Lecture 8
CS 137
Fall 2014
by Chantelle Gellert

Announcments

- Class tomorrow Friday October 3rd 8:30am MC 4065

Scientific notation

$-2.6302x10^{30}$ (30 is the exponent, 6302 is the fraction and - is the sign)
in c
float: single precision, floating point number that is 32 bits
double: double precision, floating point that is 64 bits

```c
#include <stdio.h>

int main(void){
  double x = -2.6302e30;
  printf("%g\n", x); //-2.630e+30
  printf("%2e\n", x); //-2.63e+30
  printf("%f\n",x); //2.6302000000000000001282
}
```

Value:
(-1) (sign) x fraction x $2^{exponent}$

Error in Floating point numbers:
r - real number you're trying to represent
p - approximate representation (3 decimal digits)

absolute error: $|r - p|$ $|pi - 3.14| = 0.0015927$

relative error: $\frac{|r-p|}{|r|}$ $\frac{|pi-3.14|}{|pi|} = 0.0005070$

Relative error can be small, even if absolute error is large
Minimize relative error by avoiding:
-subtracting nearly equal numbers
-dividing by very small numbers
-multiplying by very large numbers
-tests for equality

Bad: if(x==y)
Good: if(x−y < 0.0001 && y−x < 0.001)

```c
#include <stdio.h>

int main(void){
 double x = 5.0/6.0;
 double y = 1.0/2.0;
 double z = 1.0/3.0;

 if(x-y == z){
   printf("Okay\n");
 }else{
  printf("No\n");
 }
 printf("%g\n", (x-y)-z); //3.55112e^-17

 return 0;
}
```

Polynomials:

a) $3x^3 + 4x^2 + 9x + 2$
general form: f(x) = $a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$
We can represent a polynomial of degree n - 1 by an array of n elements;
double A[4] = 2.0, 7.0,4.0,3.0

```c
#include <stdio.h>
//Horner's method
//n-2 multiplications
//n-2 additions
//f[4] = {2.0, 9.0, 4.0, 3.0}

double eval(double f[], int n, double x){
  int i ;
  double y; //answer
  assert(n >0);
  y = f[n-1];

 for(i = n-2; i >= 0; --i){
  y = (y*x) + f[i];
 }

  return y;
}

int main(void){
  double f[4] = {2,9,4,3};

  printf("%g\n", eval(f,4,0); // prints: 2
  printf("%g\n", eval(f,4,1.0); // prints: 18
  printf("%g\n", eval(f,4,-1.0); //prints: -6

  return 0;
}
```

Natural way is to compute $x, x^2, x^3, ...x^{n-1}$
need n-2 multiplications to complete
then need to multiply by each of the constants
which gives another n-1 multiplications
finally add them all up
cost n-1 additions

Horner's method

$3x^3 + 4x^2 + 9x + 2$

$2 + x(9 + x(4 + 3x))$