CS 106X, Lecture 3
I/O Streams and Grids

reading:

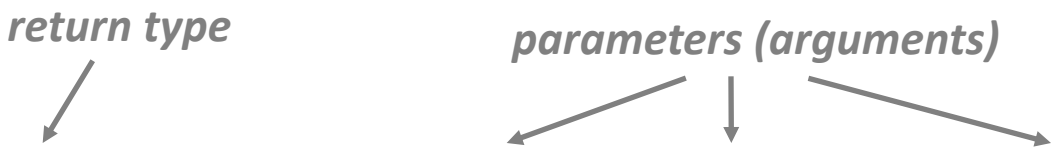
Programming Abstractions in C++, Chapter 4, 5.1

Plan For Today

- Recap: C++ Functions and Strings
- C++ Streams
  - What is a stream?
  - File streams
  - String and URL streams
- Announcements
- Stanford Library Grid

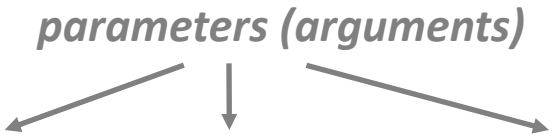
Defining a function (2.3)

- A C++ **function** is like a Java **method**.




```
type functionName(type name, type name, ..., type name) {
    statement;
    statement;
    ...
    statement;
    return expression;   // if return type is not void
}
```

- Calling a function:




```
functionName(value, value, ..., value);
```

Default parameters

- You can make a parameter optional by supplying a *default value*:
  - All parameters with default values must appear last in the list.

```
// Prints a line of characters of the given width.
void printLine(int width = 10, char letter = '*') {
    for (int i = 0; i < width; i++) {
        cout << letter;
    }
     cout << endl;
}

...

printLine(7, '?');    // ???????
printLine(5);         // *****
printLine();          // **********
```

Function prototypes (1.4)

type name(*type name*, *type name*, ..., *type name*);

- Declare the function (without writing its body) at top of program.

```
double circleArea(double r);        // function prototype

int main() {
    double a = circleArea(2.5);     // call the function
    return 0;
}

double circleArea(double r) {
    ...
}
```

With prototype, only declare default values in prototype.

Pass by Value

```
void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 17;
    int y = 35;
    swap(x, y);
    cout << x << "," << y << endl;    // 17,35
    return 0;
}
```

By default, C++ parameters are copies.

Pass by Reference

```
void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 17;
    int y = 35;
    swap(x, y);
    cout << x << "," << y << endl;    // 35,17
    return 0;
}
```

Use “&” to pass the same version.

Strings (3.1)

```
#include <string>
...
string s = "hello";
```

- A string is a (possibly empty) sequence of characters.
- Strings are *mutable* (can be changed) in C++.
- There are two types of strings in C++. :-/

Operators (3.2)

- **Concatenate** using + or += :

```
string s1 = "Dai";
s1 += "sy";                          // "Daisy"
```

- **Compare** using relational operators (ASCII ordering):

```
string s2 = "Nick";              // == != < <= > >=
if (s1 < s2 && s2 != "Joe") {    // true
    ...
}
```

- Strings are **mutable** and can be changed (!):

```
s2.append(" Troccoli");              // "Nick Troccoli"
s2.erase(6, 7);                  // "Nick T"
s2[2] = '<';                     // "Ni<k T"
```

C vs. C++ strings (3.5)

- C++ has two kinds of strings:
  - **C strings** (`char` arrays) and **C++ strings** (`string` objects)

- A string literal such as `"hi there"` is a C string.
  - C strings don't include any methods/behavior shown previously.
    - No member functions like `length`, `find`, or operators.

- Converting between the two types:
  - `string("`***`text`***`")` C string to C++ string
  - ***`string`***`.c_str()` C++ string to C string

NEW: Const parameters

- What if you want to avoid copying a large variable but don't want to change it?
- Use the **const** keyword to indicate that the parameter won't be changed
  - Usually used with strings and collections
  - Passing in a non-variable (e.g. `printString("hello")`) **does** work

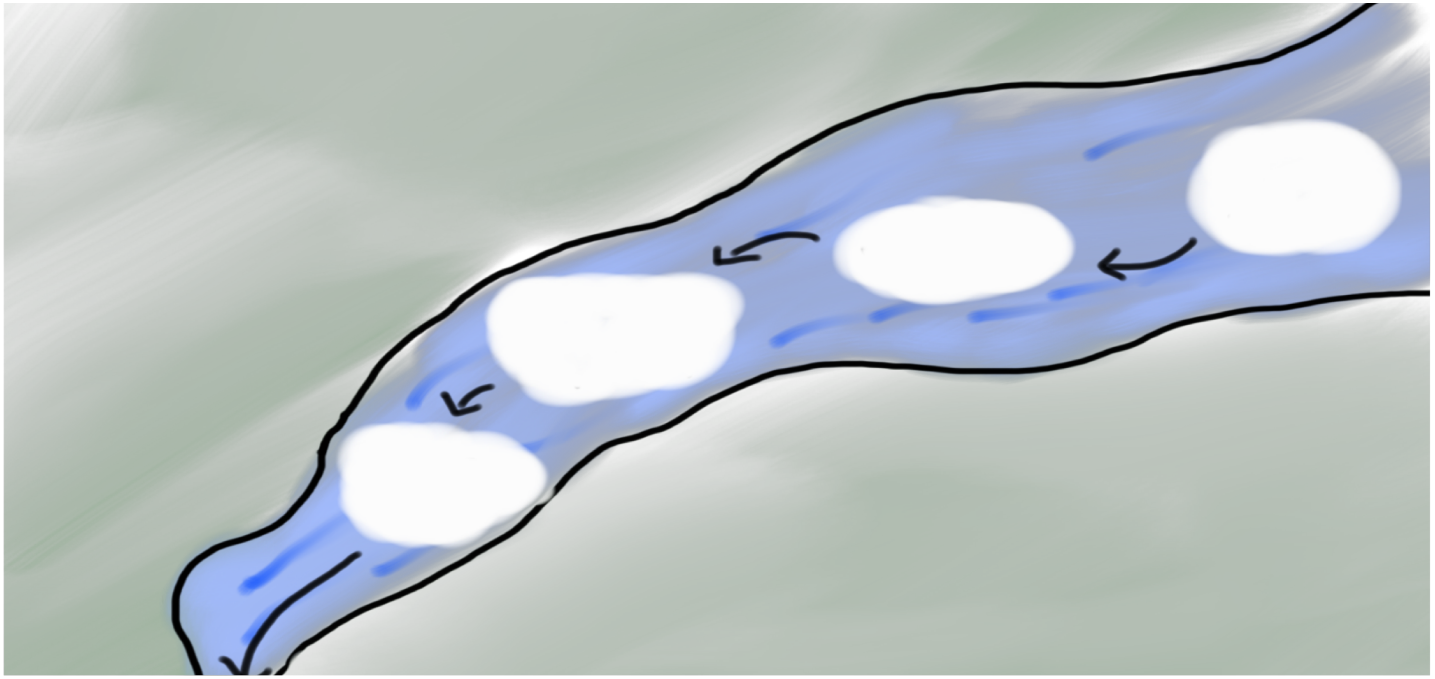
```
void printString(const string& str) {
    cout << "I will print this string" << endl;
    cout << str << endl;
}

int main() {
    printString("This could be a really really long
                string");
}
```

Plan For Today

- Recap: C++ Functions and Strings
- C++ Streams
  - What is a stream?
  - File streams
  - String and URL streams
- Announcements
- Stanford Library Grid

What is a stream?

- A stream is a type of variable that you can insert into and remove from using special *stream operations* like <<, >>, getline, get and put.



- There are *input streams* and *output* streams.

What is a stream?

- An *input stream* lets you get data from a source (like user input, a file, a webpage, etc.) and read it in your program.
- An *output stream* lets you take data *from* your program and output it to a source (like the console, a file, etc.).

ifstream members (4.3)

Member function	Description
`f.clear();`	resets stream's error state, if any
`f.close();`	stops reading file
`f.eof()`	returns `true` if stream is *past* end-of-file (EOF)
`f.fail()`	returns `true` if the last read call failed (e.g. EOF)
`f.get()`	reads and returns one character
`f.open("filename");` `f.open(s.c_str());`	opens file represented by given C string (may need to write `.c_str()` if a C++ string is passed)
`f.unget(ch)`	un-reads one character
`f >> var`	reads data from input file into a variable (like `cin`); reads one whitespace-separated token at a time
`getline(f&, s&)`	reads line of input into a string by reference; returns a true/false indicator of success

object if stream here are the different members that it has you can open a file

Operator >>

- Reads / converts next whitespace-separated token of input.
  - If unsuccessful, sets stream into *fail* state, and returns a "falsey" value.

```
ifstream input;
input.open("data.txt");
string word;
input >> word;    // "Marty"
input >> word;    // "is"
int age;
input >> age;     // 12
input >> word;    // "'years'"
input >> word;    // "old!"

if (input >> word) {    // false
    cout << "successful!" << endl;
}
```

data.txt:

```
Marty
 is   12
  'years'
  old!
```

Reading In A File

1. Open the file for reading
2. Read the file, one chunk at a time
3. Close the file

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

Step one:

Open the file for reading.

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
infile.open("filename.txt");
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
infile.open("filename.txt");
```

```
#include <fstream>;   // for file streams
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
string myFilename = ...;
infile.open(myFilename.c_str());
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
// This error-checks the filename (unlike previous)
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");
```

```
#include <fstream>;   // for file streams
#include "filelib.h"; // for promptUserForFile
```

Reading In A File

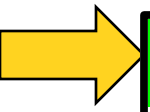
```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");
```

Step Two:

Read the file, one line at a time.

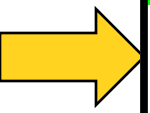
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// The animal I really dig
string line1;
getline(infile, line1);
```

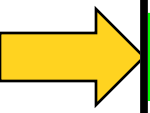
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// The animal I really dig
string line1;
getline(infile, line1);
```

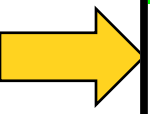
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// The animal I really dig
string line1, line2;
getline(infile, line1);
// Above all others is the pig.
getline(infile, line2);
```

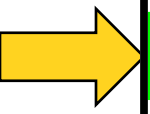
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// The animal I really dig
string line1, line2;
getline(infile, line1);
// Above all others is the pig.
getline(infile, line2);
```

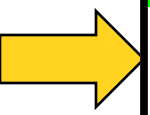
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");


...
// Pigs are noble.  Pigs are clever,
string line3;
getline(infile, line3);
```

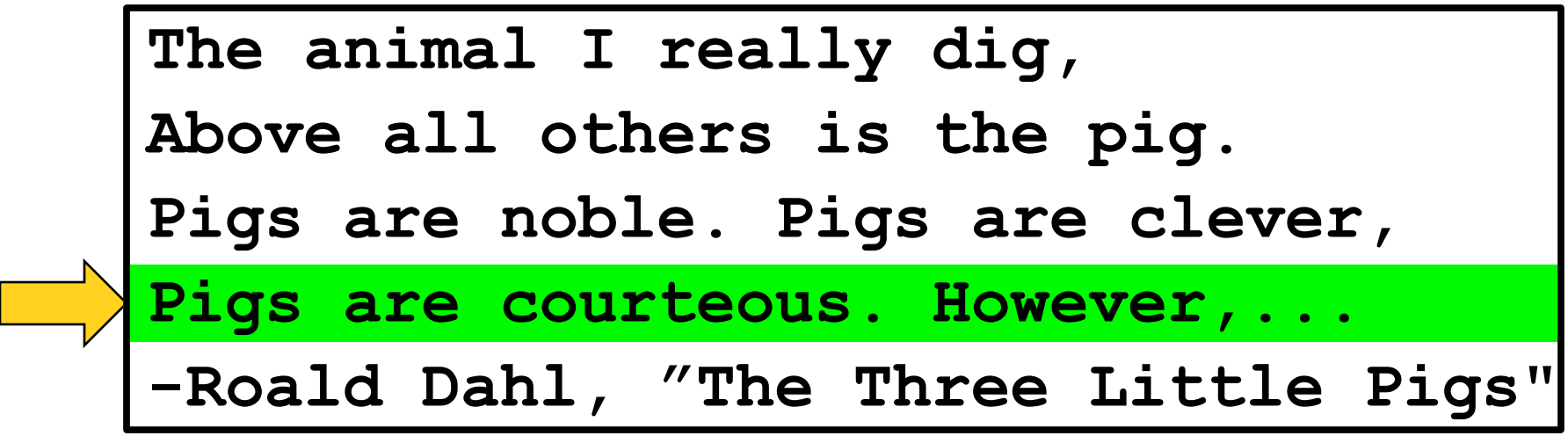
Reading In A File



```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

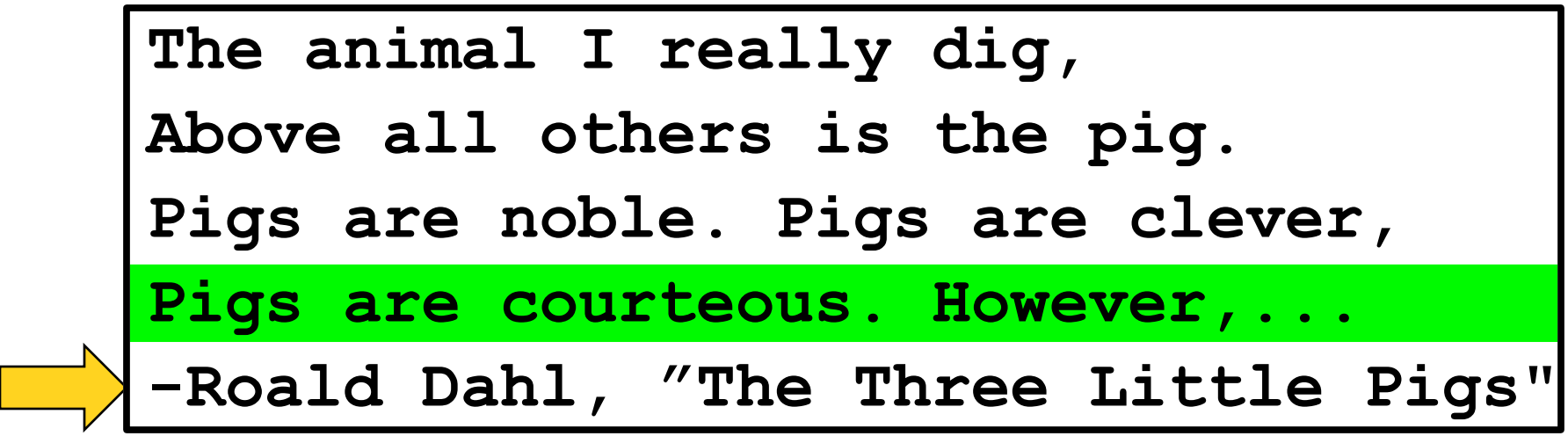
```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

...
// Pigs are noble.  Pigs are clever,
string line3;
getline(infile, line3);
```

Reading In A File




```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");


...
// Pigs are courteous.  However,...
string line4;
getline(infile, line4);
```

Reading In A File




```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");


...
// Pigs are courteous.  However,...
string line4;
getline(infile, line4);
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

...
// -Roald Dahl, "The Three Little Pigs"
string line5;
getline(infile, line5);
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, ”The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

...
// -Roald Dahl, “The Three Little Pigs”
string line5;
getline(infile, line5);
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, ”The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");


...
// -Roald Dahl, “The Three Little Pigs”
string line5;
getline(infile, line5);
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// print all lines in the file
string line;
while (getline(infile, line)) {
    cout << line << endl;
}
```

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

// print all lines in the file
string line;
while (getline(infile, line)) {
    cout << line << endl;
}
```

Step Three: close the file.

Reading In A File

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

string line;
while (getline(infile, line)) {
    cout << line << endl;
}
infile.close();
```

One Character At A Time

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

char ch;
while(infile.get(ch)) {
    // do something with ch
}
infile.close();
```

One Word At A Time

```
The animal I really dig,
Above all others is the pig.
Pigs are noble. Pigs are clever,
Pigs are courteous. However,...
-Roald Dahl, "The Three Little Pigs"
```

```
ifstream infile;
promptUserForFile(infile, "Enter a file name: ");

string word;
while(infile >> word) {
    // do something with word
}
infile.close();
```

Writing To A File

```
// Open the file for writing
ofstream outfile;
promptUserForFile(outfile, "Enter a file name: ");

// Write to the file
string word = "my cool string";
int x = 3;
outfile << word << x;

// Close the file
outfile.close();
```

That Looks Familiar…

- If file-writing syntax seems eerily similar to printing to the console, that's because it is!
  - `cin` is an `istream`; `cout` is an `ostream`
- We can take advantage of this in our code

```
void outputUserData(ostream& outputStream, string name, int
score, double health) {
    outputStream << name << endl;
    outputStream << score << endl;
    ...
}
```

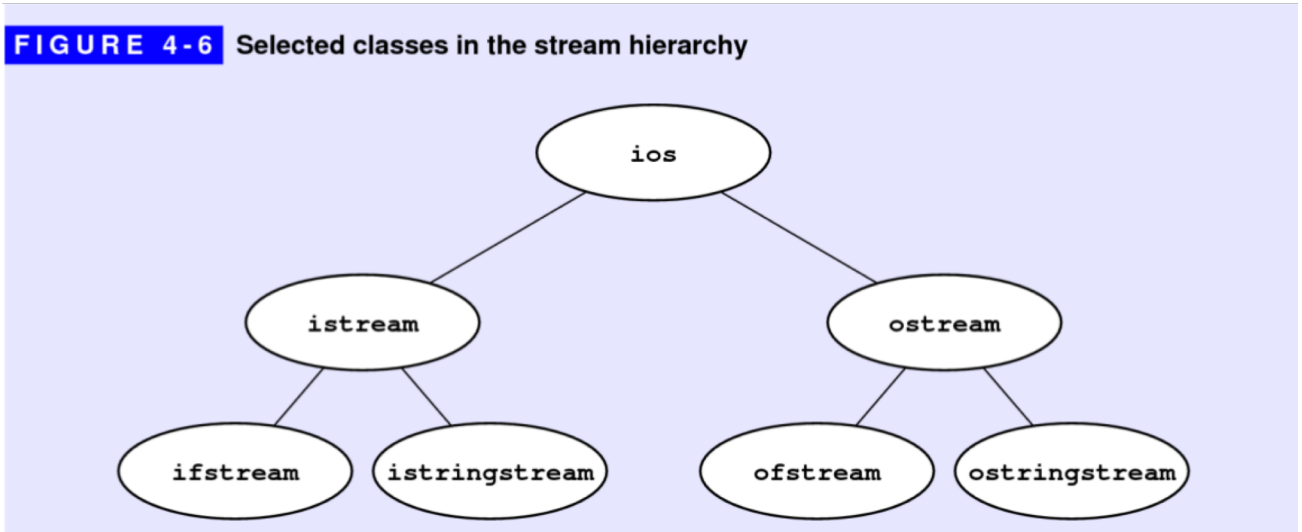
Generic Streams

```
void outputUserData(ostream& outputStream, string name,
int score, double health) {
    outputStream << name << endl;
    outputStream << score << endl;
    ...
}

int main() {
    ...
    outputUserData(cout, name, score, health);
    if (getYesOrNo("Save to file? ")) {
        outputUserData(outfile, name, score, health);
    }
}
```

Reading files (4.3)

```
#include <fstream>
```

- `ifstream`, `ofstream` classes for input/output files
  - *inheritance hierarchy,* based on parents named `istream`, `ostream`
- `cin` is an `ifstream`; `cout` is an `ofstream`



http://www.cplusplus.com/reference/istream/istream/

Plan For Today

- Recap: C++ Functions and Strings
- C++ Streams
  - What is a stream?
  - File streams
  - String and URL streams
- Announcements
- Stanford Library Grid

istringstream

```
#include <sstream>
```

- An **istringstream** lets you tokenize a string.

```
// read specific word tokens from a string
istringstream input("Jenny Smith 8675309");
string first, last;
int phone;
input >> first >> last;    // first="Jenny", last="Smith"
input >> phone;            // 8675309

// read all tokens from a string
istringstream input2("To be or not to be");
string word;
while (input2 >> word) {
    cout << word << endl;   // To \n be \n or \n not \n ...
}
```

ostringstream

```
#include <sstream>
```

- An **ostringstream** lets you write output into a string buffer.
  - Use the **str** method to extract the string that was built.

```
// produce a formatted string of output
int age = 42, iq = 95;
ostringstream output;
output << "Zoidberg's age is " << age << endl;
output << " and his IQ is " << iq << "!" << endl;
string result = output.str();

// result = "Zoidberg's age is 42\nand his IQ is 95!\n"
```

iurlstream

```
#include "urlstream.h"
```

- An **`iurlstream`** lets you read input from a web URL.
  - Use just like an ifstream when reading from a file

```
// download the text of a webpage
string url = getLine("Enter a URL: ");
iurlstream urlStream(url);     // or call .open(url) later

string line;
while (getline(urlStream, line)) {
    cout << line << endl;
}

urlStream.close();
```

iurlstream

```
#include "urlstream.h"
```

- An **`iurlstream`** lets you read input from a web URL.
  - Use just like an ifstream when reading from a file

```
// download the text of a webpage
string url = getLine("Enter a URL: ");
iurlstream urlStream(url);

int error = urlStream.getErrorCode();
// if this is non-zero, an error occurred
// try calling .open with a new URL
```

Stream Parameters

If a function takes a stream as a parameter, it MUST be passed by reference! Otherwise you will get a compiler error.

```
void outputUserData(ostream& outputStream, string name, int
score, double health) {
    outputStream << name << endl;
    outputStream << score << endl;
    ...
}
```

Stanford library (4.3)

```
#include "filelib.h"
```

Function name	Description
`createDirectory(`***name***`);`	make a new directory with the given name
`deleteFile(`***name***`);`	erase a file from the disk
`fileExists(`***name***`)`	return `true` if the given file exists on disk
`getCurrentDirectory()`	return C++ program's directory as a string
`isDirectory(`***name***`),` `isFile(`***name***`)`	return `true` based on type of file path
`openFile(`***ifstream***`&, `***name***`);`	convenience for opening file by C++ string
`promptUserForFile(` `  `***ifstream***`&, `***prompt***`)`	repeatedly prompt for existing file's name
`readEntireFile(`***ifstream***`&, `***lines***`&);`	read file data into a collection of lines
`renameFile(`***oldname***`, `***newname***`);`	change a file's name

Demo – Song Lyrics

Plan For Today

- Recap: C++ Functions and Strings
- C++ Streams
  - What is a stream?
  - File streams
  - String and URL streams
- Announcements
- Stanford Library Grid

Announcements

- Sign up for section!
- LaIR/CLaIR hours start this Sunday at 7PM
- Midterm and Final Exams
  - Alternate midterm only for academic or university conflicts, or OAE
  - No alternate final exams, except for OAE accommodations
- Assignment 1

CURIS Poster Session

TODAY **9/28 3-5PM on Packard Lawn**

See awesome summer research projects and get a taste of what CS research is like!

Plan For Today

- Recap: C++ Functions and Strings
- C++ Streams
  - What is a stream?
  - File streams
  - String and URL streams
- Announcements
- Stanford Library Grid

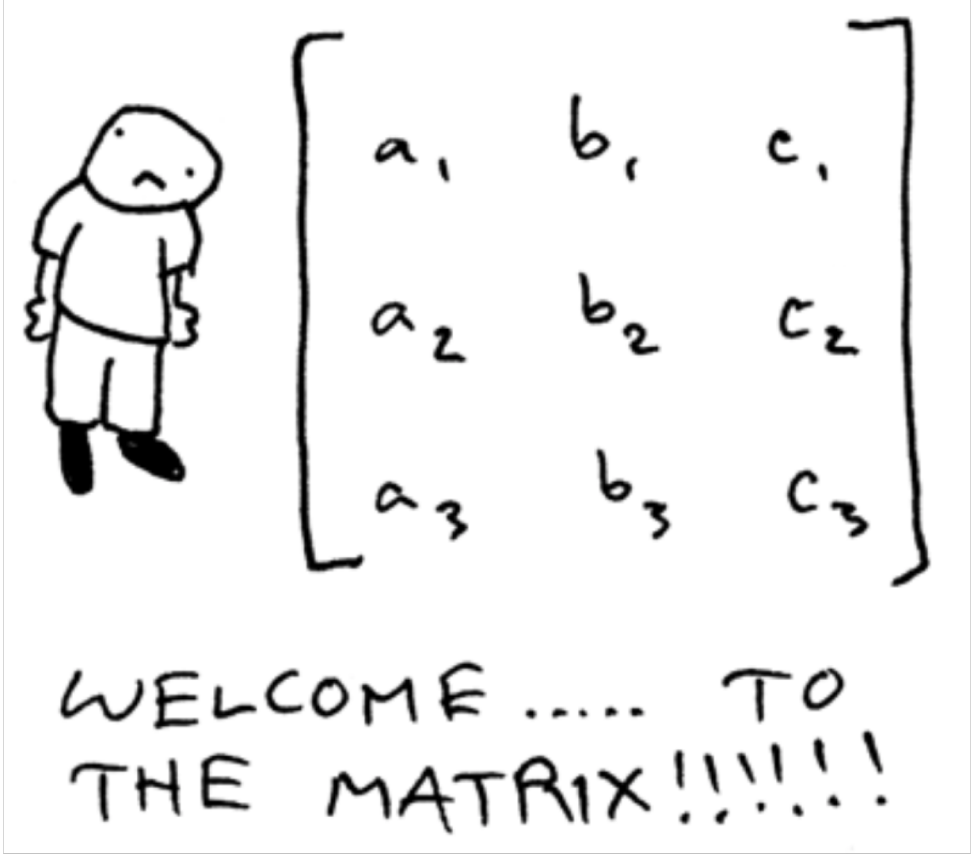
STL vs. Stanford

- **collection**: an object that stores data; a.k.a. "data structure"
  - the objects stored are called **elements**.

- **Standard Template Library (STL)**:
  C++ built in standard library of collections.
  - `vector`, `map`, `list`, …
  - Powerful but somewhat hard to use.

- **Stanford C++ library (SPL)**:
  Custom library of collections made for use in CS 106B/X.
  - `Vector`, `Grid`, `Stack`, `Queue`, `Set`, `Map`, …
  - Similar to STL, but simpler interface and error messages.

Grid


$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$
WELCOME ..... TO
THE MATRIX!!!!!!

Grid (5.1)

```
#include "grid.h"
```

- An advanced **2D array** – think spreadsheets, game boards
- must specify element type in < > (a ***template*** or a *type parameter*)

```
// constructing a Grid
Grid<int> matrix(3, 4);
matrix[0][0] = 75;
...

// or specify elements in {}
Grid<int> matrix = {
    {75, 61, 83, 71},
    {94, 89, 98, 100},
    {63, 54, 51, 49}
};
```

row \ column	0	1	2	3
0	75	61	83	71
1	94	89	98	100
2	63	54	51	49

Grid (5.1)

```
#include "grid.h"
```

- An advanced **2D array** – think spreadsheets, game boards
- must specify element type in < > (a ***template*** or a *type parameter*)

```
// constructing a Grid
Grid<int> matrix(3, 4);
matrix[0][0] = 75;

cout << matrix[10][15] << endl;
```

row \ column	0	1	2	3
0	75	61	83	71
1	94	89	98	100
2	63	54	51	49

```
***
*** STANFORD C++ LIBRARY
*** An ErrorException occurred during program execution:
*** Grid::operator [][]: (3, 2) is outside of valid range [(0, 0)..(2, 1)]
***
```

Grid members (5.1)*

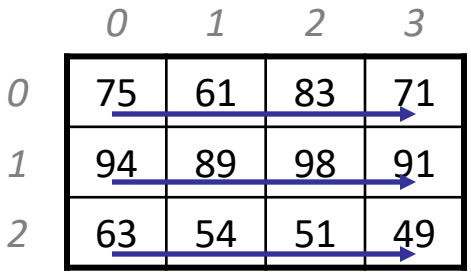
`Grid<type> name(r, c);` `Grid<type> name;`	create grid with given number of rows/cols; empty 0x0 grid if omitted
`g[r][c]` or `g.get(r, c)`	returns value at given row/col
`g.fill(value);`	set every cell to store the given value
`g.inBounds(r, c)`	returns `true` if given position is in the grid
`g.numCols()` or `g.width()`	returns number of columns
`g.numRows()` or `g.height()`	returns number of rows
`g.resize(nRows, nCols);`	resizes grid to new size, discarding old contents
`g[r][c] = value;` or `g.set(r, c, value);`	stores value at given row/col
`g.toString()`	returns a string representation of the grid such as `"{{3, 42}, {-7, 1}, {5, 19}}"`
`ostr << g`	prints, e.g. `{{3, 42}, {-7, 1}, {5, 19}}`

** (a partial list; see http://stanford.edu/~stepp/cppdoc/)*

Looping over a grid

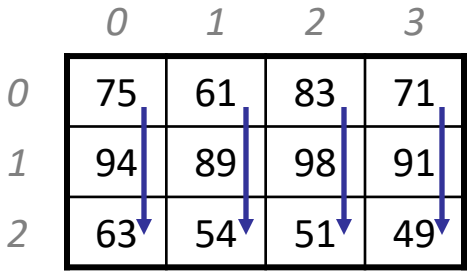
- Row-major order:

```
for (int r = 0; r < grid.numRows(); r++) {
    for (int c = 0; c < grid.numCols(); c++) {
        do something with grid[r][c];
    }
}

// "for-each" loop  (also row-major)
for (int value : grid) {
    do something with value;
}
```



- Column-major order:

```
for (int c = 0; c < grid.numCols(); c++) {
    for (int r = 0; r < grid.numRows(); r++) {
        do something with grid[r][c];
    }
}
```



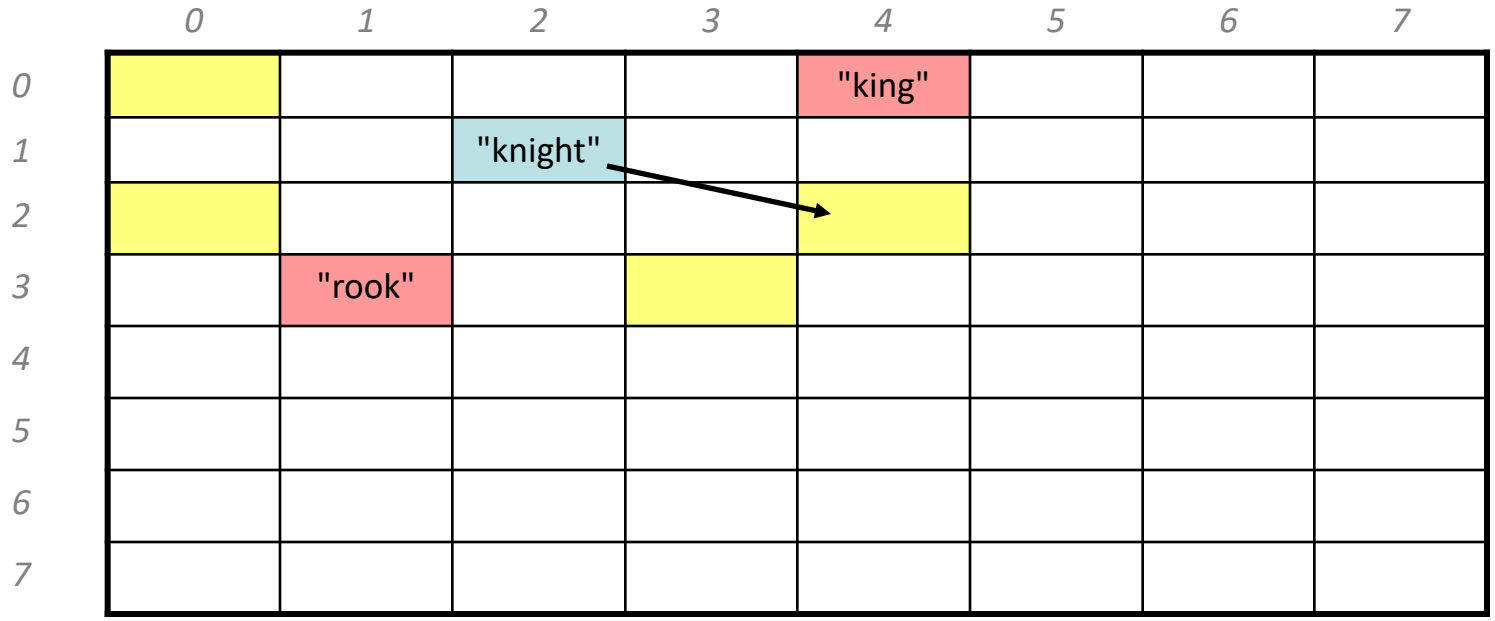
Grid as parameter

- When a `Grid` is passed by value, C++ makes a copy of its contents.
  - Copying is slow; you should usually **pass by reference** with **&**
  - If the code won't modify the grid, also pass it as **`const`**

```
// Which one is best?
A) int computeSum(Grid<int> g) {
B) int computeSum(Grid<int>& g) {
C) int computeSum(const Grid<int> g) {
D) int computeSum(const Grid<int>& g) {

// Which one is best?
A) void invert(Grid<double> matrix) {
B) void invert(Grid<double>& matrix) {
C) void invert(const Grid<double> matrix) {
D) void invert(const Grid<double>& matrix) {
```

Grid exercise

- Write a function **`knightCanMove`** that accepts a grid and two row/column pairs (*r1*, *c1*), (*r2*, *c2*) as parameters, and returns `true` if there is a knight at chess board square (*r1*, *c1*) and he can legally move to empty square (*r2*, *c2*).
  - Recall that a knight makes an "L" shaped move, going 2 squares in one dimension and 1 square in the other.
  - `knightCanMove(board, 1, 2, 2, 4)` returns `true`



Grid exercise solution

```
bool knightCanMove(Grid<string>& board, int r1, int c1,
                                        int r2, int c2) {
    if (!board.inBounds(r1, c1) || !board.inBounds(r2, c2)) {
        return false;
    }
    if (board[r1][c1] != "knight" || board[r2][c2] != "") {
        return false;
    }
    int dr = abs(r1 - r2);
    int dc = abs(c1 - c2);
    if (!((dr == 1 && dc == 2) || (dr == 2 && dc == 1))) {
        return false;
    }
    return true;
}
```

Grid solution 2

```
bool knightCanMove(const Grid<string>& board, int r1, int c1,
                                              int r2, int c2) {
    int rowChange = abs(r1 - r2), colChange = abs(c1 - c2);
    return board.inBounds(r1, c1) && board.inBounds(r2, c2)
            && board[r1][c1] == "knight" && board[r2][c2] == ""
            && ((rowChange == 1 && colChange == 2) ||
                (rowChange == 2 && colChange == 1));
}
```