Loop Unrolling

Loop unrolling is a code transformation that replicates the body of a loop and reduces the number of iterations, thereby decreasing loop overhead and increasing opportunities to improve the performance of the processor pipeline by reordering instructions. Unrolling is traditionally implemented by the code improvement phase of a compiler. It can be implemented at source level, however, if we are faced with the prospect of “hand optimizing” time-critical code on a system whose compiler is not up to the task. Unfortunately, if we replicate the body of a loop \( k \) times, we must deal with the possibility that the original number of loop iterations, \( n \), may not be a multiple of \( k \). Writing in C, and letting \( k = 4 \), we might transform the main loop of the following code from:

```c
i = 0; do {
    sum += A[i]; squares += A[i] * A[i]; i++;
} while (i < N);
```

to

```c
i=0; j=N/4; do {
    sum += A[i]; squares += A[i] * A[i]; i++;
    sum += A[i]; squares += A[i] * A[i]; i++;
    sum += A[i]; squares += A[i] * A[i]; i++;
    sum += A[i]; squares += A[i] * A[i]; i++;
} while (--j > 0);
```

In 1983, Tom Duff of Lucasfilm realized that code of this sort can be “simplified” in C by interleaving a switch statement and a loop. The result is rather startling, but perfectly valid C. It is known in programming folklore as “Duff’s device”:
i = 0; j = (N+3)/4;
switch (N%4) {
    case 0: do{ sum += A[i]; squares += A[i] * A[i]; i++;
    case 3:
    case 2:
    case 1:
        sum += A[i]; squares += A[i] * A[i]; i++;
        sum += A[i]; squares += A[i] * A[i]; i++;
        sum += A[i]; squares += A[i] * A[i]; i++;
    } while (--j > 0);
}

Duff announced his discovery with “a combination of pride and revulsion.” He noted that “Many people ... have said that the worst feature of C is that switches don’t break automatically before each case label. This code forms some sort of argument in that debate, but I’m not sure whether it’s for or against.”

Your Arguments

What do you think about “Duff’s device”? Is it reasonable to interleave a loop and a switch in this way? Should a programming language permit it? Is automatic fall-through ever a good idea? Discuss with each other, then submit your arguments (for and against) through the Google Form and be prepared to share your stance with the class.