Computer Science I

CS 135

3. File Input/Output

René Doursat

Department of Computer Science & Engineering
University of Nevada, Reno

Spring 2006

Computer Science I

CS 135

0. Course Presentation
1. Introduction to Programming
2. Functions I: Passing by Value
3. File Input/Output
4. Predefined Functions
5. If and Switch Controls
6. While and For Loops
7. Functions II: Passing by Reference
8. 1-D and 2-D Arrays

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

- ✓ Reading simple data types with **>>**
- ✓ Reading strings with **>>**

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

3.a Reading from the Keyboard

Reading simple data types with >>

- Reading variables from the keyboard with **>>**
  - ✓ **cin** is used together with **>>** (the "extraction" operator) and a variable name to collect an input value from the keyboard
  - ✓ whitespace characters (spaces, tabs, newlines, etc.) before the value are skipped
  - ✓ for example, reading one integer:
    -
```
int a;
cin >> a;
```

Values entered by the user	Effect on the variables
37	a = 37;
0	a = 0;
10000	a = 10000;

3.a Reading from the Keyboard

Reading simple data types with >>

- Reading multiple variables of the same type
  - ✓ more than one variable can be used in the same >> statement to read more than one value at a time
  - ✓ whitespace characters between the values are also skipped
  - ✓ for example, reading two integers:
    -
```
int a, b;
cin >> a >> b;
```

Values entered by the user	Effect on the variables
37 53	a = 37; b = 53;
0 180	a = 0; b = 180;
10000 2000	a = 10000; b = 2000;

3.a Reading from the Keyboard

Reading simple data types with >>

- Reading multiple variables of mixed data types
  - ✓ variables of different data types can also be read together in one **>>** statement
  - ✓ whitespaces are skipped and the other characters are read as long as they form a value of the valid type
  - ✓ for example, reading an integer, character and double:
    -
```
int a; char ch; double x;
cin >> a >> ch >> x;
```

```
72  w
   0.6
```

```
a = 72; ch = 'w'; x = 0.6;
```

Values entered by the user Effect on the variables

3.a Reading from the Keyboard

Reading simple data types with >>

- Reading multiple variables of mixed data types (cont'd)
  1. reading stops at the first whitespace character or the first character that does not belong to the requested data type
  2. if there is an error, the rest of the input is dropped entirely

✓ for example, reading an integer, double and character:

```
int a = 0; double x = 1.0; char ch = '$';
cin >> a >> x >> ch;
```

Values entered by the user	Effect on the variables
72 0.6w	a = 72; x = 0.6; ch = 'w';
72 0.6 hello	a = 72; x = 0.6; ch = 'h';
72.6 w	a = 72; x = 0.6; ch = 'w';
7.2 0.6 w	a = 7; x = 0.2; ch = '0';
72 w	a = 72; x = 1.0; ch = '$';
_72 0.6 w	a = 0; x = 1.0; ch = '$';

3.a Reading from the Keyboard

Reading simple data types with >>

- When reading with **>>**, the data type is paramount
  - ✓ for example, the extraction operator **>>** distinguishes between character **'2'** and number **2** by looking at the type of the variable on the right hand of **>>**
    - if it is of type char, the 2 is treated as character '2'

```
char ch;
cin >> ch;
```

```
2
```

```
ch = '2';
```

    - if it is an integer or floating point, the 2 is treated as the number 2

```
double z;
cin >> z;
```

```
2
```

```
z = 2.0;
```

3.a Reading from the Keyboard

Reading simple data types with >>

Rules when >> reads a character

- ```
  char ch;
  cin >> ch;
  ```

✓ the extraction operator skips leading whitespace, then finds and stores only the next character

✓ reading stops after one single character
  ```

3.a Reading from the Keyboard

Reading simple data types with >>

Rules when >> reads an integer

- ```
  int a;
  cin >> a;
  ```

✓ the extraction operator skips leading whitespace, then reads a plus or minus sign (if any), then reads only digits

✓ reading stops at the first non-digit character (for example, whitespace)
  ```

3.a Reading from the Keyboard

Reading simple data types with >>

Rules when >> reads a floating point

- ```
  double a;
  cin >> a;
  ```

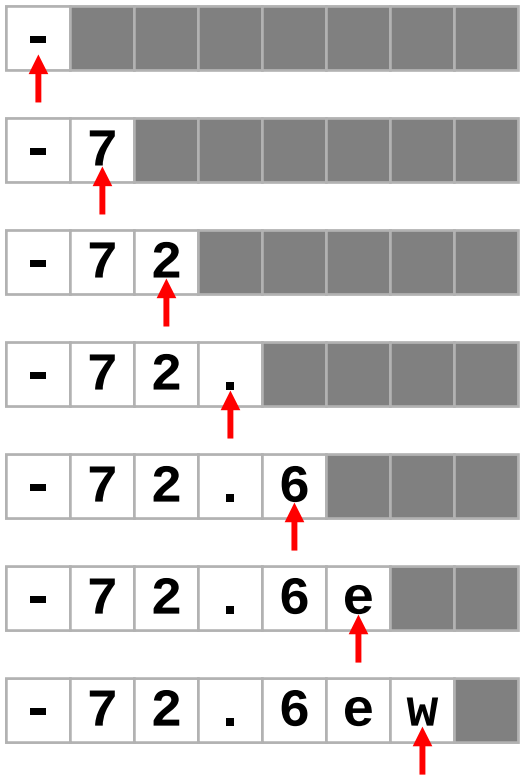
✓ the extraction operator skips leading whitespace, then reads a plus or minus sign (if any), digits, a possible '.' followed by digits (the decimal part), and/or a possible 'e' followed by digits (the exponent)

✓ reading stops whenever these rules cannot be applied, for example the next character is neither a digit, nor a dot, nor an 'e'
  ```

3.a Reading from the Keyboard

Reading simple data types with >>

➢ To apply these rules, **>>** makes use of a "pointer"



✓ I want an int: I accept **-** and I need a digit now

✓ I accept **7** → so far I have **-7**

✓ I accept **2** → so far I have **-72**

✓ I refuse **.** → the final int value is **-72** but since I now want a floating point, I have **0.** so far

✓ I accept **6** → so far I have **0.6** for my float

✓ I accept **e** → I have **0.6e**... now I need a digit

✓ I refuse **w** → this creates an error: the float value cannot be read and I drop the rest of the line

What the pointer discovers

What the **>>** operator "thinks"

3.a Reading from the Keyboard

Reading strings with >>

- What is a string?
  - ✓ a string is a sequence of zero or more characters, for example:
    - `"Please enter a value: "` *(22 chars)*
    - `"$"` *(1 char)*
    - `""` *(0 char: this one is called the empty string)*
  - ✓ a string variable can be declared as an array of type char and assigned a string value at the same time, as follows:
    - `char name[] = "John Doe";`
  - ✓ unlike other variables, however, a string of unknown size *cannot* be declared and assigned separately:
    - ~~`char name[];`~~
      ~~`name = "John Doe";`~~

3.a Reading from the Keyboard

Reading strings with >>

- Reading a string with **>>**
  - ✓ if the value of a string variable is not a hard-coded literal string, then its maximum size must also be defined as follows:
    - **char name[20];**
  - ✓ attention: the extraction operator reads only one word at a time; as usual, it separates entries based on whitespaces
  - ✓ for example, reading a full name with **>>** requires 2 strings:
    - **char first[20], last[20];**
      cin >> first >> last;

Values entered by the user	Effect on the variables
John Doe	first = "John"; last = "Doe";
72.6 w John	first = "72.6"; last = "w";

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

- ✓ Reading simple data types with **>>**
- ✓ Reading strings with **>>**

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

- ✓ Replacing `cin` with an input file stream
- ✓ Prompting the user for the filename

c. Input & Output Streams

d. Writing to an Output File

3.b Reading from an Input File

Replacing `cin` with an input file stream

- Problem: convert pounds to kilograms
  - ✓ write a C++ program that
    - prompts the user for a weight in pounds
    - displays the same weight in kilograms

→ then change the program to read the weight from a file instead of interactively from the keyboard

```
177
96
250
...
```

```
Enter weight in pounds:
177

The weight in kilos is:
80.29
```

```
The weight in kilos is:
80.29
```

The weight-converter program reading from a file and writing to the screen

3.b Reading from an Input File

Replacing `cin` with an input file stream

```
void main()
{
  // declare variables
  double pounds, kilos;

  // prompt user for weight in lbs
  cout << "Enter weight in pounds: ";
  cin >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is: ";
  cout << kilos << endl;
}
```

```
void main()
{
  // declare variables
  double pounds, kilos;
  ifstream myfile;

  // open input file and read weight
  myfile.open("lbs.txt");
  cout << "Enter weight in pounds ";
  myfile >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is ";
  cout << kilos << endl;
}
```

Changing an interactive prompt into reading from a file

3.b Reading from an Input File

Replacing **`cin`** with an input file stream

- How to read from a file

1. declare an input file variable as a mandatory **`ifstream`** type
   - **`ifstream myfile;`** *(the variable is what you want)*
2. open the file by calling the **`open`** function and passing the filename as a string in argument (no spaces or symbols!)
   - **`myfile.open("stuff.txt");`**
3. read the file: you can now use the file variable exactly like **`cin`** with the extraction operator **`>>`**
   - **`myfile >> x >> name >> a;`**
4. close the file when you are finished reading (no arguments)
   - **`myfile.close()`**

3.b Reading from an Input File

Replacing **`cin`** with an input file stream

- How to read from a file — two more things

0. **include** the following mandatory library at the top of your program if you are going to read from, or write to files
   - `#include <fstream>;`

2.5 **check** that the system was able to open the file you wanted (optional)

```
if (myfile.fail())
    cout << "Could not open file!";
```

- ✓ note that opening a file can fail for different reasons:
  - the filename does not exist
  - the filename exists but the file is protected from reading

3.b Reading from an Input File

Replacing `cin` with an input file stream

```
void main()
{
   // declare variables
   double pounds, kilos;
   ifstream myfile;

   // open input file and read weight
   myfile.open("lbs.txt");
   myfile >> pounds;

   // calculate weight in kilograms
   kilos = pounds / 2.2046;

   // display result
   cout << "The weight in kilos is ";
   cout << kilos << endl;
}
```

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

void main()
{
   // declare variables
   double pounds, kilos;
   ifstream myfile;

   // open input file and read weight
   myfile.open("lbs.txt");
   if (myfile.fail())
       cout << "Could not open file!";
   myfile >> pounds;
   myfile.close();

   // calculate weight in kilograms
   kilos = pounds / 2.2046;

   // display result
   ...
```

Complete code for reading from a file

3.b Reading from an Input File

Prompting the user for the filename

```
void main()
{
  // declare variables
  double pounds, kilos;
  ifstream myfile;

  // open input file and read weight
  myfile.open("lbs.txt");
  myfile >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is ";
  cout << kilos << endl;
}
```

```
void main()
{
  // declare variables
  double pounds, kilos;
  ifstream myfile;
  char myfilename[] = "lbs.txt";

  // open input file and read weight
  myfile.open(myfilename);
  myfile >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is ";
  cout << kilos << endl;
}
```

Putting the filename into a string variable

3.b Reading from an Input File

Prompting the user for the filename

```
void main()
{
  // declare variables
  double pounds, kilos;
  ifstream myfile;
  char myfilename[] = "lbs.txt";

  // open input file and read weight
  myfile.open(myfilename);
  myfile >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is ";
  cout << kilos << endl;
}
```

```
void main()
{
  // declare variables
  double pounds, kilos;
  ifstream myfile;
  char myfilename[32];

  // prompt user for input filename
  cout << "Enter input filename ";
  cin >> myfilename;

  // open input file and read weight
  myfile.open(myfilename);
  myfile >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is ";
  cout << kilos << endl;
}
```

Asking the user to enter the filename

3.b Reading from an Input File

Prompting the user for the filename

- How to read from any file by asking for a filename

1. declare a file and a name variable (set a maximum length)
   - ```
     ifstream somefile;
     char filename[32];
     ```
2. prompt the user for the filename
   - ```
     cout << "Enter input filename ";
     cin >> filename;
     ```
3. open the corresponding file by calling the **`open`** function and passing the filename variable in argument
   - ```
     somefile.open(filename);
     ```
4. check, read and close the file as usual
     ```

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

- ✓ Replacing `cin` with an input file stream
- ✓ Prompting the user for the filename

c. Input & Output Streams

d. Writing to an Output File

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

- ✓ Streams are sequences of characters
- ✓ Common streams: keyboard (**`cin`**) and screen (**`cout`**)
- ✓ File streams: input files and output files

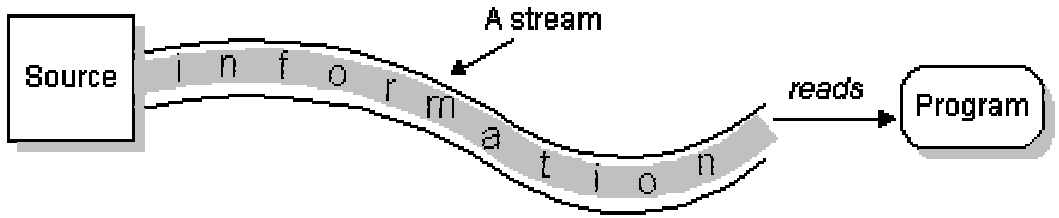
d. Writing to an Output File

3.c Input & Output Streams

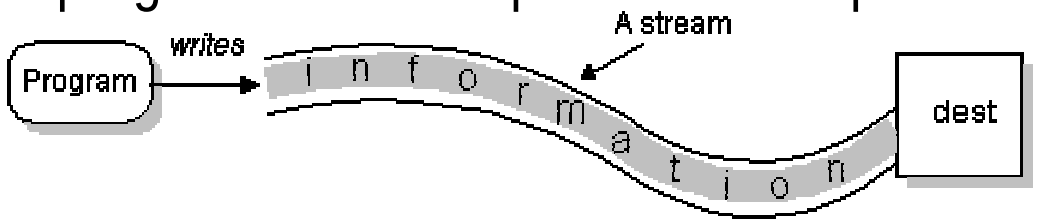
Streams are sequences of characters

Streams

- a stream is a sequence of characters from a source to a destination
  - an input stream is a sequence of characters from an input device or a file to a program in the computer



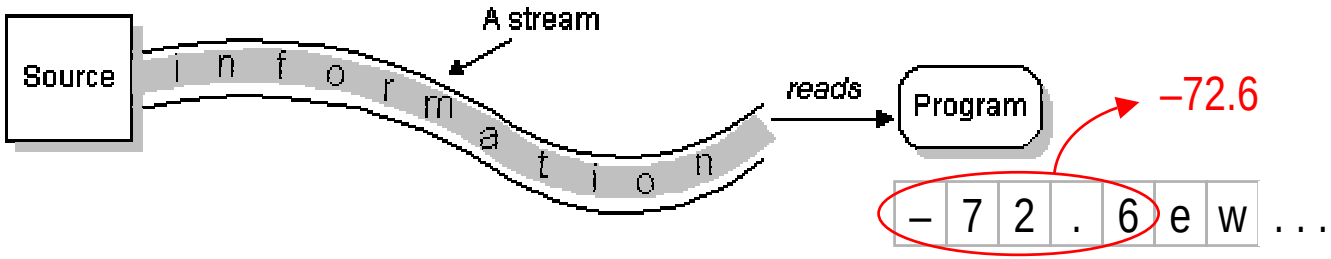
  - an output stream is a sequence of characters from a program in the computer to an output device or a file



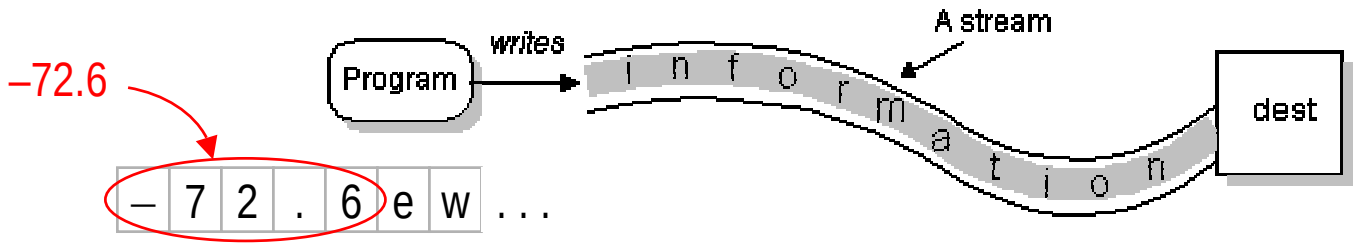
3.c Input & Output Streams

Streams are sequences of characters

- Extraction operator **>>** and insertion operator **<<**
  - ✓ these operators can be used to read from and write to streams certain combinations of characters (numbers, strings, etc.)
    - the extraction operator **>>** reads characters from an input stream and groups them into meaningful data



    - conversely, the insertion operator **<<** converts data into characters and writes these characters to an output stream



3.c Input & Output Streams

Common streams: keyboard and screen

- Use **<iostream>** to read data from the keyboard (input) and to write data to the screen (output)
  - ✓ **#include <iostream>** contains the definitions of two data types:
    - **istream** is the input stream data type
    - **ostream** is the output stream data type
  - ✓ **#include <iostream>** also contains two predefined variables:
    - **istream cin;** stands for "common input"
    - **ostream cout;** stands for "common output"

3.c Input & Output Streams

File streams: input files and output files

- Use **<fstream>** to read from and write to a file
  - ✓ **#include <fstream>** contains the definitions of two data types:
    - **ifstream** is the input file stream data type
    - **ofstream** is the output file stream data type
  - ✓ **#include <fstream>** does *not* contain predefined variables; you must declare and open your own files, for ex.:

```
ifstream weights_file;
ofstream student_records;

weights_file.open("weights.txt");
student_records.open("stu_db.xml");
```

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

- ✓ Streams are sequences of characters
- ✓ Common streams: keyboard (**`cin`**) and screen (**`cout`**)
- ✓ File streams: input files and output files

d. Writing to an Output File

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

- ✓ Replacing `cout` with an output file stream
- ✓ All in one: reading from a file and writing to another file

3.d Writing to an Output File

Replacing `cout` with an output file stream

- Problem: convert pounds to kilograms
  - ✓ write a C++ program that
    - prompts the user for a weight in pounds
    - displays the same weight in kilograms

→ then change the program to write the weight to a file instead of interactively to the screen

```
Enter weight in pounds:
177

The weight in kilos is:
80.29
```

```
Enter weight in pounds:
177
```

```
80.29
43.55
...
```

The weight-converter program reading from the keyboard and writing to a file

3.d Writing to an Output File

Replacing `cout` with an output file stream

```
void main()
{
  // declare variables
  double pounds, kilos;

  // prompt user for weight in lbs
  cout << "Enter weight in pounds: ";
  cin >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is: ";
  cout << kilos << endl;
}
```

```
void main()
{
  // declare variables
  double pounds, kilos;
  ofstream anyNameUWant;

  // prompt user for weight in lbs
  cout << "Enter weight in pounds: ";
  cin >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // open input file and write weight
  anyNameUWant.open("kilos.txt");
  anyNameUWant <<
      "The weight in kilos is: ";
  anyNameUWant << kilos << endl;
}
```

Changing an interactive display into writing to a file (compare with 3.b)

3.d Writing to an Output File

Replacing `cout` with an output file stream

- How to write to a file (compare with 3.b)
  1. declare an output file variable as an `ofstream` type
     - `ofstream foo;` *(the variable can be what you want)*
  2. open the file by calling the `open` function and passing the filename as a string in argument (no spaces or symbols!)
     - `foo.open("data_report.txt");`
  3. write into the file: you can now use the file variable exactly like `cout` with the insertion operator `<<`
     - `foo << x << name << a;`
  4. close the file when you are finished writing (no arguments)
     - `foo.close()`

3.d Writing to an Output File

Replacing `cout` with an output file stream

- How to write to a file — two more things (same as 3.b)

0. **include** the following mandatory library at the top of your program if you are going to read from, or write to files
   - `#include <fstream>;`

2.5 **check** that the system was able to open the file you wanted (optional)

```
if (foo.fail())
    cout << "Could not open file!";
```

✓ note that opening a file can fail for different reasons:

- the filename does not exist
- the filename exists but the file is protected from writing

3.d Writing to an Output File

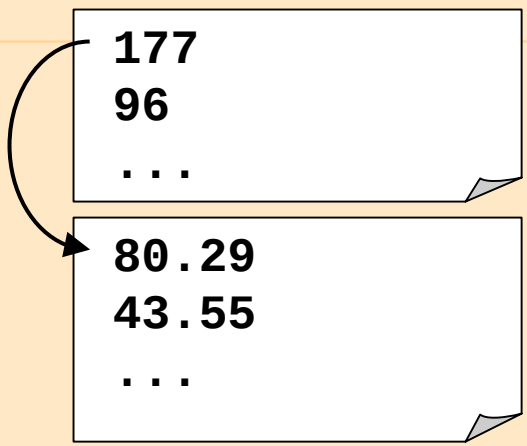
All in one: reading from a file and writing to another file

- Problem: convert pounds to kilograms
  - ✓ write a C++ program that
    - prompts the user for a weight in pounds
    - displays the same weight in kilograms

→ then change the program to read the weight in pounds from a file and write the weight in kilos into another file

```
Enter weight in pounds:
177

The weight in kilos is:
80.29
```



The weight-converter program reading from a file and writing to another file

3.d Writing to an Output File

All in one: reading from a file and writing to another file

```
#include <iostream>

void main()
{
  // declare variables
  double pounds, kilos;

  // prompt user for weight in lbs
  cout << "Enter weight in pounds: ";
  cin >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // display result
  cout << "The weight in kilos is: ";
  cout << kilos << endl;
}
```

```
#include <fstream>

void main()
{
  // declare variables
  double pounds, kilos;
  ifstream lbs_file;
  ofstream kilos_file;

  // open input & output files
  lbs_file.open("lbs.txt");
  kilos_file.open("kilos.txt");

  // read weight in pounds
  lbs_file >> pounds;

  // calculate weight in kilograms
  kilos = pounds / 2.2046;

  // write weight in kilograms
  kilos_file << kilos << endl;
}
```

one day, we will loop over this...

Changing an interactive prompt/display into reading/writing to files

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

- ✓ Replacing **`cout`** with an output file stream
- ✓ All in one: reading from a file and writing to another file

Computer Science I

CS 135

3. File Input/Output

a. Reading from the Keyboard

b. Reading from an Input File

c. Input & Output Streams

d. Writing to an Output File

Computer Science I

CS 135

0. Course Presentation
1. Introduction to Programming
2. Functions I: Passing by Value
3. File Input/Output
4. **Predefined Functions**
5. **If and Switch Controls**
6. **While and For Loops**
7. **Functions II: Passing by Reference**
8. **1-D and 2-D Arrays**