Chapter 14: Advanced Topics

Variable-Length Argument Lists

* The function prototype for "printf" is

```
        int printf(const char *format, ...);
```

* The ellipsis ("..."), which must be at the end of the parameter list, indicates that the function receives a variable number of arguments of any type.
* The macro and definitions of the variable arguments header "stdarg.h" provide the capabilities necessary to build these functions.

va_list	A type suitable for holding information needed by macros "va_start", "va_arg", and "va_end". To access the arguments in a variable-length argument list, and object type "va_list" must be declared.
va_start	A macro that is invoked before the arguments of a variable-length argument list can be accessed.
va_arg	A macro that expands to an expression of the value and type of the next argument in the variable-length argument list.
va_end	A macro that facilitates a normal return from a function whose variable-length argument list was referred to by the "va_start" macro.

* E.g.

```
        #include <stdio.h>
        #include <stdarg.h>

        double average(int, ...);

        main()
        {
            double w = 37.5, x = 22.5, y = 1.7, z = 10.2;

            printf("%s%.1f\n%s%.1f\n%s%.1f\n%s%.1f\n\n",
                    "w = ", w, "x = ", x, "y = ", y, "z = ", z);
            printf("%s%.3f\n%s%.3f\n%s%.3f\n",
            "The average of w and x is ",
            average(2, w, x),
            "The average of w, x, and y is ",
            average(3, w, x, y),
            "The average of w, x, y, and z is ",
            average(4, w, x, y, z));

            return 0;
        }


        double average(int i, ...)
        {
            double total = 0;
            int j;
            va_list ap;

            va_start(ap, i);

            for (j = 1; j <= i; j++)
                total += va_arg(ap, double);
            va_end(ap);
            return total / i;
        }
```

Using Command-Line Arguments
----
* It is possible to pass arguments to "main" from a command line by including parameter "int argc" and "char *argv[]" in the parameter list of "main".

* Parameter "argc" receives the number of command-line arguments.
* Parameter "argv" is an array of string in which the actual command-line arguments are stored.
* E.g.

```
        #include <stdio.h>

        main(int argc, char *argv[])
        {
            FILE *inFilePtr, *outFilePtr;
            int c;

            if (argc != 3)
                printf("Usage: copy infile outfile\n");
            else
            if ((inFilePtr = fopen(argv[1], "r")) != NULL)

                if ((outFilePtr = fopen(argv[2], "w")) != NULL)

                while ((c = fgetc(inFilePtr)) != EOF)
                    fputc(c, outFilePtr);

                else
                    printf("File \"%s\" could not be opened\n",
                    argv[2]);

            else
                printf("File \"%s\" could not be opened\n", argv[1]);

            return 0;
        }
```

Notes on Compiling Multiple-Source-File Programs
----
* Global variables are accessible to functions in other files, however, the global variables must be declared in each file in which they are used.

* E.g., if integer variable "flag" is defined in one file, and refer to it in a second file, the second file must contain the declaration

```
        extern int flag;
```

prior to the variable's use in that file.

* The storage class specifier "extern" indicates to the compiler that variable "flag" is defined either later in the same file or in a different file.
* The compiler informs the linker that unresolved references to

variable "flag" appear in the file.

* The function prototype can be included in each file in which the function is invoked, and compiling the files together.
* The function can be implementated in one of the files.
* E.g., "printf" and "scanf" in "stdio.h".

More on Files

* C provides capabilities for processing binary files, when the file is opened in a binary file mode.

```
        rb      Open a binary file for reading.
        wb      Open a binary file for writing.
        ...
```