COSC 251
C and C++ Cheat Sheet
Last Revision: December 23, 2012

1 Variables

```
//no initialization - BAD
int x;

//standard initialization
int y = 10;

//array, no initialization - size 10
int z[10];

//array with initialization
int z = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

//class instantiation, no parameter constructor
MyClass c;

//class instantiation, a few parameters
Rectangle r(4,3);
```

2 Control Structures

Below is a bunch of code showing an example full if statement, an if-elseif, a while loop, and a for loop. There is no enhanced for loop (a.k.a. for-each loop) in C++ previous to 2011 so we'll still assume that it isn't in quite yet. This will also cut down on confusion between C and C++.

```
// if else example - note, the else is optional
if (q < 15) {
  cout << "This is the true block" << endl;
}else{
  cout <<"This is the false block"<<endl;
  }

// if else if example - note the compound conditional, AND first, OR second
if (i < 15 && j > 20){
  printf("Reverting to C for this true block\n");
}else if (j < 10 || i > 5){
  printf("rocking it in the else if block\n");
}else{
  printf("nothing was true!\n");
}

//while loop
while(k != 12){
  cout<<"Doing stuff"<<endl;
  i++;
}

//for loop, counting from 0 to 9
int n = 0;
for (n = 0; n<10; n++){
  cout << n << endl;
}
```

One important note - C++ handles declaration of the counter in the for loop structure, C traditionally has not. To be safe, declare and initialize counters outside of the loop.

3 Functions

```
//function prototype
void nameoffunction(int);

//function
void nameoffunction(int x){
  //stuff for function
}

//passing by reference
int myotherfunction(float *f){
  //anything done to f here will be reflected in the calling function
}
```

4 Input and Output

```
//if using printf and scanf:
#include<stdio.h>

//if using cout and cin:
#include<iostream>
using namespace std;

//output in C - printf
printf("Whatever you are outputting\n");

//same output in C++ - cout
cout<<"Whatever you are outputting"<<endl;

//printing variables in C - note the placeholder %d
int numjellybeans = 10000;
printf("Number of jellybeans: %d\n", numjellybeans);

//same output in C++
count<<"Number of jellybeans: " << numjellybeans <<endl;

//input in C - scanf
scanf("%d", &numjellybeans);

```

```
//input in C++ - cin
cin>>numjellybeans;
```

In C++, you'll want to add, after your #include statements, the following:

```
using namespace std;
```

Otherwise, you will need to scope cin, cout, and endl by doing:

```
std::cin>>x;
std::cout<< "whoooo" << std::endl;
```

5 Pointers

```
//pointer declaration
int *ptr;

//using reference operator
int x = 100;
ptr = &x;

//using dereference operator
//print will print 200
int y = *x + 100;
printf("%d\n", y);
```

6 Function Pointers

```
#include <stdio.h>

int foo(int x){
  printf("%d\n", x);
  return x*x;
}

int main(){
  //declaring and initializing the pointer
  int (*ptr) (int);
  ptr = foo;

  //using the pointer
  //does the same thing as foo(4);
  ptr(4);

  return 0;
}
```

7 Classes

There are no classes in C, but there are structs:

```
typedef struct{
  int height;
  int width;
}rectangle;

//this can now be used:
rectangle r = {3, 4};
```

Example header file (Rectangle.h):

```
class Rectangle{
  private:
    int h;
    int w;
  public:
    Rectangle(int, int);
    int geth(){return h;}
    int getw(){return w;}
    void seth(int);
    void setw(int);
    int area();
};
```

Appropriate source file (Rectangle.cpp):

```
#include "Rectangle.h"

Rectangle::Rectangle(int a, int b){
  h = a;
  w = b;
}

void Rectangle::seth(int a){
  h = a;
}

void Rectangle::setw(int b){
  w = b;
}

int Rectangle::area(){
  return h*w;
}
```

8 How to Compile

Once everything is set up with the Linux machines in 160 and the servers for the CS program, you will need to handle compilation slightly differently than we have in the past, and differently than how you work with an IDE. One thing to note, you may use whatever IDE that you want (Visual Studio, Eclipse CDT, etc.) but it is your responsibility to make sure that your code works with the following compilation instructions. To begin, open a terminal window. Then navigate to where your code is located and for C:

```
gcc mycodehere.c
```

For C++:

```
g++ mycodehere.cpp mycodehere2.cpp
```

Note: if you have multiple source files (including source code for any header files you are using) you need to include those in your gcc or g++ call. Just keep tacking on source files to the end (like the g++ example). To run your code once compiled:

```
./a.out
```