```c
#include <stdio.h>

int main() {

    int i;

    for (i = 1; i < 10000; i++) {

        printf("%d,", i);

    }

}
```

Before Code: The loop is explicitly defined with `for (i = 1; i < 10000; i++)`, which repeatedly prints numbers from 1 to 9999.

```c
main(i){i < 1e4 && printf("%d,", i) + main(++i);}
```

After Code:

1. **Implicit Function Declaration:** In `main(i)`, `i` is not explicitly declared with a data type, but the compiler treats it as an `int` automatically. This approach shortens the code, aiming for conciseness, though it's not recommended in modern C for clarity.
2. **Recursion:** The `+` operator ensures both `printf("%d,", i)` and `main(++i)` are evaluated. The function calls itself until `i` reaches 9999.
    - **Initial Call:** `main(1)`
        - `i = 1`, so `1 < 1e4` is true
        - `printf` prints `1,`
        - `main(2)` is called
    - **Next Call:** `main(2)`
        - `i = 2`, so `2 < 1e4` is true
        - `printf` prints `2,`
        - `main(3)` is called
    - This continues: `main(3)`, `main(4)`, ..., `main(9999)`

        - `printf` prints each number sequentially.
    - **Termination:** `main(10000)`
        - `10000 < 1e4` is false, so no further function calls occur, ending with `9999,`.

Note: Stack Overflow Risk Recursion can lead to deep stacks, increasing the risk of memory exhaustion or stack overflow. An infinite loop could occur if no termination condition exists, but in this case, the `i < 1e4` condition ensures safe termination.

Compilation Command Explanation

```
gcc -o count2 count2.c -std=c90
```

This command compiles `count2.c` using the C90 standard and creates an executable named `count2`.

- **`gcc`:** GNU Compiler Collection for C
- **`-o count2`:** Specifies the output executable name as `count2`.
- **`count2.c`:** The source file to compile.
- **`-std=c90`:** Tells the compiler to use the C90 standard.

Features of `-std=c90`

- **Function Declarations and Definitions:** Functions must have explicitly stated return types. If omitted, `int` is assumed. Parameter types must also be specified, though "implicit function declaration" was allowed in C90.
- **Standard Library Headers:** When using the `-std=c90` option, standard library headers like `#include <stdio.h>` and `#include <stdlib.h>` are essential. Explicitly including headers helps prevent implicit declarations of functions.

This standard, also known as ANSI C, was introduced in 1990 and aimed to define clear syntax and features, ensuring portability of C code across different systems.