Approximation of e

Marcus Birkenkrahe (pledged)

May 4, 2024

Problem

The value of the mathematical constant e (Euler's number) can be expressed as an infinite series:

$e = 1 + 1/1! + 1/2! + 1/3! + \dots$

Write a program that approximates e by computing the value of

$e = 1 + 1/1! + 1/2! + 1/3! + \cdots + 1/n!$

where n is an integer entered by the user.

Bonus problem (10 points)

Modify the program to compute Euler's number so that the program continues adding terms until the current term becomes less than ϵ, where ϵ is a small (floating-point) number entered by the user.

Submission

Submit your solution(s) as an Emacs Org-mode file including the usual header matter (title, author, honor pledge) in Canvas.

To get full points, add documentation before the code blocks. Pseudocode would be good.

Solution

I/O, Constants, Variables

- Input: `n` (upper bound for the loop)
- Output: `e`

- Constants: `M_E` from `math.h`
- Floating-point variable: `e`
- Integer variables: `n`, `fac` (factorial), `i` (loop count)

Pseudo code

```
for i from 1 to n times
  Compute factorial of i // factorial(N) = 1 * 2 * ... * N
  Store factorial in fac
  Add inverse of factorial to sum
Sum is approximation to Euler's number - 1 // because 0! = 1
Compare with high precision Euler number M_E in math.h
```

C code

1. Let's begin by writing a program to compute the factorial.

   ```
   int fac = 1, i, n=5; // factorial, counting, upper bound

   for ( i = 1; i <= n; i++) {
     fac *= i;  // fac = fac * i
     printf("!%d = %d\n", i , fac);
    }
   ```

2. For the series, we need to invert the factorials and add them up.

   ```
   float e = 0.f; // Euler number
   int fac = 1, i, n=8; // factorial, counting, upper bound

   for ( i = 1; i <= n; i++) {
     fac *= i;
     printf("!%d = %d\t", fac);
     e += 1.f/fac;
     printf("e = %g\n", e + 1.f);
    }
   ```

3. How close to Euler's number are we already? **??** computes the difference to `M_E` from `math.h`.

```
#include <math.h>

float e = 0.f; // Euler number
int fac = 1, i, n=8; // factorial, counting, upper bound

for ( i = 1; i <= n; i++) {
  fac *= i;
  printf("!%d = %8d\t", i, fac);
  e += 1.f/fac;
  printf("e = %-10.10f\n", e + 1.f);
 }
printf("Diff to %.10f is %g\n", M_E, M_E-e-1.f);
```

Bonus problem

Let's modify . Note that the condition `1/fac > epsilon` implies that `fac < 1/epsilon`. When the condition is `FALSE`, we can leave the `for` loop with a `break` exit command.

```
for i from 1 to n times
 Compute factorial of i
 Store factorial in fac
 if fac < 1/epsilon
   Add inverse of factorial to sum
 else exit loop with break
Sum is approximation to Euler's number - 1 // because 0! = 1
Compare with high precision Euler number M_E in math.h
```

To compute `1/fac` or compare `fac` and `epsilon`, we need to declare `fac` as a floating-point variable.

Code:

```
float fac = 1.f, e = 0.f; // Euler number
float epsilon = 1.0e03; // threshold
int i, n=12; // factorial, counting, upper bound

for ( i = 1; i <= n; i++) {
  fac *= i;
  printf("!%d = %8g\t", i, fac);
  if ( fac < epsilon ) {
    e += 1.f/fac;
```

```
 } else {
   printf("\n** 1/!%d > 1/%g ** exit ** \n", i, epsilon);
   break;
 }
 printf("e = %-8g\n", e + 1.f);
}
```