Class 3: C++ part I

C-inherited syntax

- C++ has become very complicated (or "rich"), but it's based on C.
- C and C++ have a "nested" syntax structure.
- At the top level, C has two main possibilities:
  - Define a function, or
  - Declare something (say what it is):
    - a new "user-defined" type
    - a function (without defining it)
    - a global variable (*but avoid that*)
  - Within a function definition, other declarations can occur. (That's the nesting.)
- C++ has a few other options, such as the `using namespace` statement.

Function definitions

Basic function definitions consist of *function_type function_name* `(` *argument_list* `) { ` *function_body* `}`

where *function_type* is the type of the functions return value (like `int` or `double`), and *argument_list* is a comma-separated list of variable declarations.

For example:

```
double FahrenheitToCelsius(double T_degF)
{
  double T_degC;
  T_degC = (T_degF - 32.0)/1.8;
  return T_degC;
}
```

Pass-by-value vs. pass-by-reference

Parameters (aka arguments) in C are generally[1] "passed by value": the value is copied into a new variable to be used as the function argument.

- Changing the value in the function doesn't change any values in the calling program.

C++ also allows "pass by reference":

- The caller must provide a variable as the argument. It doesn't have to have the same name as the argument.
- Changes to the argument in the function change the value of the variable in the calling program.
- Internally, the called function is really being given the address of the variable in the calling function.

Exercise 1: pass by value vs. pass by reference

Compare these two function definitions:

```
double FahrenheitToCelsius1(double T)   // pass by value
{
  T = (T - 32.0)/1.8;  // reusing T -- bad practice!
  return T;
}

double FahrenheitToCelsius2(double & T)  // pass by reference
{
  T = (T - 32.0)/1.8;
  return T;
}
```

Try them out and see what they do. (Write a `main()` for this.) Can you call `FahrenheitToCelsius1(32.0)`? Can you call `FahrenheitToCelsius2(32.0)`?

Moral of Exercise 1

Be careful to remember which function arguments are pass-by-value and which are pass-by-reference.

Scope of variable definitions

A variable defined inside the body of one function is completely different from a variable of the same name defined in the body of another function:

```
double FahrenheitToCelsius3(double T_degF)
{
  double T_degC;
  T_degC = (T_degF-32.0)*5.0/9.0;
  return T_degC;
}

double FahrenheitToCelsius4(double T_degF)
```

[1] Exception: arrays passed to a function are passed by reference, even in C.

```
{
  double T_degC;
  T_degC = (T_degF+40.0)/1.8-40.0;
  return T_degC;
}
```

Break

Flow control

- Conditional execution (branching): `if` and `if-else`
- Looping: `while`, `do-while`, `for`
- Loop shortcuts: `continue` and `break`

basic if

```
if (expr) {
   statements;
}
```

Does statements if *expr* is non-zero, where *expr* can be any expression. Example:

```
if ( x > 100 && u > 0.0 ) {
   u= -u;
   x= 100;
}
```

basic if/else

```
if (expr) {
    statements;
}
else {
    statements;
}
```

Does first block if *expr* is non-zero, otherwise does second.

if / else if / ... / else

```
if (expr) {
    statements;
}
else if (expr2) {
    statements;
}
else if ...
```

and so on and so forth, optionally ending in:

```
else {
    statements;
}
```

looping: while and do

basic `while`	**basic** `do while`
`while (expr) { statements; }`	`do { statements; } while (expr);`
repeats statements as long as expr is non-zero	does statements, then repeats as long as expr non-zero

loop shortcuts: break, and continue

- `break` will jump out of a loop, continuing after the end of the loop
- `continue` will jump to the test and loop again if non-zero.

looping: for

loop using `for` **...**	**is exactly identical to this...**
`for (expr1; expr2; expr3) { statements; }`	`expr1; while (expr2) { statements; expr3; }`
	... with one exception.

The one way a `for` is different from a `while` is that a `continue` inside the `for` loop will execute `expr3` before going back to the beginning.

Example: print an asterisk in column n

```
void printStarAt(int n)
{
  using namespace std;
  int i;
  for (i=0; i<n; i=i+1)
```

```
    cout << ' ';
  cout << "*\n";
}
```

Exercise 2: 1-d bouncing "ball" (asterisk)

Goal:

- bounce a ball between two walls
- let x be the position of the ball
- let v be its velocity (+1 moving to the right, -1 to the left)
- start with x=0, v=+1
- reverse direction if it hits a wall at x=79 or x=0
- "animate" on terminal using the `printStar()` function

Work together to define the algorithm, and I'll code it according to your directions.

Assignment: "Adventure"

Make a simple interactive "text adventure game" that repeatedly asks user for "commands" and then gives responses.

Simple example: http://www-personal.ksu.edu/~gahs/john/adventure0.html

- Note: this was written by my 8-year-old son, with a little help from me.
- Uses JavaScript, but only works with Firefox, not Internet Explorer.

Classic example: http://www.ifiction.org/games/play.phpz?cat=&game=1&mode=html

- This is Will Crowther's 1973 version.

You don't have to set up a really big adventure, just implement a few commands in a loop and some reasonable responses. Have fun.