Session 1: Intro to C++ and Common Errors
Session Outline

1. Introductions
2. Primer on SI
3. My SI game plan
4. Skeleton of a C++ Program
5. Skeleton Examples / Exercises
6. Common Errors in Programming
7. Programming Error Examples / Exercises
8. First year classes / Note taking tips
About Me

• Adam Tilson
• 3\textsuperscript{rd}/4th Year Software Systems Engineering
• Took CS 110 in Winter 2014
• Passionate about programming
• Lifelong learner

tilson3a@uregina.ca
www.adamtilson.ca
Why am I leading SI?

- Love learning
- Love programming
- Want to share what I've learned with others
- Want to see others succeed
- Want to use the apps / programs / games you'll all be making some day.
Why are you here?

- Who are you?
- What is your major?
- Why are you taking CS 110?
- What are you hoping to get out of SI?
- What's your favourite app?
About SI

- Started in 1973 at U M KC by Dr. Martin
- Student lead study groups lead by a student who enjoys that subject
- Typically offered for “gatekeeper” courses
- Brought to U of R in 2008
- Statistics taken suggest SI attendance directly impacts your grade
Why SI?

- Provides weekly free group tutoring sessions
- Meet other students
- Get better grades
  - On average your grade increases ~1% / session for up to five sessions (and maybe more).
- Cut down on cram time
- Reinforce the key concepts in a hands on way
- Test your understanding of topics
- Friendly, low pressure environment
My SI Game Plan

1. Brief review of the material from this week
   ● Help to get all three sections on the same page

2. Hands on exercises using this material
   ● Get you working on the concepts discussed in class
   ● Test your understanding, make mistakes, learn

3. Things I wish I knew when I took CS 110
   ● General advice to help your with programming and university life
This week in a nutshell

- Anatomy of a C++ program
- 3 Common types errors
Anatomy of a C++ Program

1 #include <iostream>
2 using namespace std;
3 int main ()
{
4    cout << "Hello world" << endl;
5    return 0;
}
What is (1) and why do we need it?
What is (1) and why do we need it?

#include <iostream>
What is (1) and why do we need it?

- Preprocessor Directives
  - Tell us which parts of other people's programs we are going to include our own program.
  - Avoids reinventing the wheel, makes life easy
  - iostream lets us print streams to the screen. You'll learn more as you go.
What is (2) and why do we need it?
What is (2) and why do we need it?

using namespace std;
What is (2) and why do we need it?

- Namespaces
  - Programs can get very big, and it's possible two different parts of the program are using the same names.
  - Namespaces let us know which symbol we are referring to.
  - “std” namespace includes words like cout, cin, endl.
  - You may learn more later on.
What is (3) and why do we need it?
What is (3) and why do we need it?

```c
int main ()
{
    ...
}
```
What is (3) and why do we need it?

- **Main Function**
  - This is what runs when you run your program
  - `int main` is the entry point to your program
  - Without it, nothing will run
What is (4) and why do we need it?
What is (4) and why do we need it?

cout << "Say something!" << endl;
What is (4) and why do we need it?

- Our program actually needs to do something. :P
- cout, for example, prints something to the screen.
- We'll change this up based on what we want our program to do.
What is (5) and why do we need it?
What is (5) and why do we need it?

return 0;
What is (5) and why do we need it?

- Returns back to the operating system
- 0 means everything's good
- Other values can be returned as errors.
What's missing?

#include <iostream>

int main ( )
{
    cout << “Will this compile even?” << endl;
    return 0;
}

What's missing?

#include <iostream>
using namespace std;
int main ()
{
    cout << "Will this compile even?" << endl;
}

What is the mechanism which...

- Marks the beginning of your program?
- Marks the ending of your program?
- Lets you add other people's code to your program?
3 Common Errors

- Syntax Error
- Runtime Error
- Logical Error
What is a Syntax Error?

- You broke the rules of c++
- Your program will not compile.
- There are a LOT of ways to break the rules of c++
- Some you might know...
  - Wrong operators
  - Missing Semicolons
  - Unknown symbol
- You'll learn many more ways to break c++ in time.
What is a Runtime Error?

• Your code seems to work okay...
• It even compiles...
• But when it runs...
  – CRASH!

** You shouldn't encounter too many of these when you're starting out... hopefully **
What is a Logic Error?

- Your code seems to work okay...
- It even compiles...
- But when it runs...
  - Wrong output!

- Your program didn't do what you expected.
What type of error is this?

```cpp
#include <iostream>
using namespace std;
int main ()
{
        cout << “Hello world” << endl
        return 0;
}
```
What type of error is this?

#include <iostream>
using namespace std;
int main ()
{
    cout >> “Hello world” >> endl;
    return 0;
}
What type of error is this?

```cpp
#include <iostream>
using namespace std;
int main ()
{
    return 0;
    cout << "Hello world" << endl;
}
```
What type of error is this?

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

int main ()
{
    cout << "Hello" << " world" << endl;
    return 0;
}
```
What type of error is this?

```cpp
#include <iostream>
using namespace std;
int main ()
{
    cout << endl << "Hello world";
    return 0;
}
```
What type of error is this?

```cpp
#include <iostream>
using namespace std;
int main ()
{
    shout << "Hello world" << endl;
    return 0;
}
```
What type of error is this?

```cpp
#include <iostream>
using namespace std;
int main ()
{
    cout << 7 / 0 << endl;
    return 0;
}
```
TIWIKWITCS110 - 1st Year Classes

- Some profs take all this easy stuff for granted
- Lots of info, not a lot of time
- You will need to put in time out of class
- You may not see the big picture until the end of the class
- Textbooks are your friend
- MOOCs are your better friend
- The internet is *sometimes* your friend.
Note Taking

• Course notes are not enough to catch everything

• Leave blanks where you have missed things, and fill in the blanks later on after looking things up, asking questions

• If your prof thinks it's important enough to write on the board, it's important enough for you to write in your notes
Keeping it all together

- Starting with a strong foundation is ideal, but sometimes impossible
- Write down what you can, and fill in the blanks as you go
- Review your notes often
- Apply material you've learned later on to help you catch things you learned earlier
- Make mistakes when it doesn't matter so that you can learn from them for when it does matter.