Two topics:
Project 0 and arrays

Engin 101
Lecture 5

Lots of issues with project 0

• I wrote this project over the break.
  – Lots of things I’d change now knowing how
    the class has progressed.

Syntax you don’t know:

• a%b returns the remainder of a/b
  • So 5%3=2
• i++ is the same as i=i+1;
• if(bob)
  – bob is “true” unless bob==0
• Main is declared oddly
  – Just ignore it.
• exit(1) causes the program to exit at that point.
  – It is like a return statement for the whole program.

Other issues

• There is a comment that is wrong.
  – // Computes C(x,y)=x!/y!(x-y)! . x>=y
  – This should be
  – // Computes C(x,y)=x!/(y!*(x-y)!).
  • For x greater than or equal to y.
• And an error in the directions:
  – “You may not change the function names, arguments or
    return values”
  – That should be:
    • “You may not change the function names, arguments or return
      types”. 
Okay.

- Sorry for the errors.
  - Should be able to avoid similar problems in the future.
- Now lets look at some of this as a group.

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

// Very silly program that prompts the user to enter a set of numbers.
// The user then enters one more number and the program tells the user
// which numbers from the original set are larger than his final number.
main()
{
    const int NUM=5;   // number of values user must enter
    int list[NUM];
    int i=0;
    int ans;           // number user enters.
    int any=0;         // set to 1 if any number is greater.
    cout << "You will be prompted to enter " << NUM << " numbers" << endl;
    while(i<NUM)
    {
        cout << "Enter a number ";
        cin >> list[i];
        i=i+1;
    }
    cout << "Now pick a number ";
    cin >> ans;
    i=0;
    cout << endl;
    while(i<NUM)
    {
        if(ans<list[i])
        {
            cout << "Your number " << ans << " is less than " << list[i];
            cout << " from your list" << endl;
            any=1;
        }
        i=i+1;
    }
    if(any==0)
    {
        cout << "Your number was greater than all numbers in the list" << endl;
        cout << endl << "Bye!" << endl << endl;
    }
}
```

```cpp
double my_sin(double x)
{
    double sum=x;
    double term;
    int i=1;
    int sign;
    int value;
    while(i<STEPS)
    {
        value=1+2*i;
        sign=i%2;  // % is the mod function. In this case sign is
        // 1 if i is odd,
        // and 0 if i is even.
        term=my_intpow(x,value)*my_factorial(value);
        if(sign)
            sum=sum-term;
        else
            sum=sum+term;
    }
}
```

Continued from previous page

```cpp

while(i<NUM)
{
    if(ans<list[i])
    {
        cout << "Your number " << ans << " is less than " << list[i];
        cout << " from your list" << endl;
        any=1;
    }
    i=i+1;
}
if(any==0)
{
    cout << "Your number was greater than all numbers in the list" << endl;
    cout << endl << "Bye!" << endl << endl;
}
```
General issues

- int bob[4] creates 4 ints
  - bob[0], bob[1], bob[2], and bob[3]
- If you try to access a value that doesn’t exist (say bob[5] or bob[-1]) very strange things will happen.
  - Be very careful. Debugging such problems is probably the hardest single task for most programmers!

Arrays

- Traditional to draw the array with 0 on the top.

<table>
<thead>
<tr>
<th>M[0]</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>M[1]</td>
<td>15</td>
</tr>
<tr>
<td>M[2]</td>
<td>-12</td>
</tr>
<tr>
<td>M[3]</td>
<td>100</td>
</tr>
<tr>
<td>M[4]</td>
<td>1</td>
</tr>
</tbody>
</table>

An algorithm on an array:

sorting

- Say we have an array declared as
  - int M[5]
- We want a to print the list in order (say lowest to highest)
  - How do we do it?

Algorithm: step 1

- We know what we want to do
  - We need to figure out how to do it.
- The big trick is to think in terms of an algorithm.
  - That is, to think about the steps, but not the C++ syntax.
Step 2

• Is to code the algorithm.
  – I’m going to do step 1, we will do step 2 as a group.
• Bubble sort
  – Walk through the list from top to bottom.
  – At each step swap with element below if out of order.
  – Repeat N-1 times.

How do we know this works?

• Well, at the end of the first pass the bottom element must be in order
  – Why?
• At the end of the 2nd pass the two last elements must be in order.
  – Etc.

Example

Stolen from: knight.cis.temple.edu/~lakaemper/courses/cis068_2003/slides/cis068_09.ppt