
Computer Programming: Skills & Concepts (CP1)

Iteration, arrays and the correctness issue

Philipp Koehn

1st November 2004



Correctness of a Program

- How can you show that a program is correct?
- Similar to a mathematical proof: show that certain statements are true at all times in the program (“Invariants”)
- Brain teaser: Is it possible to write a program that checks any other program, if it is correct?

Power of a number

```
int Power(int n, int k)
/* Assumes k >= 0. Returns n^k: n raised to the power k. */
{
    int p = 1, i = k;
    /* Precondition: i >= 0 */
    while (i > 0) {
        /* Invariant: i >= 0 AND p * n^i == n^k */
        p *= n;
        --i;
    }
    /* p = n^k */
    return p;
}
```

Example: $n = 3$, $k = 4$. The answer should be $3^4 = 81$.

The computation progresses as follows. Initially, $i = k$ and $p = 1$. Note that $p \times n^i$ is invariant!

	i	p	$p \times n^i$
Initial	4	1	$1 \times 3^4 = 81$
Iteration 1	3	3	$3 \times 3^3 = 81$
Iteration 2	2	9	$9 \times 3^2 = 81$
Iteration 3	1	27	$27 \times 3^1 = 81$
Iteration 4	0	81	$81 \times 3^0 = 81$

Searching an array

```
int LinearSearch(int n, int a[], int searchKey)
/* Returns TRUE iff (if and only if) searchKey is contained
 * in the array, i.e., there exists an index i with 0 <= i < n
 * such that a[i] == searchKey.
 */
{
    int i;

    for (i = 0; i < n; ++i) {
        if (a[i] == searchKey) return TRUE;
    }
    return FALSE;
}
```

Binary search

Sometimes we quickly want to find an entry in an array.

It helps if the array is sorted.

How do you search for a name in a telephone book?

Binary search

```
int BinarySearch(int n, int a[], int searchKey)
/* Assumes the elements of the array a are in ascending order.
 * Returns TRUE iff searchKey is contained in the array, i.e.,
 * there exists an index i with 0 <= i < n and a[i] == searchKey.
 */
{
    int i, j, m;

    i = 0;
    j = n - 1;
    /* Precondition: a[0] <= a[1] <= ... <= a[n-1] */
```

```
while (i < j) {
    /* Invariant:  i <= j  AND
       if searchKey is in a[0:n-1] then searchKey is in a[i:j] */
    m = (i + j)/2;
    if (searchKey <= a[m]) {
        j = m;
    } else {
        i = m + 1;
    }
}
/* EITHER a[i] == searchKey OR searchKey is not in a[0:n-1] */
return a[i] == searchKey;
}
```