Horner’s Method for Polynomial Evaluation

Given a polynomial of the form $c_n x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \ldots + c_1 x + c_0$ and a value of x, find the value of polynomial for a given value of x. Here c_n, c_{n-1}, .. are integers (may be negative) and n is a positive integer.

Input is in the form of an array say *poly[]* where poly[0] represents coefficient for x^n and poly[1] represents coefficient for x^{n-1} and so on.

Examples:

```
// Evaluate value of 2x^3 - 6x^2 + 2x - 1 for x = 3
Input: poly[] = {2, -6, 2, -1}, x = 3
Output: 5

// Evaluate value of 2x^3 + 3x + 1 for x = 2
Input: poly[] = {2, 0, 3, 1}, x = 2
Output: 23
```

A naive way to evaluate a polynomial is to one by one evaluate all terms. First calculate x^n, multiply the value with c_n, repeat the same steps for other terms and return the sum. Time complexity of this approach is $O(n^2)$ if we use a simple loop for evaluation of x^n. Time complexity can be improved to O(nLogn) if we use O(Logn) approach for evaluation of x^n.

Horner’s method can be used to evaluate polynomial in O(n) time. To understand the method, let us consider the example of $2x^3 – 6x^2 + 2x – 1$. The polynomial can be evaluated as $((2x – 6)x + 2)x – 1$. The idea is to initialize result as coefficient of x^n which is 2 in this case, repeatedly multiply result with x and add next coefficient to result. Finally return result.

Following is C++ implementation of Horner’s Method.

```
#include <iostream>
using namespace std;

// returns value of poly[0]x(n-1) + poly[1]x(n-2) + .. +
int horner(int poly[], int n, int x)
{
    int result = poly[0];  // Initialize result

    // Evaluate value of polynomial using Horner's metho
    for (int i=1; i<n; i++)
        result = result*x + poly[i];

    return result;
}

// Driver program to test above function.
int main()
{
    // Let us evaluate value of 2x3 - 6x2 + 2x - 1 for x
    int poly[] = {2, -6, 2, -1};
    int x = 3;
    int n = sizeof(poly)/sizeof(poly[0]);
    cout << "Value of polynomial is " << horner(poly, n,
    return 0;
}
```

Output:

```
Value of polynomial is 5
```

Time Complexity: O(n)