Data Structures in C

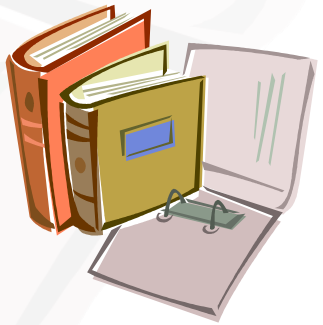
Prof. Georg Feil

Formatted Input & Output

Summer 2018

Reading Assignments

- C for Programmers (supplementary textbook)
  Chapter 9: Formatted Input/Output
  - Sections 9.1 – 9.5, 9.8, 9.11
  - Also chapter 4 section 4.12



printf() & scanf()

- Console output and input is done using the C library
  - `printf()` function for output
  - `scanf()` function for input
- Using `#include <stdio.h>` gives your program access to these input and output functions and related items in the C library
- Both of these functions accept a variable number of parameters
  - The first parameter must be a string, which may contain format specifiers like %d (indicates an integer or decimal value)
  - Additional parameters after the string are variables or expressions to print (printf), or variables to input (scanf)

printf() & scanf()

- There should be one additional parameter for each format specifier
  - When printing (printf), parameters are processed in order and their values inserted into the output string
  - When reading (scanf), values typed by the user are assigned to parameters in order

- Example:

```
printf("I am %d years old.\n", age);
```

Value of variable 'age' will be inserted here

"newline", means go to the next line

Format Specifiers for printf() & scanf()

- Format specifiers start with a **%** and are followed by one or a few letters, with no spaces in between
  - Can also include field width (see "C for Programmers" section 9.8)

Data Type	Format Specifier
int	%d
long int	%ld
float	%f
double	%lf
char	%c
C-style string	%s

See the last slide for more format specifiers

- Note that `%f` can be used for 'double' in printf, but not scanf
- Example:

```
printf("The employee makes $%f per year\n",
        salary);
```

Format Specifiers for printf() & scanf()

- Example of print and input:

```
int year = 2001;
double population;
printf("Please enter the population of Canada in %d\n",
       year);
scanf("%lf", &population);
```

- Important:
  - Variable names in `scanf` must be preceded by & (except pointers)
  - The `scanf` format string should not have `\n` in it, only conversion specifiers possibly separated by spaces
  - You can print or input more than one variable at a time using more format specifiers starting with %

scanf must use %f for float and %lf for double

printf may use %f for both

Testing if scanf() worked

- If you don't type in the right kind of data then scanf() may not be able to read it
  - In Java this would throw an exception and kill the program unless you use try-catch
  - C is more gentle, we can check if scanf() was able to read the value
- The return value of scanf() is how many variables were successfully read in or "parsed". We have only one variable in this example, so if scanf() doesn't return 1, there was a problem.

```
int year = 2001;
double population;
printf("Please enter the population of Canada in %d\n",
       year);
int parseCount = scanf("%lf", &population);
if (parseCount != 1) {
    printf("Read error\n");
}
```

Remember, no "\n" with scanf

Standard Input and Output

- Every C console program has three standard input/output device streams
  - Standard input (stdin): For reading text input
  - Standard output (stdout): For printing regular text
  - Standard error (stderr): For printing error messages

- By default printf() uses the standard output (stdout), and scanf uses the standard input (stdin).
- In Linux you can redirect the standard input and output streams when you run a console program from the shell (>, <, |)
  - Redirecting means sending the output to a file or another program, or reading the input from a file or another program, instead of the console
- You can discard buffered input (text typed by the user but not yet read by the program) like this:

```
fflush(stdin); // Useful to recover after "junk" input
```

Example 1: Extending Hello World

- Extend Hello World to:
  - Calculate Pi using a simple formula (355/113)
  - Input a floating point factor to multiply by
  - Calculate and print out Pi multiplied by the factor

Example 1: Extending Hello World

```
// The mathematical constant Pi (we will calculate it)
// This is a global variable, so name starts with upper case
double Pi;

int main(int argc, char** argv)
{
    printf("Hello this is the Pi Program\n");

}
```

Example 1: Extending Hello World

```
// The mathematical constant Pi (we will calculate it)
// This is a global variable, so name starts with upper case
double Pi;

int main(int argc, char** argv)
{
    printf("Hello this is the Pi Program\n");

    Pi = 355.0 / 113.0;
    printf("Pi is about %f\n", Pi);

}
```

Example 1: Extending Hello World

```
// The mathematical constant Pi (we will calculate it)
// This is a global variable, so name starts with upper case
double Pi;

int main(int argc, char** argv)
{
    printf("Hello this is the Pi Program\n");

    Pi = 355.0 / 113.0;
    printf("Pi is about %f\n", Pi);

    // Input how much to multiply by (this is a local variable)
    double multiplier = 0;
    printf("Please enter how much to multiply Pi by\n");
    scanf("%lf", &multiplier);
}
```

Example 1: Extending Hello World

```
// The mathematical constant Pi (we will calculate it)
// This is a global variable, so name starts with upper case
double Pi;

int main(int argc, char** argv)
{
    printf("Hello this is the Pi Program\n");

    Pi = 355.0 / 113.0;
    printf("Pi is about %f\n", Pi);

    // Input how much to multiply by (this is a local variable)
    double multiplier = 0;
    printf("Please enter how much to multiply Pi by\n");
    scanf("%lf", &multiplier);

    double manyPi = multiplier * Pi;
    printf("Pi multiplied by %f is %f\n", multiplier, manyPi);

    return 0;
}
```

Exercise 1: Extending Hello World

- The program on the previous slide doesn't check if scanf actually worked
  - Update the program to print an error message and quit on bad input

- If you're done early… update the program to ask for input again on errors (careful, it's a bit tricky!)

Additional Format Specifiers

- Some additional useful format specifiers
  (see the module on variables for information on C data types)

Data Type	Format Specifier
unsigned int	%u
unsigned long int	%lu
display int in Hex	%x
long long int	%lld
print float or double in scientific notation	%e
print float or double in "auto" format	%g

On Windows may need to use %I64d