

# 计算机科学基础II

## 实验十一 引用与复制构造函数

曹鹏

Email: [caopeng@seu.edu.cn](mailto:caopeng@seu.edu.cn)

Tel: 13851945861

# 实验目的

---

## 实验十

1. 掌握引用概念及应用，掌握引用作为函数参数的方法。
2. 学会编写与应用复制构造函数。

## 4. 编程（分数类）

将实验十的分数类：

- 添加复制构造函数
- 参数改为引用

//Exp10\_2.cpp

fraction makeCommond(fraction);

fraction add(fraction);

fraction sub(fraction);

fraction mul(fraction);

fraction div(fraction);

fraction reciprocal();

bool equal(fraction);

bool greaterThan(fraction);

bool lessThan(fraction);

void display();

void input();



//Exp11\_4-2.cpp

void makeCommond(fraction&);

fraction(fraction& a);

fraction add(fraction &);

fraction sub(fraction &);

fraction mul(fraction &);

fraction div(fraction &);

fraction reciprocal();

bool equal(fraction &);

bool greaterThan(fraction &);

bool lessThan(fraction &);

void display();

void input();

## 4. 编程（分数类）

将实验十的分数类：

- 添加复制构造函数
- 参数改为引用

//Exp10\_2.cpp

fraction fraction::makeCommond

(fraction b){

int temp;

reduction();

b.reduction();

above\*=b.below;

b.above\*=below;

temp=below\*b.below;

below=b.below=temp;

return b;

}

b=a.makeCommond(b);

a=a.makeCommond(a);//特例

| a  |    | b (值传递) |    |
|----|----|---------|----|
| 分子 | 分母 | 分子      | 分母 |
| 1  | 2  | 1       | 2  |
| 1  | 2  | 1       | 2  |
| 2  | 2  | 2       | 2  |
| 2  | 2  | 2       | 2  |
| 2  | 2  | 2       | 2  |
| 2  | 4  | 2       | 4  |

正确

## 4. 编程（分数类）

将实验十的分数类：

- 添加复制构造函数
- 参数改为引用

//Exp10\_2.cpp

fraction fraction::makeCommond

(fraction & b){

int temp;

reduction();

b.reduction();

above\*=b.below;

b.above\*=below;

temp=below\*b.below;

below=b.below=temp;

return b;

}

a=a.makeCommond(a);//特例

|  | a  |    | b (引用传递) |    |
|--|----|----|----------|----|
|  | 分子 | 分母 | 分子       | 分母 |
|  | 1  | 2  | 1        | 2  |
|  | 1  | 2  | 1        | 2  |
|  | 2  | 2  | →2       | 2  |
|  | 4← | 2  | 4        | 2  |
|  | 4  | 2  | 4        | 2  |
|  | 4  | 4  | 4        | 4  |

错误

## 4. 编程（分数类）

```
class fraction {  
    int above;      //分子  
    int below;     //分母  
public:  
    fraction(int a=0,int b=1){    //构造函数  
        above=a;  
        below=b;  
    }  
    fraction(fraction& a){        //复制构造函数  
        above=a.above;  
        below=a.below;  
    }  
};
```

## 4. 编程（分数类）

//Exp10\_2.cpp

fraction makeCommond(fraction);

fraction add(fraction);

fraction sub(fraction);

fraction mul(fraction);

fraction div(fraction);

fraction reciprocal();

bool equal(fraction);

bool greaterThan(fraction);

bool lessThan(fraction);

void display();

void input();



//Exp11\_4-2.cpp

void makeCommond(fraction&);

fraction(fraction& a);

fraction add(fraction &);

fraction sub(fraction &);

fraction mul(fraction &);

fraction div(fraction &);

fraction reciprocal();

bool equal(fraction &);

bool greaterThan(fraction &);

bool lessThan(fraction &);

void display();

void input();

## 4. 编程（分数类）

//错误代码，参数引用时不可以与自己通分  
//a=a.makeCommomd(a);

```
fraction fraction::makeCommomd
(fraction &b){
    int temp;
    reduction();    b.reduction();
    above*=b.below;
    b.above*=below;
    temp=below*b.below;
    below=b.below=temp;
    return b;
}
```

//正确代码，参数引用时可以与自己通分  
//a.makeCommomd(a);

```
void fraction::makeCommomd
(fraction &b){
    int temp,t1,t2;
    reduction();    b.reduction();
    t1=above*b.below;
    t2=b.above*below;
    temp=below*b.below;
    below=b.below=temp;
    above=t1;
    b.above=t2;
}
```



## 4. 编程（集合类）

自定义一个集合类set，采用数组存放集合的元素。

- 添加复制构造函数
- 参数改为引用

自定义集合运算包括以下操作：

| 集合类set成员函数功能                        | 成员函数名     |
|-------------------------------------|-----------|
| 1) 判断元素elem是否为集合set的元素。             | Member    |
| 2) 为集合添加一个元素elem。                   | AddElem   |
| 3) 从集合中删除一个元素elem。                  | RmvElem   |
| 4) 复制一个集合，其元素与原集合完全一样。              | Copy      |
| 5) 显示集合中的所有元素。                      | Equal     |
| 6) 求两个集合中相同的元素，即求两个集合的交集。           | Intersect |
| 7) 求两个集合中所有的元素，即求两个集合的并集。           | Union     |
| 8) 判断两集合包含的元素是否完全相同。                | Equal     |
| 9) 判断一个集合是否被包含在另一个集合之中，即是否为另一个集合的子集 | Contain   |

## 4. 编程（集合类）

自定义一个集合类set，采用数组存放集合的元素。

- 添加复制构造函数
- 参数改为引用

//Exp10\_1.cpp

set();

bool Member(char);  
ErrCode AddElem(char);  
void RmvElem(char);  
void Copy(set);  
bool Equal(set);  
void print();

set Intersect(set);  
set Union(set);  
bool Contain(set);



//Exp11\_4-1.cpp

set();

set(set&); //复制构造函数

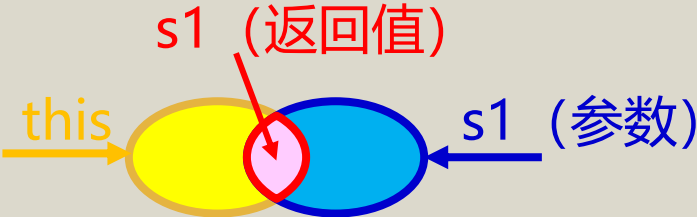
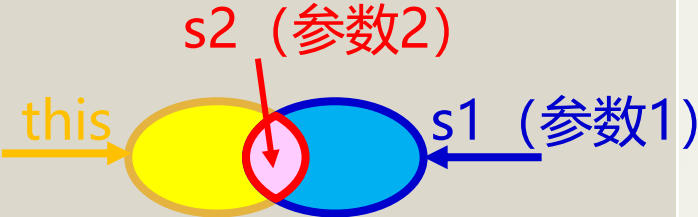
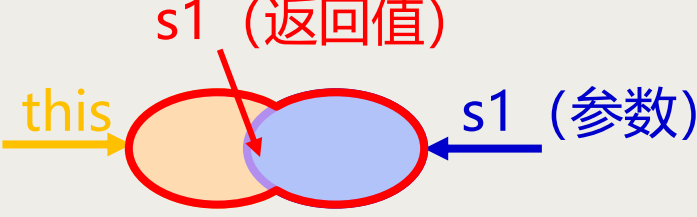
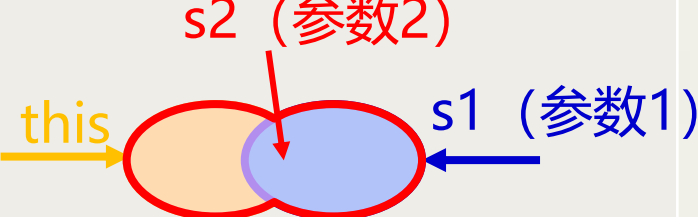
bool Member(char&);  
ErrCode AddElem(char&);  
void RmvElem(char&);  
void Copy(set&);  
bool Equal(set&);  
void print();

void Intersect(set, set&);  
ErrCode Union(set, set&);  
bool Contain(set&);

## 4. 编程（集合类）

自定义一个集合类set，采用数组存放集合的元素。

- 添加复制构造函数
- 参数改为引用

|    | 返回值+值传递(Exp10_1.cpp)   | 参数+引用传递(Exp11_4-1.cpp)   |
|----|--|--|
| 交集 | <pre>set Intersect(set s1);<br/>//求本集合与s1集合的交，并且作为返回值</pre>  <p>The diagram shows two overlapping circles. The left circle is yellow and labeled 'this' with a yellow arrow. The right circle is blue and labeled 's1 (参数)' with a blue arrow. The intersection of the two circles is highlighted with a red border and labeled 's1 (返回值)' with a red arrow.</p> | <pre>void Intersect(set s1, set&amp;s2);<br/>//求本集合与s1集合的交，并存入s2集合中</pre>  <p>The diagram shows two overlapping circles. The left circle is yellow and labeled 'this' with a yellow arrow. The right circle is blue and labeled 's1 (参数1)' with a blue arrow. The intersection of the two circles is highlighted with a red border and labeled 's2 (参数2)' with a red arrow.</p> |
| 并集 | <pre>set Union(set s1);<br/>//求本集合与s1集合的并，并且作为返回值</pre>  <p>The diagram shows two overlapping circles. The left circle is orange and labeled 'this' with a yellow arrow. The right circle is blue and labeled 's1 (参数)' with a blue arrow. The union of the two circles is highlighted with a red border and labeled 's1 (返回值)' with a red arrow.</p>          | <pre>ErrcCode Union(set s1, set&amp; s2);<br/>//求本集合与s1集合的并，并存入s2集合中</pre>  <p>The diagram shows two overlapping circles. The left circle is orange and labeled 'this' with a yellow arrow. The right circle is blue and labeled 's1 (参数1)' with a blue arrow. The union of the two circles is highlighted with a red border and labeled 's2 (参数2)' with a red arrow.</p>     |

## 4. 编程（集合类）

```
class set{
    char elements[maxnum];
    int num;
public:
    set(){num=0;}
    set(set& a){
        int i;
        num=a.num;
        for(i=0;i<num;i++)
            elements[i]=a.elements[i];
    }
};
```

## 4. 编程（集合类）

参数（值传递）

- ◆ 形参分配新的存储空间
- ◆ 类对象参数调用复制构造函数

//Exp10\_1.cpp

set();

```
bool Member(char);  
ErrCode AddElem(char);  
void RmvElem(char);  
void Copy(set);  
bool Equal(set);  
void print();  
set Intersect(set);  
set Union(set);  
bool Contain(set);
```



参数（引用传递）

- ◆ 函数代码不变
- ◆ 形参引用实参地址，形参和实参同一存储空间
- ◆ 类对象参数不调用复制构造函数

//Exp11\_4-1.cpp

set();

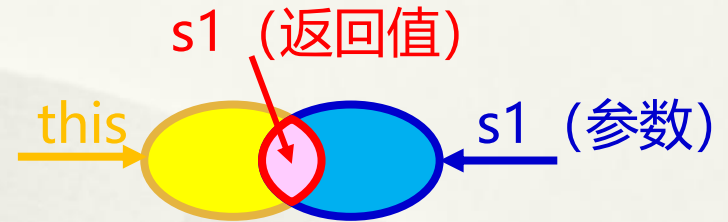
set(set&);

```
bool Member(char&);  
ErrCode AddElem(char&);  
void RmvElem(char&);  
void Copy(set&);  
bool Equal(set&);  
void print();  
void Intersect(set, set&);  
ErrCode Union(set, set&);  
bool Contain(set&);
```

## 4. 编程（集合类）

//求本集合与s1集合的交，并且作为返回值

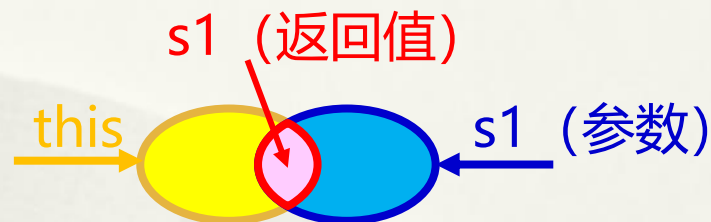
```
set set::Intersect(set s1){  
    int i,j;  
    set s;  
    for(i=0;i<num;i++)  
        for(j=0;j<s1.num;j++)  
            if(elements[i]==s1.elements[j]){  
                s.elements[s.num++] = elements[i];  
                break;  
            }  
    return s;  
}
```



## 4. 编程（集合类）

//求本集合与s1集合的交，并且作为返回值

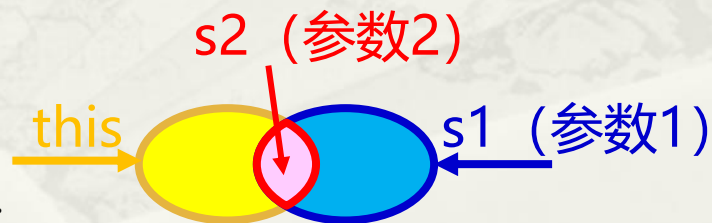
```
set set::Intersect(set s1){  
    int i,j;  
    set s;  
    for(i=0;i<num;i++){  
        for(j=0;j<s1.num;j++){  
            if(elements[i]==s1.elements[j]){  
                s.elements[s.num++] = elements[i];  
                break;  
            }  
        }  
    }  
    return s;  
}
```



//求本集合与参数s1集合的交，并存入参数s2集合中

//这里的s2必须引用，才能改变s2实参将运算结果传递出去

```
void set::Intersect(set s1, set& s2){  
    int i,j;  
    s2.num=0;  
    for(i=0;i<num;i++){  
        for(j=0;j<s1.num;j++){  
            if(elements[i]==s1.elements[j]){  
                s2.elements[s2.num++] = elements[i];  
                break;  
            }  
        }  
    }  
}
```



## 4. 编程（集合类）

//求本集合与s1集合的并，并且作为返回值

```
set set::Union(set s1){
```

```
    int i;
```

```
    set s;
```

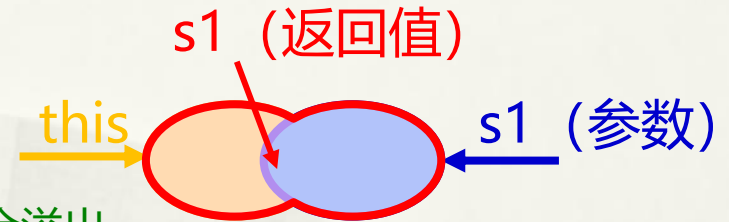
```
    s.Copy(s1);
```

```
    for(i=0;i<num;i++)
```

```
        s.AddElem(elements[i]); //这里忽略了集合溢出
```

```
    return s;
```

```
}
```

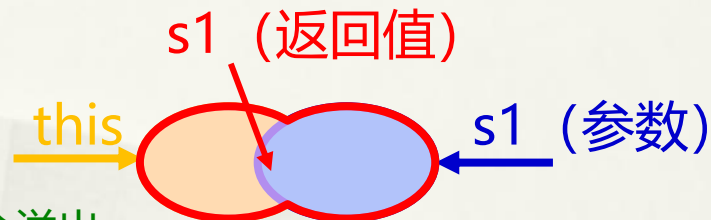




## 4. 编程（集合类）

//求本集合与s1集合的并，并且作为返回值

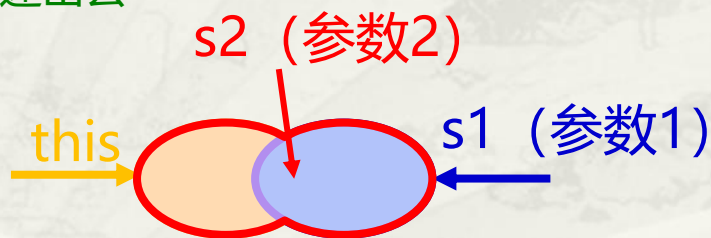
```
set set::Union(set s1){  
    int i;  
    set s;  
    s.Copy(s1);  
    for(i=0;i<num;i++)  
        s.AddElem(elements[i]); //这里忽略了集合溢出  
    return s;  
}
```



//求本集合与参数s1集合的并，并存入参数s2集合中

//这里的s2必须引用，才能改变s2实参将运算结果传递出去

```
ErrCode set::Union(set s1, set& s2){  
    int i;  
    s2.Copy(s1);  
    for(i=0;i<num;i++)  
        if(s2.AddElem(elements[i]) == overflow)  
            return overflow;  
    return noErr;  
}
```



---



# End