iostream version:

```
#include <iostream>

int main(int argc, char **argv) {

  int parity = 0;
  int x;

  while (std::cin >> x)
    parity ^= x;
  std::cout << parity << std::endl;

  return 0;
}
```

scanf version

```
#include <stdio.h>

int main(int argc, char **argv) {

  int parity = 0;
  int x;

  while (1 == scanf("%d", &x))
    parity ^= x;
  printf("%d\n", parity);

  return 0;
}
```

Result: Using a third program, I generated a text file containing 33,280,276 random numbers. The execution times are:

```
iostream version:  24.3 seconds
scanf version:      6.4 seconds
```

The speed difference is largely due to the iostream I/O functions maintaining synchronization with the C I/O functions. We can turn this off with a call to

```
std::ios::sync_with_stdio(false);
```

```
#include <iostream>

int main(int argc, char **argv) {

  int parity = 0;
  int x;

  std::ios::sync_with_stdio(false);

  while (std::cin >> x)
    parity ^= x;
  std::cout << parity << std::endl;

  return 0;
}
```

New result:

```
iostream version:                        21.9 seconds
scanf version:                            6.8 seconds
iostream with sync_with_stdio(false):     5.5 seconds
```

C++ iostream wins! It turns out that this internal syncing / flushing is what normally slows down iostream i/o. If we're not mixing cstdio and iostream, we can turn it off, and then iostream is fastest.