数组

数组的基本概念

1. **数组**：数组是由类型相同的元素构成的有序集合，每个元素都可以看作下标和值的偶对。
2. **n维数组**：n维数组是受n组线性关系约束的线性表，即n维数组的每个元素受$n(n \geq 1)$个线性关系的约束，每个元素在n个线性关系中的序号$(i_1, i_2, \ldots, i_n)$称为该元素的下标，可以通过一组下标唯一确定一个元素。n维数组可以理解为每个元素是n-1维数组的定长线性表。一维数组可以看成一个线性表，二维数组可以看作元素是线性表的线性表。例如，一个m行n列的矩阵，当其用二维数组进行表示的时候，可以看作由m个行向量组成的线性表，即
   $\mathbf{A} = (\mathbf{A}_0, \mathbf{A}_1 \ldots, \mathbf{A}_{m-1})$，其中每个元素$\mathbf{A}_i$是一个行向量：$\mathbf{A}_i = (a_{i0}, a_{i1}, \ldots, a_{i,n-1})$，$0 \leq i \leq m-1$，即：

   $$\mathbf{A} = ((a_{00}, a_{01}, \ldots, a_{0,n-1}), (a_{10}, a_{11}, \ldots, a_{1,n-1}), \ldots, (a_{m0}, a_{m1}, \ldots, a_{m-1,n-1}))$$

   也可以看成是由n个列向量组成的线性表，不做赘述。

   $$\mathbf{A} = \begin{bmatrix} (a_{00} & a_{01} & \cdots & a_{0,n-1}) \\ (a_{10} & a_{11} & \cdots & a_{1,n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1}) \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ \vdots \\ a_{m-1,0} \end{pmatrix} & \begin{pmatrix} a_{01} \\ a_{11} \\ \vdots \\ a_{m-1,1} \end{pmatrix} & \begin{pmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \end{pmatrix} & \begin{pmatrix} a_{0,n-1} \\ a_{1,n-1} \\ \vdots \\ a_{m-1,n-1} \end{pmatrix} \end{bmatrix}$$

   二维数组中每个元素a_{ij}既属于第i行的行向量，又属于第j列的列向量，元素a_{ij}在行上的前驱和后继是$a_{i,j-1}$和$a_{i,j+1}$，在列上的前驱和后继是$a_{i-1,j}$和$a_{i+1,j}$
3. **数组的顺序存储方式**：数组一般不做插入和删除操作，即结构中元素个数和元素间关系不变化。一般采用顺序存储结构表示数组。那么如何将多维数组的元素排成线性序列后存入存储器呢?
   - 行主序（行优先顺序）。将数组元素按照行向量排列，第i个行向量后面紧接着存储第i+1个行向量。
   - 列主序（列优先顺序）。将数组元素按照列向量排序，第j个列向量后面紧接着存储第j+1个列向量。
4. **数组中元素存储地址的计算**：在已知数组首地址的情况下，根据给定的一组下标便可以求得相应元素的存储位置。
   - 一维数组：假设一维数组$A[c_i \ldots d_i]$存放在一块连续的存储单元中，每个元素占据C个连续字节，如果数组中元素$A[c_1]$的首地址是$Loc(c_1)$，则$A[c_1+1]$的首地址是$Loc(c_1)+C$，对于$c_1 \leq i \leq d_1$，都有：

     $$Loc(i) = Loc(c_1) + (i - c_1) \times C$$

     数组的起始位置也称为基地址，在一般程序语言中基地址为0，带入$c_1 = 0$可得：$Loc(i) = Loc(0) + i \times C$
   - 二维数组：行主序和列主序的访问均无需着笔讨论，很简单。
5. **二维数组的抽象数据类型的定义**：数组是定长线性表。数组一旦被定义，数组的维数和维界（下标的取值范围）及元素间的关系都不能被改变。对于数组的操作一般只有两种：求给定下标的元素的值、对给定下标的元素赋值。二维数组的抽象数据类型的定义如下：

```cpp
template<class T>
class Array{
public:
    virtual void setValue(int row, int col, const T& data) = 0;
    virtual T getValue(int row, int col) = 0;
    virtual ~Array(){};
};
```

非重点，不做过多探讨。