register int count_fast;  // request for a register variable

You've probably noticed the qualifying words "hint" and "request." The compiler doesn't have to honor the request. For example, the registers may already be occupied, or you might request a type that doesn't fit in a register. Many programmers feel that modern compilers are often smart enough not to need the hint. If you write a for loop, for example, the compiler might take it upon itself to use a register for the loop index.

If a variable is stored in a register, it doesn't have a memory address; therefore you can't apply the address operator to a register variable:

```c
void gromb(int *);
...
int x;
register int y;
gromb(&x);   // ok
gromb(&y);   // not allowed
```

Using register in the declaration is enough to invoke this restriction, even if the compiler doesn't actually use a register for the variable.

In short, an ordinary local variable, a local variable declared using auto, and a local variable declared using register all have automatic storage duration, local scope, and no linkage.

```c
short waffles;          // auto variable by default
auto short pancakes;    // explicitly auto
register int muffins;   // register variable
```

Declaring a local variable without a specifier is the same as declaring it with auto, and such a variable is typically handled by being placed on a memory stack. Using the register specifier is a hint that the variable will be heavily used, and the compiler may choose to use something other than the memory stack to hold it; for example, a CPU register.

### Static Duration Variables

C++, like C, provides static storage duration variables with three kinds of linkage: external