ENG 101: Day 3

Monday --10/1/05

Functions

Administrative

- GSI office hours updated
  - Central campus hours are:
    - Monday 6:30pm-8:30pm Alina
      - Espresso Royale on State street, she will have a net-connected portable and a paper sign indicating she is the 101 GSI.
    - Wednesday 2:30pm-4:30pm Nadine
      - Undergrad. library basement, in or near the CAEN workstations. She will have a sign.
  - Maps on the webpage.

Administrative

- Homework box
  - In EECS, directions on the webpage under "homework"
- Midterms are from 6:30-8:30
  - Feb 10 and March 30 (as announced before)
  - Make-ups are on the following Saturday at 9:30am.
  - Contact Wanda Dobberstein <wldobber@engin.umich.edu> for make-up exam requests.
    - Requests are due 14 days before the date of the regular exam!

Administrative

- Notice I'm giving you all "hole-punched" paper (as much as I can).
  - I strongly recommend you keep a 3-ring binder with all the papers. A ¾" binder should be plenty.
  - Keep handouts, returned homework, in-labs, and exams.
    - It is often the case with code that you will want to find an example you've seen before. The binder helps.
- Books are out-of-stock everywhere around here.
  - Can find them on-line.
  - Stores are claiming "Wednesday" as arrival time.

Review

- You should be comfortable with:
  - Assignments
    - Including interactions of types double and int
  - If/else statements
    - Including curly brackets
  - While loops
    - Including curly brackets
  - cout to print
    - Can print things in quotes, doubles, ints.

ex0.cc

```
#include<iostream>
using namespace std;

main()
{
    int i=1;
    int fact=1;
    int max;

    cout << "Enter a non-negative integer less than 20 ";
    cin >> max;


    while(i<max)
    {
        i++;
        fact=fact*i;
    }
    cout << max << " factorial is equal to " << fact << endl;
}
```

ex1.cc
Part 1

```
#include<iostream>
using namespace std;

int factorial (int value)
{
    int i=1;
    int fact=1;

    while(i<value)
    {
        i++;
        fact=fact*i;
    }
    return(fact);
}
```

ex1.cc
Part 2

```
main()
{
    int max, a;

    cout << "Enter a non-negative integer less than 20 ";
    cin >> max;

    a=factorial(max);
    cout << max << " factorial is equal to " << a << endl;

    if(max<19)
    {
        max=max+1;
        a=factorial(max);
        cout << max << " factorial is equal to " << a << endl;
    }
}
```

Functions

- A function has a return type, and an argument list.
  - These all must have a type (**`int`** or **`double`** for us so far)
  - Code is just like main() but now we can get values from someplace else.
  - Return statement is the value that will be returned.
    - It also ends the function.

Using a function

- We "invoke" the function by stating its name and its arguments.
  - So factorial(4) or factorial(a) are both fine.
  - In general arguments should be of the same type.
    - So int for int and double for double
    - It will convert on the fly, but generally bad idea.
  - The function evaluates to whatever the return value is.
  - ***The arguments don't change!***

Why functions? (1/2)

- Useful if don't want to write the same code over and over again.
  - So if using factorial a lot in a program, you don't want to have to type in the code again.
  - Also nice if you need to add a feature (say you want -1! to be 0 for some reason) as you only have to change it once.

Why functions? (2/2)

- But a big reason is to make things easier for the reader and the writer.
  - Causes "functional decomposition"
    - You can break a problem down into parts.
    - Each part can be a function.
  - Can write functions first, or write calling code first.
- Breaking problems into smaller pieces is perhaps the most important idea in this whole class!

And some problems with our code

- One icky thing is that we use the value 20 in two different places without explaination.
  - Well really 20 and 19
  - These are called “magic values” or “magic numbers” because the reader has no clue where they came from or if they are connected.
- The idea was that since the int type can only represent certain ranges, at some point the value of n! is too big.
  - When is that?

ex2.cc
Part 2

```
main()
{
    const int MAX_FACTORIAL=20;
    int max, a;


    cout << "Enter a non-negative integer less than " <<
            MAX_FACTORIAL << endl;
    cin >> max;

    a=factorial(max);
    cout << max << " factorial is equal to " << a << endl;

    if(max<MAX_FACTORIAL-1)
    {
        max=max+1;
        a=factorial(max);
        cout << max << " factorial is equal to " << a << endl;
    }
}
```

ex3.cc
Part 1

```
#include<iostream>
using namespace std;

// Finds the 2 roots of a polynomial.  "which" should be
// only 0 or 1.  Different values of which give you the
// different roots.  Doesn't work if imaginary roots.
double qroot (double a, double b, double c, int which)
{
    double inside, top, bottom;

    inside=b*b - 4*a*c;
    if(which==0)
        top=-b + sqrt(inside);
    else
        top=-b - sqrt(inside);
    bottom=2*a;

    return(top/bottom);
}
```

ex3.cc
Part 2

```
main()
{
    double n2coef, n1coef, n0coef;
    double root1, root2;

    cout << "Enter the n squared coefficient ";
    cin >> n2coef;
    cout << "Enter the n coefficient ";
    cin >> n1coef;
    cout << "Enter the constant coefficient ";
    cin >> n0coef;

    root1=qroot(n2coef,n1coef,n0coef,0);
    root2=qroot(n2coef,n1coef,n0coef,1);

    cout << endl << "The roots are " << root1 << " and "
         << root2 << endl;
}
```

mc.cc
(from last time)

```
#include<iostream>
#include<cstdlib>
using namespace std;
main(int argc, char * argv[])
{
    const int trials=9000000;
    double x1, y1;
    double distance;
    int count=0; // number of hits
    int i=0;
    double value;

    while(i<trials)
    {
        x1=(1.0)*rand()/RAND_MAX;  // rand() generates an int [0.0, RAND_MAX]
        y1=(1.0)*rand()/RAND_MAX;
        distance=x1*x1+y1*y1; // Square of distance from home.
        if(distance<1)
            count++;
        i++;
    }
    value=(4.0)*count/trials;

    cout << "count= " << count << endl;
    cout << "value= " << value << endl;
}
```