

Slide 1

Administrivia

- Reminder: Homework 1 due Thursday.
All homework is considered pledged work. Write “pledged” on hardcopy work, and include it in comments for programming assignments.
- Tentative dates for quizzes on Web. First one next Tuesday.

Slide 2

Now Back to C — Functions

- C programs are organized in terms of *functions*.
- More about this later; for now, they're a little like mathematical functions, except that evaluating them can have “side effects”.
For example, evaluating the library function `printf` has the side effect of writing some text to standard output (by default, displaying it in the terminal window).
- A complete C program must contain a function called `main`. When you type `a.out`, the operating system calls this function. The return value can be used to indicate whether the program succeeded.
- Let's look again at the “hello world” program . . .

Slide 3

“Hello World” Program, One More Time

- Historical/cultural aside: Among computer programmers, it's considered traditional that the first program one writes in a new language just prints “hello world” to the screen — maybe not the simplest possible program, but close. Particularly apt for C, because the tradition was begun by an early and still authoritative work on C (*The C Programming Language*, Kernighan and Ritchie).
- For now, okay to regard everything but the comments and what's inside the function `main` as stuff you have to have but aren't expected to understand yet.

Slide 4

Variables

- To do anything interesting in a program, we need some place to store input and intermediate values. (E.g., consider a simple program that asks the user for two numbers, adds them, and prints the result. It needs a place to hold the numbers and (maybe) their sum.)
- For this we use *variables*. Can think of them as boxes holding values. Each has a *name* and a *type*.
- Variable names follow rules for *identifiers* — letters, numbers, and underscores only, must start with letter or underscore, preferably letter. Case-sensitive.
- Variable types? To the computer, “it's all ones and zeros”; types say how we want to interpret them (integers? characters?), define what kinds of things we can do with a variable. Textbook lists C's built-in types. Some will work in `gcc` only with the `-std=c99` option.

Sidebar — Compiler Options

Slide 5

- Earlier I showed the simplest way to use `gcc` to compile a program. But there are many variations — *options*. Specify on the command line, ahead of name of input file.
- Some of the most useful:
 - `-Wall` and `-pedantic` warn you about dangerous and non-standard things.
 - `-std=c99` allows you to use full C99.
 - `-o` allows you to name the output file (default `a.out`).

Variables, Continued

Slide 6

- To use a variable in a program, we have to tell the compiler about it — *declare* it, giving its name and type. In C, declarations must come before code.
Examples:

```
int x;
int sum;
float number_with_fractional_part;
```
- We can then give it a value. Simplest way is with *assignment*.
- How to specify values? Textbook gives details. Examples:

```
x = 5;
number_with_fractional_part = 1.1;
```

On the right can also be an *expression*, something like a mathematical formula. (More next time.)

Output

- The “hello world” used `printf` to print some text. `printf` can do a lot more.
- For example, we can use it to print integers, e.g.,

```
printf("the value of x is %d\n", x);
```

Slide 7

Sidebar — Man Pages, Revisited

- As mentioned earlier, most commands — and many library functions — have “man pages” (short for “manual”). These are meant as online references rather than tutorials, so not always easy reading, but usually very complete.
- `man` program shows its output to you using a program intended for paging through text. On our systems, default is `less`. Keystroke commands include space to go forward, `b` to go back, `q` to quit. `h` for help — or, of course, you could read all about it (how?).
- Sometimes there are multiple commands/functions with the same name. `printf` is one. `man printf` tells you about the (command-line) command, not the C library function. To get all possibilities, `man -a printf`. To get the one for the library function, `man 3 printf`.

Slide 8

Input

- How to get values into a program? `scanf` library function. Example:

```
scanf ("%d", &x);
```

(What's the ampersand for? `scanf` needs to know not the value of `x`, but its location in memory. More about this when we talk about pointers later.)

Slide 9

Minute Essay

- Make your best guess at writing the lines of code that would be needed (just the "insides" of the `main` program) to declare an integer variable called `my_age`, give it a value of your age, and print its value in the form

```
I am 21 years old.
```

(Replace 21 with your age.)

Slide 10

Minute Essay Answer

- The needed lines of code would look something like this:

```
int my_age;  
my_age = 21;  
printf("I am %d years old\n", my_age);
```

Slide 11