CS 210 **Sept. 29, 2016**
Day 10

Example – Parameter passage
Write a function to simulate throwing two dice. The function will returns a void and the value of the two dice will come back as output parameters. Your main program will print the value of the two dice on the console.

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void RollTwoDice(int *d1, int *d2);
int main()
{
    int d1, d2;
    //seed random number genrator with present time
    srand((unsigned)time(NULL));
    RollTwoDice(&d1, &d2);
    printf("%d, %d \n", d1, d2);
    return 0;
}
void RollTwoDice(int *d1,int *d2)
{
    *d1 = rand() % 6 + 1;
    *d2 = rand() % 6 + 1;
}
```

Ch 11 File pointers and text files

Input/Output Files

- text file
  - a named collection of characters saved in secondary storeage

- input (output) stream
  - continuous stream of character codes representing textual input (or output) data

The keyboard and Screen as Text Streams

- stdin
  - system file pointer for keyboard's input stream
- stdout, stderr
  - system file pointers for screen's output stream

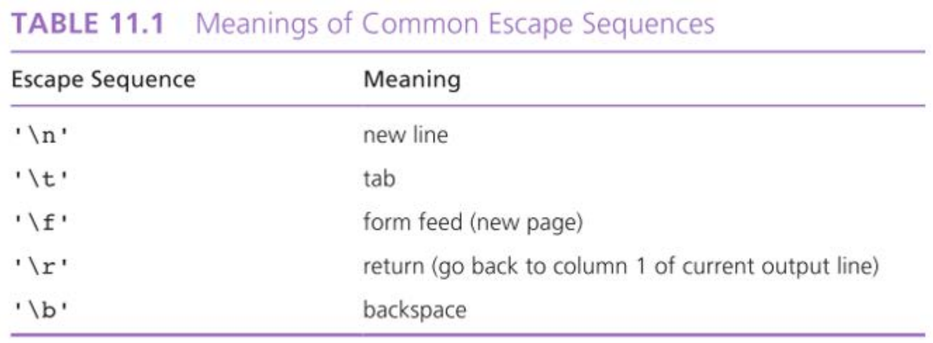

TABLE 11.1 Meanings of Common Escape Sequences

Escape Sequence	Meaning
`'\n'`	new line
`'\t'`	tab
`'\f'`	form feed (new page)
`'\r'`	return (go back to column 1 of current output line)
`'\b'`	backspace

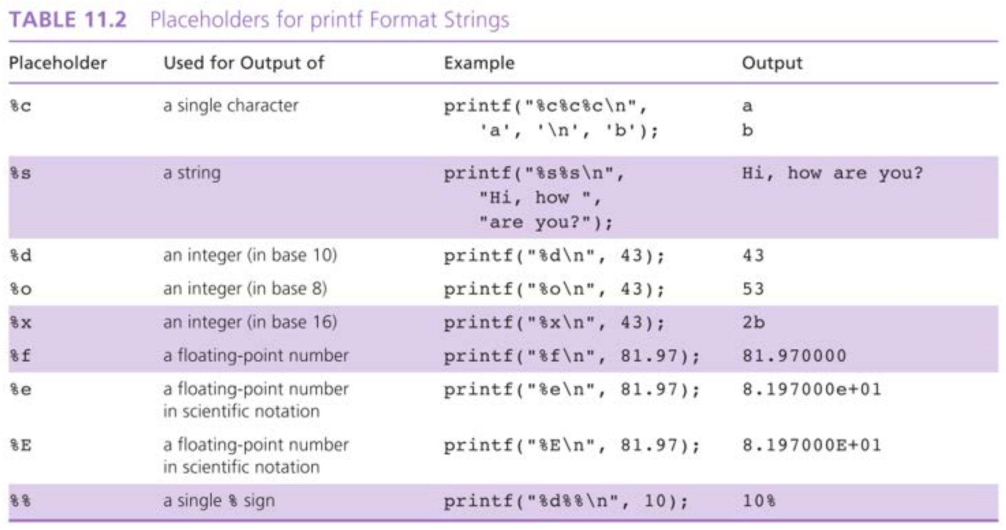

TABLE 11.2 Placeholders for printf Format Strings

Placeholder	Used for Output of	Example	Output
`%c`	a single character	`printf("%c%c%c\n", 'a', '\n', 'b');`	`a` `b`
`%s`	a string	`printf("%s%s\n", "Hi, how ", "are you?");`	`Hi, how are you?`
`%d`	an integer (in base 10)	`printf("%d\n", 43);`	`43`
`%o`	an integer (in base 8)	`printf("%o\n", 43);`	`53`
`%x`	an integer (in base 16)	`printf("%x\n", 43);`	`2b`
`%f`	a floating-point number	`printf("%f\n", 81.97);`	`81.970000`
`%e`	a floating-point number in scientific notation	`printf("%e\n", 81.97);`	`8.197000e+01`
`%E`	a floating-point number in scientific notation	`printf("%E\n", 81.97);`	`8.197000E+01`
`%%`	a single % sign	`printf("%d%%\n", 10);`	`10%`

TABLE 11.3 Designating Field Width, Justification, and Precision in Format Strings

Example	Meaning of Highlighted Format String Fragment	Output Produced
`printf("%5d%4d\n", 100, 2);`	Display an integer right-justified in a field of five columns.	`▮▮100▮▮▮2`
`printf ("%2d with label\n", 5210);`	Display an integer in a field of two columns. Note: Field is too small.	`5210▮with▮label`
`printf("%-16s%d\n", "Jeri R. Hanly", 28);`	Display a string left-justified in a field of 16 columns.	`Jeri▮R.▮Hanly▮▮▮28`
`printf("%15f\n", 981.48);`	Display a floating-point number right-justified in a field of 15 columns.	`▮▮▮▮▮981.480000`
`printf("%10.3f\n", 981.48);`	Display a floating-point number right-justified in a field of ten columns, with three digits to the right of the decimal point.	`▮▮▮981.480`
`printf("%7.1f\n", 981.48);`	Display a floating-point number right-justified in a field of seven columns, with one digit to the right of the decimal point.	`▮▮981.5`
`printf("%12.3e\n", 981.48);`	Display a floating-point number in scientific notation right-justified in a field of 12 columns, with three digits to the right of the decimal point and a lowercase `e` before the exponent.	`▮▮▮9.815e+02`
`printf("%.5E\n", 0.098148);`	Display a floating-point number in scientific notation, with five digits to the right of the decimal point and an uppercase `E` before the exponent.	`9.81480E-02`

TABLE 11.4 Comparison of I/O with Standard Files and I/O with User-Defined File Pointers

Line	Functions That Access stdin and stdout	Functions That Can Access Any Text File
1	`scanf("%d", &num);`	`fscanf(infilep, "%d", &num);`
2	`printf ("Number = %d\n", num);`	`fprintf(outfilep, "Number = %d\n", num);`
3	`ch = getchar();`	`ch = getc(infilep);`
4	`putchar(ch);`	`putc(ch, outfilep);`

Pointers to Files

- C allows a program to explicitly name a file for input or output.
- Declare file pointers:
  - FILE *inp;   /* pointer to input file */
  - FILE *outp;   /* pointer to output file */
- Prepare for input or output before permitting access:
  - inp = fopen(“infile.txt”, “r”);
  - outp = fopen(“outfile.txt”, “w”);

Pointers to Files

- fscanf
  - file equivalent of scanf
  - fscanf(inp, “%1f”, &item);
- fprintf
  - file equivalent of printf
  - fprintf(outp, “%.2f\n”, item);
- closing a file when done
  - fclose(inp);
  - fclose(outp);

CS 210 **Fall, 2016**

Reading and writing files with VS 2015

The textbook (pp. 320-322) shows how to read and write files using pointers with `fopen`. If you use `fopen` in Visual Studio you will get an error indicating that `fopen` is not safe and suggesting you use `fopen_s` in its place. You can still use fopen but you need to turn off the errors and warnings first. To do this you need to add the following line at the top of your program which uses pointers to files:

```
#pragma warning(disable:4996)
```

Here is an example of a C program which uses `fopen` to create a file, write some ints into it, read the file, and print its contents to the console.

```
#include<stdio.h>
#pragma warning(disable:4996)
int main()
{
    int i;
    FILE *inp;  //Declare pointers to in
    FILE *outp; //  and out files
    int dataIn, status;
    //Open the file for output
    outp = fopen("MyFile.txt", "w");
    //Write five ints to the file
    for(i=0;i<5;i++)
       fprintf(outp, "%d\n", i);
    fclose(outp);   //close the file
    //reopen for input
    inp = fopen("MyFile.txt", "r");
    status = fscanf(inp, "%d", &dataIn);
    //read the file and print to console
    while(status == 1)
       {
        printf("%d\n", dataIn);
        status = fscanf(inp, "%d", &dataIn);
       }
    fclose(inp);
}
```

We can also use `fopen_s` which is a secure open statement. The syntax is slightly different. The program below is the same as that above but it uses `fopen_s` in place of `fopen`.

```c
#include<stdio.h>
int main()
{
    int i, err;
    FILE *inp;  //Declare pointers to in
    FILE *outp; //  and out files
    int dataIn, status;
    //Open the file for output
    err = fopen_s(&outp, "MyData.txt", "w");
    //Write five ints to the file
    for(i=0;i<5;i++)
       fprintf(outp, "%d\n", i);
    fclose(outp);   //close the file
    //reopen for input
    err = fopen_s(&inp, "MyData.txt", "r");
    status = fscanf_s(inp, "%d", &dataIn);
    //read the file and print to console
    while(status == 1)
       {
        printf("%d\n", dataIn);
        status = fscanf_s(inp, "%d", &dataIn);
       }
    fclose(inp);
}
```

Example
Write a function which writes 1000 random integers in the range 0 ≤ x ≤ 100 to file called "Numbers.txt". Your function should return a status variable that is 0 only if the file is successfully written. Otherwise it should return a 1. Write a main program which calls your function. The main program should write a message to the screen to indicate whether or not the file write was successful.

```
#include<stdio.h>
#include<stdlib.h>
int WriteNumbers();
int main()
{
    if(WriteNumbers() == 0)
        printf("File written successfully. \n");
    else
        printf("Error writting file.\n");
}
int WriteNumbers()
{
    //Writes 1000 random ints between 0 and 100
    //  to "Numbers.txt"
       int err, i, r;
    FILE *outp; //  and out files
       err = fopen_s(&outp, "Numbers.txt", "w");
    if(err != 0)
        return 1;
    srand(23);
    for(i=0;i<1000;i++)
       {
        r = rand();
        r = r % 101;  //0 to 100
        fprintf(outp, "%d\n", r);
    }
    fclose(outp);
    return 0;

}
```

CS 210 **September 29, 2016**
File Operations

Down load the source file for the example above from the website at:
http://csserver.evansville.edu/~blandfor/CS210/Day10WriteNumbers.docx

Create a project with this source file, compile it, and verify that it runs successfully.

Add a second function to the project which reopens the file "Numbers.txt" for input. Read the numbers in the file and print their average. Your function should add the numbers in the file and count the number of random numbers that were read in. Your function should return the average value of all of the ints in the file as a double. Print this double in the main program.

Turn in a printed copy of your source file.