Lecture 15

CSE 110

27 July 1992

1 `fgets`

`fgets` is an awfully useful function. It takes three arguments. The third argument is a stream. `fgets` reads characters from the stream. The first argument is a pointer to a space in which `fgets` can store the characters it reads. The second argument is an <`int`>; it says how big that space is. If the second argument is n, then `fgets` will read no more than $n - 1$ characters from the stream and store them into the buffer. It might read fewer, because `fgets` stops after it reads a newline character.

After storing the characters it read into the buffer, it sticks a `NUL` character on the end. (That's why it reads at most $n-1$ characters; it needs to save space for the `NUL`.)

If it succeeds, `fgets` returns its first argument, which is not usually very useful since we already know what the first argument was. But if it fails, `fgets` returns the `NULL` pointer.

In short, `fgets` reads a line of input from a certain stream, stopping when the buffer it's storing the input into is full.

1.1 `gets`

`fgets` has a dysfunctional little brother, `gets`. `gets` only reads from the standard input, its only argument is a pointer to the space to in which you want the input stored.

The question: How does `gets` know how much space you've arranged for it?

The answer: It doesn't.

If the line that `gets` is reading is too big to fit in the array you've provided, `gets` happily writes past the end of the array and destroys who-knows-what.

For this reason, serious programmers never use `gets`.

2 Debugging Facilities

Turbo-C++ has reasonably good debugging facilities. You can step through part of your program one statement at a time, watching how certain variables change, and you can run your program normally and have it pause when control reaches a certain line.

This is all covered in detail in your texbook, pages 723–733.

2.1 Stepping

The `F7` and `F8` keys will run your program one step at a time; each time you press one, your program will run one more statement. When you press `F7` or `F8`, the program runs, and stops again when control reaches the next line.

The difference between `F7` and `F8` is that if the current line contains a function call, the pressing `F7` will go stop at the next executed line, which is inside the called function, but `F8` will run the function calls and stop only when control reaches the next line in the calling function. `F8` skips the details of the function calls; they're run, but you don't have to step into them and go through the one step at a time like you do with `F7`.

2.2 Inspecting Variables

Under the `Debug` menu is a selection called `Inspect`. If you select this menu item, you'll get a dialog box that lest you enter an expression. A window will pop up on the screen that displays the value of the expression. For example, if you wanted to see the current value of the expression `*ptr`, type `*ptr` and a window will pop up with the value of `*ptr`. The value in this window will change if the value of `*ptr` changes as you step through the program.

2.3 Setting Breakpoints

Normally when you pick `Run` from the `Run` menu, the program runs without stopping. You can set a *breakpoint* at a line of code, and execution will pause when control reaches that line. Then you can examine the values of variables with the inspector, step through critical sections of code one step at a time, or set or clear more breakpoints.

To set a breakpoint on a line, move the cursor to that line and choose `Toggle Breakpoint` from the `Debug` menu. When you `Run` the program, control will stop when control reaches the line with the breakpoint. To continue execution normally, choose `Run` again; the program will pick up where it left off and run until it hits another breakpoint.

To remove a breakpoint, move the cursor to the line with the breakpoint and choose `Toggle Breakpoint` again.

2.4 Restarting the Program

The `Run` command only starts your program from the beginning if you've changed it or if it's completely finished. If you've stopped at a breakpoint or stepped partway through the program, the `Run` command tells Turbo-C++ to continue running the program where it left off.

If you really to want to start all over again from the beginning, choose `Program Reset` from the `Run` menu.

2.5 Setting Command-Line Arguments

You don't have to break into a DOS shell to run your program with command-line arguments. If you choose `Arguments` under the `Run` menu, you can enter the arguments you want to run your program with.