باٮ

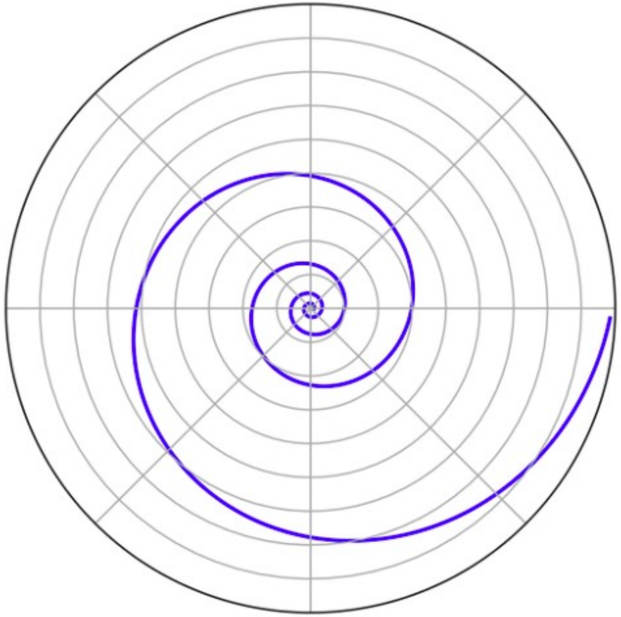
Fundamentals of Programming

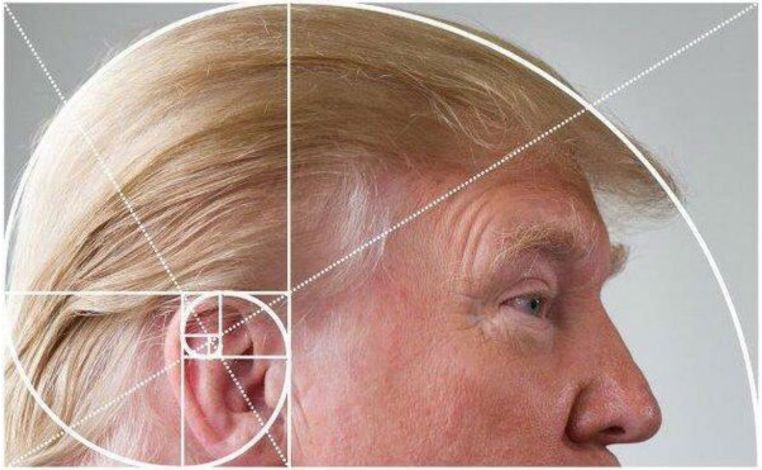
Bu-Ali Sina University, Tuyserkan Faculty of Engineering

Mohammad R. Tayyebi

Session 1: Dec 7 2019

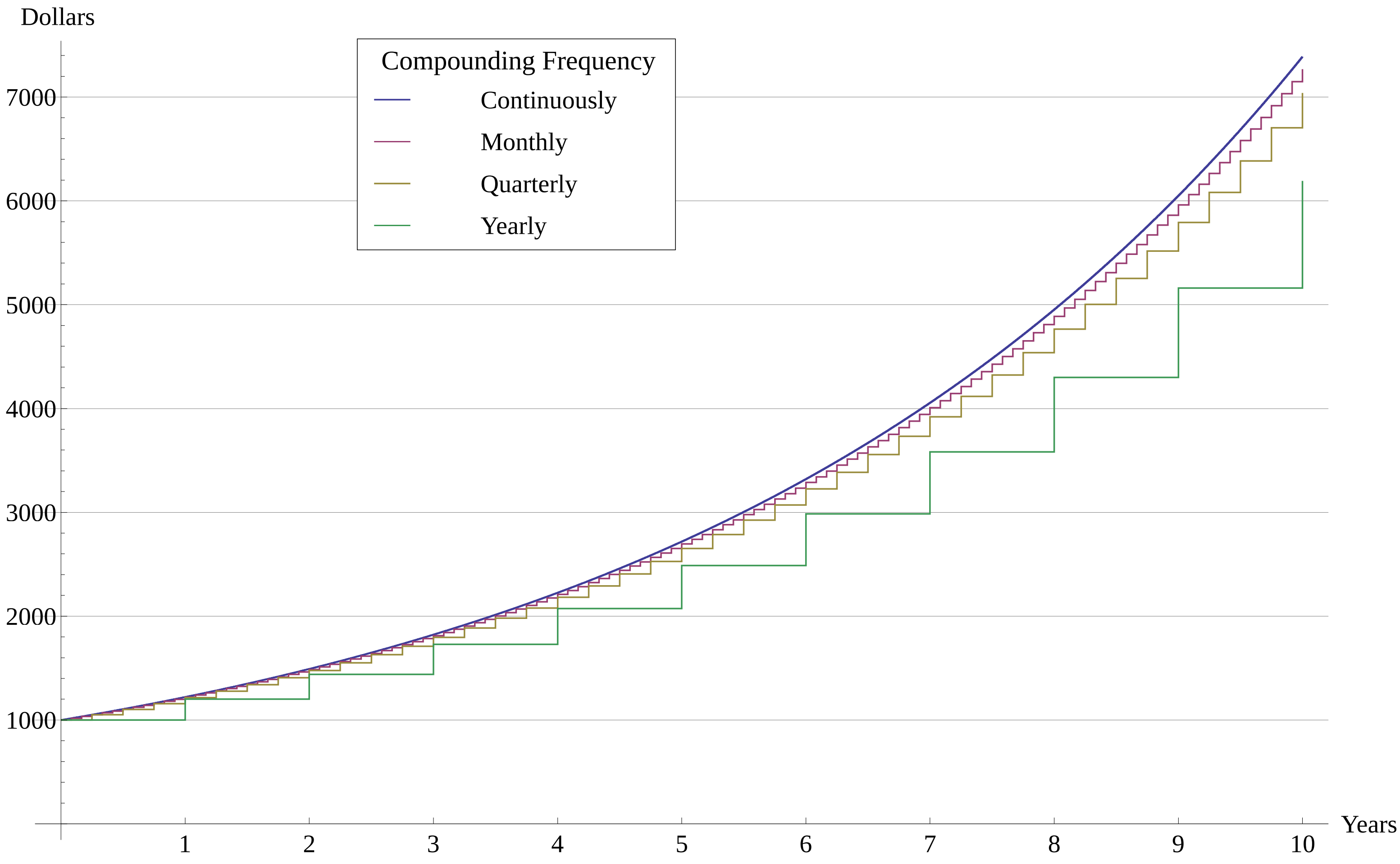
Logarithmic spiral





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 $\$1.00 \times (1.5 \hat{} 2) = \$2.25$ at the end of the year. Compounding quarterly yields $\$1.00 \times (1.25 \hat{} 4) = \$2.4414\ldots$, and compounding monthly yields $\$1.00 \times (1 + 1/12) \hat{} 12 = \$2.613035\ldots$ 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 $\$1.00 \times (1 + 1/n) \hat{} n$.


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

Euler’s constant

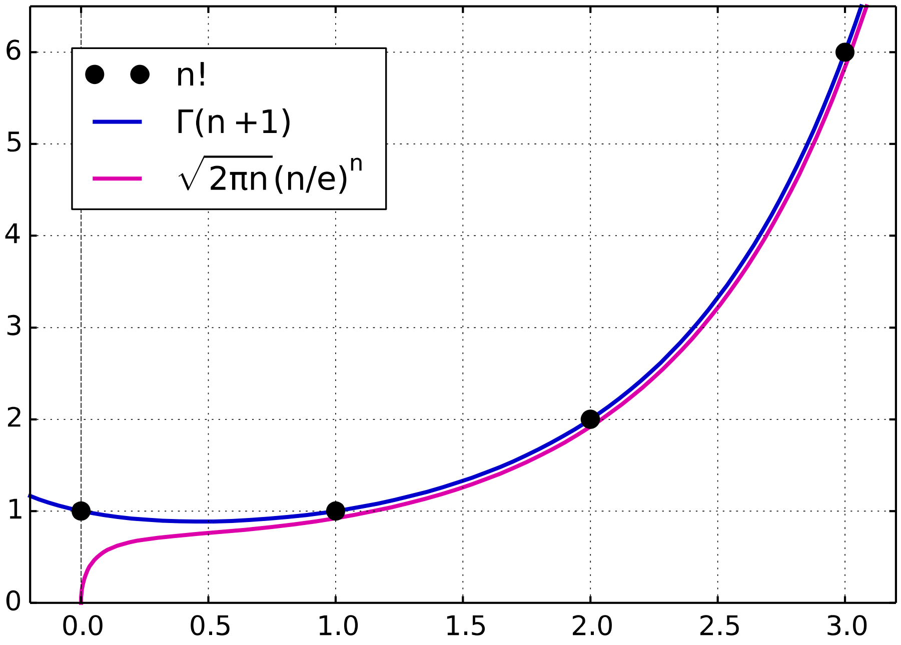
2.7 1828 1828 45 90 45

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

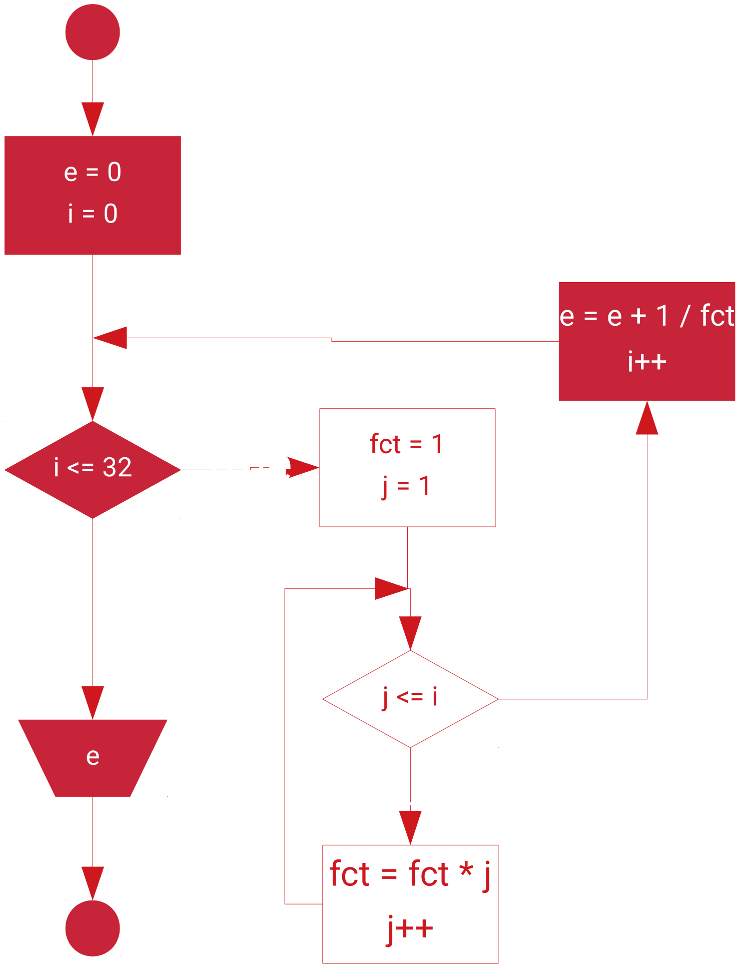


Euler’s constant

$$\sum_{n=0}^{\infty} \frac{1}{n!}$$

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

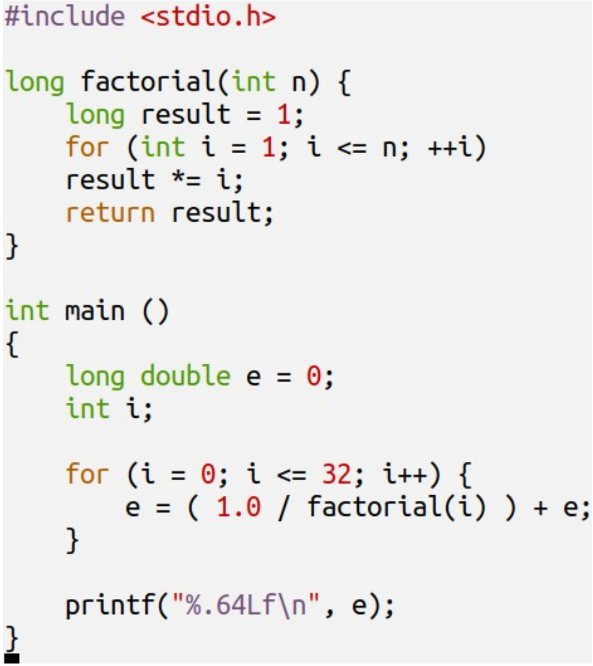
Flowchart



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);
}
```



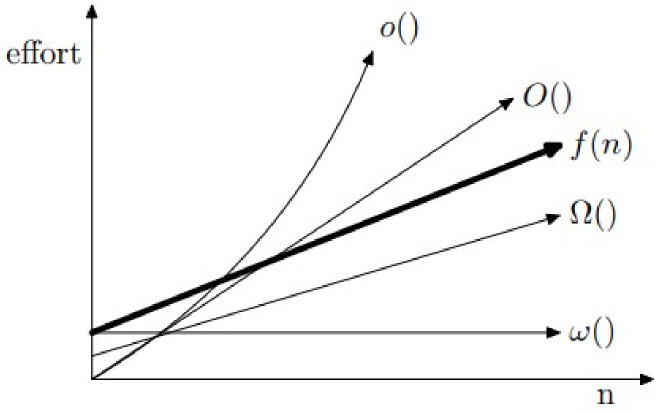
```
#include <stdio.h>

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)

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

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



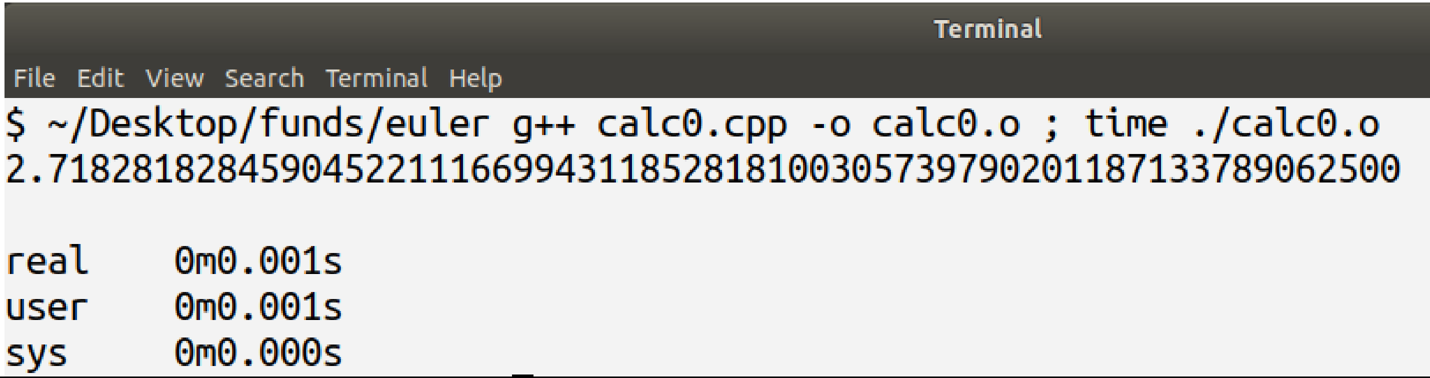
O(n): in worst situations, your algorithm has a complexity of n

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

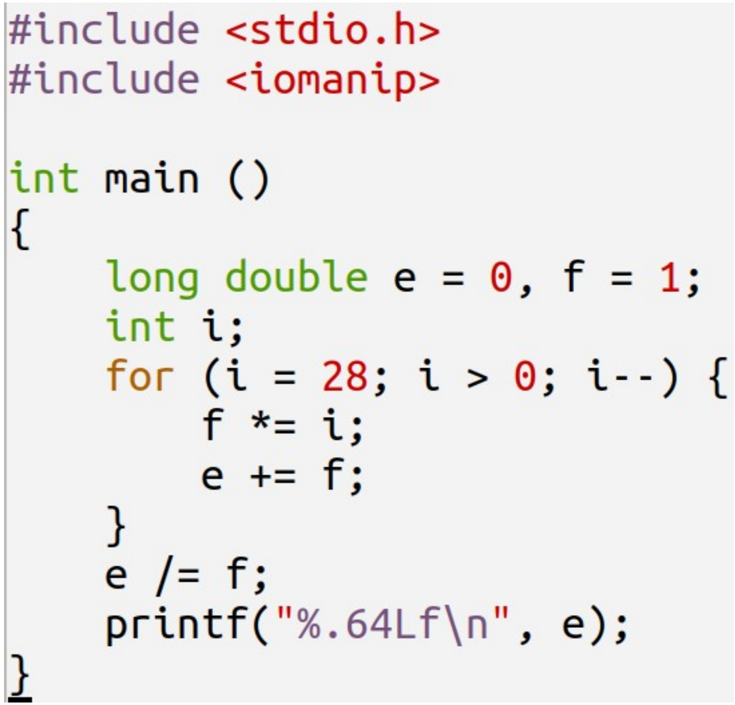
g++ *calc0.cpp* -o *calc0.o* ; **time** *./calc0.o*



```
$ ~/Desktop/funds/euler g++ calc0.cpp -o calc0.o ; time ./calc0.o
2.718281828459045221166994311852818100305739790201187133789062500

real	0m0.001s
user	0m0.001s
sys	0m0.000s
```

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

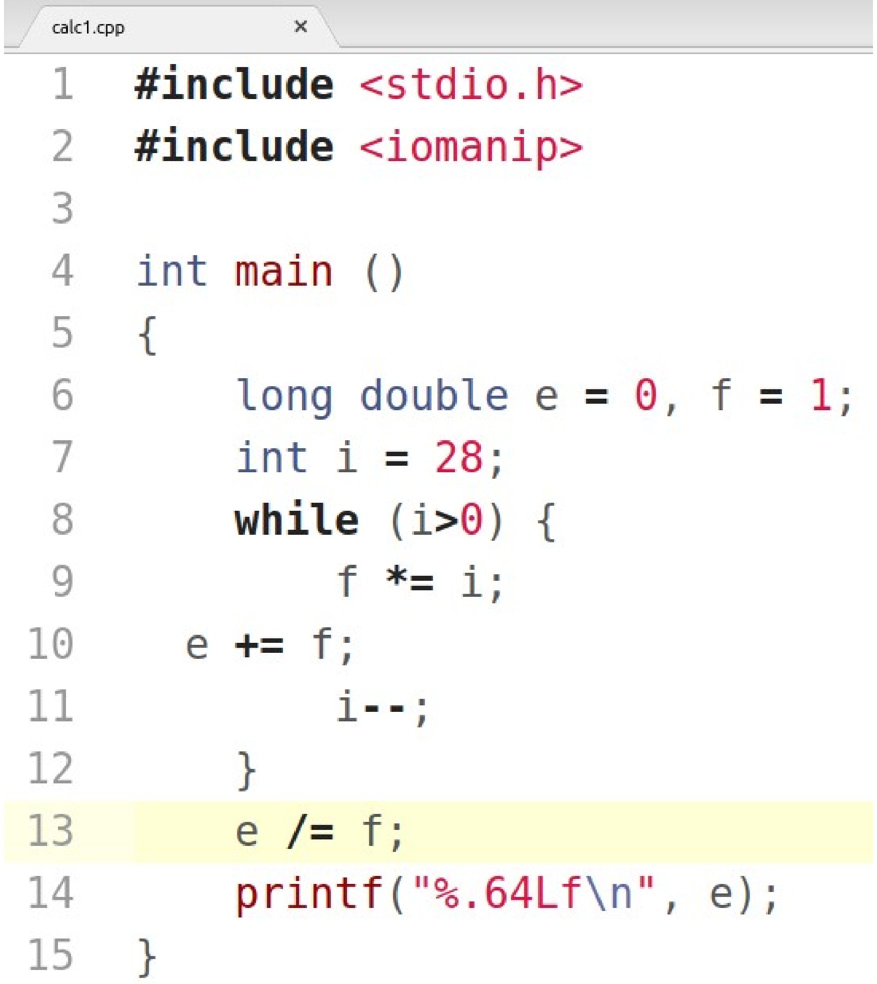
i	f	e	f = f * i	e = e + f	*e / f*
8	**1**	**0**	1 * 8 = **8**	0 + **8** = 8	**1.00000**
7	8	8	8 * 7 = 56	8 + 56 = 64	**1.14286**
6	56	64	56 * 6 = 336	64 + 336 = 400	**1.19048**
5	336	400	1680	2080	**1.23810**
4	1680	2080	6720	8800	**1.30952**
3	6720	8800			**1.43651**
2	20160	28960			**1.71825**
1	40320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

Programming Languages

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.

C++

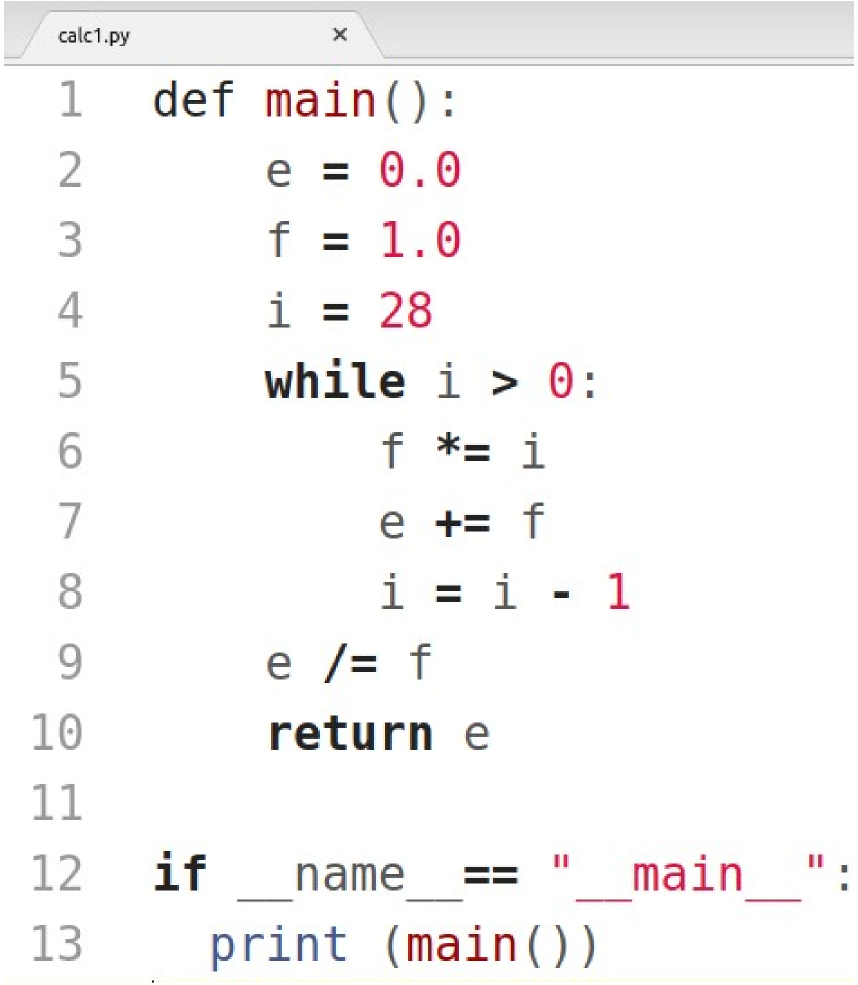
```
#include <stdio.h>
#include <iomanip>

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);
}
```



```
#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);
}
```

Python

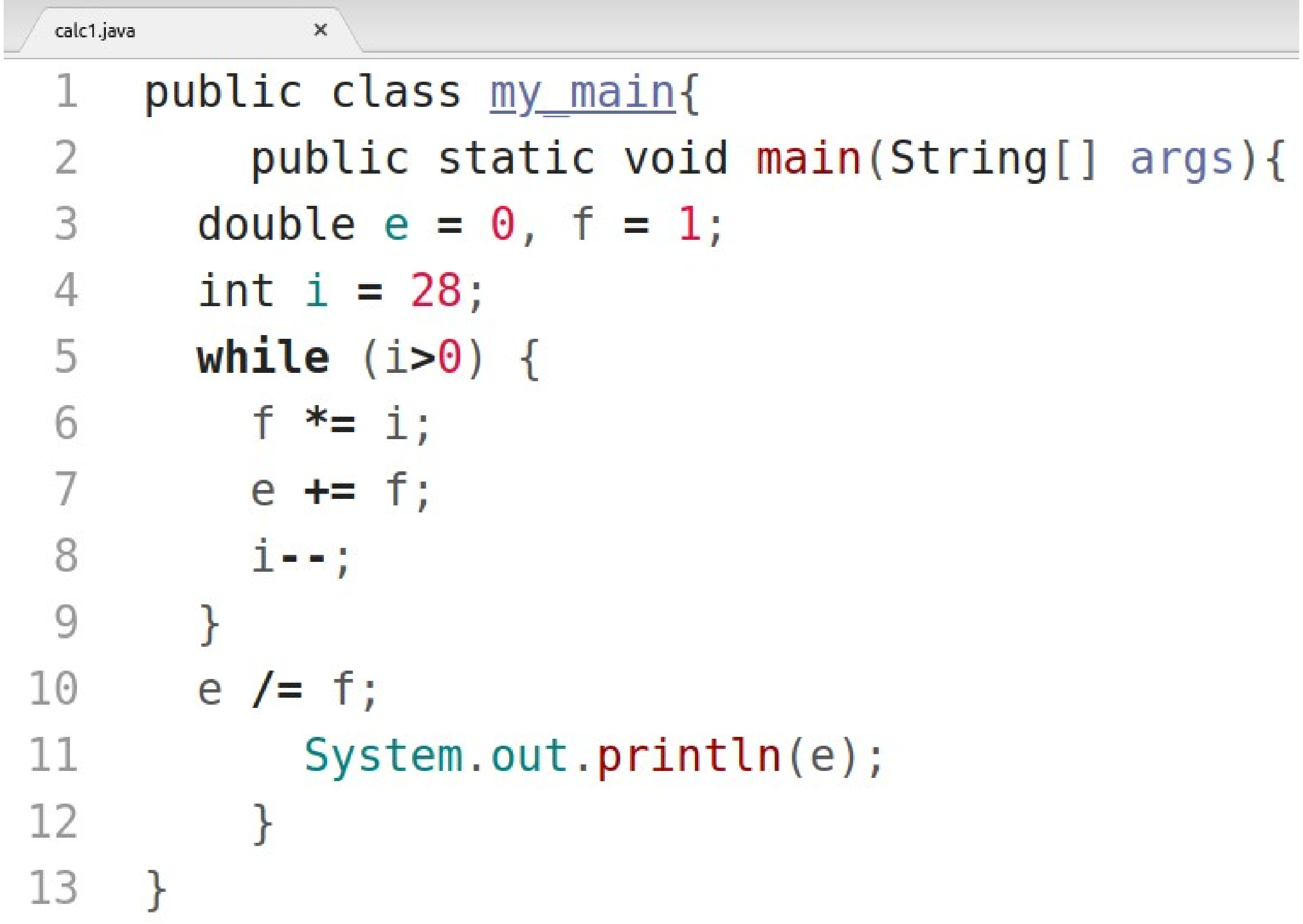
```
def main():
    e = 0.0
    f = 1.0
    i = 28
    while i > 0:
        f *= i
        e += f
        i = i - 1
    e /= f
    return e

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



```
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())
```

Java

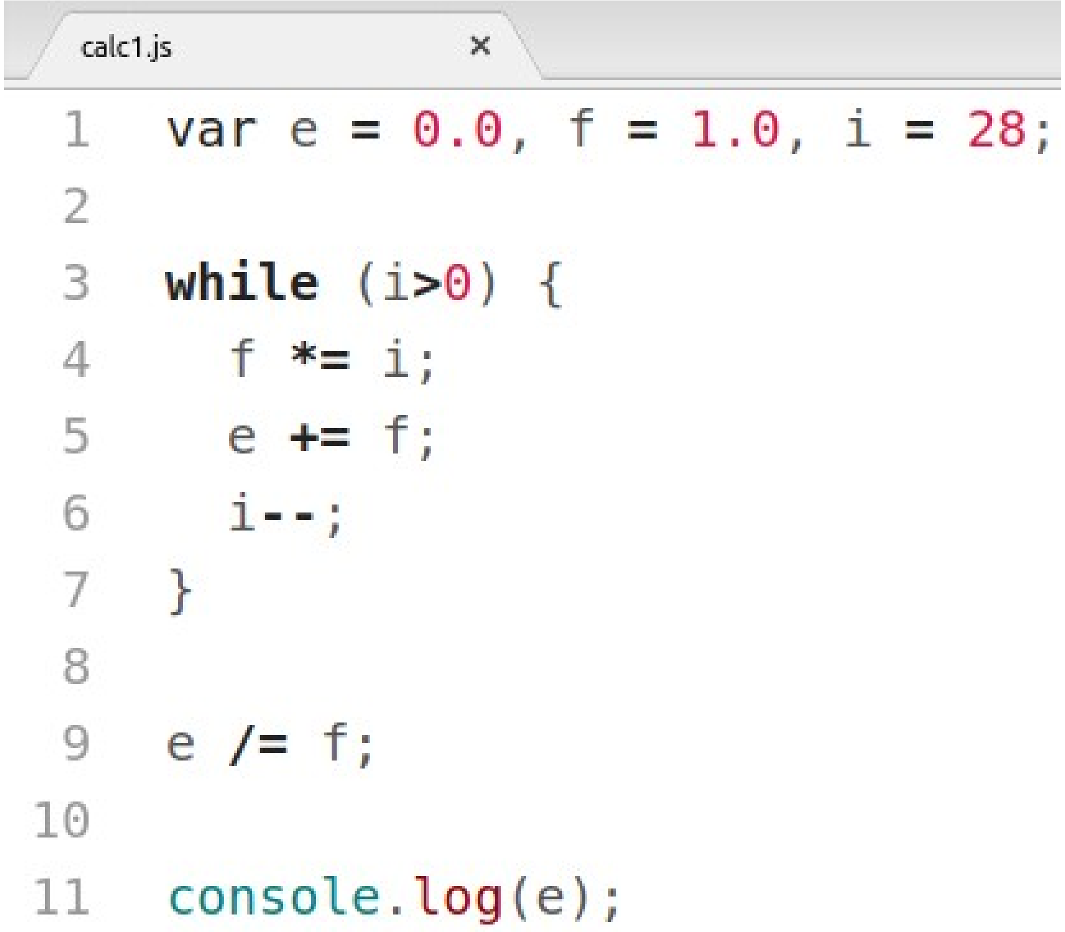
```
public class my_main{
  public static void main(String[] args){
        double e = 0, f = 1;
        int i = 28;
        while (i>0) {
                f *= i;
                e += f;
                i–;
        }
        e /= f;
        System.out.println(e);
  }
}
```



```
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);
      }
}
```

JavaScript

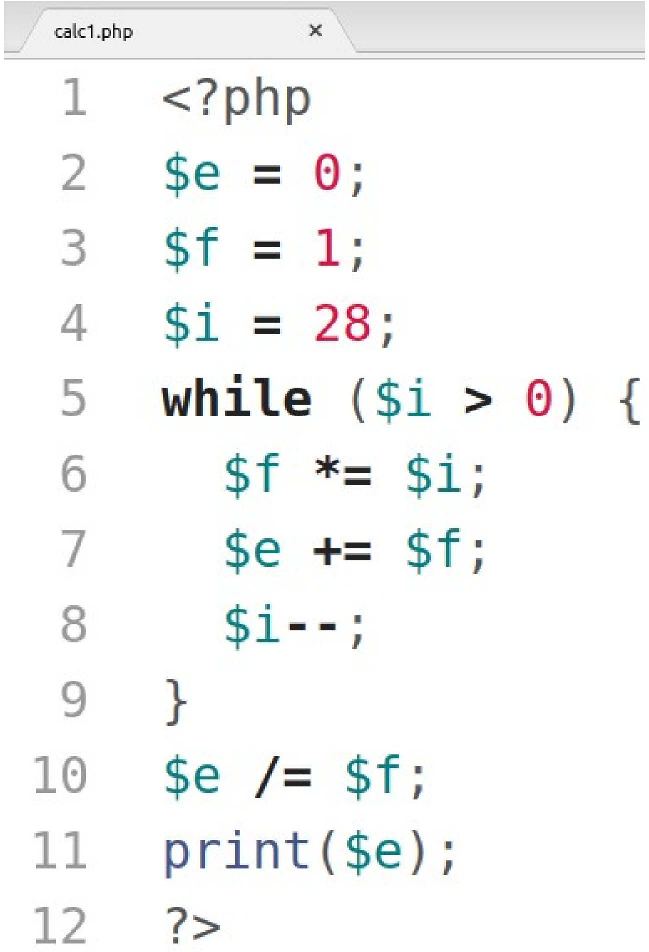
```
var e = 0.0, f = 1.0, i = 28;

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

e /= f;

console.log(e);
```



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

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

e /= f;

console.log(e);
```

PHP

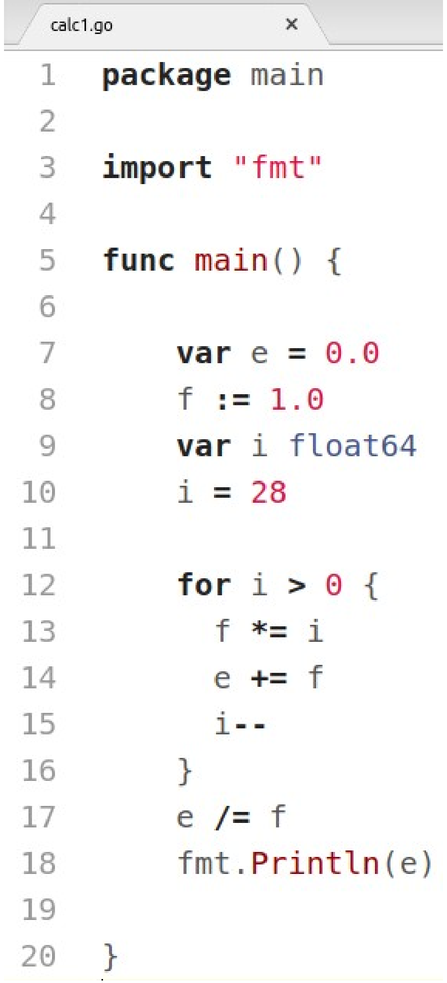
```
<?php
$e = 0;
$f = 1;
$i = 28;
while ($i > 0) {
        $f *= $i;
        $e += $f;
        $i–;
}
$e /= $f;
print($e);
?>
```



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

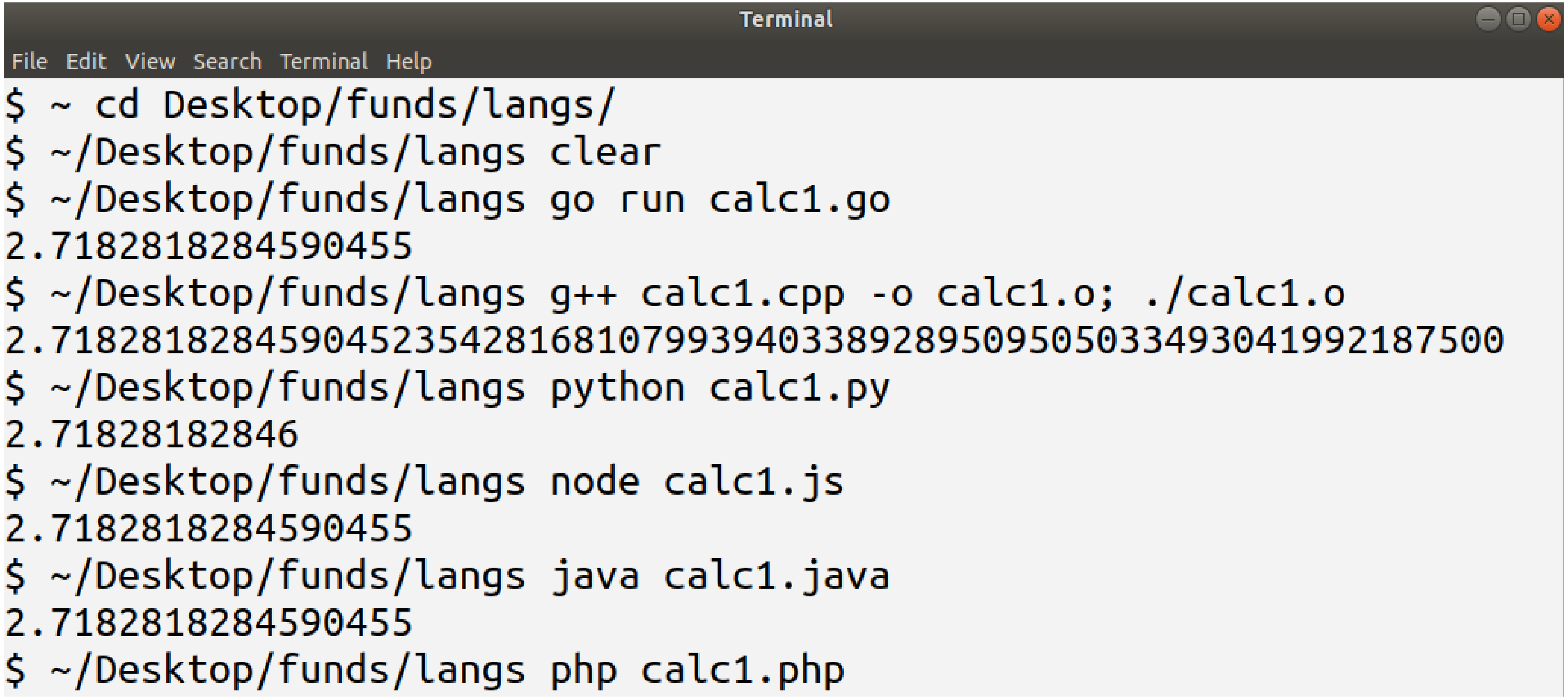
Go

```
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)

}
```



```
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)

}
```

Linux is your friend



```
$ ~ cd Desktop/funds/langs/
$ ~/Desktop/funds/langs clear
$ ~/Desktop/funds/langs go run calc1.go
2.718281828459045
$ ~/Desktop/funds/langs g++ calc1.cpp -o calc1.o; ./calc1.o
2.718281828459045235428168107993940338928950950503349304199218750 0
$ ~/Desktop/funds/langs python calc1.py
2.71828182846
$ ~/Desktop/funds/langs node calc1.js
2.718281828459045
$ ~/Desktop/funds/langs java calc1.java
2.718281828459045
$ ~/Desktop/funds/langs php calc1.php
```

Sepas <3

Contacts:

http://kouy.ir/tayyebi

Proudly a Sariab Blogger:

www.sariab.ir, info@sariab.ir

Reference:

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

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

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