بار

Fundamentals of Programming

Bu-Ali Sina University, Tuyserkan Faculty of Engineering

Mohammad R. Tayyebi

Undergraduate, BASU

Blogger, Sariab.ir

Introduction

Session 1, Dec 7 2019

Session Goals:

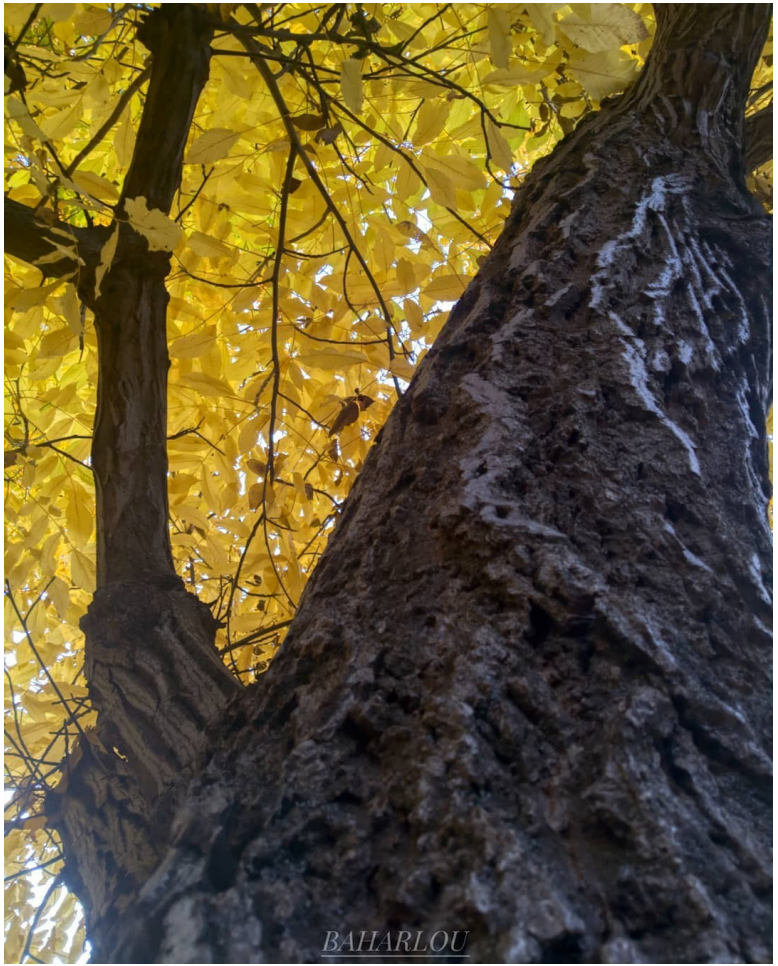
Greetings and Rules,

Algorithmic Thinking,

Enjoying Nature,

Become Familiar with Some Keywords

Greetings



Rules

2nd : Keep the bathroom trips to minimum

3rd : Don’t be too comfortable [that you fall sleep and start snoring]

4th : No kids

Rules

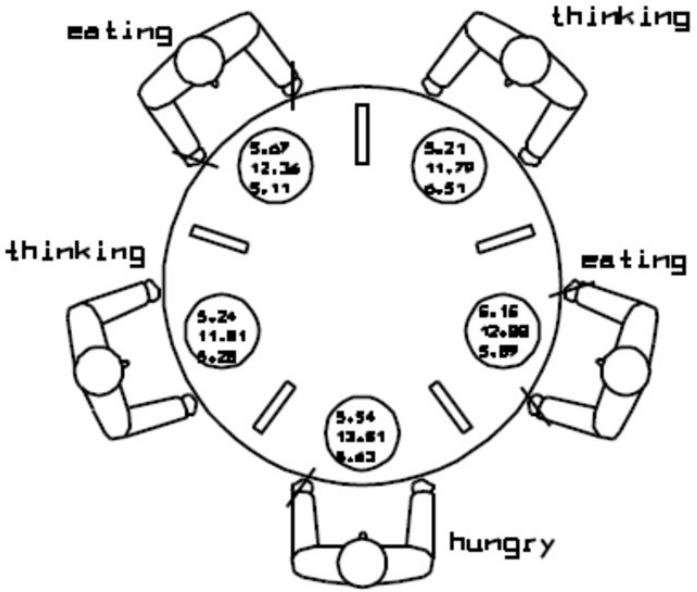
1st and the most important: ***ENJOY THE CLASS*** !

5th : Listen carefully, then write it down

6th : Practice, Try, Try, Try…

7th : “If you are born *philosopher*, it’s not your fault. But if you die *philosopher*, it’s your mistake.” - William Henry Gates III; And notice that old programmers never die, they just cast into **void**!

Dining philosophers problem



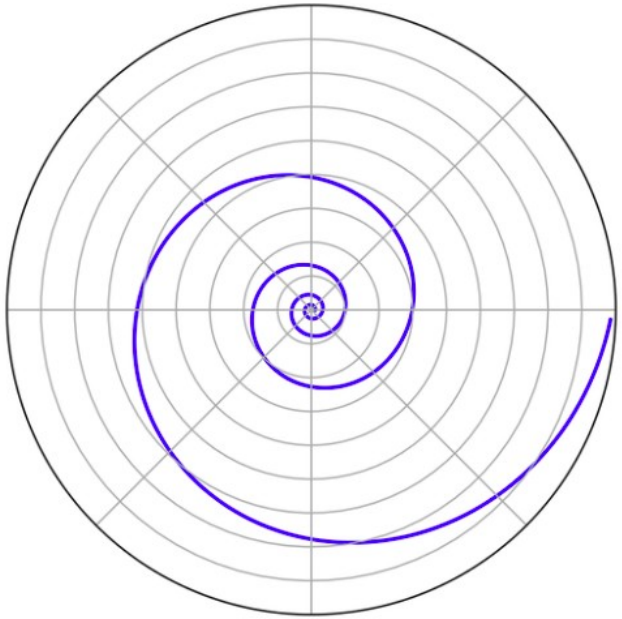
This court is now in session!

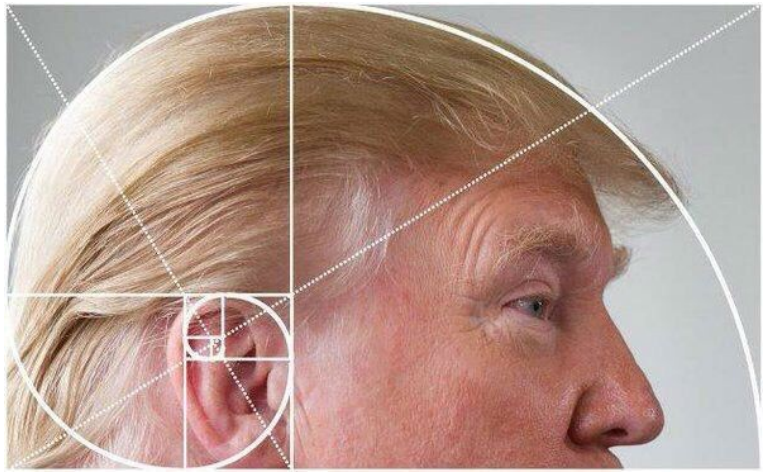
دیباچه

ارْحَمُوا عَزِيزاً ذَلَّ
وَ غَنِيّاً افْتَقَرَ
وَ عَالِماً ضَاعَ فِي زَمَانِ جُهَّالٍ.

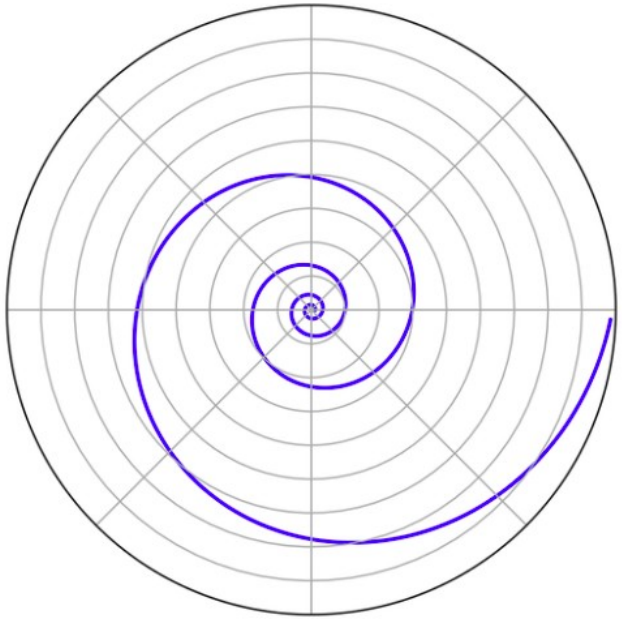
گفت پیغمبر که بر این سه گروه رحم آرید ار ز سنگید ار ز کوه/ آنکه او بعد از عزیزی خوار شد و آنکه بد با مال و بی دینار شد
و آن سوم آن عالمی کاندر جهان مبتلا گردد میان ابلهان/ ز آنکه از عزت به خواری آمدن همچو قطع عضو باشد از بدن

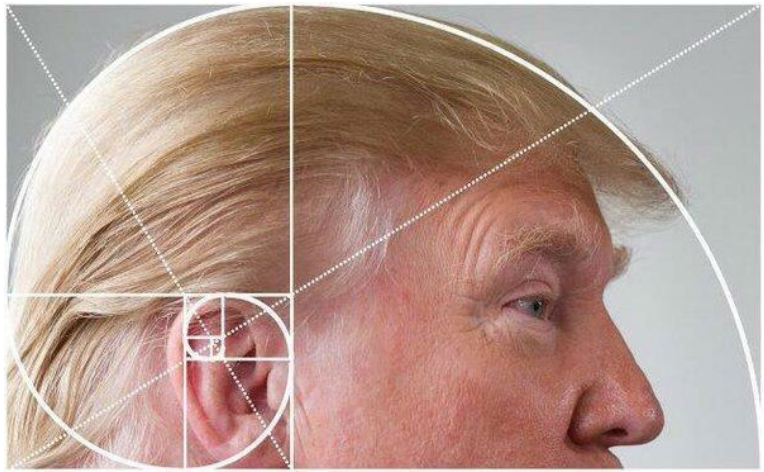
Logarithmic spiral





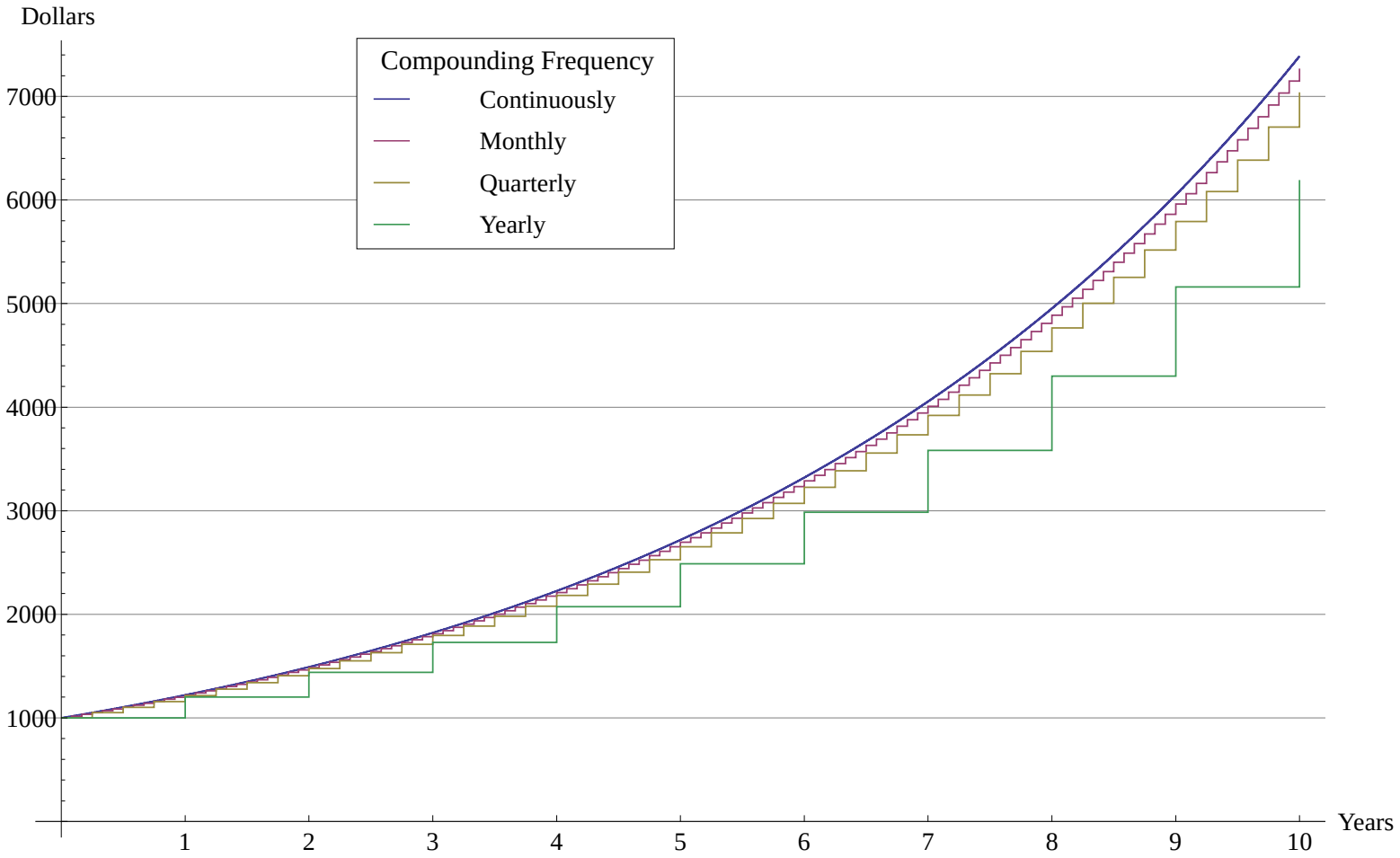
Logarithmic spiral





Compound interest

An account starts with $1.00 and pays 100 percent interest per year. If the interest is credited once, at the end of the year, the value of the account at year-end will be $2.00. If the interest is credited twice in the year, the interest rate for each 6 months will be 50%, so the initial $1 is multiplied by 1.5 twice, yielding $1.00 × (1.5 ^ 2) = $2.25 at the end of the year. Compounding quarterly yields $1.00 × (1.25 ^ 4) = $2.4414..., and compounding monthly yields $1.00 × (1 + 1/12) ^ 12 = $2.613035... If there are n compounding intervals, the interest for each interval will be 100%/n and the value at the end of the year will be $1.00 × (1 + 1/n) ^ n.

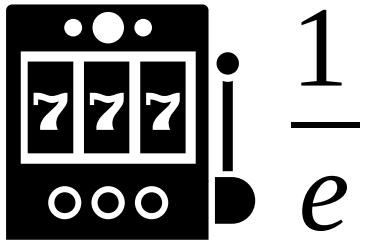

Dollars
Compounding Frequency
Continuously
Monthly
Quarterly
Yearly
7000
6000
5000
4000
3000
2000
1000
1
2
3
4
5
6
7
8
9
10
Years

Euler’s constant

2.7 1828 1828 45 90 45

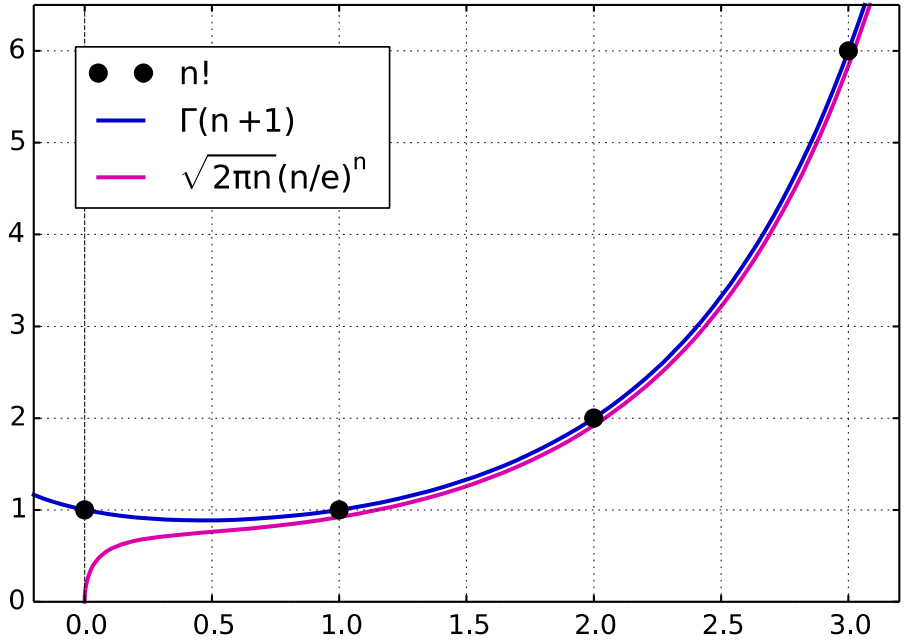
تاریخچه عدد طبیعی

اولین اشاره به این عدد، در جدولی که در ضمیمهٔ مقالهٔ مربوط به لگاریتم جان **نپر** در سال ۱۶۱۸ انتشار یافته بود مشاهده می‌شود. با این حال، این مقاله توضیحی راجع به این عدد نمی‌داد بلکه تنها لیستی از لگاریتم‌های حساب شده در مبنای این عدد را نشان می‌داد. به نظر می‌رسد که این جدول توسط ویلیام **اوترد** (مخترع خط‌کش محاسبه) تهیه شده‌است. اما «کشف» این عدد توسط ژاکوب **برنولی** به انجام رسید.



$$\frac{1}{e}$$

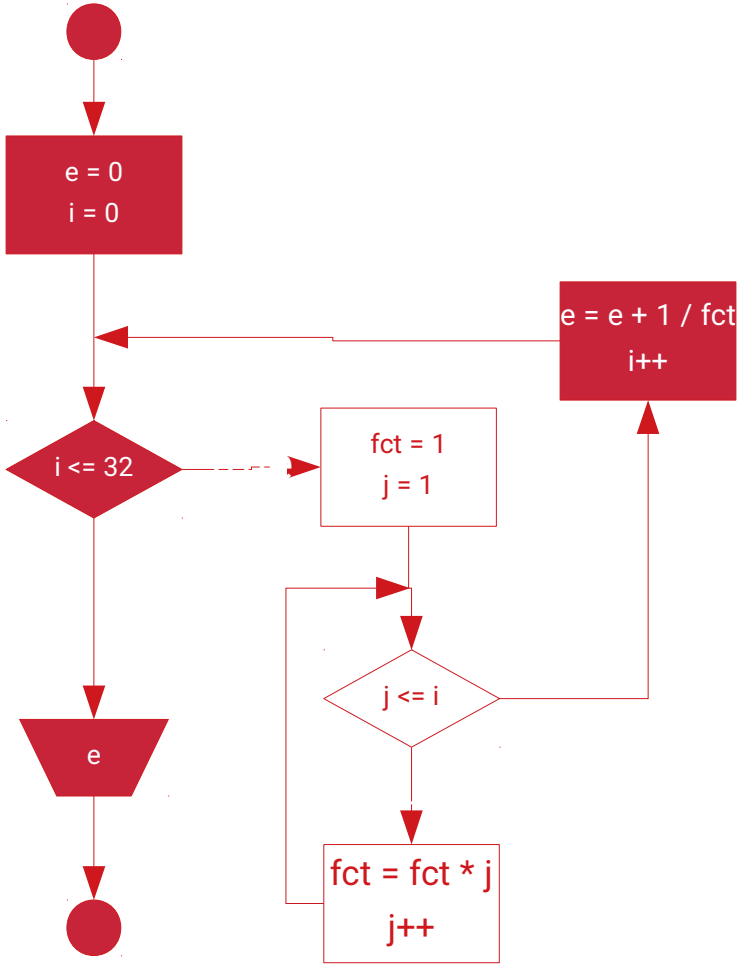
Strirling's Approximation



Euler’s constant

$$\sum_{n=0}^{\infty} \frac{1}{n!} \qquad \lim_{n=0} \left(1+\frac{1}{n}\right)^{n}$$

Flowchart



Code

```
#include <stdio.h>

long factorial(int i) {
    long result = 1;
    for (int j = 1; j <= i; ++j)
        result *= j;
    return result;
}
```

```
int main ()
{
    long double e = 0;
    int i;
    for (i = 0; i <= 32; i++) {
        e = ( 1.0 / factorial(i) ) + e;
    }
    printf("%.64Lf\n", e);
}
```

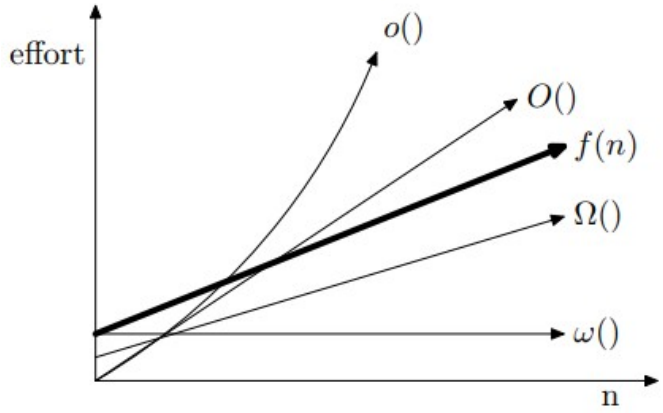
```
#include <stdio.h>

long factorial(int n) {
    long result = 1;
    for (int i = 1; i <= n; ++i)
    result *= i;
    return result;
}

int main ()
{
    long double e = 0;
    int i;

    for (i = 0; i <= 32; i++) {
        e = ( 1.0 / factorial(i) ) + e;
    }

    printf("%.64Lf\n", e);
}
```

Algorithm Complexity (Big Ohhh!)

O(N^2)

$$f(n) = O(g(n)) \Leftrightarrow (\exists c)(\exists n_0)(\forall n > n_0)(f(n) \leq c \cdot g(n))$$

$$f(n) = \omega(g(n)) \Leftrightarrow (\forall c)(\exists n_0)(\forall n > n_0)(f(n) > c \cdot g(n))$$



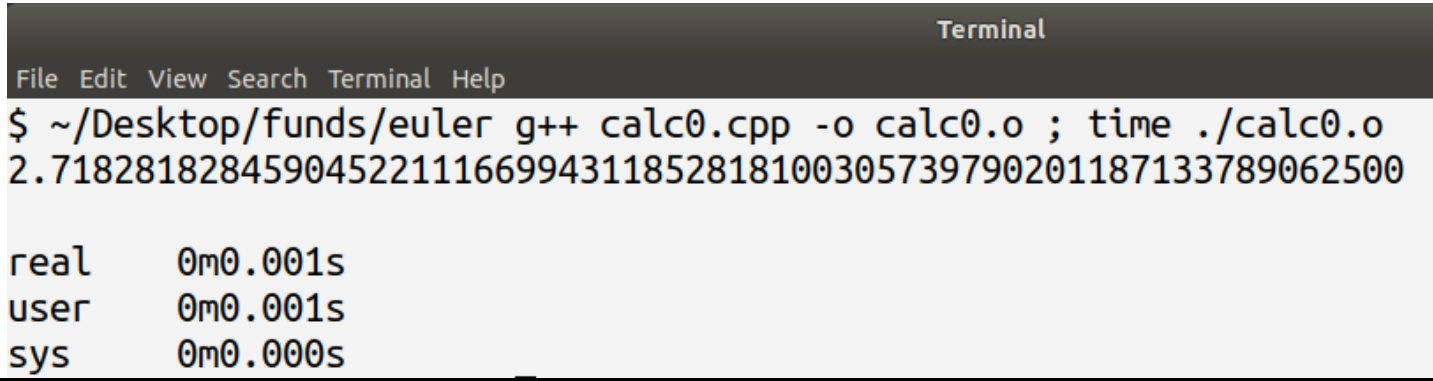
O(n): in worst situations, your algorithm has a complexity of n

Ω(n): in best case, your algorithm has a complexity of n

Θ(n): in every situation, your algorithm has a complexity of n

How to run the code

g++ *calc0.cpp* -o *calc0.o* ; **time** *./calc0.o*



Optimized Code

```
long double e = 0, f = 1;
int i;
for (i = 8; i >= 1; i--) {
    f = f * i;
    e = e + f;
}
e = e / f;
```

```
#include <stdio.h>
#include <iomanip>

int main ()
{
    long double e = 0, f = 1;
    int i;
    for (i = 28; i > 0; i--) {
        f *= i;
        e += f;
    }
    e /= f;
    printf("%.64Lf\n", e);
}
```

Optimized Code Traced

i	f	e	f = f * i	e = e + f	*e / f*
8	1	0	1 * 8 = 8	0 + 8 = 8	1.00000
7	8	8	8 * 7 = 56	8 + 56 = 64	1.14286
6	56	64	56 * 6 = 336	64 + 336 = 400	1.19048
5	336	400	1680	2080	1.23810
4	1680	2080	6720	8800	1.30952
3	6720	8800			1.43651
2	20160	28960			1.71825
1	40320	69280			2.71825
0			END OF LOOP	END OF LOOP	e = e / f = 2.7182539682

Mathematical Method

```
12  def main(n):
13      e = 0.0
14      f = 1.0
15      r = -1
16      while r <= n:
17          e += P (n, r)
18          r += 1
19      e /= P (n, r - 2)
20      return e
```

https://gist.github.com/tayyebi/5bd1bfedbc71acb5801c049c5054de63

Programming Languages

A programming language is a formal language, which comprises a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. Most programming languages consist of instructions for computers. - Wikipedia

C++

```
#include <stdio.h>
#include <iomanip>

int main ()
{
   long double e = 0, f = 1;
   int i = 28;
   while (i>0) {
      f *= i;
            e += f;
      i--;
   }
   e /= f;
   printf("%.64Lf\n", e);
}
```

calc1.cpp

```
1  #include <stdio.h>
2  #include <iomanip>
3
4  int main ()
5  {
6      long double e = 0, f = 1;
7      int i = 28;
8      while (i>0) {
9          f *= i;
10   e += f;
11         i--;
12     }
13     e /= f;
14     printf("%.64Lf\n", e);
15 }
```

Python

```
def main():
    e = 0.0
    f = 1.0
    i = 28
    while i > 0:
        f *= i
        e += f
        i = i - 1
    e /= f
    return e

if __name__== "__main__":
  print (main())
```

calc1.py

```
1   def main():
2       e = 0.0
3       f = 1.0
4       i = 28
5       while i > 0:
6           f *= i
7           e += f
8           i = i - 1
9       e /= f
10      return e
11
12  if __name__== "__main__":
13    print (main())
```

Java

```
public class my_main{
    public static void main(String[] args){
            double e = 0, f = 1;
            int i = 28;
            while (i>0) {
                    f *= i;
                    e += f;
                    i--;
            }
            e /= f;
            System.out.println(e);
    }
}
```

calc1.java

```
1   public class my_main{
2       public static void main(String[] args){
3     double e = 0, f = 1;
4     int i = 28;
5     while (i>0) {
6       f *= i;
7       e += f;
8       i--;
9     }
10    e /= f;
11        System.out.println(e);
12      }
13  }
```

JavaScript

```
var e = 0.0, f = 1.0, i = 28;

while (i>0) {
        f *= i;
        e += f;
        i--;
}

e /= f;

console.log(e);
```

calc1.js

```
1   var e = 0.0, f = 1.0, i = 28;
2
3   while (i>0) {
4     f *= i;
5     e += f;
6     i--;
7   }
8
9   e /= f;
10
11  console.log(e);
```

PHP

```
<?php
$e = 0;
$f = 1;
$i = 28;
while ($i > 0) {
        $f *= $i;
        $e += $f;
        $i--;
}
$e /= $f;
print($e);
?>
```

calc1.php

```
1  <?php
2  $e = 0;
3  $f = 1;
4  $i = 28;
5  while ($i > 0) {
6    $f *= $i;
7    $e += $f;
8    $i--;
9  }
10 $e /= $f;
11 print($e);
12 ?>
```

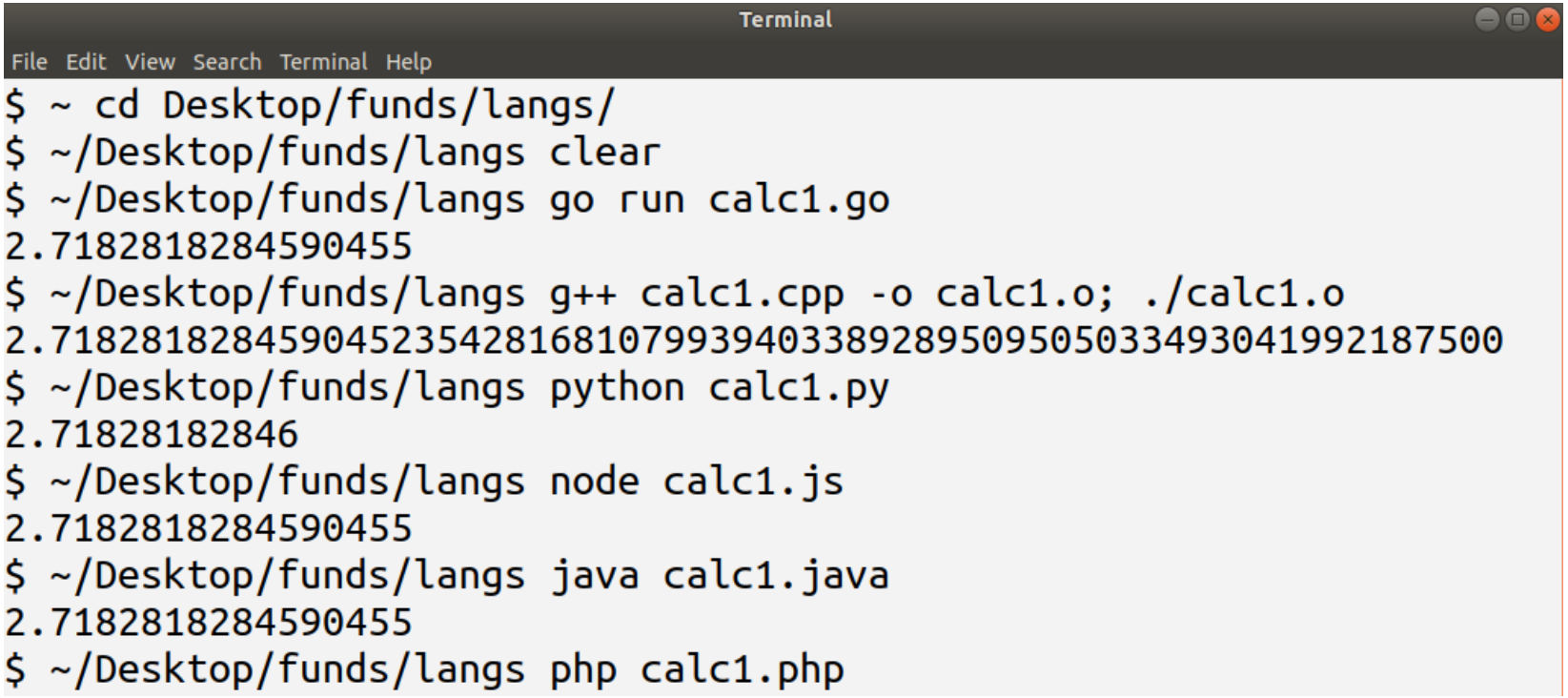
Go

```
package main

import "fmt"

func main() {

    var e = 0.0
    f := 1.0
    var i float64
    i = 28

    for i > 0 {
            f *= i
            e += f
            i--
    }
    e /= f
    fmt.Println(e)

}
```

calc1.go

```
1  package main
2
3  import "fmt"
4
5  func main() {
6
7      var e = 0.0
8      f := 1.0
9      var i float64
10     i = 28
11
12     for i > 0 {
13       f *= i
14       e += f
15       i--
16     }
17     e /= f
18     fmt.Println(e)
19
20 }
```

Linux is your friend



Whats Next

Tomorrow

CPP: Main syntax, Functions, Variables, and Arrays

Sepas <3

Contacts:

http://kouy.ir/tayyebi

Proudly a Sariab Blogger:

www.sariab.ir, info@sariab.ir

Reference and Images:

https://en.wikipedia.org/wiki/E_(mathematical_constant)

https://www.onstageblog.com/onscreen/2018/6/13/9-rules-for-the-movie-theater

https://www.sne-journal.org/benchmarks/c10

https://www.commpro.biz/court-is-in-session-the-law-of-marketing-small-brands-cannot-overlook/36419114-hand-about-to-bang-gavel-on-sounding-block-in-the-court-room/

https://codereview.stackexchange.com/questions/33015/

https://stackoverflow.com/questions/27873104/