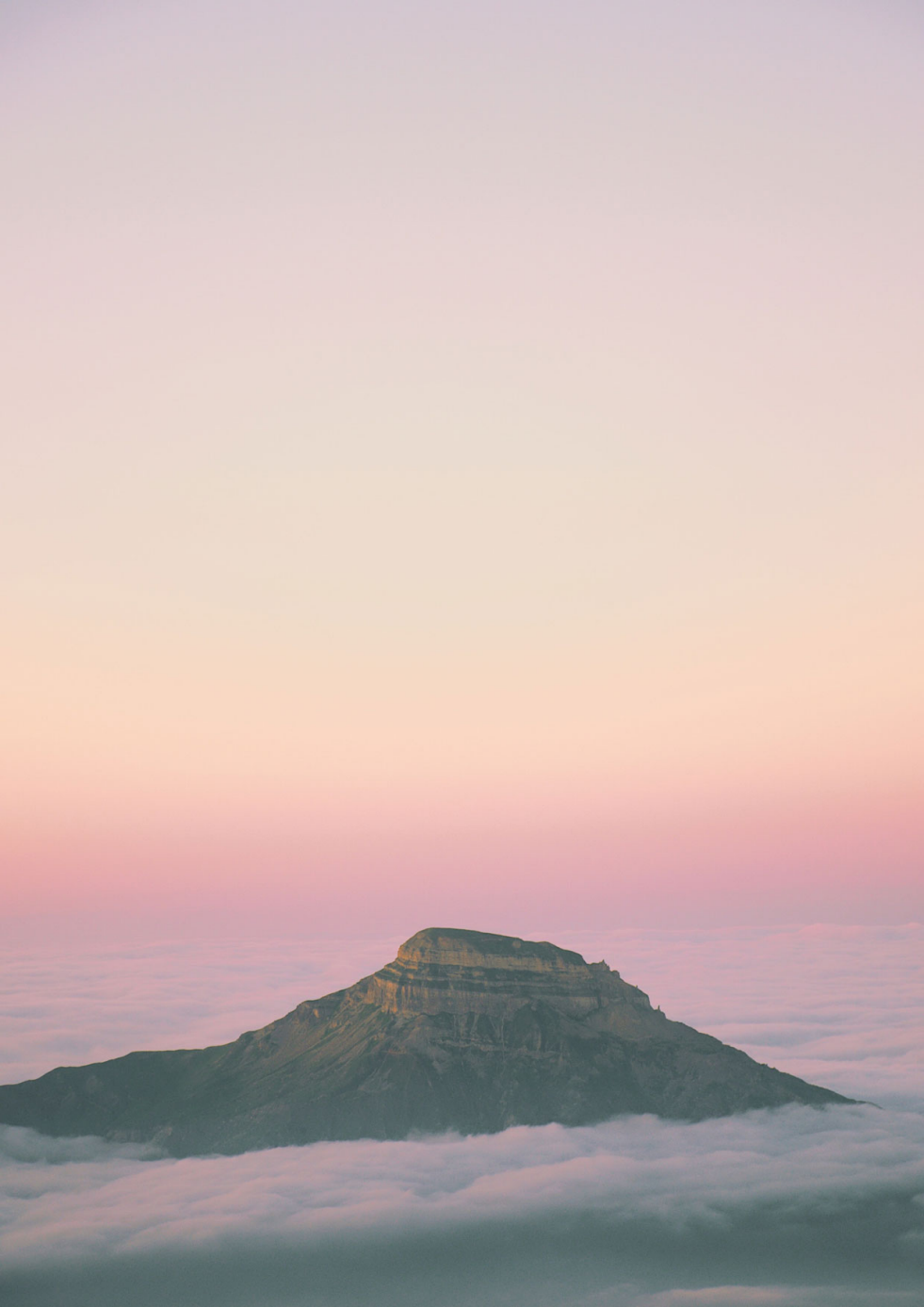

- 单链表的存储结构

```
typedef struct LNode{
    ElemType          data;
    struct  LNode     *next;
}LNode, *LinkList;
```

```
//---线性表的双链式存储结构----
typedef  struct  DuLNode{
        ElemType  data;
        struct DuLNode *next;
        struct DuLNode *prior;
        } DuLNode,*DuLinkList;
```

一、顺序存储结构

```
#define LIST_INIT_SIZE   100  //线性表存储空间的初始分配量
#define LISTINCREMENT   10  //线性表存储空间的分配增量
typedef  struct{
    Elemtype   *elem;     //存储空间基址
    int            length;    //当前长度
    int            listsize;  //当前分配存储容量
}SqList;
```

一、 顺序栈：

```
# define STACK_INIT_SIZE    100;
# define STACKINCREMENT    10;
typedef  struct {
SElemType   *base; //在栈构造之前和销毁之后,base=NULL
SElemType   *top;   //栈顶指针
int  stacksize;
 } SqStack;
```

```
#define MAXQSIZE 100  // 最大队列长度
typedef struct {
        QElemType *base; // 初始化的动态分配存储空间
        int rear;          // 队尾指针，指向队尾元素的下一个位置
        int front;         // 队头指针，指向队头元素的位置
} SqQueue;
```

- 二叉链表的存储表示

```
typedef  struct  BiTNode{
    TElemType  data;
    struct  BiTNode  *lchild, *rchild;
}BiTNode ,*BiTree;
```

❖三叉链表的存储表示

```
typedef  struct  TriTNode{
    TElemType  data;
    struct  TriTNode  *lchild, *rchild, *parent;
} TriTNode ,*TriTree;
```

❖三叉链表的存储表示

```
typedef  struct  TriTNode{
    TElemType  data;
    struct  TriTNode  *lchild, *rchild, *parent;
} TriTNode ,*TriTree;
```

```
    TElemType  data;
    struct  TriTNode  *lchild, *rchild, *parent;
} TriTNode ,*TriTree;
```

二叉树的存储结构

顺序存储结构

- 用一组地址连续的存储单元存储二叉树中的数据元素。
  - 完全二叉树，只要从根起按层序存储即可。将完全二叉树上编号为 i 的结点元素存储在一维数组中下标为 i-1 的分量中。
  - 一般的二叉树，可对照完全二叉树的编号进行相应的存储，但在没有结点的分量中需填充空白字符(例如填充0)。
- 顺序存储表示

```
#define  MAX_TREE_SIZE  100            //最大结点数
Typedef   TElemType SqBiTree[MAX_TREE_SIZE]; // 0号根结点
SqBiTree  bt;
```

双亲表示法

- 以一组连续的存储空间存放树的结点，每个结点中附设一个指针指示其双亲结点在这连续的存储空间中的位置。
- 形式说明

```
#define MAX_TREE_SIZE = 100;
typedef struct PTNode{    // 结点结构
        TElemType data;
        int parent;          // 双亲位置域
} PTNode;
typedef struct {             // 树结构
        PTNode nodes[MAX_TREE_SIZE];
        int r, n;        // 根的位置和结点数
} PTree;
```

孩子表示法

- 把每个结点的孩子排列起来，看成一个线性表，以单链表存储；令其头指针和结点的数据元素构成一个结点，并将所有这样的结点存放在一个地址连续的存储空间里。
- 形式说明

```
typedef struct CTNode {          // 孩子结点
        int child;
        struct CTNode *next;
} *ChildPtr;
typedef struct {
        ElemType data;                  // 结点的数据元素
        ChildPtr firstchild;            // 孩子链表头指针
} CTBox;
typedef struct {
        CTBox nodes[MAX_TREE_SIZE];
        int n, r;                       // 结点数和根的位置
} CTree;
```

孩子兄弟表示法

用二叉链表作树的存储结构，链表中每个结点的两个指针域分别指向其第一个孩子结点和下一个兄弟结点。

形式说明

```
typedef struct CSNode{
        ElemType data;
        struct CSNode *firstchild, *nextsibling;
} CSNode, *CSTree;
```

一图的数组(邻接矩阵)存储表示

```
#define  INFINITY   INT_MAX;          // 最大值∞
#define  MAX_VERTEX_NUM 20;          // 最大顶点个数
typedef enum {DG,DN,UDG,UDN} GraphKind;
                  // {有向图,有向网,无向图,无向网}
typedef struct ArcCell {
  VRType adj;    // VRType是顶点关系类型。
                    对无权图，用1或0
                  // 表示相邻否；对带权图，则为权值类型。
  InfoType  *info; // 该弧相关信息的指针
  } ArcCell, AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
```



```
typedef struct {
   VertexType vexs[MAX_VERTEX_NUM];      // 顶点信息
   AdjMatrix arcs;                                  // 邻接矩阵
   int vexnum, arcnum;          // 图的当前顶点数和弧(边)数
   GraphKind          kind;                    // 图的种类标
   志
   } MGraph;
```

typedef struct {
VertexType vexs[MAX_VERTEX_NUM]; // 顶点信息
AdjMatrix arcs; // 邻接矩阵
int vexnum, arcnum; // 图的当前顶点数和弧(边)数
GraphKind kind; // 图的种类标
志
} MGraph;

```
#define MAX_VERTEX_NUM  20
typedef struct ArcNode {
   int          adjvex;       // 该弧所指向的顶点的位置
   struct ArcNode *nextarc;
                              // 指向下一条弧的指针
   InfoType  *info;           // 该弧相关信息的指针
   } ArcNode;
```



```
typedef struct VNode {
   VertexType      data;  // 顶点信息
   ArcNode *firstarc;     // 指向第一条依附该顶点的弧
   } AdjList[MAX_VERTEX_NUM];
typedef struct {
   AdjList vertices;          // 顶点数组
   int vexnum, arcnum; // 图的当前顶点数和弧数
   int kind;                  // 图的种类标志
   } ALGraph;
```

日期: /

静态表

```
typedef int KeyType;

typedef struct{
   KeyType key;
   ......
}SElemType;
```

顺序表的类型描述：

```
typedef struct{
   ElemType *elem; // 数据元素存储空间基址，建表
                   // 时按实际长度分配，0号单元留空
   int length;     // 表中元素个数
   } SSTable;
#define LQ(a,b) ((a)=<(b))
```

顺序表的类型描述：

```
typedef struct{
   ElemType *elem; // 数据元素存储空间基址，建表
                   // 时按实际长度分配，0号单元留空
   int length;     // 表中元素个数
   } SSTable;
```

日期: /

哈希表

```
#define NULLKEY  -1   /*  根据关键字类型定义空标识  */
typedef  struct
  {  KeyType   key ;          /*  关键字域  */
     otherType  otherinfo ;  /*  记录的其它域  */
  }RecType ;
```

待排记录的数据类型定义如下：

```
typedef  int  KeyType;  // 关键字类型为整数类型
#define MAXSIZE  1000 // 待排顺序表最大长度
 typedef  struct {
    KeyType   key;           // 关键字项
    InfoType  otherinfo;  // 其它数据项
 } RcdType;                  // 记录类型
 typedef  struct {
    RcdType   r[MAXSIZE+1]; // r[0]闲置
    int           length;          // 顺序表长度
 } SqList;                      // 顺序表类型
```