Software Development Cycle (Simplified)

- Edit
- Compile/Link
- Execute
- Debug
/*
 * Converts distance in miles to kilometers.
 */
#include <stdio.h>              /* printf, scanf definitions */
define KMS_PER_MILE 1.609      /* conversion constant */

int
main(void)
{
    double miles,    /* input - distance in miles. */
            kms;    /* output - distance in kilometers */

    /* Get the distance in miles. */
    printf("Enter the distance in miles> ");
    scanf("%lf", &miles);

    /* Convert the distance to kilometers. */
    kms = KMS_PER_MILE * miles;

    /* Display the distance in kilometers. */
    printf("That equals %f kilometers.\n", kms);

    return (0);
}
Simple Data Types

- void
- Integer types (signed or unsigned):
  - char, short int, int, long int
  - char is an 8 bit (=1 byte) number
- Floating-point types:
  - float, double
- No Boolean types
  - Use 0=False and anything else(usually 1)=True
Initializers

- int j=1;
- char c='a';
- Declare type of variable and assign a value to it
Assignment Statements

- **FORM:** variable = expression;
- **Example:** x = y + z + 2.0;
- a=b=c=1;
- Same as a=(b=(c=1));
- The whole expression c=1 has value equal to 1
- The whole expression b=c=1 has value equal to 1
- Also a=b=c=d+1;
  - Whole expression c=d+1 has value d+1 etc.
  - But cannot write a=b=c+1=d+1
    - Left hand side of any assignment must be “lvalue”; c + 1 is not a valid lvalue
More on Assignment Statements

- Example 1:

  ```
  a+=1;  /* same as a = a +1; */
  ```

- Example 2:

  ```
  b*=5;  /same as b=b*5; */
  ```
Operators and “Punctuators”

- These are special characters with particular meanings, potentially depending on context.
- `+`, `−`, `∗`, `/` are the usual arithmetic operators, and can be applied to integer types as well as floating-number types.
- `%` is “mod operator”: `a % b` is equal to the remainder when `a` is divided by `b`, and returns a value in the range `0 .. (b − 1)`. We won’t worry about what happens when either `a` or `b` is non-positive.
- `%` character has different meaning in `printf( “%d”, a );`
- Punctuators include parentheses, braces, semicolons …
Precedence of Operators

- You may have learned about this in the third grade:
  1 + 2 * 3 has the value of 1 + (2 * 3)
- If we want the addition to be performed first, must parenthesize: (1 + 2) * 3.
- We say that * has a higher precedence than +.
- See page 63 and 64 for the “Rules for Evaluating Expressions”
Compound Statement

```c
{
    definitions-and-declarations (optional)
    Statement-list
}
```

- Used for grouping as function body and to restrict identifier visibility
- Note: no semicolon after closing bracket
  - But every statement in C must be followed by ;
#include <stdio.h>

void main()
{
    int x = 7, y;
    char id;

    printf( "Enter a letter, by which I will
identify you\n" );
    scanf( "%c", &id );
    printf( "Now enter a number!\n" );
    scanf( "%d", &y );
    printf( "%c, if nothing else, I know this:" , id);
    printf( "%d times %d is %d\n", x, y, x*y );
}
Input & Output functions

- Use printf, putchar etc. to write to the screen
- Use scanf, getchar etc. to read from the keyboard or console
- Note the & associated with the variable name in scanf
  scanf(%d, &number);
  scanf(%lf,&x);
Formatting the program output

- Type int: use \%d, \%2d, \%3d ...
- What happens if num=1000 and you write: printf("Number = \%1d\n", num);

- Type float or double: use \%f, \%6.2f, \%7.4f
  - First number indicates total number of digits (including the decimal point)
  - Second number indicates number of digits below the decimal point
Interactive vs Batch mode

- In interactive mode, user interacts with the program and types in data when it is running.
- In batch mode, the program scans its data from a data file prepared beforehand and (possibly writes the results to another data file)
I/O Redirection

- In Unix or MS-DOS operating system, use
  - `<` for redirecting input
  - `>` for redirecting output
- Example: `convert < myinputdata`

- Will talk about the syntax for opening and closing files after we’ve discussed pointer. (See pp.78 of text in case you are curious.)
Common Programming Errors

- Syntax errors: such as missing (  
- Run-time errors: such as divide by zero error  
- Undetected errors  
- Logic errors
Top-down Design

- Decomposing a complex problem into smaller, easier to solve unit.
Your Homework

- Read Chapter 1 and 2 of textbook
- Write a program to convert a temperature in degrees Fahrenheit to degrees in Celsius. (P.91, HW 2.1)

Program input: int fahrenheit;
Program output: double celsius;
Note: celius = 5/9 (fahrenheit –32)