20190408 左士海——作业 2

1. 试编写算法，从顺序表中删除具有最小值的元素并由函数返回最小值，空出的位置由最后一个元素填补，若顺序表为空则显示出错信息并退出运行。

```
134  template<class T, int MaxSize>
135  T SeqList<T, MaxSize>::delMin() {
136      if (length < 1) {
137          std::cerr << "顺序表为空";
138          exit(1);
139      }
140      T min = data[0];
141      int pos = 0;
142      for (int i = 1; i < length; ++i) {
143          if (data[i] < min) {
144              min = data[i];
145              pos = i;
146          }
147      }
148      data[pos] = data[length - 1];
149      length--;
150      return min;
151  }
```

解：只需要遍历一遍循序表，并记录最小结点的索引，再直接将其替换为最后一个元素。
时间复杂度：$O(n)$。

2. 试编写算法，从顺序表中删除具有给定值 *x* 的所有元素。

```
153  template<class T, int MaxSize>
154  void SeqList<T, MaxSize>::delAllX(T x) {
155      int index = 0
156      for (int i = 0; i < length; ++i) {
157          if (data[i] != x) {
158              data[index++] = data[i];
159          }
160      }
161      length = index;
162  }
```

解：遍历一遍顺序表的过程实现表的重新覆盖。
时间复杂度：$O(n)$。

3. 试编写算法，从有序表中删除其值在给定值 *s* 和 *t*(要求 *s* 小于 *t*)之间的所有元素。

```
164  template<class T, int MaxSize>
165  void SeqList<T, MaxSize>::delByRange(T s, T t) {
166      int index = 0;
167      for (int i = 0; i < length; ++i) {
168          if(data[i] <= s || data[i] >= t) {
169              data[index++] = data[i];
170          }
171      }
172      length = index;
173  }
```

解：与上一题思路相同，只有判断条件的差异。
时间复杂度：$O(n)$。

4. 试编写算法，从顺序表中删除所有其值重复的元素，使所有元素的值均不同。如对于线性表(2，8，9，2，5，5，6，8，7，2)，则执行此算法后变为(2，8，9，5，6，7) 。注意：

表中元素未必是排好序的，且每个值的第一次出现应当保留。

```
template<class T, int MaxSize>
int SeqList<T, MaxSize>::find(const T& e, int lo, int hi) {
    // 从后向前找
    while ((lo < hi--) && (e != data[hi]));
    return hi;
}

template<class T, int MaxSize>
void SeqList<T, MaxSize>::remove(int i) {
    int j = i + 1;
    while (j < length) {
        data[i++] = data[j++];
    }
    length = i;
}

template<class T, int MaxSize>
void SeqList<T, MaxSize>::deduplicate() {
    int index = 1;
    while (index < length) {
        (find(data[index], 0, index) < 0) ? index++ : remove(index);
    }
}
```

解：无序唯一化，主要借助 find 与 remove 方法。find 返回表中左后一个查找值的下标。依次遍历，向前查找重复值，若有则删除当前值。每次迭代所需时间为：$O(n)$。
总时间复杂度：$O(n^2)$。

5.试编写算法，根据一个元素类型为整型的单链表生成两个单链表，使得第一个单链表中包含原单链表中所有元素值为奇数的结点，使得第二个单链表中包含原单链表中所有元素值为偶数的结点，原有单链表保持不变。

```
// 生成奇偶链表
template<typename T>
void genList(LinkList<T>& L1, LinkList<T>& L2, LinkList<T>& L3) {
    Node<T> * p1 = L1.head->next;
    while (p1 != nullptr) {
        (p1->data % 2 != 0) ? L2.insert(L2.listLength() + 1, p1->data) : L3.insert(L3.listLength() + 1, p1->data);
        p1 = p1->next;
    }
}
```

解：遍历链表 L1，若当前节点结点元素值为奇数，则尾插至 L2，否则插入 L3。每次迭代时间复杂度：$O(n)$。
总时间复杂度：$O(n^2)$。

```
linklist: [ 21 4356 54 675 4 78 43 78 4 78 3 5 56 324 768 5287 348 ]
linklist: [ 21 675 43 3 5 5287 ]
linklist: [ 4356 54 4 78 78 4 78 56 324 768 348 ]
```

6.设表 *L* 用数组表示，且各元素值递增有序。试写一算法，将元素 *x* 插入到表 *L* 的适当位置，使得表中元素仍保持递增有序。

```cpp
template<class T, int MaxSize>
void SeqList<T, MaxSize>::insertOrder(const T& e) {
    int lo = 0, hi = length - 1, mid = 0;
    T x = data[(lo + hi) >> 1];
    while (lo <= hi) {
        mid = (lo + hi) >> 1;
        if (e > data[mid]) {
            lo = mid + 1;
        } else if (e < data[mid]) {
            hi = mid - 1;
        } else {
            break;
        }
    }
    for (int i = length; i > mid; --i) {
        data[i + 1] = data[i];
    }
    data[mid+1] = e;
    length++;
}
```

解：由于数组是有序数组，所以可以通过二分查找获得要插入元素的索引：O(logn)。
总时间复杂度：$O(n)$。

7. 已知一个单链表，设计一个复制单链表的算法。

```cpp
template<class T>
LinkList<T>::LinkList(const LinkList<T> &A) {
    head = new Node<T>;
    Node<T> * rear = head;
    Node<T> * t = A.head->next;
    while (t != nullptr) {
        auto * temp = new Node<T>;
        temp->data = t->data;
        rear->next = temp;
        rear = temp;
        t = t->next;
    }
    rear->next = nullptr;
}
```

```cpp
template<class T>
LinkList<T> & LinkList<T>::operator=(const LinkList<T> & A) {
    if (this != &A) {
        Node<T> * p = head->next;
        Node<T> * q;
        while (p != nullptr) { q = p; p = p->next; delete q;}
    }
    Node<T> * rear = head;
    Node<T> * t = A.head->next;
    while (t != nullptr) {
        auto * temp = new Node<T>;
        temp->data = t->data;
        rear->next = temp;
        rear = temp;
        t = t->next;
    }
    rear->next = nullptr;
    return *this;
}
```

解：即设计拷贝构造函数与重载单链表等于符号，二者实现原理相同，差异在于是否有对原链表的删除。

8. 已知一个无序单链表，表中结点的 data 字段为正整数。设计一个算法按递增次序打印表中结点的值。

```cpp
template<class T>
void LinkList<T>::printByOrder() {
    vector<T> v;
    auto * p = head->next;
    while (p != nullptr) {
        v.push_back(p->data);
        p = p->next;
    }
    sort(v.begin(), v.end());
    for(auto it = v.begin(); it != v.end(); ++it)
    {
        cout<<*it<<" ";
    }
    cout << endl;
}
```

解：先进行一次遍历：$O(n)$ 。并将数据取出至数组，再利用快速排序对数组排序：$O(nlogn)$。最后迭代输出。
总时间复杂度：$O(n)$。