IOStreams II

Reminder

- Final exam
  - The date for the Final has been decided:

  - Saturday, November 16th
  - 8am – 10am
  - 01-2000

Announcement

- Exam 2
  - Has been moved to Monday October 28th

Project

- Questions?

- Farmer Problem: due Oct 30th

New plan

- Today: IOStreams 2

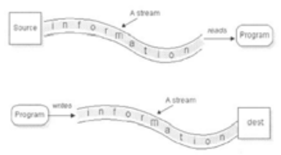
- Monday: Exam 2
- Tuesday: Exceptions
- Thursday: Files I

IOStreams

- Suite of classes for performing I/O in C++
- Reading and Writing:
  - Standard input and output
  - File input and output
  - Input and Output to strings

Streams

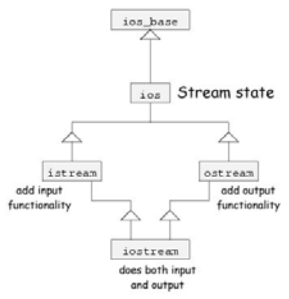
- Like Java, Basic low level mechanism for I/O is the stream
  - Stream is a sequence of bytes



Streams

- Unlike Java, the basic stream in C++ is buffered.
  - Used to increase efficiency
  - When the program writes something, it is put into a buffer in memory.
  - The output doesn't appear on the screen until the buffer is flushed (written)

IOStream class inheritance



`cin` is an istream

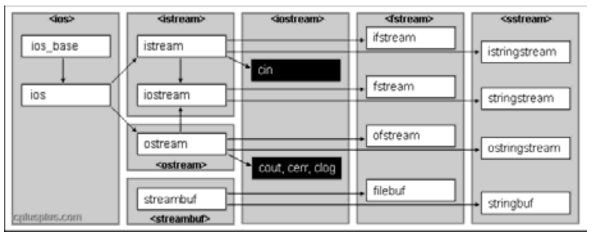
`cout` and `cerr` are ostreams

Other classes inherit from these and add I/O to/from a device, string, or memory

Today

- Today we will look at:
  - fstream – I/O to/from files
  - sstream – I/O to and from strings
  - Writing your own << and >> operators.

IOStream Class Hierarchy



fstream

- Reading and Writing from Files
  - `ifstream` (inherits from `istream`) for input files
  - `ofstream` (inherits from `ostream`) for output files
  - `fstream` (inherits from `iostream`) for files that can support both input and output.

Constructors

- Default constuctors
  - `ifstream(), ifstream(), fstream()`
  - Creates an unopened file.
    - Can use `open()` to attach to a file
- Construct and Open
  - `ifstream( const char *name, int mode = ios::in, int perm = 0644 );`
  - `ofstream( const char *name, int mode = ios::out, int perm = 0644 );`
  - `fstream( const char *name, int mode, int perm = 0644 );`
  - `badbit` set if open fails

Open / Close

- `open( const char *name, int mode, int perm = 0644 );`
  - Opens a file and attaches to a stream
- `close();`
  - closes the file associated with a stream. The stream can be reopened with another file after this.

Using fstreams

- Since fstreams are derived from istream and ostream, I/O is the same as using `cin` and `cout`.

Using fstreams

- fstreams support random access
  - `istream &istream::seekg( streamoff offset, ios::seek_dir where );`
  - `ostream &ostream::seekp( streamoff offset, ios::seek_dir where );`
- `ios::seek_dir = { ios::begin, ios:end, ios::cur};`
  - these are actually implemented in `istream` and `ostream`, and can be used on any stream that is associated with a seekable device

fstreams

- Questions?

sstreams

- adds functionality to do “input” and “output” from/to arrays of characters in memory.
- no actual I/O is done; however, the conversions performed are exactly the same as those we have already covered.
- C++ means of doing `atoi(), itoa()`

sstreams

- E.g. commandline arguments

```
main (int argc, char *argv[])
{
    int intArg;
    float floatArg;

    istringstream ints (argv[1]);
    istringstream floats (argv[2]);
    ints >> intArg;
    floats >> floatArg;
    …
}
```

sstreams

- One reason to use an istrstream is to do conversions on data that have already been read in.

sstreams

- Example: consider a program whose input is supposed to contain four values on each line:

  1 2 3 4
  5 6 7 8
  9 10 11 12
  - Consider the obvious approach:
    - cin >> a >> b >> c >> d;
    - The extractions will skip white space automatically. If the input is erroneous, for example:

      1 2 3 4
      5 6 7
      9 10 11 12

sstreams

```
char buffer[BUFLEN];
int a,b,c,d;
while( cin.getline( buffer, BUFLEN ) ) {
    istringstream S (buffer);
    S >> a >> b >> c >> d;
    …
}
```

sstreams

- E.g. itoa (toString)

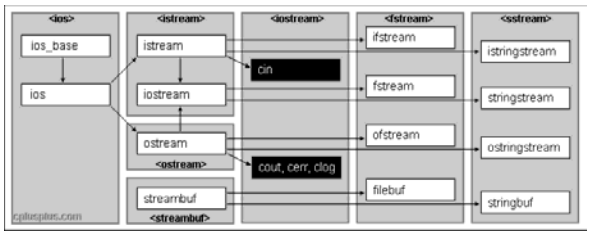
```
char buffer[20];
ostringstream outs (buffer);

int i = 20;
outs << I;
```

sstreams

- Questions?

IOStream Class Hierarchy



Insertion and Extraction – writing your own

- Recall how operators work:

```
Point P (0,0);
 cout << P;
```

is the same as:

```
cout.operator<< (P);
```

however, who can predict the need for an operator for each class?

Insertion and Extraction – writing your own

- Instead, the compiler looks for:
  - `operator<<( cout, P );`
- Note that this can't be a member of Point since `cout` is the first argument.
- Take home message: `operator<<` and `operator>>` are defined outside of a class.

Writing an inserter (`operator<<`)

1. The first argument should be a reference to an `ostream`. The second argument should be a constant reference to your class.
2. The function should return a reference to an `ostream` so that insertions can be chained.
3. The body of the function should perform whatever output is appropriate for your class, but nothing more!
4. If you need to access the private data members of your class directly, then your class must declare this function to be a `friend`

Writing an inserter (`operator<<`)

```
friend ostream &operator<<( ostream &out, const
  Point &p )
{
   out << '(' << p.x << ',' << p.y << ')';
   return out;
}
```

Output:

```
(0,0)
```

Writing an extractor (`operator>>`)

- Like inserter except:
  - Must handle possible errors
  - Argument cannot be const reference.
  - Almost surely will have to declare as a `friend`.

Writing an extractor (`operator>>`)

```
friend istream &operator>> (istream &in, Point &p)
{
     char c;
     int ok = FALSE;

     in >> c;
     if (c == ‘(‘)) {
         in >> p.x >> c;
         if (c == ‘,’) {
             in >> p.y >> c;
             if (c == ‘)’) ok = TRUE;
         }
      }
      if (!ok) in.clear (in.rdstate() | ios::failbit

     return in;
}
```

Writing an extractor (`operator>>`)

- Things to note:
  - Checks to see if in same format as output.
  - Stops reading as soon as an error is found.
  - Sets `failbit` if format is not correct.

- Questions?

Demo code

- Handouts

- Other examples linked on Web site.

Summary

- fstream – I/O to/from files
- sstream – I/O to/from strings (char arrays)

- Rolling your own extractors and inserters.

- Questions?
  - Have a nice weekend.