Average of a stream of numbers

Difficulty Level: Rookie
Given a stream of numbers, print average (or mean) of the stream at every point. For example, let us consider the stream as 10, 20, 30, 40, 50, 60, …

```
Average of 1 numbers is 10.00
Average of 2 numbers is 15.00
Average of 3 numbers is 20.00
Average of 4 numbers is 25.00
Average of 5 numbers is 30.00
Average of 6 numbers is 35.00
.................
```

To print mean of a stream, we need to find out how to find average when a new number is being added to the stream. To do this, all we need is count of numbers seen so far in the stream, previous average and new number. Let *n* be the count, *prev_avg* be the previous average and x be the new number being added. The average after including *x* number can be written as *(prev_avg*n + x)/(n+1)*.

```
#include <stdio.h>

// Returns the new average after including x
float getAvg(float prev_avg, int x, int n)
{
    return (prev_avg*n + x)/(n+1);
}

// Prints average of a stream of numbers
void streamAvg(float arr[], int n)
{
```


```
    float avg = 0;
    for(int i = 0; i < n; i++)
    {
        avg  = getAvg(avg, arr[i], i);
        printf("Average of %d numbers is %f \n", i+1, avg);
    }
    return;
}

// Driver program to test above functions
int main()
{
    float arr[] = {10, 20, 30, 40, 50, 60};
    int n = sizeof(arr)/sizeof(arr[0]);
    streamAvg(arr, n);

    return 0;
}
```

The above function getAvg() can be optimized using following changes. We can avoid the use of prev_avg and number of elements by using static variables (Assuming that only this function is called for average of stream). Following is the oprimnized version.

```
#include <stdio.h>

// Returns the new average after including x
float getAvg (int x)
{
    static int sum, n;

    sum += x;
    return (((float)sum)/++n);
}

// Prints average of a stream of numbers
void streamAvg(float arr[], int n)
{
    float avg = 0;
    for(int i = 0; i < n; i++)
    {
        avg  = getAvg(arr[i]);
        printf("Average of %d numbers is %f \n", i+1, avg);
    }
    return;
}
```


Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
// Driver program to test above functions
int main()
{
    float arr[] = {10, 20, 30, 40, 50, 60};
    int n = sizeof(arr)/sizeof(arr[0]);
    streamAvg(arr, n);

    return 0;
}
```

Thanks to Abhijeet Deshpande for suggesting this optimized version.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix

- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

Writing code in comment? Please use **ideone.com** and share the link here.

16 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Progod • 2 years ago

float getAvg (int x)
{
static int sum, n;

sum += x;
return (((float)sum)/++n);
}
here sum can give you overflow problem.
to solve this problem you can use following approach.

```
 float getAvg (int x)
{
    static float oldAvg, n;

    fload balance = n - oldAvg;
    oldAvg += (balance/++n);
```


Recent Comments

Abhi You live US or India?
Google (Mountain View) interview · 17 minutes ago

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 57 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

```
    return oldAvg;
}
```

Progod ↗ Progod • 2 years ago

Sorry for the typo.

```
 float getAvg (int x)
{
static int sum, n;

sum += x;
return (((float)sum)/++n);
}
here sum can give you overflow problem.
to solve this problem you can use following approach.

float getAvg (int x)
{
    static float oldAvg, n;

    fload balance = x - oldAvg;
    oldAvg += (balance/++n);

    return oldAvg;
}
```

Abhijeet Deshpande ➦ Progod • 2 years ago

I think you meant

```
 float balance = x - oldAvg;
oldAvg += (balance/++n);
```

Progod ➦ Abhijeet Deshpande • 2 years ago

Yes it should be float balance = x - oldAvg; That was a typo

```
 /* Paste your code here (You may delete these lines if
```

Progod ➦ Abhijeet Deshpande • 2 years ago

Sorry for typo It should be float balance = x - oldAvg;

```
/* Paste your code here (You may delete these lines if r
```

kartik ➦ Progod • 2 years ago

It doesn't seem to work. The following program prints

Average of 1 numbers is 0.000000
Average of 2 numbers is 0.500000
Average of 3 numbers is 1.000000
Average of 4 numbers is 1.500000
Average of 5 numbers is 2.000000
Average of 6 numbers is 2.500000

```
#include <stdio.h>

float getAvg (int x)
{
    static float oldAvg, n;

    float balance = n - oldAvg;
    oldAvg += (balance/++n);
```

see more

Abhijeet Deshpande • 2 years ago

This code does not exploit the inherent property of the problem, that is, the ave
So, the possible optimizations are:

1. You need not pass the number of elements every time.
2. You need not pass the previous average. (again, assuming this function is c
stream)
3. Perhaps not relevant here, but n++ is more optimal compared to n+1

Hence, the code:

```
float get_incremental_avg (int x)
{
        static int sum, n;

        sum += x;
        return (((float)sum)/++n);
}
```

kartik ➦ Abhijeet Deshpande • 2 years ago

@Abhijeet Deshpande: Great! Your approach looks good to be added t

Could you explain why n++ is more optimal compared to n+1. Any link

Abhijeet Deshpande ➦ kartik • 2 years ago

@Kartik: Thank you.

Well, in C language, n=n+1 (Assigning to "n", only for comparis
not assigned to n) n++ does not make any difference. The resu
code is compiled and an assembly code is generated, n+1 con

Load R1, n /* Load n in some register */
Load_Imm R2, 1 /* Load an immediate value 1 in reg R2 */
Add R1, R2 /* Add R1 and R2 and store result in R2 - for Intel t

You need 2 registers and 3 instructions (~3 cycles for a RISC p

Whereas n++ translates to..

Load R1, n /* Load n in some register */
Incr R1 /* Increment R1 */

You need 1 regsiter and 2 instructions here.

Typically Increment (and decrement) operations are available ir

see more

Rushi Agrawal ➦ Abhijeet Deshpande • 2 years ago

But abhijeet, the C/C++ compiler does these optimizatic
write n=n+1 or n++, so effectively both the statements a
it.

kartik ➚ Abhijeet Deshpande • 2 years ago

@Abhijeet Deshpande: Thanks!!

Venki • 2 years ago

Average over past few X numbers will be interesting, where X is function parar
than X return average so far.

Abhijeet Deshpande ➚ Venki • 2 years ago

@Venki: Does this answer your question? What you ask for is a FIR fil
or a moving average filter, wherein all the co-efficients of the filter are 1
I have not compiled the code. There might be errors, but this code sho

```
float get_incremental_avg (int x, int n_of_taps)
{

        static int *p, *base_addr;
        static int sum, n;

        if (p != NULL)
        {
                p = (int*)calloc (n_of_taps, sizeof(int));
                base_addr = p;
        }
        *p = x;
```

see more

Doom • 2 years ago

how could we take care of the overflow(if it occurs) for prev_avg*n?

```
/* Paste your code here (You may delete these lines if not writing co
```

kartik ➚ Doom • 2 years ago

I think, following expression can be used to avoid overlfow.

prev_avg*(n/(n+1)) + x/(n+1);

Daniel Cheng ➚ kartik • 2 years ago

Either way will have large error when n is large. If you want som
summation algorithm and avoid the *(n/(n+1)) step.

Subscribe Add Disqus to your site