Passing Streams to Functions

Passing Streams to Functions

- **One Rule:** always pass a stream as a reference

file: fileopen.h

```
// Pre: template parameter T must be either ifstream or ofstream type.
template <typename T>
void fileopen (T & filestr, const string promptpart)
{
    const int MAX_TRIES = 5;
    int count = 0;
    string filename;
    cout<<"enter name of "<<promptpart<<" file:    ";
    cin>>filename;
    filestr.open(filename.c_str());

    while (!filestr)
    {
        filestr.clear(); // may be necessary on your platform
        cout<<"ERROR: file not connected. Try again..."<<endl;
        cout<<"enter name of "<<promptpart<<" file: ";
        cin>>filename;
        filestr.open(filename.c_str());
        count++;

        if (count > MAX_TRIES)
        {
            cout<<"NOT CONNECTING AFTER "<<MAX_TRIES<<" ATTEMPTS...BAILING OUT" <<"..."<<endl;
            exit(1);
        }
    }
    return;
}
```

```
#include <fstream>
#include “fileopen.h”

int main()
{
    ifstream in;
    fileopen(in, “input”);
```

Operator Overloading

```
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Operator Overloading

```
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}

// example 1
point p1, p2;
cout << p1;
cout << p1 << " " << p2;
```

Operator Overloading

```
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
// example 1
point p1, p2;
cout << p1;
cout << p1 << " " << p2;
// example 2
ofstream fout;
fout << p1;
fout << p1 << " " << p2;
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
cout << " "
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

System Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
cout << " "
returns cout with the space added to the stream
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
cout << " "
returns cout with the space added to the stream
cout << p2
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
cout << " "
returns cout with the space added to the stream
cout << p2
executes overloaded operator<<
```

Chaining

```
// example 1
cout << p1 << " " << p2;
...
ostream& operator << (ostream & out, const point & p)
{
    out << "(" << p.m_X << ", " << p.m_Y << ")";
    return out;
}
```

Stream Processing...

```
cout << p1
executes overloaded operator<<
returns cout with the points data added to the stream
cout << " "
returns cout with the space added to the stream
cout << p2
executes overloaded operator<<
returns cout with the points data added to the stream
```

Final Note

- iostream and fstream are of the same family
- getline, ignore, get, putback, etc are all available for filestreams as well!

```
ifstream fin;
char input;
fin.open("input.dat");
while (in.get(input))
{
    process_data(input);
```

End of Session