CSCI 135
I/O (Standard and File)

Basic Language Constructs

Recall: each statement in an imperative language updates the value of some variable.

⇒ Classes of Constructs:

- Declaration of variables: How does the variable map onto memory?
- Updating variable: How do we update the variable, and what do we update it with?
- I/O: How do we input and output data into the system?
- Control: How do we control which statement gets executed next?
- Modularity and Object Orientation: How do we organize the program to enable proper software engineering practices?
- Comments: Used to describe code; ignored by compiler, but code unmaintainable without good comments!
  C/C++: on line beginning with // or surrounded by /*, */

I/O Statements (C++ only)

cout is an instruction that receives expressions, evaluates them, and sends them to the console output. Ex:

```
cout << "Three times 2 is " << 3*2 << endl;
```

1. Evaluate string "Three times 2 is " and output
2. Evaluate int 3*2 (to 6) and output
3. Evaluate endl and output newline character

⚠Good idea to always output endl (or the *char* '\n') at end since some platforms may not output a line before receiving the newline

I/O Statements (C++ only)

`cout` is an instruction that receives expressions, evaluates them, and sends them to the console output. Ex:

```
cout << "Three times 2 is " << 3*2 << endl;
```

1. Evaluate string "Three times 2 is " and output
2. Evaluate int 3*2 (to 6) and output
3. Evaluate endl and output newline character

⚠ Good idea to always output endl (or the *char* '\n') at end since some platforms may not output a line before receiving the newline

---

`cin` is an instruction that receives data from console input (*i.e.*, keyboard) and sends it to a variable. Ex:

```
string name,surname; int age; cin >> name >> surname >> age;
```

⚠ C++ uses whitespace to delimit name and surname

⇒ Can't use cin to input strings with spaces.

(!) `getline(cin,name)` will allow name to contain spaces

`cerr` is used to output error messages.

Libraries

One problem: cin/cout/cerr are in the input/output library, but we haven't said where to find them!
⇒ Use `#include <iostream>`

- Makes functions in `iostream` library visible to your program
- Called "preprocessor directive" (executed before compiler)
- C++ (and C) have many libraries (*e.g.*, I/O, math, strings, ...)

⚠There are slight naming and syntax inconsistencies across compilers (especially older ones).

Namespaces

A **namespace** is a collection of name definitions. The `std` namespace is the set of all definitions.
So, we precede the program with:

```
#include <iostream> using namespace std;
```

to include `cin, cout, ...`
What if we only want to use some parts of library?

```
#include <iostream> using std::cin;
```

would include `cin` but not `cout, ....`

Return Value of cin

`cin` is actually a function that has an argument (x in below example) and returns a boolean that is false iff at end of file. (EOF character is ^D in linux)
⇒ useful for reading until end of input.

```
#include <iostream>
using namespace std;
while (cin >> x) {          exit loop on EOF
  do something with next x input;
  };
do something after all input received
```

⚠EOF character differs across OSs

Streams

A possibly infinite sequence of arbitrary data, [intended to be] accessed linearly.
$\Rightarrow$ can only access next element in stream!
Typical Uses:

- Input from keyboard, output to screen
- Input from file, output to file
- Inter-process/task communication (in other languages)

Stream Usage (C++ only)

Standard IO | **File I/O**

1. Include appropriate libraries:

```
#include <iostream>;
```

```
#include <fstream>;
```

2. Declare stream variable/object (implicitly named cin for keyboard)

```
ifstream istr;
```

3. Establish path connecting input (keyboard, file, etc.) to stream (default for standard; see next slide for file)
4. Read from stream to variable

```
cin >> x;
```

```
istr >> x;
```

(and similarly for output, but using type `ofstream`)

Functions in fstream Library

Recall:

- `open(filename)`: a file **must** be opened before reading/writing.
- <<, >>, `getline`: Used similarly to cout/cin
- `close()`: a file should be closed after done reading/writing (operating system *should* close it after program terminates, but explicit close recommended)

⚠Program can only access streams; file name is invisible and needs to be linked to the stream

Example - Average of Data

```
#include <fstream>
#include <iostream>          still want cin/cout
  using namespace std;
main() {
  ifstream instr;            instr is stream name
  instr.open("myfile");      'connects' file, stream
  int x, sum=0, n=0;
  while (instr >> x) {
    cout << x << endl;
    sum += x;
    n++;
    };
  instr.close();
  cout << "\nAverage of input is: "
       << (double) sum / (double) n << endl;
  instr.open("anotherfile");
  ...                        same stream name, another file!
}
```

Example (output to file)

Print even and odd numbers in input sequence to 2 different files

```
ofstream ostr1,ostr2;
ostr1.open("evens");
ostr2.open("odds");
while (cin >> v) {
  if (v%2 == 0)
    ostr1 << v << endl;
  else
    ostr2 << v << endl;
  };
ostr1.close(); ostr2.close();
```

Example: Using Getline

Given file "data" containing <name,emplid> pairs, output john's emplid.

```
ifstream istr;
string line, name;
int emplid;  bool done = false;
istr.open("data");
while (!done && !(istr.eof())) {    new function: eof
  getline(istr,line);
  insert code to parse line into name, emplid here
  if (name == "john") {
    cout << "John's emplid is: "
         << emplid << endl;
    done = true;
  }};
if (done == false)
  cout << "John is not in file" << endl;
```

File Open Complications

File opens can fail!

- Input file doesn't exist
- No write permission to output file (or read permission for input file)
- Various other reasons

`fstream.fail()` returns true iff open failed.

```
istr.open("myfile");
if (istr.fail())
{
  cout << "File open failed\n";
  exit(1);
}
// Assert: istr is open for input
...
```

Other fstream Features

Just FYI (partial list)

- Peek ahead in stream
- Checking if at eof
- Character read/write (get, put)
- Formatted I/O (width of fields, precision of numbers, etc.)
- Appending (vs. overwriting) file when writing

Finding the Second Largest Element

Problem: Given a file of distinct ints, determine the second largest one.

Approaches:

1. Two pass:
   1. Scan through file, to determine largest number, m
   2. Scan through file (again), to determine largest number $\neq$ m.
2. One pass: Keep track of two largest elements, making just one pass through file.

Two Pass Algorithm for Second Largest

```
ifstream istr;
int max = INT_MIN;      defined in climits.h
istr.open("myfile");
if (!istr.fail()) {
  while (istr >> elem)
    if (elem>max) max=elem;
  }
istr.close();
int max2 = INT_MIN;
istr.open("myfile");   reusing same stream
if (!istr.fail()) {
  while (istr >> elem)
    if ((elem>max2) && (elem != max)) max2=elem;
  }
istr.close();
cout << "Second largest element: "
     << max2 << endl;
```

One Pass Algorithm For Second Largest

```
int max=INT_MIN, max2=INT_MIN;
istr.open("myfile");
if (!istr.fail()) {
  while (istr >> elem)  >max2, >max
    if (elem>max) {
      max2=max;
      max=elem;
      };
    else if (elem>max2) max2=elem;  >max2, <max
  }
istr.close();
cout << "Second largest element: "
     << max2 << endl;
```

General Guidelines For Stream Processing

- Determine what information you need to keep around for each element in stream.
  Ex: sum of entries, largest entry
- Is it possible to keep all needed information?
  yes $\Rightarrow$ one pass algorithm suffices
  no $\Rightarrow$ need multi-pass algorithm (and resulting overhead)
- Answer to above depends on available data structures.
- Don't forget to handle errors.
- Bigger picture: can you redefine spec so that file stream is structured to allow one-pass algorithm?

Above applies to all linearly structured data (though overhead might matter less or not at all for memory-resident data).

A Few Other File IO Issues

? How do I open a file with a name determined at run time?

1. C++ pre 2011: only possible with C-strings (not C++ strings)
2. C++ 2011: open can take a string as argument – compile with `-std=c++11` (or later) option

A Few Other File IO Issues

? How do I open a file with a name determined at run time?

1. C++ pre 2011: only possible with C-strings (not C++ strings)
2. C++ 2011: open can take a string as argument – compile with `-std=c++11` (or later) option

---

? How do I handle type errors in input?

1. Input as string and do your own parsing
2. Check return value of >> and reinput data (after calling `cin.clear()` to clear 'bad input flag' and `cin.ignore(...)` to discard input)

Exercises

1. Given: a file called people with ⟨name,age⟩ pairs (assume file has no errors, and ages are distinct).
   Output:
   - Number of people
   - Name of oldest person
   - Average age
   - Names of second and third oldest people
2. Given: 2 sorted files with data as above
   Output:
   - Median person of each file
   - Create third file with contents of input files (in sorted order).