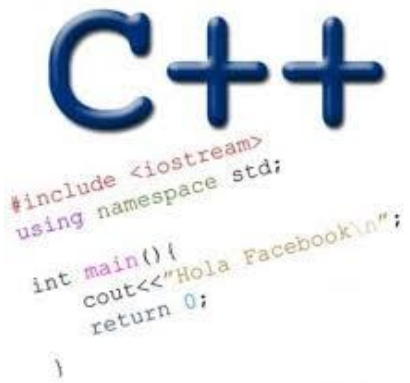
FILE IO

Problem Solving with Computers-I

https://ucsb-cs16-sp17.github.io/



Clickers out – frequency AB

Announcements

- Midterm next Wed (04/26)
- All material until (and including) lecture 5, homeworks (1-6), lab00 – lab02
- You may bring a single sheet of handwritten notes (both sides)

Demo- Lab 03 Counting ducks

- Read the lab description here:
  https://ucsb-cs16-sp17.github.io/lab/lab03/

- First order goal is to understand the behavior of the given countDucks.cpp program
- Next, we would like to understand the provided implementation, and understand how C++ enables File IO
- Note that the FILE IO functions that you see in this example program (used from the C++ standard library) are far from exhaustive.

Let’s look at how C++ programs interface with files

```
ifstream ifs; // the stream we will use for the input file
```

Q: Assume you are working with a file directly on a unix shell (not in the context of a C++ program). If you wanted to read the content of an existing file, which of the following would you do first?

A. Copy it using the ‘cp command’
B. Open it with an editor like vim or emacs

Opening a C++ file

```
ifstream ifs; // the stream we will use for the input file
ifs.open("animals01.txt"); //Opening a file in C++
```

Q: Again assume your on a unix shell. Which of the following would happen if you tried opening a file (with vim or emacs) that doesn't exist?

A. A new file is created
B. An error message is displayed

What if opening the file failed!

```
ifstream ifs; // the stream we will use for the input file
ifs.open(“animals01.txt”); //Opening a file in C++
```

Why would opening the file fail?

- You don’t have the correct permissions (true both for C++ programs and editors)
- The file does not exist (happens only if you are trying to read from the file)

But how do we detect failures in a C++ program?

Doing a status check in your C++ program

```
ifstream ifs; // the stream we will use for the input file
ifs.open("animals01.txt"); //Opening a file in C++

ifs.fail() //This function call returns a TRUE if there was
           //a failure, and a false otherwise
```

If there was a failure when trying to open a file, your program is not going to crash, but proceeding to do other things would be futile!

Activity 1: With your peer group write some C++ code that causes your program to exit if there was a failure when opening a file.

File was opened successfully, let's read one line!

```
string thisline;
getline(ifs,thisLine); //Important to understand how this
                       //actually works
```

What is the output of this program for the provided code, and the given file?

```
string thisline;
getline(ifs,thisLine);
cout<< thisline <<“ ”;
getline(ifs,thisLine);
cout<< thisline;
```

A. duck duck
B. duck goose
C. goose duck
D. goose goose
E. duck goose duck

animals01.txt

duck
goose
duck

What if I had one too many getline() statements?

```
string thisline;
for( int i=0; i<4; i++){
    getline(ifs,thisLine);
    cout<< thisline <<“ ”;
}
```

animals01.txt

duck
goose
duck

The behavior is indeterminate.

What are our options to avoid reading past the end of the file?

Status check on getline()

```
string thisline;
for( int i=0; i<4; i++){
    getline(ifs,thisLine);
    cout<< thisline <<“ ”;
    cout<< ifs.eof()<<endl;
}
```

animals01.txt

duck
goose
duck

ifs.eof() : returns a true if getline previously tried to read past the last line, false otherwise

Activity 2: With your peer group modify the above code, so that the program prints out all the lines in the file, without attempting to read past the last line. Hint use ifs.eof()

```
#include <iostream>
#include <fstream>

using namespace std;

int main( int argc, char** argv )
{
  ifstream theFile;
  string thisLine;
  theFile.open( argv[1] );
  while ( 1 ) {
    _____A_____;
    if (theFile.eof()) break;
    _____B_____;
    cout << thisLine;
  }
  theFile.close();
}
```

Where should we place the following C++ statement in order to correctly read and print all lines in a file (one at a time)

```
getline(theFile, thisLine);
```

C. Either A or B

D. None of the above

for loop OR while loop? Which one should you use?

```
for (int i = 0; i < 15; i++){
   cout << i << endl ;
}

int j =0;
int n;
while(j < 15){
   cout << "Enter a number" << endl ;
   cin>>n;
   j = j+n;
   cout<<"Current value of j is:"<<endl;
}
```

Infinite loops

```
for(int y=0;y<10;y--)
    cout<<"Print forever\n";
```

```
int y=0;
for(;;y++)
    cout<<"Print forever\n";
```

```
int y=0;
for(;y<10;);
    y++;
```

```
int y=0;
while(y<10)
    cout<<"Print forever\n";
```

```
int y=0;
while(y=2)
    y++;
```

Function call mechanics

What is the output of the following code

```
int sum(int a, int b){
        cout<< a+b;
}
int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result+30;
}
```

Function call mechanics

What is the output of the following code

```
int sum(int a, int b){
        return a+b;
}
int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result+30;
}
```

Function call mechanics

What is the output of the following code

```
int sum(int a, int b){
        int result= a+b;
        exit(0);
}

int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result+30;
}
```

Next time

- Data representation