函数模板

西安交通大学 仇国巍

函数重载是最佳方案吗？

- 假如设计一个求两参数最大值的函数，在实践中可能需要定义四个函数：

```
int max ( int a, int b ) { return ( a > b ) ? a , b; }
long max ( long a, long b ) { return ( a > b ) ? a , b;}
double max ( double a, double  b ) { return ( a >b)? a , b; }
char max ( char a, char b ) { return ( a > b ) ? a , b; }
```

- 这些函数几乎相同，唯一的区别就是形参类型不同
- 需要事先知道有哪些类型会使用这些函数，对于未知类型这些函数不起作用

比如：string

专家的解决方案

```
template < class  T >     //class 某种类型
T max(T a , T b)
{
        return ( a > b ) ? a , b;
}
```

T 是什么？

答：是可以比较大小的某种类型。

这种抽象的东西是什么？

答：就像一个模板，所以叫函数模板

函数模板

函数模板是用类型做参数，设计出的通用的函数。

其定义形式为：

```
template <class T1,class T2,,,>
函数返回类型 函数名(函数参数表)
{
        //函数模板定义
}
```

T1，T2等等用在其中

其中template表示定义的是模板，＜ ＞里是模板的类型参数，可以是一个或多个。

模板工作方式

- 函数模板只是说明，不能直接执行，需要实例化为模板函数后才能执行
- 在说明了一个函数模板后，当编译系统发现有一个对应的函数调用时，将根据实参中的类型来确认是否匹配函数模板中对应的形参，然后生成一个重载函数。

函数模板如下：

```
template <class T>
T  Max(T a,T b)
{ return a>b? a:b;  }
```

编译时发现：

```
……
Max(6 , 7);
…….
```

编译系统生成：

```
int  Max(int a, int b)
{ return a>b? a:b;  }
```

1 + 2 = 3

函数模板举例——求数组中最小值

```
#include <iostream>
template <class T>
T min(T a[],int n)
{
    int i;
    T minv=a[0];
    for( i=1;i<n ; i++)
    {
        if(minv>a[i])
            minv=a[i];
    }
    return minv;
}
```

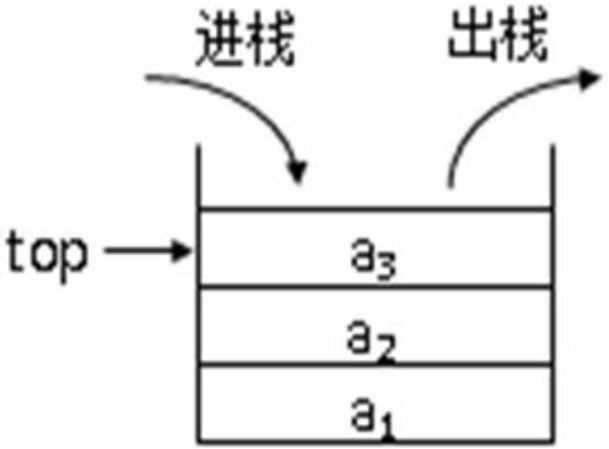
```
void main()
{
    int a[]={1,3,0,2,7,6,4,5,2};
    double b[]={1.2,-3.4,6.8,9,8};
    cout<<"a数组的最小值为: "
        <<min(a,9)<< endl;
    cout<<"b数组的最小值为: "
        <<min(b,4)<<endl;
}
```

a数组的最小值为：0
b数组的最小值为：-3.4

类模板

从栈说起

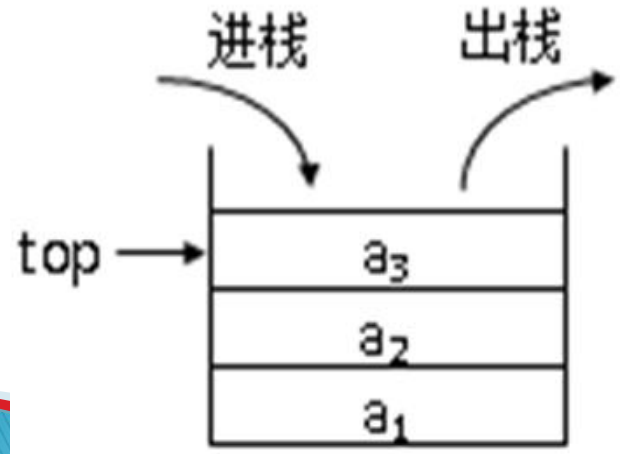
- 栈限制在结构的一端进行插入和删除操作
- 允许插入和删除操作的一端称为栈顶，另一端称为栈底



假定用int数组来存储栈里的数据，能否建立一个“栈”类？

建立int型的栈类

- 属性(数据成员)：
  - int型数组(存储)
  - top(栈顶位置)
- 操作(成员函数)：
  - 进栈(插入)
  - 出栈(删除)




```
class stack
{
    int dat[100];   //设元素数<=100
    int top;
public:
    void push(int e){     //进栈
        将数据放入dat;
        改top;  //向上移
    }
    int  pop( ) {          //出栈
        改top;  //向下移
        返回删除的值;
    }
}
```

建立char, double型的栈类

```
class stack
{
    char dat[100];
    int top;
public:
    //进栈
    void push(char e)
    {   ……   }
    //出栈
    char pop( )
    {   ……   }
}
```

```
class stack
{
    double dat[100];
    int top;
public:
    //进栈
    void push(double e)
    {    ……    }
    //出栈
    double pop( )
    {    ……    }
}
```

换成其他类型可以吗？ 比如string，比如某种struct, ……

建立抽象的栈类模板

```
class stack
{
    char dat[100];
    int top;
public:
    //进栈
    void push(char e)
    {   ……   }
    //出栈
    char pop( )
    {   ……   }
}
```

```
template <class T>
class stack
{
    T dat[100];
    int top;
public:
    //进栈
    void push(T e)
    {   ……   }
    //出栈
    T  pop( )
    {   ……   }
}
```

此处 class 指某种类型，不是C++中的类

类模板

- 类是对问题空间的抽象，而类模板则是对类的抽象，是对一批仅仅成员数据类型不同的类的抽象
- 程序中可以首先定义一个类模板，然后通过使用不同的实参生成不同的类。
- 类模板的定义格式：

```
template <class <类型参数>>
class <类名>
{
        ……
};
```

类模板使用方法

格式： 类模板名 <数据类型> 对象名;

比如： stack <int> s1; // 定义一个整数类型栈

stack <char> s2; // 定义一个字符类型栈

在C++的标准模板库(STL)中，定义了大量类模板（其中也包括栈），使用这些优秀的类模板可以我们提高编程效率，提高程序可靠性。