

C++

- C++ IO I/O
 - IO
- - <<
- class
 - =
- STL
 - vector

C++ IO I/O

IO

iostream

```
fstream fp; //  
fp.open("fileName",openmode); // openmode fileName
```

ifstream ofstram /

```
ifstream in; //  
in.open("fileName"); // , ios::in  
in.open("fileName",openmode); //  
  
ofstream out; //  
out.open("fileName"); // , ios::out  
out.open("fileName",openmode); //  
  
//  
ifstream in("fileName");  
ifstream in("fileName",openmode); //  
  
ofstream out("fileName");  
ofstream out("fileName",openmode); //
```

in out .open() in out False

openMode	
ios::app	
ios::ate	
ios::in	ifstream
ios::out	ofstream
ios::trunc	

openMode	Flags
ios::binary	ios::binary
ios::nocreate	ios::nocreate
ios::noreplace	ios::noreplace False

Flags openmode |



C++ (stream) IO

```
// , in
char c;
while(in >> c){
    cout<<c;
}

//
string s = "mokemore"
out << s;
```



```
fp.close();
in.close();
out.close();
```



```
// fp n
fp.seekg( n );

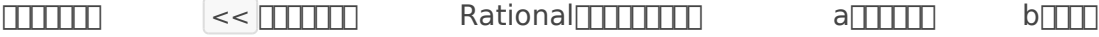
// fp n
fp.seekg( n, ios::cur );

// fp n
fp.seekg( n, ios::end );
```

```
// 文件流 fp 打开
fp.seekg( 0, ios::end );
```

ios 成员函数	文件流
ios::beg	文件开头
ios::cur	文件当前位置
ios::end	文件末尾





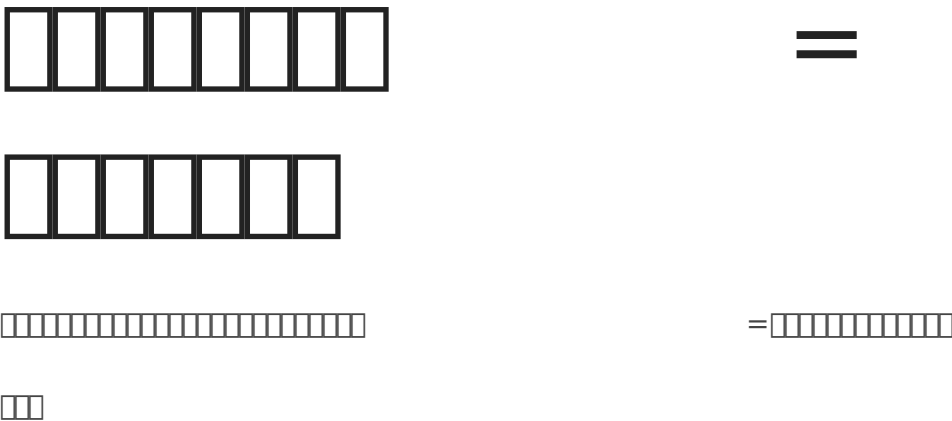
```
class Rational{
private:
    int a;
    int b;
    //[]
    friend std::ostream& operator<<(std::ostream& os , Rational r);

    Rational(int a,int b);    //[]
}

std::ostream& operator<<(std::ostream& os , Rational r){
    os<<r.a<<"/"<<r.b;
    return os;
}
```

class □

class 1



```
/*1111 class Test11111 int* ptr*/
class Test a;
a.ptr=new int(3);
b=a; //11 b11 ptr1111 a1 ptr1111
delete a.ptr;
cout<<*b.ptr; //11111111
```

111111

```
class Test{
public:
    int* ptr;

    ~Test(){
        delete this->ptr;
    }

    Test &operator=(const Test &b){
        if(this!=b){ //1111111111111111
            delete this->ptr;
            this->ptr=new int(*b.ptr);
        }
        return *this;
    }
};
```

STL

vector

int

```
vector<int> v1;           //空容器
vector<int> v2(n);        //容器大小 n 个元素 0
vector<int> v3(n,elem);   //容器大小 n 个元素 elem
vector<int> v4(start,end); //从 start 到 end 的 int* 元素 start 到 end 的元素
vector<int> v5(v4);       //容器大小 v4 的大小
vector<int> v6={1,2,3,4,5}; //容器大小 5
```

vector

```
vector<int> v1={1,2,3},v2={4,5,6},v3; //空容器

v3.assign(n,elem);           //容器大小 n 个元素 elem
v3.assign(v1.begin(),v1.end()); //从 v1 的 begin 到 end 的元素 v3 的元素
v3=v1;                       //v1 的元素 v3 的元素 =v1
v1.swap(v2);                 //v1 和 v2 交换
```

```
vector.size();//容器大小
vector.empty();//容器是否为空
vector.resize(n);//容器大小 n 个元素
vector.resize(n,elem);//容器大小 n 个元素 elem
```

vector

```
vector[n];           //容器大小 n 个元素
vector.at(n);        //vector 的 at 方法 n 个元素
for(vector<int>::iterator it=vector.begin();it!=vector.end();it++){ //遍历
    cout<<*it;
```

}

vector

```
vector.push_back(elem); // 在末尾添加元素 elem
vector.pop_back(); // 删除末尾元素
vector.erase(pos); // 删除 pos 位置的元素
vector.insert(pos, elem); // 在 pos 位置插入元素 elem
vector.insert(pos, n, elem); // 在 pos 位置插入 n 个元素 elem
vector.insert(pos, begin, end); // 在 pos 位置插入 [begin, end) 范围内的元素
```

vector

函数	说明
begin()	返回指向容器第一个元素的迭代器
end()	返回指向容器末尾元素的迭代器
rbegin()	返回指向容器末尾元素的逆序迭代器
rend()	返回指向容器第一个元素的逆序迭代器

vector

vector

```
vector.insert(pos, elem) // 在 pos 位置插入元素 elem
vector.insert(pos, n, elem) // 在 pos 位置插入 n 个元素 elem
vector.insert(pos, begin, end) // 在 pos 位置插入 [begin, end) 范围内的元素
```