

Trabalho 2 - Séries Infinitas

INF 110 Programação 1

Professor: André Santos
Alunos: Vinícius Zopelar - 108202
Yuri Bragine - 108199

1. Serie infinita para cálculo de e:

- Código:

```
trabalho > g++ euler.cpp > main0
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main(){
6
7      long double ntermos, euler, fatorial;
8
9      cin >> ntermos;
10     fatorial = 1.0;
11     euler = 1.0;
12
13
14     for(int i = 1.0; i<=ntermos; i++){
15         fatorial = i*fatorial;
16         euler += 1.0/(fatorial);}
17
18     cout << fixed << setprecision(62) << euler << endl;
19
20     return 0;}
```

- Resultados:

[illegible]

- Análise dos resultados:

n	Saída
1	2,0000000000
2	2,5000000000
3	2,6666666667
4	2,7083333333
5	2,7166666667
7	2,7182539683
10	2,7182818011
50	2,7182818285
100	2,7182818285

- Considerações:

O programa para calcular o número de Euler é bem simples, contendo apenas um for que faz a conta do fatorial e soma o termo atual ao Euler.

Esse programa começa a ficar bem próximo do resultado exato a partir dos 10 termos aumentando poucas casas decimais de precisão quando a conta é feita com 100 termos.

Sendo assim, percebe-se que o resultado já é consideravelmente confiável com relativamente poucos termos, nesse caso 10.

2. Series infinitas para cálculo de seno e cosseno:

- Código:

```
6  int main(){
7
8      long double ntermos, sen, cos, fatorial, angulo, rad;
9      long double pi = 3.141592653589;
10
11     cin >> ntermos >> angulo;
12
13     //calculando o angulo em rad
14     rad = pi/(180/angulo);
15
16     //calculando o seno
17     sen = rad; //primeiro termo da série
18     fatorial = (1*2)*3;
19
20     for(int i = 3; i<=4*ntermos; i+=4){
21         //      termo positivo      termo negativo
22         sen += (pow(rad,i+2)/((fatorial*(i+1))*(i+2))) - (pow(rad,i)/fatorial);
23         fatorial = (fatorial*(i+1)*(i+2)*(i+3)*(i+4)); //calculando o fatorial inicial que vai ser utilizado na próxima conta deste for.
24     }
25
26     //calculando o cosseno
27     cos = 1; // primeiro termo da série
28     fatorial = 1*2; //como o contador do for começa em 2, o fatorial inicial deve ser 2
29
30     for(int u = 2; u<=4*ntermos; u+=4){
31         //      termo positivo      termo negativo
32         cos += (pow(rad,u+2)/((fatorial*(u+1))*(u+2))) - (pow(rad,u)/fatorial);
33         fatorial = (fatorial*(u+1)*(u+2)*(u+3)*(u+4)); //calculando o fatorial inicial que vai ser utilizado na próxima conta deste for.
34     }
35
36     cout << "O seno de " << (int)angulo << " e: " << fixed << setprecision(62) << sen << endl;
37
38     cout << "O cosseno de " << (int)angulo << " e: " << fixed << setprecision(62) << cos << endl;
39
40
41     return 0;}
```

- Resultados:

```
1 60
0 seno de 60 e: 0.86629528378670215974302615258295645617181435227394104003906250
0 cosseno de 60 e: 0.50179620150040721141861121967764347573393024504184722900390625

C:\Users\vinic\OneDrive\Documentos\programas\trabalho>a
2 60
0 seno de 60 e: 0.86602544509964903500899915944621909602574305608868598937988281
0 cosseno de 60 e: 0.50000043343314388554626814809012103069107979536056518554687500

C:\Users\vinic\OneDrive\Documentos\programas\trabalho>a
5 60
0 seno de 60 e: 0.86602540378430643581292303068153159983921796083450317382812500
0 cosseno de 60 e: 0.50000000000022899612978424799258903021836886182427406311035156

C:\Users\vinic\OneDrive\Documentos\programas\trabalho>a
10 60
0 seno de 60 e: 0.86602540378430643581292303068153159983921796083450317382812500
0 cosseno de 60 e: 0.50000000000022899612978424799258903021836886182427406311035156

C:\Users\vinic\OneDrive\Documentos\programas\trabalho>a
100 60
0 seno de 60 e: 0.86602540378430643581292303068153159983921796083450317382812500
0 cosseno de 60 e: 0.50000000000022899612978424799258903021836886182427406311035156
```

- Análise dos resultados:

n	60°	
1	sen	0,866295283787
	cos	0,501796201500
2	sen	0,866025445100
	cos	0,500000433433
5	sen	0,866025403784
	cos	0,500000000000
10	sen	0,866025403784
	cos	0,500000000000
100	sen	0,866025403784
	cos	0,500000000000

- Considerações:

Esse programa é mais complexo quando comparado com o cálculo do número de Euler, já que alterna o sinal da conta entre subtração e adição, fazendo com que dois termos fiquem incluídos em somente um for.

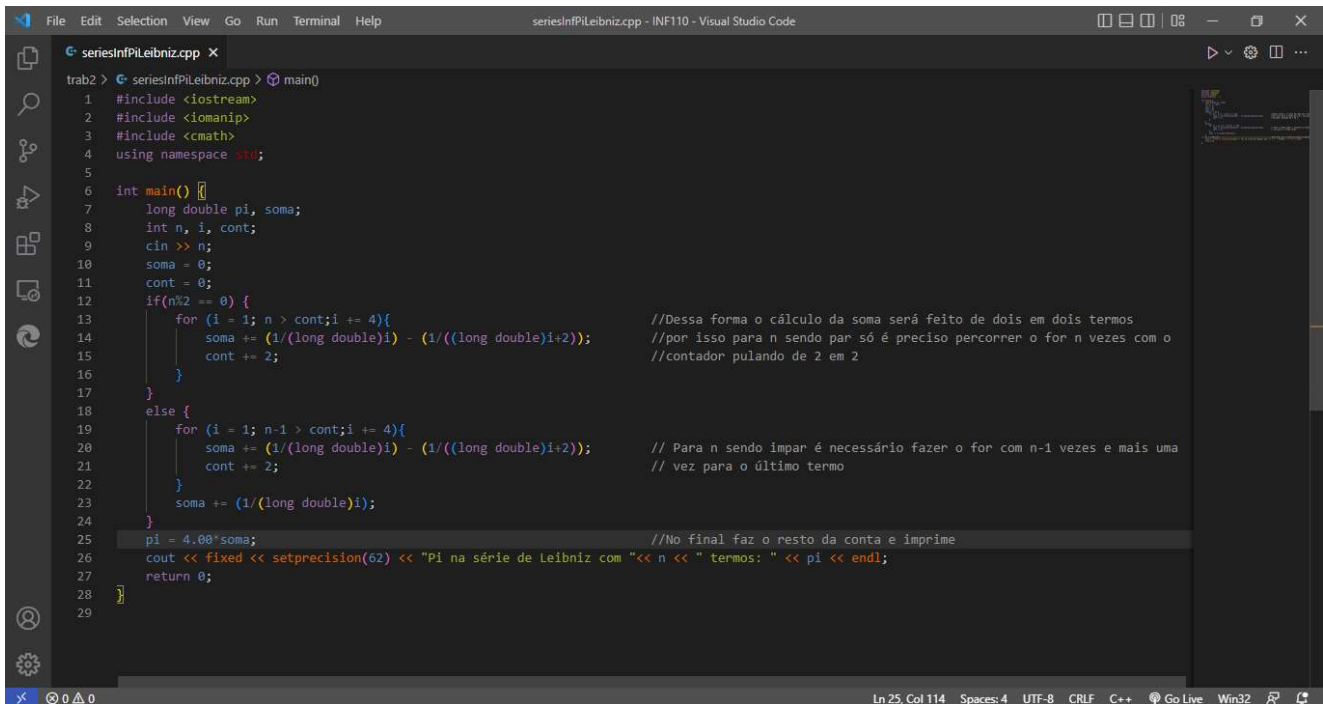
O resultado do cálculo começa a ficar preciso cedo nesse caso, já aos 5 termos temos uma precisão considerável, que aumenta cada vez mais de acordo com o número de termos.

Nesse caso, o resultado também já é preciso com poucos termos, atingindo a precisão mais cedo ainda do que o programa do número de Euler, sendo, nesse caso 5 termos suficientes.

3.14... Series infinitas para cálculo do π :

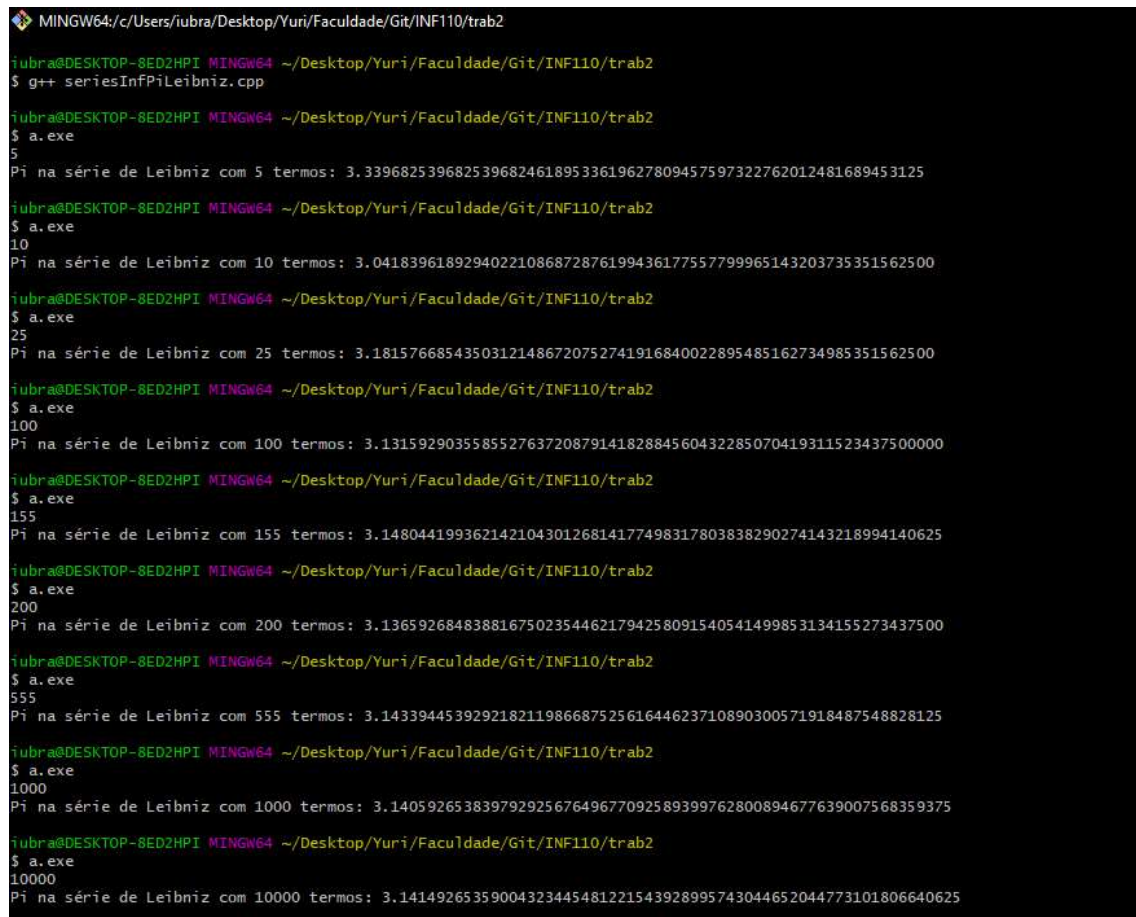
3.1 Série de Leibniz:

- Código:



```
trab2 > g++ seriesInfPiLeibniz.cpp & main()
1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4  using namespace std;
5
6  int main() {
7      long double pi, soma;
8      int n, i, cont;
9      cin >> n;
10     soma = 0;
11     cont = 0;
12     if(n%2 == 0) {
13         for (i = 1; n > cont; i += 4) {
14             soma += (1/(long double)i) - (1/((long double)i+2)); //Dessa forma o cálculo da soma será feito de dois em dois termos
15             cont += 2; //por isso para n sendo par só é preciso percorrer o for n vezes com o
16         } //contador pulando de 2 em 2
17     }
18     else {
19         for (i = 1; n-1 > cont; i += 4) {
20             soma += (1/(long double)i) - (1/((long double)i+2)); // Para n sendo impar é necessário fazer o for com n-1 vezes e mais uma
21             cont += 2; // vez para o último termo
22         }
23         soma += (1/(long double)i);
24     }
25     pi = 4.00*soma; //No final faz o resto da conta e imprime
26     cout << fixed << setprecision(62) << "Pi na série de Leibniz com "<< n << " termos: " << pi << endl;
27     return 0;
28 }
29
```

- Resultados:



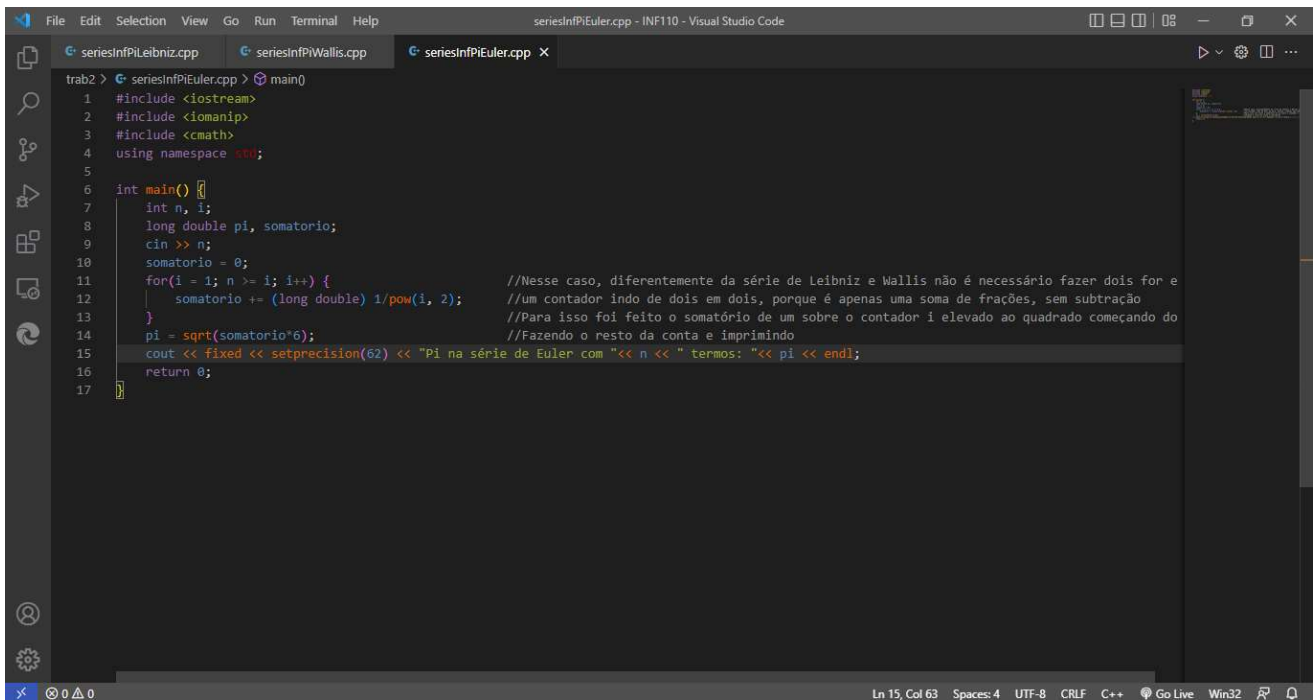
```
MINGW64:/c:/Users/iubra/Desktop/Yuri/Faculdade/Git/INF110/trab2
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ g++ seriesInfPiLeibniz.cpp
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
5
Pi na série de Leibniz com 5 termos: 3.33968253968253968253968246189533619627809457597322762012481689453125
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
10
Pi na série de Leibniz com 10 termos: 3.04183961892940221086872876199436177557799965143203735351562500
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
25
Pi na série de Leibniz com 25 termos: 3.18157668543503121486720752741916840022895485162734985351562500
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
100
Pi na série de Leibniz com 100 termos: 3.13159290355855276372087914182884560432285070419311523437500000
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
155
Pi na série de Leibniz com 155 termos: 3.14804419936214210430126814177498317803838290274143218994140625
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
200
Pi na série de Leibniz com 200 termos: 3.13659268483881675023544621794258091540541499853134155273437500
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
555
Pi na série de Leibniz com 555 termos: 3.14339445392921821198668752561644623710890300571918487548828125
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
1000
Pi na série de Leibniz com 1000 termos: 3.14059265383979292567649677092589399762800894677639007568359375
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
10000
Pi na série de Leibniz com 10000 termos: 3.14149265359004323445481221543928995743044652044773101806640625
```


- Código:

[illegible]

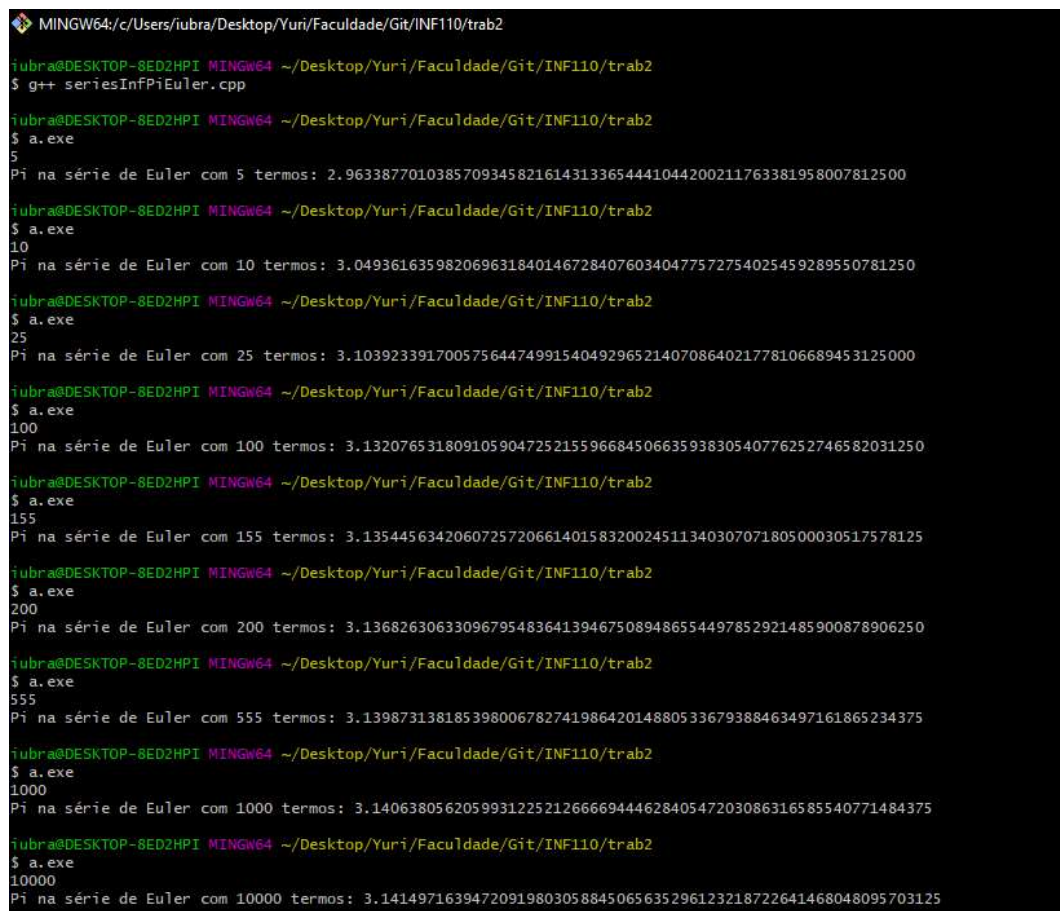
3.3 Série de Euler:

- Código:



```
1 #include <iostream>
2 #include <iomanip>
3 #include <cmath>
4 using namespace std;
5
6 int main() {
7     int n, i;
8     long double pi, somatorio;
9     cin >> n;
10    somatorio = 0;
11    for(i = 1; n >= i; i++) {
12        somatorio += (long double) 1/pow(1, 2); //Nesse caso, diferentemente da série de Leibniz e Wallis não é necessário fazer dois for e
13    } //um contador indo de dois em dois, porque é apenas uma soma de frações, sem subtração
14    pi = sqrt(somatorio*6); //Para isso foi feito o somatório de um sobre o contador i elevado ao quadrado começando do
15    cout << fixed << setprecision(62) << "Pi na série de Euler com "<< n << " termos: "<< pi << endl; //Fazendo o resto da conta e imprimindo
16    return 0;
17 }
```

- Resultados:



```
MINGW64:/c:/Users/iubra/Desktop/Yuri/Faculdade/Git/INF110/trab2
iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ g++ seriesInfPiEuler.cpp

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
5
Pi na série de Euler com 5 termos: 2.96338770103857093458216143133654441044200211763381958007812500

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
10
Pi na série de Euler com 10 termos: 3.04936163598206963184014672840760340477572754025459289550781250

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
25
Pi na série de Euler com 25 termos: 3.10392339170057564474991540492965214070864021778106689453125000

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
100
Pi na série de Euler com 100 termos: 3.13207653180910590472521559668450663593830540776252746582031250

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
155
Pi na série de Euler com 155 termos: 3.13544563420607257206614015832002451134030707180500030517578125

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
200
Pi na série de Euler com 200 termos: 3.13682630633096795483641394675089486554497852921485900878906250

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
555
Pi na série de Euler com 555 termos: 3.13987313818539800678274198642014880533679388463497161865234375

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
1000
Pi na série de Euler com 1000 termos: 3.14063805620599312252126666944462840547203086316585540771484375

iubra@DESKTOP-8ED2HPI MINGW64 ~/Desktop/Yuri/Faculdade/Git/INF110/trab2
$ a.exe
10000
Pi na série de Euler com 10000 termos: 3.14149716394720919803058845065635296123218722641468048095703125
```


- Comparação dos resultados:

n	Leibniz	Wallis	Euler	Variação	pi
5	3,3396825	3,4133333	2,9633877	0,1782049	3,1415927
10	3,0418396	3,0021760	3,0493616	0,0922310	
25	3,1815767	3,2025774	3,1039234	0,0376693	
100	3,1315929	3,1260789	3,1320765	0,0095161	
155	3,1480442	3,1516779	3,1354456	0,0061470	
200	3,1365927	3,1337875	3,1368263	0,0047663	
555	3,1433945	3,1444191	3,1398731	0,0017195	
1000	3,1405927	3,1400238	3,1406381	0,0009546	
10000	3,1414927	3,1414356	3,1414972	0,0000955	

- Considerações:

Após realizar as séries de cálculos para pi, a partir do método de Leibniz, Wallis e Euler pode-se perceber que todos foram se tornando mais precisos conforme o número de termos aumentava. E além disso, o método de Euler mesmo com poucos termos, em todos os casos, se aproximou mais do valor real de pi em comparação a Leibniz e Wallis. E mesmo assim, a variação entre o valor de pi em relação ao método de Euler é demasiadamente grande, chegando a 0,0000955 com 10.000 termos na série. É interessante perceber que, para Leibniz e Wallis, com uma quantidade par de termos a série resulta em um valor menor que pi e uma quantidade ímpar de termos resulta em um valor maior, diferentemente de Euler que é um valor crescente.

- Vinícius (Exercícios 1 e 2)

- Yuri (Exercício 3)