

# C/C++程序设计案例实战

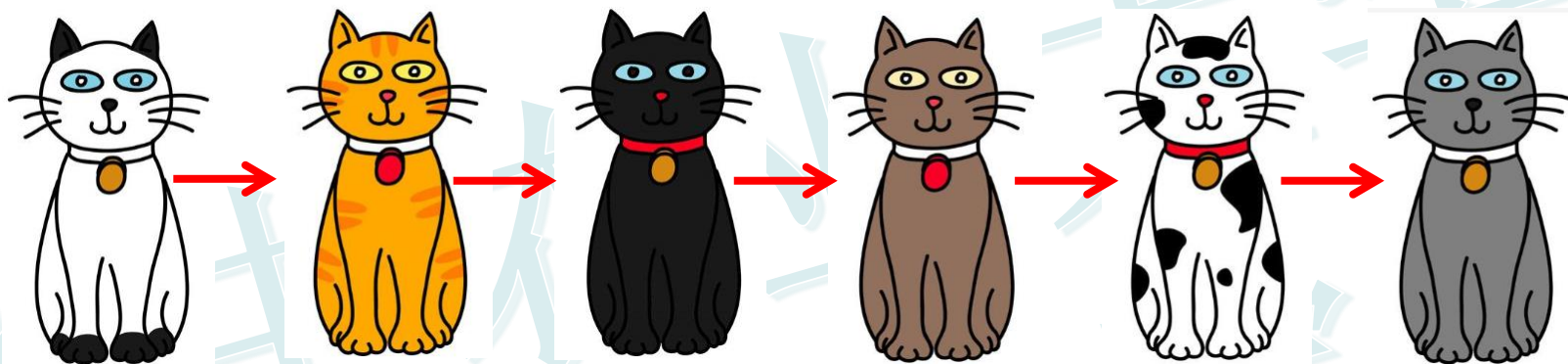
## ——手拉手做游戏之克隆

华中农业大学信息学院 章 英

# 问题引入



# 问题引入



到来

离开

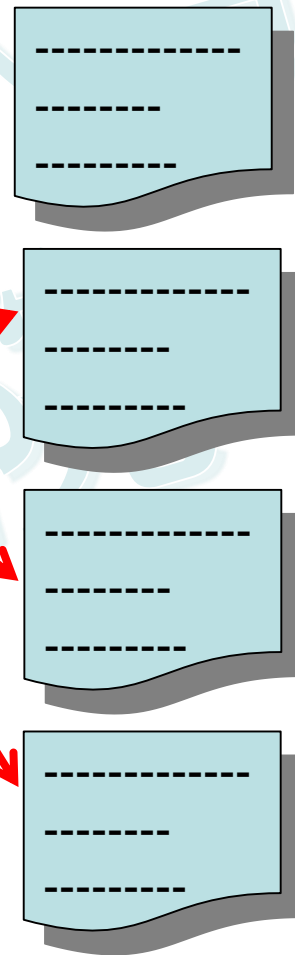
查找

一物多用

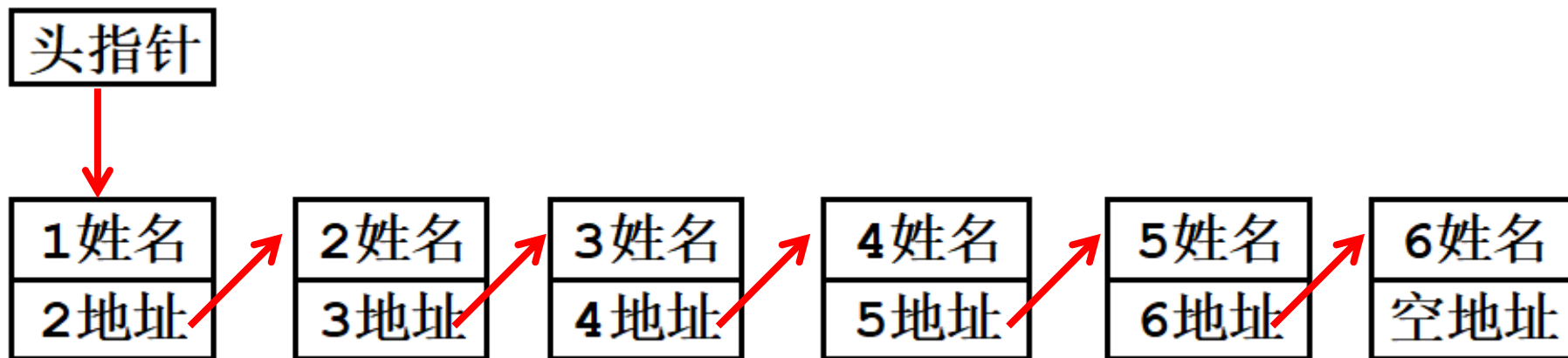
类模板

整形  
实数型  
字符串型  
学生类类型

链表类



# 链表类



插入

删除

查找

遍历

# 链表类——代码实现

```
5 struct Node {  
6     int data;  
7     Node *next;  
8 };  
9  
10 class List {  
11     private:  
12         int *head;  
13     public:  
14         List() { head=NULL; }  
15         void Insert(int &x);  
16         void Delete(int &x);  
17         Node *Search(int &x);  
18         void PrintList();  
19         ~List();  
20     };
```

1 姓名

2 地址

头指针

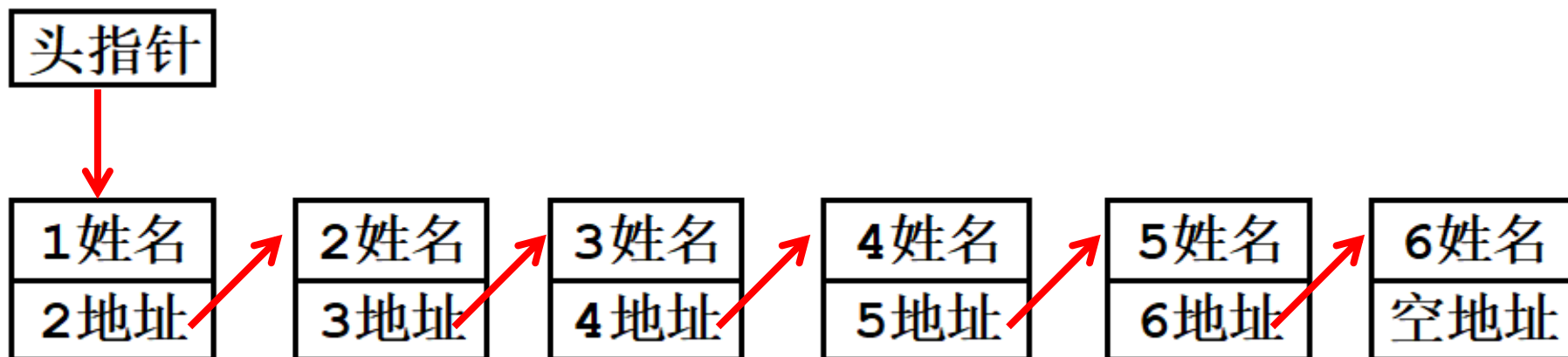
插入

删除

查找

遍历

# 实例化链表类对象



**List t;**

**t.head**

**NULL**

# 链表类模板——代码实现

```
27  template <class T>
28  struct Node {
29      Node<T> *next;
30      T data;
31  };
32
33  template <class T>
34  class List {
35  private:
36      Node<T> *head;
37  public:
38      List() { head=NULL; }
39      void Insert(T &t);
40      void Delete(T &t);
41      Node<T> *Search(T &t);
42      void PrintList();
43      ~List();
44  };
```

template <class T>

开始  
克隆



# 链表类模板——代码实现

```
27  template <class T>
28  struct Node {
29      Node<T> *next;
30      T data;
31  };
32
33  template <class T>
34  class List {
35  private:
36      Node<T> *head;
37  public:
38      List() { head=NULL; }
39      void Insert(T &t);
40      void Delete(T &t);
41      Node<T> *Search(T &t);
42      void PrintList();
43      ~List();
44  };
```

**Node <T>**



# 链表类模板——代码实现

```
27  template <class T>
28  struct Node {
29      Node<T> *next;
30      T data;
31  };
32
33  template <class T>
34  class List {
35  private:
36      Node<T> *head;
37  public:
38      List() { head=NULL; }
39      void Insert(T &t);
40      void Delete(T &t);
41      Node<T> *Search(T &t);
42      void PrintList();
43      ~List();
44  };

```

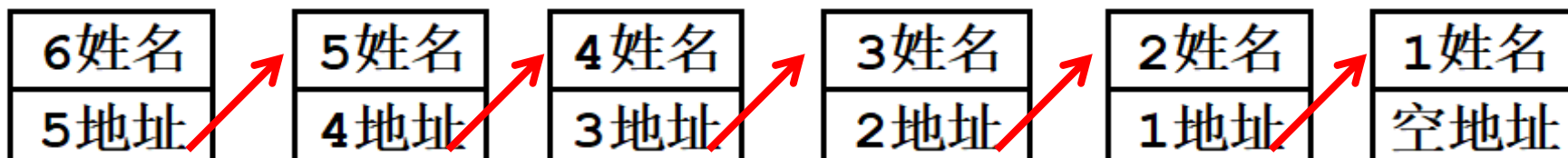
The diagram illustrates the relationship between the template parameter **T** and its usage in the code. A large **T** is positioned on the right, with three red arrows pointing to its occurrences: **Node<T>** in line 29, **T &t** in line 39, and **T &t** in line 41.

# 头插法创建单链表

```
47  template <class T>
48  void List<T>::Insert (T &t)
49  {
50      Node<T> *pnew;
51      pnew=new Node<T>;
52      pnew->data=t;
53      pnew->next=head;
54      head=pnew;
55  }
```

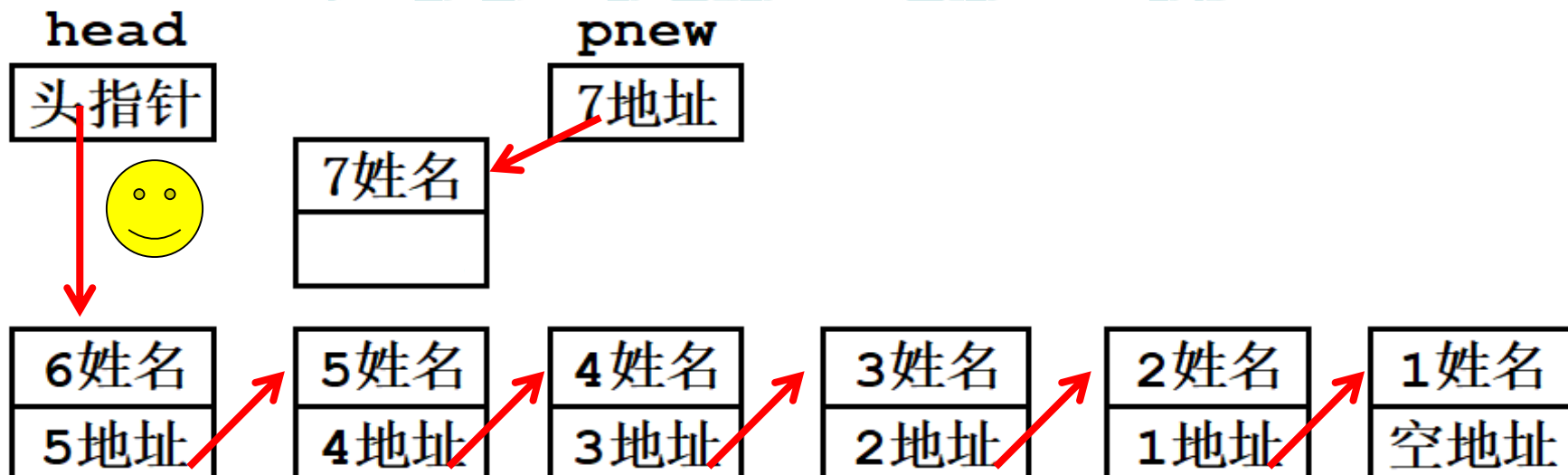
head

头指针



# 头插法创建单链表

```
47  template <class T>
48  void List<T>::Insert (T &t)
49  {
50      Node<T> *pnew;
51      pnew=new Node<T>;
52      pnew->data=t;
53      pnew->next=head;
54      head=pnew;
55  }
```



# 头插法创建单链表

```
47  template <class T>
48  void List<T>::Insert (T &t)
49  {
50      Node<T> *pnew;
51      pnew=new Node<T>;
52      pnew->data=t;
53      pnew->next=head;
54      head=pnew;
55  }
```



# 头插法创建单链表

```
47  template <class T>
48  void List<T>::Insert (T &t)
49  {
50      Node<T> *pnew;
51      pnew=new Node<T>;
52      pnew->data=t;
53      pnew->next=head;
54      head=pnew;
55  }
```



# 类模板——模板类

一物两用

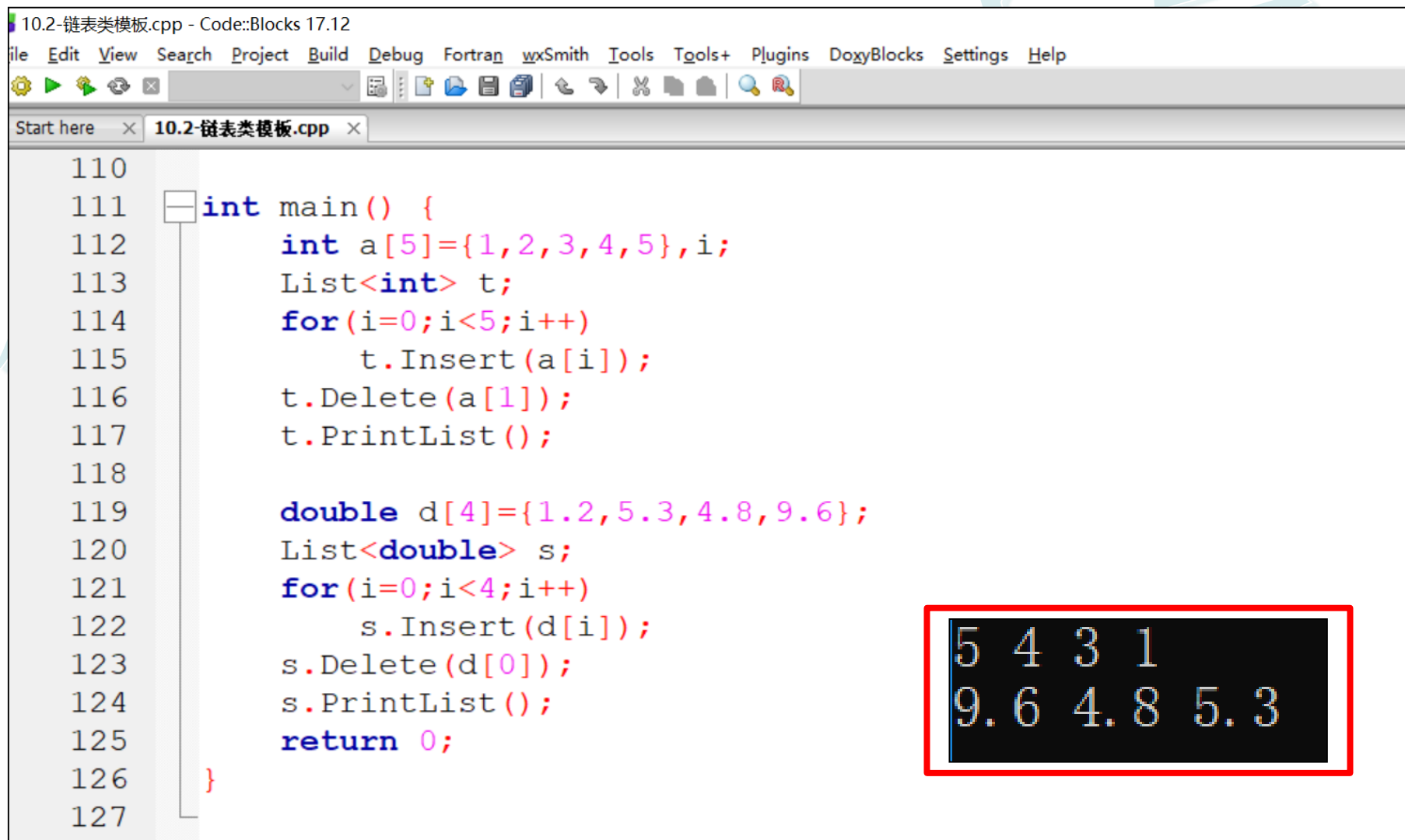
```
111 int main() {  
112     int a[5]={1,2,3,4,5},i;  
113     List<int> t;  
114     for(i=0;i<5;i++)  
115         t.Insert(a[i]);  
116     t.Delete(a[1]);  
117     t.PrintList();  
118  
119     double d[4]={1.2,5.3,4.8,9.6};  
120     List<double> s;  
121     for(i=0;i<4;i++)  
122         s.Insert(d[i]);  
123     s.Delete(d[0]);  
124     s.PrintList();  
125     return 0;  
126 }
```

5 4 3 2 1  
5 4 3 1

9.6 4.8 5.3 1.2  
9.6 4.8 5.3



# 案例——代码运行结果

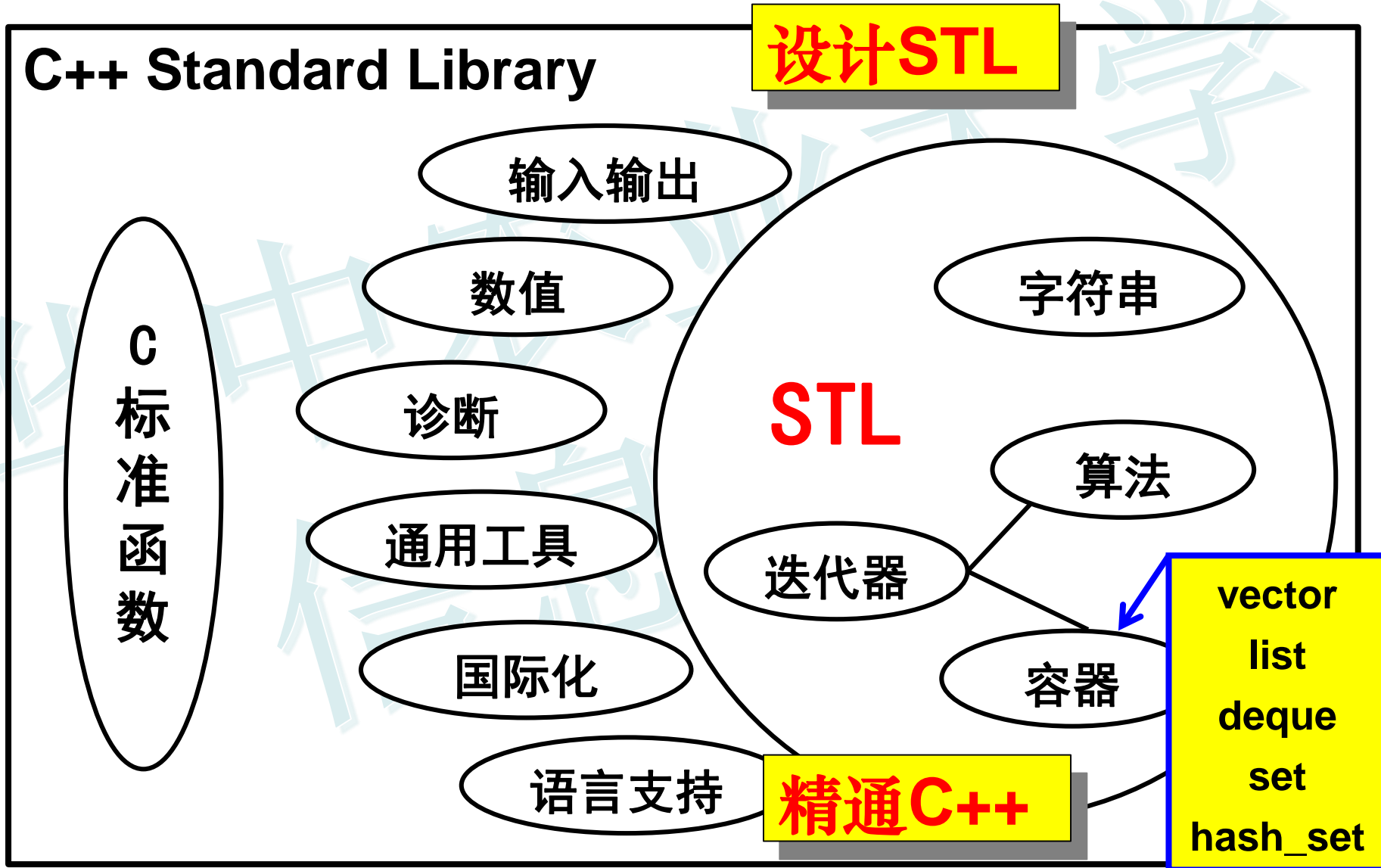


```
110
111 int main() {
112     int a[5]={1,2,3,4,5},i;
113     List<int> t;
114     for(i=0;i<5;i++)
115         t.Insert(a[i]);
116     t.Delete(a[1]);
117     t.PrintList();
118
119     double d[4]={1.2,5.3,4.8,9.6};
120     List<double> s;
121     for(i=0;i<4;i++)
122         s.Insert(d[i]);
123     s.Delete(d[0]);
124     s.PrintList();
125     return 0;
126 }
127
```

5 4 3 1  
9.6 4.8 5.3



# STL——Standard Template Library



# 链表类模板

```
template <class T>  
class List
```

整形

实数型

字符串型

学生类类型

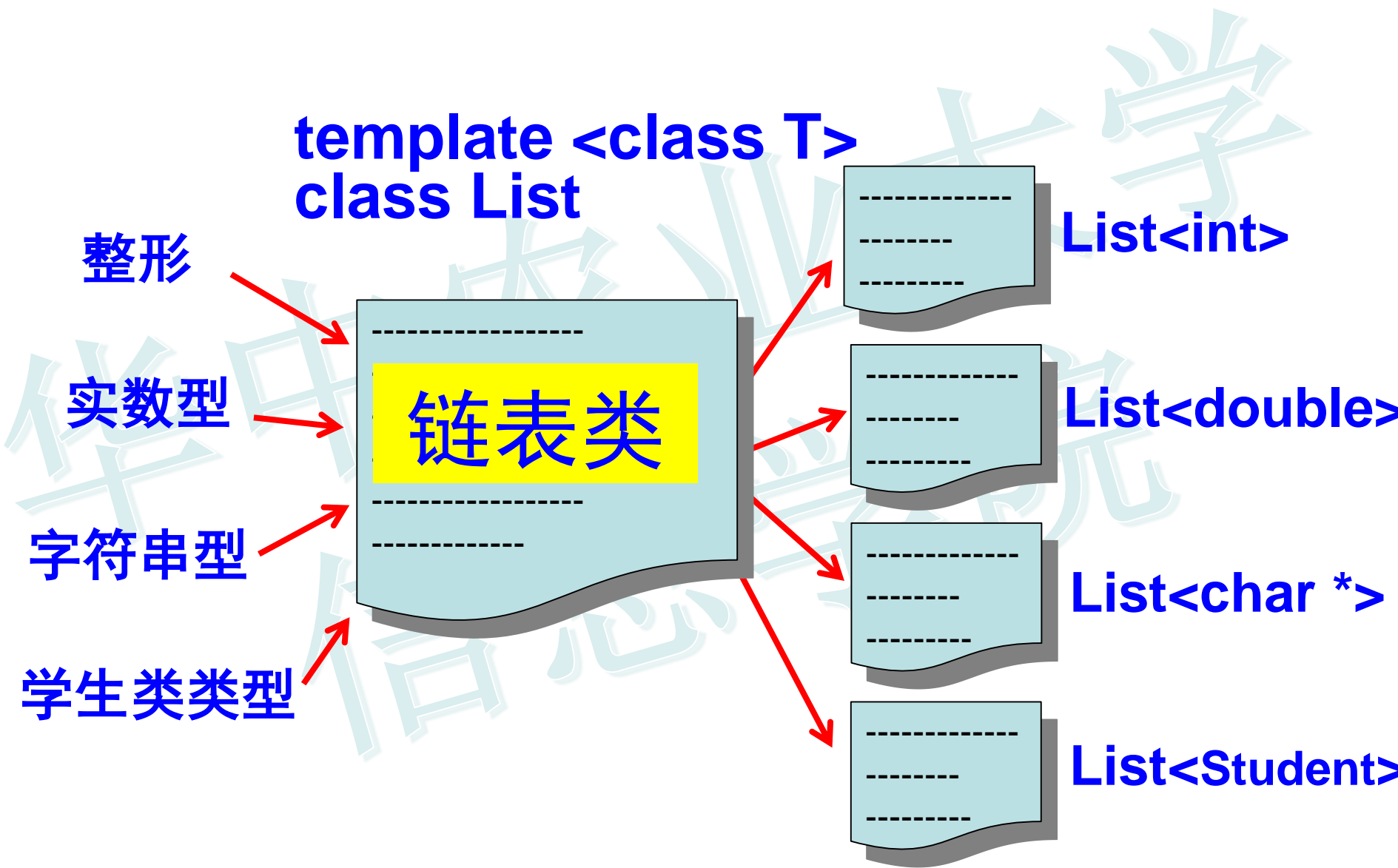
链表类

List<int>

List<double>

List<char \*>

List<Student>



## 小结

- (1) 能够写出头插法创建单链表的过程
- (2) 能够写出链表类模板的具体实现

## 延伸

请继续完善案例代码的链表类模板，  
实现字符串链表类和学生链表类。