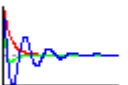


Discussion 2

- Functions
- Comparing equality of floating point values
- Complex Numbers
 - `complex()`
 - `polar()`
- Matrices and linear algebra

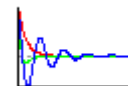


Functions

- Function Prototype

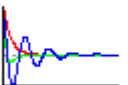
```
return_type function_name ( { argument_type [argument_name] , } );
```
- Function Definition

```
return_type function_name ( {argument_type [argument_name] , } )  
{  
    statements;  
    return return_value;  
}
```
- Common Errors
 - return_value type does not match return_type
 - Function name for prototype does not match definition
 - Argument types in prototype do not match those in definition



Function Example: Function Definition

```
int is_prime(int num); /* Define this function later */  
  
/* This function returns '1' if input argument is prime. Otherwise, it returns  
 * zero. */  
int is_prime(int num)  
{  
    int i;  
    for(i = 2; i < (sqrt(num) + 1); i++) {  
        if ( (num % i) == 0) { // If this is zero, then the number cannot be prime  
            return 0;  
        }  
    }  
    return 1;  
}
```



Function Example: Function Usage

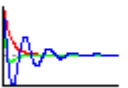
```
int main()
{
    int num_primes = 0;
    int i = 1;
    int n;
    printf("Which prime number do you wish to find? : ");
    scanf("%d", &n);
    printf("Calculating...\n");
    for(i = 1; num_primes < n; i++) {
        /* For each iteration, check whether i is prime. If it
is, increment
        * our variable keeping track of the number of primes */
        if( is_prime(i) ) {
            num_primes++;
        }
    }

    i--;
    printf("The prime is %d\n", i);
    return 0;
}
```



Function Example: Output

```
$ ch ./nth_prime.c
Which prime number do you wish to find? : 100
Calculating...
The prime is 541
$ ch ./nth_prime.c
Which prime number do you wish to find? : 200
Calculating...
The prime is 1223
$
```



Floating Point Numbers

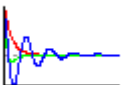
- Note that the standard equality operator, “==”, is rarely used with floating point numbers.
- Due to rounding and precision errors, two floating point numbers may not evaluate to being equal on a digital computer, although they may be equal from a mathematical standpoint.



Floating Point Numbers

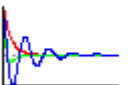
- The preferred way to test the equality of floating point numbers on a computer is by using code similar to the following:

```
#include <math.h> /* For FLT_EPSILON */  
float x1, x2;  
/* ... */  
/* Test to see if x1 == x2 */  
if (fabs(x1-x2) < FLT_EPSILON) {  
    /* Code to perform if x1 == x2 */  
}
```



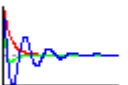
Floating Point Numbers

- Different floating point types have different epsilons:
- float : FLT_EPSILON
- double : DBL_EPSILON
- long double: LDBL_EPSILON
- They are all defined in <math.h>



Complex Numbers

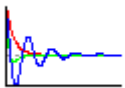
- There are two commonly used methods for initializing complex numbers.
 - The `complex()` function initializes a complex number by specifying its real and imaginary parts.
 - The `polar()` function initializes a complex number by specifying its magnitude and angle.



Complex Numbers

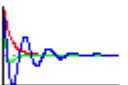
- Example
 - Write a program to determine the real part, imaginary part, magnitude, and phase angle of the following equation:

$$z = \frac{1+2i}{3-4i} + (5-6i)(7+8i)e^{i\pi/2}$$



Complex Numbers

```
/* *****  
 * File: ex4.c *  
 * Author: Yu-Cheng Chou *  
 * Date: 2007/1/24 *  
 ***** */  
#include <stdio.h>  
#include <tgmath.h>  
#include <complex.h>  
  
int main() {  
    double complex z;  
  
    /* The following two lines are equivalent */  
    z = complex(1, 2)/complex(3, -4) + complex(5, -6)*complex(7, 8) + exp(I*M_PI/2);  
    //z = complex(1, 2)/complex(3, -4) + complex(5, -6)*complex(7, 8) + polar(1, M_PI/2);  
  
    printf("real(z) = %.3f\n", real(z));  
    printf("imag(z) = %.3f\n", imag(z));  
    printf("abs(z) = %.3f\n", abs(z));  
    printf("carg(z) = %.3f\n", carg(z));  
  
    return 0;  
}
```



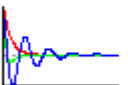
Complex Numbers

```
$ ch ./ex4_complex_numbers2.c  
real(z) = 82.800  
imag(z) = -0.600  
abs(z) = 82.802  
carg(z) = -0.007  
$
```



Matrices and Linear Algebra

- Computational arrays may be initialized in Ch with the 'array' keyword
- Computational arrays may be added, subtracted, multiplied, and divided with each other. Each operation is performed as matrix operations.



Matrices and Linear Algebra

- Example:
 - Find $x = 2ABb + Ab$ given the following:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

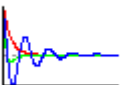
$$B = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



Matrices and Linear Algebra

```
/*  
 *   File: ex6.ch  
 * Author: Yu-Cheng Chou  
 *   Date: 2007/1/24  
 */  
  
#include <stdio.h>  
#include <array.h>  
  
int main()  
{  
    array int A[2][2] = {1, 2, 3, 4},  
              B[2][2] = {3, 4, 1, 2},  
              b[2][1] = {1, 2},  
              x[2][1];  
  
    x = 2*A*B*b + A*b;  
  
    printf("x =\n%d\n", x);  
  
    return 0;  
}
```



Matrices and Linear Algebra

- Example Output

```
$ ch ./ex6_matrix_operations.ch  
x =  
47  
117  
$
```

