From Inductive Definition to Loop

Definition

Consider the factorial function $n!$. It may be viewed as a sequence $0!, 1!, 2!, 3!, 4!, \cdots = 1, 1, 2, 6, 24, \cdots$. The n^{th} term of the sequence t_n can be defined as follows:

$$t_n = \begin{cases} 1 & \text{if } n = 0 \text{ (basis)}, \\ nt_{n-1} & \text{if } n > 0. \end{cases}$$

Converting the Definition to Code

1. Initialize a variable `fact` to the value of the basis. Also initialize a variable `i` as the sequence index of the next element to compute.
2. Repeat the following steps until `i` exceeds the input n.
   (a) multiply `fact` by `i`,
   (b) increment `i` to the next index.

A Code

```
fact = i = 1 ;
while(i<=n) fact *= i++ ;
```

Definition

The n^{th} term t_n of the Fibonacci sequence $0, 1, 1, 2, 3, 5, 8, 13, \cdots$ is defined inductively as follows:

$$t_n = \begin{cases} n & \text{if } n = 0, 1 \text{ (basis)}, \\ t_{n-1} + t_{n-2} & \text{if } n > 1. \end{cases}$$

Converting the Definition to Code

1. Initialize two variables `f0` and `f1` to zero and one respectively (two values of the basis). Initialize the sequence index `i` to two.
2. Repeat the following until the value of `i` exceeds the input n.
   (a) update `f0` and `f1` for current two values,
   (b) increment `i`.

A Code

```
f0 = 0, f1 = 1, i = 2;
while(i<=n) {
        f1 += f0 ;
        f0 = f1 - f0 ;
        ++i ;
}
```

The variable `f1` has the value of t_n.

Definition

Consider the series $1^2 + 2^2 + 3^2 + 4^2 + \cdots$. The sum upto the n^{th} term, S_n, gives the sequence $1, 5, 14, 30, \cdots$. Let us call the n^{th} term to be t_n.

Definition

Both t_n and S_n are defined inductively as

$$t_n = \begin{cases} 1 & \text{if } n = 1 \text{ (basis)}, \\ t_{n-1} + 2n - 1 & \text{if } n > 1. \end{cases},$$

$$S_n = \begin{cases} t_1 & \text{if } n = 1 \text{ (basis)}, \\ S_{n-1} + t_n & \text{if } n > 1. \end{cases}$$

Converting the Definition to Code

1. Initialize two variables `term` and `sum` to one (values of their basis). Initialize the sequence index `i` to two.
2. Repeat the following until the value of `i` exceeds the input n.
   (a) update `term` and `sum`
   (b) increment `i`.

A Code

```
sum = term = 1, i = 2;
while(i<=n) {
        term += 2*i++ - 1 ;
        sum += term ;
}
```

The variable `sum` has the value of s_n and the variable `term` has the value of t_n.

Definition

Consider the following infinite series for computing the value of e^x,

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Let us call the n^{th} term to be t_n, and the sum upto the n^{th} term to be S_n.

Definition

The inductive definitions of t_n and S_n are as follows:

$$t_n = \begin{cases} 1 & \text{if } n = 0 \text{ (basis)}, \\ \frac{xt_{n-1}}{n} & \text{if } n > 0. \end{cases},$$

$$S_n = \begin{cases} t_0 & \text{if } n = 0 \text{ (basis)}, \\ S_{n-1} + t_n & \text{if } n > 0. \end{cases}$$

Converting the Definition to Code

1. Initialize `term` and `sum` to one, term index `i` is also one. Use the variable `oldS` to store the previous value of `sum`.
2. Repeat the following steps while `oldS` is less than `sum`. Number of term can be controlled in other ways.

   Update `oldS`, `term` and `sum`

A Code

```
sum = term = 1;
i = 1.0 ;
do {
        term *= x/i ;
        oldS = sum ;
        sum += term ; i += 1.0 ;
} while (oldS < sum) ;
```

Note

Note that the previous code does not work for -ve value of x due to our terminating condition. If x is -ve, we calculate e^{-x} and then $\frac{1}{e^{-x}}$.

Definition

Consider the following infinite series for computing the value of $\ln x$,

$$\frac{\ln x}{2} = \frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \cdots$$

Let us call the n^{th} term to be t_n and the sum upto the n^{th} term to be S_n.

Definition

Let $y = \frac{x-1}{x+1}$ and $f_n = y^{2n-1}$, so $t_n = \frac{f_n}{2n-1}$. The inductive definitions of f_n and S_n are as follows.

$$f_n = \begin{cases} y & \text{if } n = 1 \text{ (basis)}, \\ y^2 f_{n-1} & \text{if } n > 1. \end{cases},$$

$$S_n = \begin{cases} t_1 & \text{if } n = 1 \text{ (basis)}, \\ S_{n-1} + f_n/(2n-1) & \text{if } n > 1. \end{cases}$$

Converting the Definition to Code

1. Initialize `y` to $\frac{x-1}{x+1}$, `i` to `3`, and both `fn` and `sum` to `y`. Another variable `oldS` is used to store the previous value of `sum`.
2. Repeat the following steps while the absolute value of `oldS` is less than the absolute value of `sum`.

   Update `oldS`, `fn`, `sum` and `i`.

A Code

```
#define ABS(X) (((X) < 0.0)? -(X) : (X))
    y = (x-1)/(x+1) ;
    sum = fn = y;
    i = 3.0 ;
    do {
          fn *= y*y;
          oldS = sum ;
          sum += fn/i ; i += 2.0 ;
    } while (ABS(oldS) < ABS(sum)) ;
```