Priority Queue (优先级队列)

- 优先级队列　每次从队列中取出具有最高优先权的元素
- Ex：任务优先级及执行顺序的关系

任务编号	1	2	3	4	5
优先级	20	0	40	30	10
执行顺序	3	1	5	4	2

数字越小，优先级越高

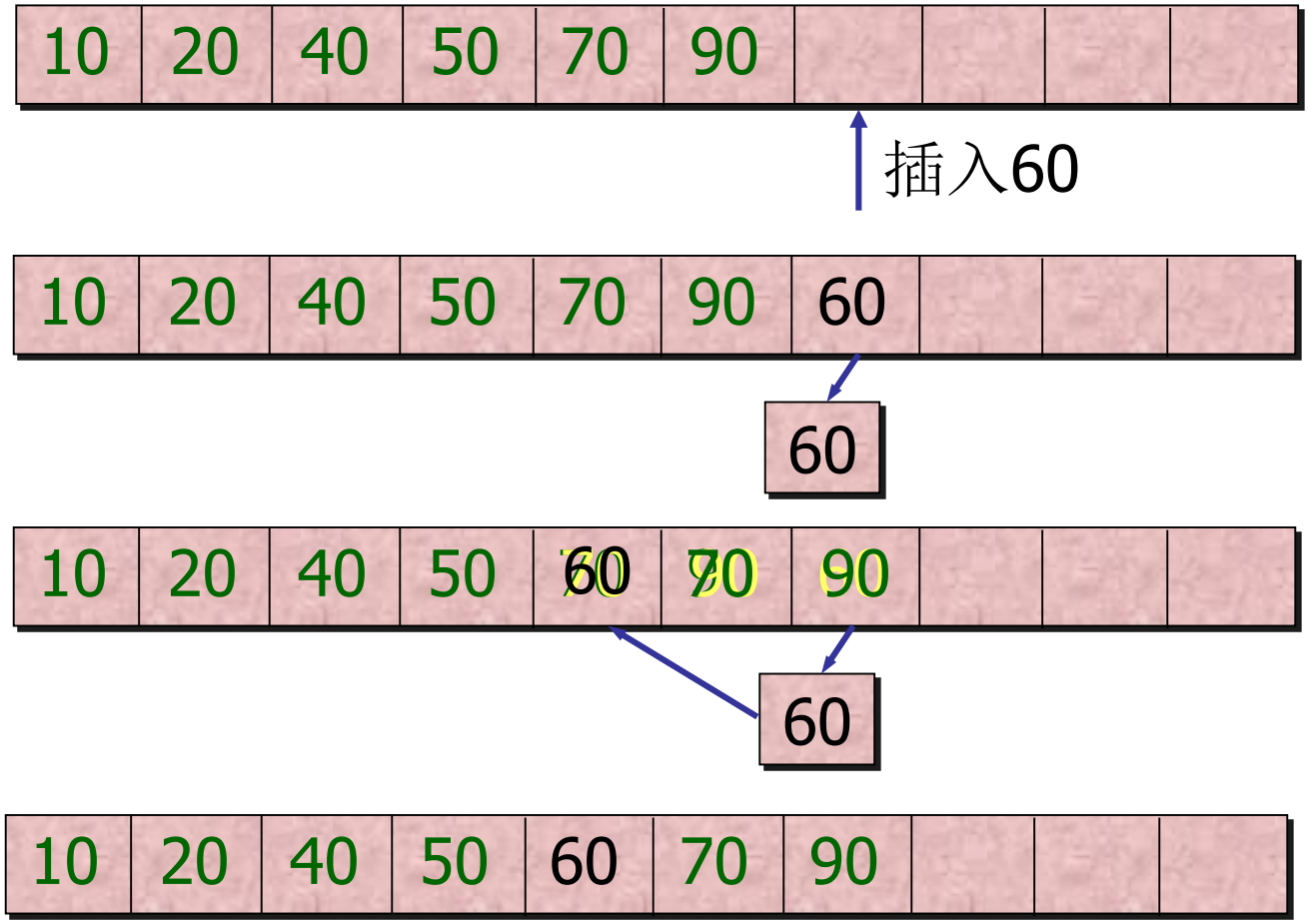
ADT of Priority Queue

```
#include <assert.h>
#include <iostream.h>
#include <stdlib.h>

template <class E>
class PQueue {
private:
    E *pqelements;          //存放数组
    int count;              //队列元素计数
    int maxPQSize;              //最大元素个数
    void adjust();              //调整
```

```
public:
    PQueue(int sz = 50);
    ~PQueue()
        { delete [ ] pqelements; }
    bool Insert(E x);
    bool RemoveMin(E& x);
    bool GetFront(E& x);
    void MakeEmpty()
        { count = 0; }
    bool IsEmpty() const
        { return count == 0; }
    bool IsFull() const
        { return count == maxPQSize; }
    int Length() const
        { return count; }
};
```

优先级队列部分成员函数的实现

```
template <class E>
PQueue<E>::PQueue(int sz) {
    maxPQSize = sz;  count = 0;
    pqelements = new E[maxPQSize];
    assert (pqelements != NULL);
}

template <class E>
bool PQueue<E>::GetFront (E& x) {
    if (IsEmpty() == true) return false;
    x = pqelements[0];
    return true;
}
```


10 20 40 50 70 90
插入60
10 20 40 50 70 90 60
60
10 20 40 50 60 70 90
60
10 20 40 50 60 70 90

```
template <class E>
bool PQueue<E>::Insert(E x) {
    if (IsFull() == true) return false;   //判队满断言
    pqelements[count++] = x;              //插入
    adjust();
return true;
}
```

```
template <class E>
void PQueue<E>::adjust()
{    E temp = pqelements[count-1];
     //将最后元素暂存再从后向前找插入位置
     for (int j = count-2;  j >= 0;  j--)
          if (pqelements[j] <= temp) break;
          else pqelements[j+1] = pqelements[j];
     pqelements[j+1] = temp;
}
```

```
template <class E>
bool PQueue<E>::RemoveMin(E& x)
{
    if (IsEmpty()) return false;
    x = pqelements[0];        //取出0号元素
    for (int i = 1; i < count; i++)
        pqelements[i-1] = pqelements[i];
        //从后向前逐个移动元素填补空位
    count--;
    return true;
}
```

双端队列

- 双端队列:在队列的两端进行插入和删除

3个队列头部函数：

– getHead, EnQueueHead, DeQueueHear

3个队列尾部函数：

– getTail, EnQueueTail, DeQueueTail

2个队列函数：

– EnQueue: // EnQueueTail

– DeQueue: // DeQueueHear

- getHead: // SeqQueue::getFront(x)
- DeQueueHear: // SeqQueue::DeQueue(x)
- EnQueueTail: // SeqQueue::EnQueue(x)

- getTail: 判非空?; 返回队尾元素;
- EnQueueHead: 判满?;不满则在队首插入;
- DeQueueTail: 判空?;不空则在队尾删除;

The END