单链表基本操作的 C++代码

```
// 单链表的构造函数, 生成一个只有哨位结点的空表
template <class T>
SLList<T>::SLList()
{
    head=tail=currptr=new SLNode<T>();          // 创建哨位结点
    size=0;
};
// 单链表的构造函数, 生成含有哨位结点和一个表结点的表
template <class T>
SLList<T>::SLList(T &item)
{
    tail=currptr=new SLNode<T>(item);   // 生成一个表结点
    head=new SLNode<T>(currptr);        // 生成哨位结点
    size=1;
};
// 单链表的析构函数
template <class T>
SLList<T>::~SLList()
{
    while(!IsEmpty())
    {
        currptr=head->next;
        head->next=currptr->next;
        delete currptr;
    }
    delete head;
};
```

算法 Insert

```
// 在当前结点后插入一个 data 域值为 item 的结点
template<class T>
void SLList<T>::Insert(const T &item)
{
    currptr->next=new SLNode<T>(item, currptr->next);
    if(tail==currptr)
        tail=currptr->next;
    size++;
};
// 在表尾插入一个 data 域值为 item 的结点
template<class T>
void SLList<T>::InsertFromTail(const T &item)
{
    tail->next=new SLNode<T>(item, NULL);
    tail=tail->next;
    size++;
};
// 在哨位结点后插入
template<class T>
void SLList<T>::InsertFromHead(const T &item)
{   if(IsEmpty())
    {   head->next=new SLNode<T>(item, head->next);
        tail=head->next;
    }
    else
```

```
            head->next=new SLNode<T>(item, head->next);
        size++;
};
```

算法 Delete

```
// 删除当前结点的后继结点并将其 data 值返回给变量 de_item
template<class T>
bool SLList<T>::Delete (T &de_item)
{
        if(currptr==tail||IsEmpty())
        {       cout<<"No next node or empty list!";
                return false;
        }
        SLNode<T> *temp=currptr->next;
        currptr->next=temp->next;
        size--;
        de_item=temp->data;
        if(temp==tail) tail=currptr;    // 考察被删除结点是否为原表尾
        delete temp;
        return true ;
};
// 删除哨位结点后的第一个真正表结点并将其 data 值返回给变量 de_item
template<class T>
bool SLList<T>::DeleteFromHead (T &de_item)
{
        if(IsEmpty())
        {       cout<<"Empty list!";
                return false;
        }
        SLNode<T> *temp=head->next;
        head->next=temp->next;
        size--;
        de_item=temp->data;
        if(temp==tail) tail=head;               // 若原表中除了哨位结点外只有一个表结点
        delete temp;
        return true;
};

// 删除表尾结点并将其 data 值返回给变量 de_item
template<class T>
bool SLList<T>::DeleteFromTail (T &de_item)
{
        if(IsEmpty())
        {       cout<<"Empty list!";
                return false;
        }
        // 令当前指针指向表尾结点的前驱结点
        SetEnd();
        Prev();
        de_item=tail->data;
        currptr->next=NULL;
        size--;
        delete tail;
        tail=currptr;
        return true;
};
```