# Adventures in C Programming

Noah Bedford

May 5, 2013

## Things I've learned this semester:

- The ternary if operator has the format

  `variable = condition? value_if_true : value_if_false`

- A pointer is an address in memory
- C's errors are cryptic
- How to do mutually recursive functions

## I liked how these things worked:

C is much faster than Python (the only other language I've spent much time with) The XOR swap (`x = x ^ y, y = x ^ y, x = x ^ y`) is a very clever way to not store anything in memory while swapping two variables. I like the way the ternary `if` operator works.

## I would have done this differently:

If I were Dennis Ritchie I would have put in tail recursion, duct-typing, closures, objects, the ability to declare arrays without having to specify how much memory they use, etc. To be fair, when he was writing the language, all of this either didn't exist or was super new, and the memory thing means that C would be higher level, so it's a lot like any sort of time travel question– maybe implementing C that way would have ruined modern programming.

## My favourite programs from this semester:

Here's my program for computing factorials:

```c
#include <stdio.h>

int main(void){
  int input, output;
  scanf("%d", &input);
  int factorial(input){
    if (input == 1){
      return 1;
    }
    else {
      return input * factorial(input - 1);
    }
  }
  output = factorial(input);
  printf("%d\n", output);
  return 0;
}
```

Here's my program which approximates e (it doesn't use a bignum library):

```c
#include <stdio.h>
// compute e using lim_{n→∞} (1 + 1/n)^n
float mypow(float x, int y){
  if (y == 0){
    return 1;
    }
  if (y == 1){
    return x;
    }
  else {
    return x * mypow(x, y - 1);
  }
}

float e(float n){
  return mypow((1+1/n),n);
}

int main(){
  float n = 1000;
  printf("%f\n",e(n));
  return 0;
}
```

## The hardest thing I did:

I learned how to use basic functions of libGMP (a big number library). It was difficult because it's documented mostly by listing its functions and what arguments they take and not with any explanation. Here's a short example program:

```c
// stores ≈ π and ≈ √π using GMP
#include <stdio.h>
#include <gmp.h>

int main(){
  mpf_t pie;
  mpf_t sqrtpie;
  mpf_set_default_prec(1000); // set default precision to 1000 bits
  mpf_init(pie);
  mpf_init(sqrtpie);
  mpf_set_str(pie, "3.1415926535897932384626433832795028841971693993751058209749445
  gmp_printf ("%.40Ff\n", pie);
  //mpf_out_str(NULL, 10, 40, pie); // give me 40 digits of a base 10 string of a
  mpf_sqrt (sqrtpie, pie);
  gmp_printf ("%.40Ff\n", sqrtpie); // Turns out gmp_printf is way easier
  return 0;
}
```