```c
// This is a very simple example with comments for discovering where
// the gcc compiler places certain C storage class variables

// initialized variables
static int foo1 = 42;
int foo2 = 43;
const int foo3 = 76;

// un-initialized variables
static int foo4;
int foo5;

void main()
{
    static volatile const int * const foo8;
}

// after compiling, hunt for the symbols using objdump
// objdump -C -t -j.data myfoo.exe | grep foo
// run this for each section header of interest

// DATA SECTION //
// the .data section is where your initialized data goes.
// This data consists of global and static variables that are
// initialized at compile time. It also includes string literals.
// Local variables are located on a thread's stack, and take no
// room in the .data or .bss sections.
// foo1, foo2

// RDATA (RODATA) SECTION //
// constants that are initialized may go here, or the text section
// foo3

// BSS SECTION //
// The .bss section is where any uninitialized static and global
// variables are stored. The linker combines all the .bss sections
// in the OBJ and LIB files into one .bss section in the EXE.
// foo4, foo5, foo6

// TEXT SECTION //
// The .text section is where all general-purpose code emitted by
// the compiler or assembler ends up.

// EDATA SECTION //
// The .edata section is a list of the functions and data that the
// PE file exports for other modules.
```