The Cornerstone

Volume 10, Issue 7 Tuesday November 6, 2012 MI-Gamma

Inside this issue:

migamma.org

YOU START TYPING BUT SEE "YOUR FRIEND IS TYPING" SO YOU STOP
SO DO THEY
memegenerator.net

Don't be a Socially Awkward Penguin!

TAU BETA PI—THIRD ACTIVES

Your Future As An Engineer

The Stereotype of the Engineer as socially awkward and professionally uncommunicative person is deeply pervaded in our culture both inside & outside of the College. This is a real problem we must work together to correct!

Too often I've seen the most amazing geniuses struggle at conveying their brilliant ideas to their cohorts. I've seen people with game-changing business concepts that never get out of the gate because their failure to communicate with the right people.

Hopefully by the time you've reached this paragraph you know which camp you fall into. Maybe you're a huge socialite and don't need this, that's great. Maybe you're a total bumbling nerd which is also great but not when it hinders your communication skills. Maybe you're in denial. Or maybe you're like most of us who just fall somewhere in between. So chances are, you could benefit from Engineering Futures, a very valuable program that you can gain access to for FREE through Tau Beta Pi! Just sign up online.

Basically it is a series of intelligently designed workshops for professional development… "professional development" sounds like a buzzword but really it's to help YOU in YOUR LIFE, what've you got to lose? Hope to see you there

~ *Nick*

C++11: Two Cool New Features by Justine Kunz

First off, to use C++ 11 you must get a compiler that is compatible with the new C++11 features. GCC version 4.7.0 is C++ 11 compatible, and you can access it from a CAEN computer by typing the following command in a terminal:

```
module load gcc/4.7.0
```

And then you will be able to compile your files as you would expect ie. "g++ *your_files your_command_flags*", without the quotes, of course.

Now, time to implement some features! C++ 11 has two types of new features, some that make your code run faster, and then others that make you code faster. I'll talk about a feature from each category, namely, the rule of 5 and lambdas.

C++ 11 makes code run faster by stealing (rather than copying) any data that is about to be deleted. Let's look at the following example:

```
// function declaration

vector<int> make_vector(int i, int j, int k);

// in int main()

vector<int> simple_vec = make_vector(1, 2, 3);
```

When you write a function that returns a variable by value and then you store it in another variable, the compiler actually makes a copy of the return value. In this example, the vector that is returned is typically copied into the simple_vec vector. However, the C++11 compiler knows that the vector returned is just a temporary value, called an r-value (r to indicate that it can only appear on the right side of an assignment operator), and so the C++11 compiler can just steal vector information (rather than copy it) to save time.

These are called "Move" operations. C++ 11 implements the "Rule of 5", rather than the old Rule of 3, adding the "Move Constructor" and the "Move Assignment" to the old 3

(Pop Quiz: Which three operations make up the old rule?).

The C++ Standard Library containers have already implemented these, and so if you have C++ code using any STL containers and you recompile them with a C++11 compiler, then your code may run faster! (Note: if you pass everything by reference/pointer then you may not notice a difference).

Now, onto the amazing lambda! A lambda is basically a temporary function. It acts like a function and takes arguments but it doesn't have a name (but it can be stored in a named function pointer). My favorite place to use a lambda is in a `for_each` (`#include <algorithm>`), when I am iterating through a container and I want to apply a function to every element, but I'm too lazy to define an entire function or function object. Check out the following code for example:

```
vector<int> my_vec = {1, 2, 3, 4, 5};

int add_constant = 10;

// lambdas are often too long to fit on one line, like the one below

for_each (my_vec.begin(), my_vec.end(), [add_constant, &cout]

     (int &i) { i += add_constant; cout << i << " "; });

cout << endl;
```

Assuming that the necessary libraries are included and the code is in some sort of function (main or otherwise), then this code will produce the following output:

```
11 12 13 14 15
```

This works because we used captured variables. The captured variables are included between the [square brackets] and indicate any variables that you would like to use within the lambda (the & indicates the variable should be captured by reference). The arguments for the lambda function are included in (parentheses).

Since the for_each calls the lambda on a dereferenced iterator for each item in the container, this lambda must take ints as an arguments and since we want to actually change the container, I passed the ints into the lambda by reference.

If you would like to learn more look no further than google.com! There is a lot of information about the new C++ 11 features, as well as plenty of problems/solutions that users have faced on stackoverflow.com.

CHECK OUT MIGAMMA.ORG TO SEE OUR AWESOME WEBSITE & PREVIOUS NEWSLETTERS!

LOGIC PUZZLES

(1) There are three switches downstairs. Each corresponds to one of the three light bulbs in the attic. You can turn the switches on and off and leave them in any position.
How would you identify which switch corresponds to which light bulb, if you are only allowed one trip upstairs?

(2) Three Masters of Logic wanted to find out who was the wisest amongst them. So they turned to their Grand Master, asking to resolve their dispute.
"Easy," the old sage said. "I will blindfold you and paint either red, or blue dot on each man's forehead. When I take your blindfolds off, if you see at least one red dot, raise your hand. The one, who guesses the color of the dot on his forehead first, wins."
And so it was said, and so it was done. The Grand Master blindfolded the three contestants and painted red dots on every one. When he took their blindfolds off, all three men raised their hands as the rules required, and sat in silence pondering. Finally, one of them said: "I have a red dot on my forehead."
How did he guess?

(3) The Grand Master takes a set of 8 stamps, 4 red and 4 green, known to logicians A, B and C, who can see each other, and loosely affixes two to the forehead of each logician so that each logician can see all the other stamps except those 2 in the Grand Master's pocket and the two on her own forehead. He asks them in turn if they know the colors of their own stamps:
A: "No."
B: "No."
C: "No."
A: "No."
B: "Yes."
What color stamps does B have?

ANSWERS

(1) Keep the first bulb switched on for a few minutes. Let it heat up then switch it off and turn another one on. Now you can walk into the room and figure it out!

(2) The wisest one must have thought like this:
I see all hands up and 2 red dots, so I can have either a blue or a red dot. If I had a blue one, the other 2 guys would see all hands up and one red and one blue dot. So they would have to think that if the second one of them (the other with red dot) sees the same blue dot, then he must see a red dot on the first one with red dot. However, they were both silent (and they are wise), so I have a red dot on my forehead.

(3) B says: "Suppose I have red-red. A would have said on her second turn: 'I see that B has red-red. If I also have red-red, then all four reds would be used, and C would have realized that she had green-green. But C didn't, so I don't have red-red. Suppose I have green-green. In that case, C would have realized that if she had red-red, I would have seen four reds and I would have answered that I had green-green on my first turn. On the other hand, if she also has green-green, then B would have seen four greens and she would have answered that she had two reds. So C would have realized that, if I have green-green and B has red-red, and if neither of us answered on our first turn, then she must have green-red.
"'But she didn't. So I can't have green-green either, and if I can't have green-green or red-red, then I must have green-red.'
So B continues:
"But she (A) didn't say that she had green-red, so the supposition that I have red-red must be wrong. And as my logic applies to green-green as well, then I must have green-red."
So B had green-red, and we don't know the distribution of the others certainly.