# Arrays provide no safety

Unlike most programming languages, C++ does not protect you from mistakes like reading or writing from/to memory outside of the boundaries of an array. This is because checking that the index is inside of the bounds of the array takes an extra processing step and C++ is language that prioritizes performance. What happens when you access memory outside of the allocated space is unpredictable, your operating system might detect the illegal access and halt the application, or you might read a bunch of junk values into your application. In the example below I intentionally read values well past the end of an array of size 10, in a real application the results can be disastrous.

|  |  |
| --- | --- |
| #include <iostream>  #include <array>  using namespace std;  int main() {    const int SIZE = 10;    array<long, SIZE> items = {};    for (size\_t i=0; i<100; i++) {      cout << items[i] << " ";    }    return 0;  }  /\* Here is the output of one example run of the above program on a desktop  \* running Windows 10.  \*/   |  | | --- | | 0 0 0 0 0 0 0 0 0 0 93 0 1926720 10 14 0 1907168 0 4199367 0 0 0 93 0  4237712 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  -1836441553 30777168 0 0 0 0 0 0 0 0 0 0 0 0 4199675 0 0 0 0 0 0 0 0  0 0 0 841382260 32761 0 0 0 0 0 0 0 0 0 0 887267953 32761 0 0 0 0 | |