Tutorial: Programming in C

Important Note: Non- ANSI Compilers

Some of you may have a tough time compiling the program given in Chapter 4. In most cases this will be because your compiler does not support ANSI Standard C. Instead, it most likely supports the older K&R First Edition standard. This special note provides an explanation of how to adapt the code of Chapter 4, and subsequent lessons, for non-ANSI compilers.

First of all, though, notes on specific machines and compilers:

-- Those of you with IBM compatibles, Macintoshes, Amigas and the like already have ANSI C in practically all cases. You can ignore this note.

-- Those of you running Unix systems who find that the compiler you access by typing "cc" is not an ANSI compiler (on many machines it is, try it on the program of Chapter 4 first before you worry) should try using "gcc" or "acc". If either succeeds, you have an ANSI compiler-- in the first case the Gnu gcc compiler, which is free, in the second case a commercial ANSI C compiler (such as Sun's). Use acc or gcc from now on instead of cc.

Now, if none of the above does the trick for you, here are notes on how to adapt the programs for non-ANSI use.

1. Function prototypes:

The program of Chapter 4 provides a "prototype" for each function. These prototypes look like this (for example):

```
void card_print(int c);
```

Unfortunately, pre-ANSI compilers do not support the list of arguments in a prototype. Replace the prototype with this:

```
void card_print();
```

(In other words, the same thing, but with the list of arguments removed.)

2. Function definitions:

The program of Chapter 4 also includes the actual code for several new functions. Pre-ANSI compilers have a different way of specifying the arguments to a function, however.

An example:

```
void deck_draw(int deck[], int *card_top) { ... code ... }
```

should look like this in pre-ANSI C:

```
void deck_draw(deck, card_top) int deck[]; int *card_top; { ... code ... }
```

(Note that the opening "{" does not appear until *after* the argument types are given.)

There are other differences, but these two are the most problematic, and by making the changes specified above you should be able to make the program in Chapter 4 (and subsequent programs) work with your non-ANSI compiler. But if your system completely lacks an ANSI compiler, I strongly encourage you to ask your sysadmin to get GNU gcc, which is freely available for Unix systems.