

# More on File I/O

## Strings in C++

CS 16: Solving Problems with Computers I  
Lecture #10

Ziad Matni  
Dept. of Computer Science, UCSB

# Announcements

---

- Heads-Up: Midterm #2 is on Tuesday 11/14
- Found evidence that students are COPYING from each other.
  - That's a bad thing...
  - It's time to remind everyone of the **plagiarism policy**

(Students: read the **syllabus**, if you forgot it!!

Also read the UCSB webpage on **Academic Integrity**

<http://judicialaffairs.sa.ucsb.edu/academicintegrity.aspx> )

**STUDENT X**

**← 96% Match! →**

**STUDENT Y**

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

const double loan_interest(0.06), loan_reduct(0.03), income_rate(0.35);
double cal_interest(double home_price, double down_payment);
//calculate the interest
double cal_cost(double interest, double home_price, double down_payment);
//calculate the after tax cost

main(){
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    double home_price(1.0),down_payment;
    double interest, aftax_cost;
    while (home_price != 0)
    {
        cout<<"Enter the home price (or zero to quit):\n";
        cin>>home_price;
        if (home_price != 0)
        {
            cout<<"Enter the down payment:\n";
            cin>>down_payment;
            interest=cal_interest(home_price, down_payment);
            aftax_cost=cal_cost(interest, home_price, down_payment);
            cout<<"The after-tax cost is $"<<aftax_cost<<" annually.\n";
        }
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
const double loan_interest(0.06), loan_reduct(0.03), income_rate(0.35);

double find_interest(double home_price, double down_pay);
// this function will find the interest

double total_cost(double interest, double home_price, double down_pay);
//This function will calculate for the total cost after tax.

int main(){
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    double home_price(1.0),down_pay;
    double interest,result;
    while (home_price != 0){
        cout<<"Enter the home price (or zero to quit):\n";
        cin>>home_price;
        if (home_price !=0){
            cout<<"Enter the down payment:\n";
            cin>>down_pay;
            interest=find_interest(home_price, down_pay);
            result= total_cost(interest, home_price, down_pay);
            cout<<"The after-tax cost is $"<<result<<" annually.\n";
        }
    }
    return 0;
}
```

**Starting with Lab 5, we will be taking a close look at ALL student lab exercise submissions and checking for cheating/copying from one another or from an internet source.**

**You will get a ZERO, if caught, and be reported, per the plagiarism policy.**

# Lecture Outline

---

- **More on File I/O (Ch. 6)**
- File I/O Use in Functions
- End of File Detection
- Using *get*, *getline*, *put*, *putback*
  
- **More on Strings (Ch. 8.1, 8.2)**
- More manipulators!
- Built-in string functions

```
#include <fstream>
```

```
...
```

```
int main()
```

```
{
```

```
    ifstream LovelyInput;
```

```
    ofstream AwesomeOutput;
```

```
    LovelyInput.open("ThatInputFile.txt");
```

```
    if (LovelyInput.fail())
```

```
    {
```

```
        cerr << "Bad Mojo!\n";
```

```
        exit(1);
```

```
    }
```

```
...
```

```
LovelyInput >> MyVar1;
```

```
LovelyInput >> MyVar2;
```

```
LovelyInput.close();
```

```
...
```

```
AwesomeOutput.open("OutWithIt.txt");
```

```
AwesomeOutput << "Dodgers :( \n";
```

```
AwesomeOutput << 521;
```

```
AwesomeOutput << sqrt(521);
```

```
...
```

```
AwesomeOutput.close();
```

# Let's Look at a Demo...

---

## **RWDemo.cpp**

Found in your demo folder under **demo\_lect9**



# Can I Call a Function to do File I/O?

---

- Yes!
- But there are strict rules about it:
  - Mainly: stream objects must be passed by reference into functions

## Stream Names as Arguments

---

- Streams can be arguments to a function
  - The function's formal parameter for the stream **must** be call-by-reference

- Example:

```
void make_neat(ifstream& messy_file,  
              ofstream& neat_file);
```



## Detecting the End of a File

---

- Input files used by a program may vary in length
  - Programs may not be able to **correctly assume** the number of items or lines in the file
  - You may not know either!
- C++ provides 2 methods that can tell you if you have reached the end of a file that you are reading

## Detecting the End of a File

- The Boolean expression **(in\_stream.eof( ))**
  - Utilizes the member function **eof()** ... or end-of-file
  - **True** if you have reached the end of file
  - **False** if you have not reached the end of file
- The Boolean expression **(in\_stream >> next)**
  - Does 2 things:
    - \* Reads a value from **in\_stream** and stores it in variable **next**
    - \* Returns a Boolean value
  - **True** if a value **can** be read and stored in next
  - **False** if **there is not a value to be read** (i.e. b/c of the end of the file)

## End of File Example

*using **while (ifstream >> next)** method*

- To calculate the average of the numbers in a file that contains numbers of type double:

```
ifstream in_stream;
in_stream.open("inputfile.txt")

double next, sum(0), average;
int count = 0;

while(in_stream >> next) {
    sum = sum + next;
    count++;
}
average = sum / count;
```


## End of File Example

using **while ( !ifstream.eof() )** method

---

- To read each character in a file,  
and then write it to the screen:

*More  
on .get() later*



```
in_stream.get(next);  
while ( ! in_stream.eof( ) ) {  
    cout << next;  
    in_stream.get(next);  
}
```

# Which of the 2 Should I Use?!

In general:

See demo file:  
**changeCtoCPP.cpp**

- Use **eof** **when input is treated as text**  
and using a member function `get` to read input
- Use the **extraction operator (>>)** method  
**when processing numerical data**

## Member Function `get(char)`

- Member function of every input stream
  - i.e. it works for **cin** *and* for **ifstream**
- Reads ***one character*** from an input stream
- Stores the character read in a variable of **type char**, which is the single argument the function takes
- Does **not** use the extraction operator (`>>`)
- Does **not** skip whitespaces, like blanks, tabs, new lines
  - *Because these are characters too!*



## Using `get`

- These lines use **`get`** to read a character and store it in the variable **`next_symbol`**

```
char next_symbol;  
cin.get(next_symbol);
```

- Any character will be read with these statements
  - Blank spaces too!
  - `'\n'` too! (The newline character)
  - `'\t'` too! (The tab character)

## get Syntax

See demo file:  
[get\\_example.cpp](#)

```
input_stream_object.get(char_variable);
```

- Examples:

```
char next_symbol;  
cin.get(next_symbol);
```

```
ifstream in_stream;  
in_stream.open("infile.txt");  
in_stream.get(next_symbol);
```

## More About `get`

- Given this code: `char c1, c2, c3;` and this input: `AB`  
`cin.get(c1);` `CD`  
`cin.get(c2);`  
`cin.get(c3);`
- Results: in `c1 = 'A'` `c2 = 'B'` `c3 = '\n'`
- On the other hand: `cin >> c1 >> c2 >> c3;`  
would place `'C'` in `c3` because `>>` operator skips newline characters

## The End of The Line using `get`

- To read and echo an entire line of input by collecting all characters before the newline character
- Look for `'\n'` at the end of the input line:

```
cout << "Enter a line of input and I will echo it.\n";  
char symbol;  
do {  
    cin.get(symbol);  
    cout << symbol;  
} while (symbol != '\n');
```

- All characters, including `'\n'` will be output

## NOTE: '\n ' vs "\n "

- '\n'
  - A value of type char
  - **Can** be stored in a variable of type char
- "\n"
  - A string containing only one character
  - **Cannot** be stored in a variable of type char
- In a **cout** statement they produce the same result

# getline function

See demo file:  
**getline\_example.cpp**

- For standard inputs, **cin** is fine: *but it ignores space, tabs, and newlines*
- Sometimes, you want to get the *entire line of data!*
- Best to use the function **getline** for that purpose.
- You have to include the **<iostream>** library (which you likely already do!)
- Popular Usage:

```
getline(ifstream_object, string);  
getline(cin, string);
```



## Member Function **put**

---

- Member function for **ofstream**
- Requires **one argument of type char**
- Places its argument of **type char** in the output stream
- Not very popular...

## put Syntax

---

- `output_stream_object.put(char_variable);`
- Examples:

```
ofstream out_stream;  
out_stream.open("outfile.dat");  
out_stream.put('Z');
```

## Member Function **putback**

- The **putback** member function puts a **char** in the input stream
- **putback** is a member function of every input stream
  - cin, ifstream
- Useful if you want to assess a character and decide what to do from there (but still want to re-use that character)
- Character *placed* in the stream does not have to be a character *read* from the stream!

# putback Example

Also see demo file:  
**putback\_example.cpp**

- The following code reads up to the first blank in the input stream *fin*, and writes the characters to the file connected to the output stream *fout*

```
fin.get(next);  
while (next != ' '){  
    fout.put(next);  
    fin.get(next);  
}  
fin.putback(next);
```

- The blank space read to end the loop is put back into the input stream

# Character Functions

---

- Several predefined functions exist to facilitate working with characters
- The **cctype** library is required for most of them

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

# The **toupper** Function

---

- **toupper** returns the argument's upper case character
  - `toupper( 'a' )` returns 'A'
  - `toupper( 'A' )` returns 'A'

DOES NOT WORK WITH STRINGS!  
IT'S FOR CHARACTERS ONLY!



## The **tolower** Function

---

- Similar to **toupper** function...
- **tolower** returns the argument's lower case character
  - `tolower('a')` returns 'a'
  - `tolower('A')` returns 'a'

# The `isspace` Function

- **`isspace`** returns *true* if the argument is **whitespace**
  - Whitespace is: spaces, tabs, and newlines
    - So, **`isspace(' ')`** returns true, so does **`isspace('\n')`**
  - Example:

```
if (isspace(next) )  
    cout << '-';  
else  
    cout << next;
```

Prints a '-' if next contains a space, tab, or newline character

### Some Predefined Character Functions in ctype (part 2 of 2)

Function	Description	Example
<code>isupper(Char_Exp)</code>	Returns <i>true</i> provided <i>Char_Exp</i> is an uppercase letter; otherwise, returns <i>false</i> .	<pre>if (isupper(c))     cout &lt;&lt; c &lt;&lt; " is uppercase."; else     cout &lt;&lt; c         &lt;&lt; " is not uppercase.";</pre>
<code>islower(Char_Exp)</code>	Returns <i>true</i> provided <i>Char_Exp</i> is a lowercase letter; otherwise, returns <i>false</i> .	<pre>char c = 'a'; if (islower(c))     cout &lt;&lt; c &lt;&lt; " is lowercase.";</pre> <b>Outputs:</b> a is lowercase.
<code>isalpha(Char_Exp)</code>	Returns <i>true</i> provided <i>Char_Exp</i> is a letter of the alphabet; otherwise, returns <i>false</i> .	<pre>char c = '\$'; if (isalpha(c))     cout &lt;&lt; c &lt;&lt; " is a letter."; else     cout &lt;&lt; c         &lt;&lt; " is not a letter.";</pre> <b>Outputs:</b> \$ is not a letter.
<code>isdigit(Char_Exp)</code>	Returns <i>true</i> provided <i>Char_Exp</i> is one of the digits '0' through '9'; otherwise, returns <i>false</i> .	<pre>if (isdigit('3'))     cout &lt;&lt; "It's a digit."; else     cout &lt;&lt; "It's not a digit.";</pre> <b>Outputs:</b> It's a digit.
<code>isspace(Char_Exp)</code>	Returns <i>true</i> provided <i>Char_Exp</i> is a whitespace character, such as the blank or new-line symbol; otherwise, returns <i>false</i> .	<pre>//Skips over one "word" and //sets c equal to the first //whitespace character after //the "word": do {     cin.get(c); } while (! isspace(c));</pre>

# String Basics

- Use the **+** operator to *concatenate* 2 strings

```
string str1 = "Hello ", str2 = "world!", str3;  
str3 = str1 + str2;    // str3 will be "Hello world!"
```
- Use the **+=** operator to *append* to a string

```
str1 += "Z";    // str1 will be "Hello Z"
```
- Call out a character in the string based on **position**, using [ ] braces
  - Recall array indices in C++ start at zero (0)

```
cout << str1[0];    // prints out 'H'  
cout << str2[3];    // prints out 'l'
```

# Character Manipulators Work Too!

- Include **<cctype>** to use with, for example, **toupper()**

```
string s = "hello";  
s[0] = toupper(s[0]);  
cout << s;    // Will display "Hello"
```

- ...or to use with **tolower()**

```
string s = "HELLO";  
for (int i=0; i < 5; i++) s[i] = tolower(s[i]);  
cout << s; // Will display "hello"
```

# Built-In String Member Functions

---


- Search functions
  - `find`, `rfind`, `find_first_of`, `find_first_not_of`
- Descriptor functions
  - `length`, `size`
- Content changers
  - `substr`, `replace`, `append`, `insert`, `erase`



## Search Functions: **find** 1

- You can search for the *first occurrence* of a string in a string with the **.find** function

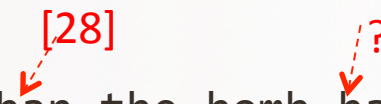
```
string str = "With a banjo on my knee and ban the bomb-ban!";  
int position = str.find("ban");  
cout << position;    // Will display the number 7
```



## Search Functions: **find** 2

- You can also search for the ***first occurrence*** of a string in a string, starting at position ***n***, using a slight mod to **.find()**

```
string str = "With a banjo on my knee and ban the bomb-ban!";  
int position = str.find("ban", 12);  
cout << position;    // Will display the number 28
```



## Search Functions: **find** 3

- You can use the **find** function to make sure a substring is **NOT** in the target string using the “no position” value


**string::npos** is returned if no position exists

```
if (MyStr.find("piano") == string::npos)
    cout << "There is no piano there!"
// This will happen if “piano” is NOT in the string MyStr
```

## Search Functions: **rfind**

- You can search for a the *last occurrence* of a string in a string with the **.rfind** function

```
string str = "With a banjo on my knee and ban the bomb-ban!";  
int rposition = str.rfind("ban");  
cout << rposition;    // Will display the number 41
```



## Search Functions: **find\_first(\_not)\_of**

- **find\_first\_of**
  - Finds 1<sup>st</sup> occurrence of ***any*** of the characters included in the specified string
- **find\_first\_not\_of**
  - Finds 1<sup>st</sup> occurrence of a character that is ***not any*** of the characters included in the specified string
- Example:

See demo file:  
**non\_numbers.cpp**

## Descriptor Functions: **length** and **size**

- The **length** function returns the length of the string
- The member function **size** is the same exact thing...
  - So, if **string str1 = “Mama Mia!”**,  
then **str1.length() = 9**  
and **str1.size() = 9** also

Example – what will this code do?:

```
string name = “Bubba Smith”;  
for (int i = name.length(); i > 0; i--)  
    cout << name[i-1];
```



## Content Changers: **append**

- Use function **append** to append one string to another

```
string name1 = " Max";  
string name2 = " Powers";  
cout << name1.append(name2); // Displays " Max Powers"
```
- Does the same thing as: **name1 + name2**

## Content Changers: **erase**

- Use function **erase** to clear a string to an empty string
- One use is:  
**name1.erase()** -- Does the same thing as: **name1 = ""**
- Another use is:  
**name1.erase(*start position, how many chars to erase*)**
  - Erases only part of the string
  - Example:  

```
string s = "Hello!";  
cout << s.erase(2, 2); // Displays "Heo!"
```

## Content Changers: **replace** and **insert**

- Use function **replace** to replace part of a string with another
  - Popular Usage:  
`string.replace(start position, # of places after start position to replace, replacement string)`
- Use function **insert** to insert a substring into a string
  - Popular Usage:  
`string.insert(start position, insertion string)`

### **Example:**

```
string country = "Back in the USSR"; // length is 16
cout << country.replace(14, 2, "A");    // Displays "Back in the USA"
cout << country.insert(15, "BC");      // Displays "Back in the USABC"
```

## Content Changers: **substr**

- Use function **substr** (short for “substring”) to extract and return a substring of the **string** object

– Popular Usage:

`string.substr(start position, # of places after start position)`

### **Example:**

```
string city = “Santa Barbara”;  
cout << city.substr(3, 5)    // Displays “ta Ba”
```

# YOUR TO-DOs

---

- ☐ HW 6 due Thu. 11/9
- ☐ Lab 5 due tomorrow Fri. 11/3
- ☐ New Lab 6 will be posted for Mon. 11/6 start
- ☐ Visit Prof's and TAs' office hours if you need help!
- ☐ Eat at least one salad this week!

**</LECTURE>**