Algorithm in Programming

In programming, algorithm are the set of well defined instruction in sequence to solve a program. An algorithm should always have a clear stopping point.

Qualities of a good algorithm

1. Inputs and outputs should be defined precisely.
2. Each steps in algorithm should be clear and unambiguous.
3. Algorithm should be most effective among many different ways to solve a problem.
4. An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.

Algorithm Examples
Algorithm to add two numbers
Algorithm to find the largest among three numbers
Algorithm to find all the roots of quadratic equation
Algorithm to find the factorial
Algorithm to check prime number
Algorithm of Fibonacci series

Examples Of Algorithms In Programming

Write an algorithm to add two numbers entered by user.

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2
```

```
Step 5: Display sum
Step 6: Stop
```

Write an algorithm to find the largest among three different numbers entered by user.

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a>b
              If a>c
                  Display a is the largest number.
              Else
                  Display c is the largest number.
          Else
              If b>c
                  Display b is the largest number.
              Else
                  Display c is the greatest number.
Step 5: Stop
```

Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.

```
Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate discriminant
           D←b2-4ac
Step 4: If D≥0
                  r1←(-b+√D)/2a
                  r2←(-b-√D)/2a
                  Display r1 and r2 as roots.
          Else
                  Calculate real part and imaginary part
                  rp←b/2a
                  ip←√(-D)/2a
                  Display rp+j(ip) and rp-j(ip) as roots
Step 5: Stop
```

```
Step 1: Start
Step 2: Declare variables n,factorial and i.
Step 3: Initialize variables
            factorial←1
            i←1
Step 4: Read value of n
Step 5: Repeat the steps until i=n
      5.1: factorial←factorial*i
      5.2: i←i+1
Step 6: Display factorial
Step 7: Stop
```

Write an algorithm to check whether a number entered by user is prime or not.

```
Step 1: Start
Step 2: Declare variables n,i,flag.
Step 3: Initialize variables
          flag←1
          i←2
Step 4: Read n from user.
Step 5: Repeat the steps until i<(n/2)
      5.1 If remainder of n÷i equals 0
               flag←0
               Go to step 6
      5.2 i←i+1
Step 6: If flag=0
              Display n is not prime
          else
              Display n is prime
Step 7: Stop
```

Write an algorithm to find the Fibonacci series till term≤1000.

```
Step 1: Start
Step 2: Declare variables first_term,second_term and temp.
Step 3: Initialize variables first_term←0 second_term←1
Step 4: Display first_term and second_term
Step 5: Repeat the steps until second_term≤1000
      5.1: temp←second_term
      5.2: second term←second term+first term
```

```
Step 6: Stop
```

Algorithm is not the computer code. Algorithm are just the instructions which gives clear idea to you idea to write the computer code.

Related Articles

- Why every programmer should learn to optimize algorithms
- self in Python, Demystified
- Increment ++ and Decrement -- Operator as Prefix and Postfix
- Interpreter Vs Compiler : Difference Between Interpreter and Compiler
- Algorithm in Programming
- Flowchart In Programming