Desempenho Sequencial

a) Escreva um codigo simples em Python (ou Matlab, Octave, ou Java) que seja principalmente um Loop (como o exemplo da aula 1) e calibre as entradas para que ele demore ao menos 2 min executando em sua maquina

Foi desenvolvido o código a seguir na linguagem de programação *Python*, que calcula o fatorial de um número natural. O seu tempo de execução registrado está apresentado na tabela 1, no item c deste relatório.

Listing 1: Código em Python para cálculo do fatorial de um número natural n e impressão na tela do seu tempo de execução

```
def fatorial (n):
    fat = 1
    for i in range (1,n+1):
        fat*=i
    return fat

def main ():
    start = time.time()
    fatorial(900000)
    end = time.time()
    print(end - start)

main()
```

Este código foi avaliado em duas situações: A primeira, utilizando a versão Python2, foi executada com a linha de comando a seguir:

```
$ python2 main.py
```

A segunda avaliação foi feita utilizando a versão Python3, conforme a seguinte linha de comando:

```
$ python3 main.py
```

b) Reescreva o mesmo condigo em C e compare o desempenho (tempo de execução) com o da versão anterior.

A seguir está apresentado o mesmo código desenvolvido para o item a traduzido para a linguagem C. O tempo de execução registrado está apresentado na tabela 1 no item c deste relatório.

Listing 2: Código em C para cálculo do fatorial de um número natural n e impressão na tela do seu tempo de execução

```
float fatorial(int n){
    float fat = 1.;
    for (int i=1; i<=n; i++){
        fat*=i;
    }
    return fat;
}
```

```
int main(){
    clock_t t = clock();
    fatorial(900000);
    t = clock()-t;
    printf("%lf", ((double)t)/((CLOCKS_PER_SEC));
    return 0;
}
```

Este código foi executado conforme as linhas de comando a seguir:

```
$gcc main.c -o main
$./main
```

c) Recompile agora usando a flag -O3 (ex. gcc - O3 ...). Execute o mesmo codigo e compare o desempenho (tempo de execução) com os das versões anteriores

Por fim, o código apresentado no item b foi compilado utilizando a flag de otimização $-O3$, conforme a seguinte linha de comando:

```
$gcc -O3 main.c -o main
$./main
```

Os tempos de execução registrados para os casos apresentados nos itens *a, b* e c deste relatório estão apresentados na tabela a seguir. Para todas as execuções o parâmetro n utilizado foi 900000.

	Tempo de execução (em segundos)
Python2	404.755491018
Python3	405.51060724258423
C	0.007757
C $\|\|gcc - O3$	0.000009

Tabela 1: Tempos de execução em segundos registrados para as diferentes execuções descritas

É possível notar significativa diferença entre os tempos de execução do código em linguagem Python para o código em linguagem C. O tempo de execução do código executado em C sem utilização de flag de otimização representou 0.0019% do tempo necessário para a execução em Python3. Percebe-se ainda que a utilização da flag de otimização $-O3$ tornou o desempenho do código na linguagem C ainda maior. A sua execução foi mais de 800 vezes mais rápido em relação à compilação sem o uso da flag de otimização.