

A Problem

Suppose each line of my input consists of a student's exam scores. The final exam is optional, so each line may have either 2 or 3 scores:

mickey mouse 78 95

donald duck 85 93 63

winnie pooh 68 58 93

Problem Statement: Read the exam scores for each student and print their average

Why cin fails

- A possible code fragment to read a student's name and exam scores

```
cin >> fname >> lname;
```

```
cin >> exam1 >> exam2 >> final;
```

- What happens if there are only two scores?

```
mickey mouse 78 95
```

```
donald duck 85 93 63
```

A Solution

1. Read a full line of input (getline)
2. Break the line of input into name and exam fields (string streams)

getline() reads an entire line

- Syntax:
 - From cin: `getline(cin, string s)`
 - From a file: `getline ifstream file, string s)`
- Example
 - `string line;`
 - `getline(cin, line); // read a line from the console`

 - `ifstream fromFile;`
 - `fromFile.open("data.txt");`
 - `getline(fromFile, line); // reads a line from a file`

string streams

- string streams allow you to
 - extract fields from a string using the input operator >>
 - create a formatted string using the output operator <<
- string streams behave just like cin and cout: they support << and >>, and you can test them for eof

including string streams

- `#include<sstream>`: includes a string stream in your program

input stringstream

- `istringstream buffer;` -- declares an “input” stringstream named `buffer`
- `str(string s)` member function: assigns the string to `istringstream` which you want to break into fields
- `>>`: reads the next field from the string and converts it to the appropriate value
- `clear()`: clears the current string out of the string stream object so that a new one can be assigned to it via the `str()` function.

the average exam problem

```
int exams[3], count = 0; double sum = 0;
istringstream buffer;
string fname, lname, line;

getline(cin, line);           // read a student
buffer.str(line);            // assign the line to buffer
buffer >> fname >> lname;    // extract first and last names
while (buffer >> exams[count]) { // extract exams until end of line
    count++;
    sum += exams[count];
}
cout << "average = " << sum / count << endl;
```


the average exam problem— processing all lines of the file

```
int exams[3], count = 0; double sum = 0;
istream buffer;
string fname, lname, line;

while (getline(cin, line)) {           // read a student
    sum = 0;
    count = 0;
    buffer.clear();
    buffer.str(line);                 // assign the line to buffer
    buffer >> fname >> lname;         // extract first and last names
    while (buffer >> exams[count]) { // extract exams until end of line
        count++;
        sum += exams[count];
    }
    cout << "average = " << sum / count << endl;
}
```

output stringstream

- `ostringstream line;` -- declares an “output” stringstream that allows you to create a formatted string
- `<<`: allows you to write variables into the string
- `str()`: returns the formatted string
- `clear()`: prepares the `ostringstream` object for another formatted string. You must also call `.str(“”)` to clear the formatted string before starting on a new formatted string.

ostream example

- Write a function that takes the integer variables hours, minutes, and seconds as parameters, and creates a string formatted as “hh:mm:ss hours”
string formatTime(int hours, int minutes, int seconds) {
 ostream time;
 time << hours << ":" << minutes << ":"
 << seconds << " hours";
 return time.str();
}