```c
#include <stdio.h>

// Variables declared outside any function are said to be in global scope
int p = 22;

// All functions can see global variables
void bar(int i){
    printf("Inside bar, p = %d\n", p);
    p = 25; // bar changes the global variable
}

// Unless those functions choose to shadow the global variable
// by defining a local variable of their own with the same name
// The new shadowing variable can be of any type, not necessarily
// The same type as the global variable
void foo(int i){
    float p = 33.5;
    i = 11;
    printf("Inside foo p = %f\n", p);
}

// Static variables retain their value across function calls
// They are not destroyed when the function returns
// Moreover, the statement declaring and initializing them is
// only executed the first time the function is called
void count(){
    static int calls = 1;
    printf("The function count was called %d times\n", calls++);
}

int main(){
    // Global variables are visible everywhere
    printf("Global p = %d\n", p);

    int i = 100;
    foo(i); // foo shadows the global variable p
    bar(i); // bar does not shadow the global variable p and changes it
    printf("Global p is now changed = %d\n", p);
    foo(p); // foo will not be affected by change of global variable

    int a = 42;
    printf("Outside loop, a = %d\n", a);

    // Good practice to shadow the counter variable of loops
    // This way we can prevent accidentally overwriting some useful
    // value the loop counter variable may have stored earlier
    for(int i = 0; i < 5; i++){
        double a = -1.50; // a shadows the a defined on line 41
        printf("Loop counter i = %d", i);
        printf(" inside loop a = %lf\n", a);
    }
    printf("Outside loop, a = %d\n", a);
    printf("Precious value of i is still intact = %d\n", i);

    // Main is not a reserved keyword so we can define a variable
    // named main as well
    int main = 77;
    printf("The value of the variable main = %d\n", main);

    count();
    count();
    count();

    int count = 90; // This will shadow the global function name count
    // Now count is a variable and not a function so the following
    // statement, if uncommented, will give a compilation error
    // count();
    return 0;
}
```