Pitfall #2: "Functions" are Not Algorithms

The **`main`** function is **not necessarily an algorithm**.

For example, we can **write a program that runs forever** (never terminates, and never returns a value).

Algorithms must be finite
(see Patt & Patel).

Variable Declarations Allocate and Name Sets of Bits

Variable declarations
- allow the programmer to **name sets of bits**
- and to **associate a data type**

The declaration `int answer = 42;`

tells the compiler…
- to make space for a **32-bit 2's complement** number (an **`int`**),
- to initialize the bits to the bit pattern for 42,
- and to make use of those bits whenever a statement uses the **symbolic name answer**.

Variables in C are Sets of Bits (0s and 1s)

In C, a variable is a name for a set of bits.

The bits will (of course!) **always be 0s and 1s**.

But **variables in C can change value as the program executes**.

Other properties of a variable must be inferred from the program (in the example program, **answer** is always 42, because no statement changes **answer**).

Each Variable Has a Specific Data Type

Many languages (such as **C**) require that the programmer **specify a data type for each variable**.

A **C** compiler uses a variable's data type to interpret statements using that variable.

For example, a "+" operation in **C** might mean to add two sets of bits
- as **unsigned** bit patterns,
- as **2's complement** bit patterns, or
- as **IEEE single-precision floating-point** bit patterns.

The compiler generates the appropriate instructions.

