Time Complexity Analysis

1. Code Snippet 1:

```
int c = 0;
for(int i = n; i > 0; i /= 2) {
   c++;
}
```

Time Complexity: O(log n)
Explanation: The loop divides i by 2 in each iteration, resulting in logarithmic iterations.

2. Code Snippet 2:

```
int c = 0;
for(int i = n; i > 1; i /= i) {
   c++;
}
```

Time Complexity: O(1)
Explanation: Since i is divided by itself, the loop runs only once.

3. Code Snippet 3:

```
int c = 0;
for(int i = 0; i < n; i += k) {
   c++;
}
```

Time Complexity: O(n)
Explanation: The loop increments by a constant k, resulting in n/k iterations, simplified to O(n).

4. Code Snippet 4:

```
int c = 0;
for(int i = 1; i < n; i *= 2) {
   c++;
}
```

Time Complexity: O(log n)
Explanation: The loop multiplies i by 2, making the iterations logarithmic to n.

5. Code Snippet 5:

```
int c = 0;
for(int i = 0; i < n; i++) {
   c += i;
}
```

Time Complexity: O(n)
Explanation: Although there is a summation inside the loop, the overall complexity remains linear.

6. Code Snippet 6:

```
int c = 0;
for(int i = 0; i < n; i++) {
   for(int j = 0; j < i; j++){
      c++;
   }
}
```

Time Complexity: $O(n^2)$
Explanation: The inner loop runs i times, creating a triangular number pattern, resulting in quadratic time complexity.