SENG 1120/6120 (Data Structures)

Lecturer:

- Dr. Alexandre Mendes (Callaghan)
  - room ES236 – ES Building
  - phone 4921 6172
  - email alexandre.mendes@newcastle.edu.au

A little bit about me

- Born in Brazil, grew up in Rio

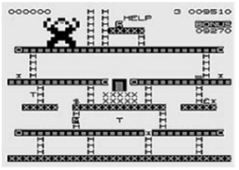
Ancient history

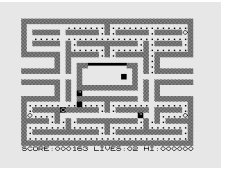
- Got my first computer at age 12
- Programmed in BASIC (Beginner's All-purpose Symbolic Instruction Code)

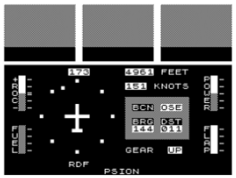
```
 10 INPUT "What is your name: "; U$
 20 PRINT "Hello "; U$
 25 REM
 30 INPUT "How many stars do you want: "; N
 35 S$ = ""
 40 FOR I = 1 TO N
 50 S$ = S$ + "*"
 55 NEXT I
 60 PRINT S$
 65 REM
 70 INPUT "Do you want more stars? "; A$
 80 IF LEN(A$) = 0 THEN GOTO 70
 90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT
```

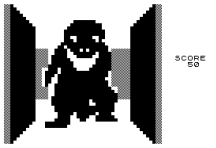
Ancient history

- This is what video games looked like.









Not so ancient history

- Went to university at UNICAMP (State University of Campinas)

Not so ancient history

- Went to university at UNICAMP (State University of Campinas)

Not so ancient history

- BSc in Applied Maths and Computing
- MSc in Electrical Eng. (Automation)
- PhD in Electrical Eng. (Automation)

Recent history

- Arrived in Newcastle in 2003
  - Research Assistant until 2006
  - Lecturer in 2007
  - Senior Lecturer in 2012
- Now
  - Married, one child
  - Still prefer soccer to rugby

Course details

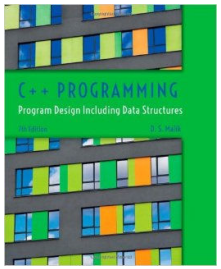
- Lectures:
  - Tuesday 10AM – 12PM, room ES204
- Assignments:
  - 3 assignments, deadlines in course outline
- Late assignments will lose 10% per calendar day
- Assessment:
  - Assignments are worth 15%, 10% and 15%
  - Mid-semester exam is worth 10%
  - Final exam is worth 50%

Additional information

- Computer labs (ES409):
  - Tuesday 2pm-4pm
  - Wednesday 1pm-3pm
- Lab sessions are not compulsory, but attendance is highly recommended
- Students who don't attend to the labs and/or do not complete the tasks will NOT lose marks

Textbook

- D.S. Malik. "*C++ Programming: Program Design Including Data Structures*", 7th Edition, Cengage Learning, 2014.
  - This book is available at the bookstore in the Union Building



Teaching assistants

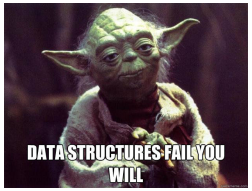
- Two persons have agreed to be teaching assistants. They are:
  - Mr. Daniel Bell
  - Mr. Timothy Pitts
- Laboratories commence in week 1 of semester

TO DO! (Really important)

- Read the course outline, which is available on Blackboard

Pause...Time for Student Feedback!

- *"Absolutely no courses in this degree have managed to explain how they help me with my goals"*
- *"Great course, great workshop. Keep on NOT spoon feeding future students. We need to have graduates that are not naive of what it's like to fail in the industry. Better to fail now in uni than at a workplace."*
- *"[Most unsatisfactory] Course of my uni experience so far."*



What is this course about?

- This course will teach you two things:
  - A new language (C++) – 4 weeks
  - Data structures – 8 weeks
- SENG1120 is assumed knowledge for:
  - SENG2200: Programming Languages and Paradigms
  - COMP2230: Introduction to Algorithms
  - COMP2240: Operating Systems
  - COMP2270: Theory of Computation
  - COMP3260: Data Security
  - COMP3330: Machine Intelligence
  - COMP3290: Compiler Design
  - COMP3320: Computer Graphics

C++ language

- This course is taught using the programming language C++
  - Your previous programming instruction was in Java, so this represents a change of language
- You can install public domain C++ on your home computer. Options include:
  - Cygwin (www.cygwin.com) which implements a Unix emulation environment for Windows PCs.
  - Mac users can download the Mac OS X Developer Tools from the Apple site.
  - In ES409, we will use Cygwin and your assignment must be Cygwin-compatible.

C++ language

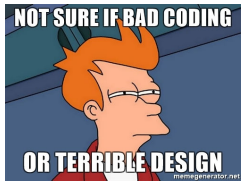
- Examples provided in class will comply with GNU C++ and are compiled using the compiler g++
- All labs and assignments must be runnable in GNU C++, which means NON-STANDARD LIBRARIES SHOULD NOT BE USED!
- Do not use C++11. The syntax is somewhat different and programs will not compile when using the GNU C++ compiler. Make sure your code compiles in the lab before you submit it.
- MAC users: Check your code. It might not work with Cygwin

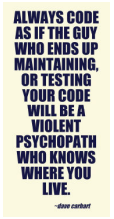
C++ language

- Computers in ES409 have Cygwin and Visual Studio installed. If you develop your code using Visual Studio, NetBeans, Eclipse, Borland C++, etc, just make sure it compiles and runs with Cygwin.
- This is VERY important. When marking, if your code does not compile and runs, you will lose 100% of the marks straightaway, because we won't be able to test it.
- Plagiarism:
  - The University has a strict policy on copyright and plagiarism (see the Course Outline which includes a link to the Policy)

Pause...Time for Student Feedback!

- *"The marking criteria is ridiculous, we gain marks and then just have them taken off of us..."*
- *"The assessment items make me feel like I'm walking on a razors edge."*
- *"My assignments did what was asked, but since it was not done the way he wanted, I received zeros."*
- *"Spend more time figured[sic] how to implement totally unnecessary code in assignments just to please the markers."*





C++ Language

Read chapter 2 of the textbook!

These slides will cover some of the contents, but the textbook is much more detailed.

What is C++?

- A programming language based on the language "C" which was developed by Dennis Richie in 1972
- C++ was created by Bjarne Stroustrup in 1979
- C++ *is not* C. It includes modern constructs that provide support for object-oriented programming techniques
- C++ had a big influence on James Gosling's development of the Java programming language

C++ vs Java

- Java does automatic garbage collection. This is not the case for C++, resulting in the possibilities of "memory leaks" and existence of pointers to non-existent (or worse, still not intended) objects.
- Java does array bound checking. C++ does not check array indexes by default, and this has been used by hackers to overwrite data values and code, and to obtain data.
- All Java variables are initialised on creation (zero for primitive types and null for object references). This is not the case for C++. Additionally non `void` functions are not checked to ensure they return something.

Program template

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello" << endl;
    return 0;
}
```

- The `include` directive in the first line tells the compiler to insert code from another file.
  - In this case the `< >` says the filename is a *system file*
- The `using` directive is like an `import` in Java. This directive makes the `std` classes available (equivalent to the automatic availability of `java.lang.*` in Java)
- The standard output stream is `cout`. The program directs the message followed by an end-of-line character (or "`\n`") to output
- The `return` informs the OS that all went well

Compiling & Running a Program

- To compile the previous program, (assuming it is stored in a file called `test.cpp` you would type:

```
g++ -o test test.cpp
```

  - Alternately a `Makefile` could be used
    - `make test` or simply `make`, depending on the `Makefile`
- This would produce a file called `test.exe` (in Windows) or `test.out` (in Linux)
- The program would be run in the usual way
  - Though it may be necessary to provide the path, e.g. `./test`

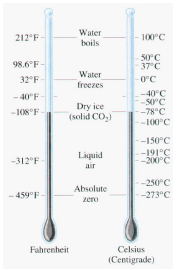
Software Development

- The software development process includes the following phases:
  - Task specification
  - Solution design
  - Implementation of the solution (coding)
  - Analysis of the solution
  - Testing and debugging
  - Maintenance and evolution
  - Obsolescence
  - plus DOCUMENTATION
- The phases may be performed together, and sometimes in a different order. Sometimes you will revisit phases and do them again.
- You are expected to practice some of these phases through this course (and beyond)

Sample Problem

- Let us use a simple problem to demonstrate the stages:
- "Given an interval set by the user (e.g. -10, 40), display a table for converting Celsius temperatures to Fahrenheit as displayed below":

```
CONVERSIONS FROM -10.0 to 40.0C
  Celsius      Fahrenheit
  -10.0C       Display degrees
  -9.0C        F in this column
  .            after they
  .            have been
  .            computed
  35.0C
  36.0C
  37.0C
  38.0C
  39.0C
  40.0C
```



The Algorithm

- This is a set of instructions for solving the problem
  - Typically expressed in pseudocode, a mix of English and program code
- Designing the algorithm involves decomposing the problem into sub-tasks
  - Then treating each sub-task as a problem that can be broken into sub-tasks
- Eventually the sub-tasks become trivial enough to be easily coded
- For example, our problem has the sub-tasks:
  - Input the temperature range from the user
  - Loop through the input range
  - Convert a temperature from degrees C to degrees F
  - Print the line of the conversion table

Pre- and Post-Conditions

- It is the most basic documentation for interface definition
- When we define a function, we specify *how* it performs its task
- When we use a function, we are only concerned with *what* it does
- Documenting the *what* for a function is done using pre and post conditions
- A *pre-condition* states conditions that must be true when the function is called
  - Otherwise it is not guaranteed to perform correctly
- A *post-condition* states what will be true when the function call has completed
  - Provided the pre-condition was satisfied and the function is correctly designed and implemented

Example of Pre- and Post-Conditions

- For our temperature conversion problem we could state the following:

```
double celsius_to_fahrenheit(double c);
// Precondition:  c is a celsius
// temperature that is not less than
// absolute zero (-273.15 degrees C)
// Postcondition: The return value is
// the input temperature c converted
// to Fahrenheit degrees
```

- As a first step in designing any function you should write out its signature (return type, name and parameter list followed by a semi-colon), then the pre- and post-conditions as comments
  - This may need modification if further development shows it is insufficient or incorrect
- The use of pre- and post-conditions is extremely important in team situations
  - Because it forms a contract between the user of a function and its writer

The STL and Standard Namespace

- C++ now has compiler requirements defined in the ANSI/ISO C++ Standard (American National Standards Institute/International Organization for Standardization)
  - Providing much better portability of C++ source code than was previously the case
- This standard includes the STL (Standard Template Library)
- Library facilities can be used after an *include directive* has been placed at the top of the file using the facility
- The items in the STL are identified using names in the *standard namespace* or `std`. Thus `using namespace std;` must follow the `include` directives
- Alternatively, you can use `std::` before any function that belong to that library, e.g.
- `cout` → `std::cout`
- `endl` → `std::endl`

Declared Constants

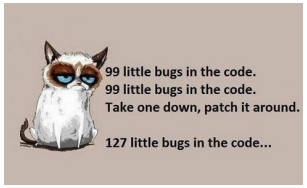
- Declared constants are defined using the keyword `const` before the declaration, e.g.

```
const double LOW_TEMP_LIMIT = -273.15;
```

- Note the convention that capital letters are used for the constant names
- The value of `LOW_TEMP_LIMIT` can never change during program execution

Testing and Debugging

- When test data reveals erroneous output, the first step is to *understand why the bug happened.* Then *correct the known error(s).* Finally *re-run all the test cases.*
- DO NOT:
  - Change suspicious code just hoping things will get better
  - Assume that even controlled rectification will not change things that previously worked properly
    - It is a truism that it is more important when performing maintenance to ensure that what previously worked continues to work than it is to ensure the new feature works properly.



DO's and DONT's

- DO always have a .cpp file with a "main" function listed under the "SOURCES" section of your makefile.
- DO always put your name and student number in each file as comment lines at the top.
- DO always backup your files in two or more devices.
- DO always code in steps, with compilation and testing between them. Only proceed to implement the next functionality after the previous one is working.
- DON'T overwrite old versions of your project (e.g. assignment). If you make a mistake and 'break' your code, you need a 'restore' point. Later in your program, you will learn to use a version control software (probably GitHub).

Your first code

DEMO

- Write a pseudo-code for the temperature transformation problem. Include error checking procedures
- All elements required for the task are included in this lecture slides
- Think about the documentation
  - Pre- and post-conditions
  - A brief explanation about the variables and constants
  - Any limitations

---

1. Input the temperature range from the user and a step for displaying intermediate values
2. Perform all error checking procedures
3. Display the labels at the top of the table
4. For each line in the table (start, intermediate and end temperature values), print the temperature in Celsius, then calculate its equivalent in Fahrenheit and print it
5. Ask the user if he/she wants to repeat the operation

See you next week!