C++: No bounds checking

• When you use a value as an array subscript, C++ does not check it to make sure it is a valid subscript.
• In other words, you can use subscripts that are beyond the bounds of the array.

```cpp
const int SIZE = 3;
int values[SIZE];

for (int i=0; i < 5; i++) {
    values[i] = 100;
}
```
What the code does

The way the values array is set up in memory. The outlined area represents the array.

Memory outside the array (Each block = 4 bytes)

values[0] values[1] values[2]

Memory outside the array (Each block = 4 bytes)

How the numbers assigned to the array overflow the array’s boundaries. The shaded area is the section of memory illegally written to.

100 100 100 100 100

(Does not exist) (Does not exist)

Watch out

- Be careful not to use invalid subscripts.
- Doing so can, without warning:
  - corrupt other memory locations
  - crash program
  - lock up computer
  - cause elusive bugs
Watch out: off by one

- It’s easy to get the loop index off by one, especially if you
  - start at 1 instead of 0
  - use <= instead of <

// This code has an off-by-one error.
const int SIZE = 100;
int numbers[SIZE];
for (int count = 1; count <= SIZE; count++)
    numbers[count] = 0;

Parallel Arrays

- Parallel arrays: two or more arrays that contain related data
- A subscript is used to relate arrays: elements at same subscript are related, belong to the same entity
- Arrays may be of different types
Parallel Arrays

• Example: Employee hours worked and payrate

```
const int NUM_EMPS = 5;   // Number of Employees
int hours[NUM_EMPS];      // Holds hours worked
double payRate[NUM_EMPS]; // Holds pay rates

cout << "Enter the hours worked and pay rates:\n";
for(int i = 0; i < NUM_EMPS; i++)  {
    cout << "Hours worked by employee " << i+1 << ": ";
    cin >> hours[i];
    cout << "Hourly pay rate for employee " << i+1 << ": ";
    cin >> payRate[i];
}
```
Parallel Arrays

• Example: Cont.

cout << “Here is the gross pay for each employee:\n”;  
cout << fixed << setprecision(2);  
for(int i = 0; i < NUM_EMPS; i++) {  
    double grossPay = hours[i] * payRate[i];  
    cout << “Employee “ << i+1 << “: $”;
    cout << grossPay << endl;
}