**UNIVERSIDADE FEDERAL DE VIÇOSA**

**CAMPUS FLORESTAL**

PABLO FERREIRA - 3480

SAMUEL SENA - 3494

**TRABALHO PRÁTICO II**

**RELATÓRIO TAREFA A**

FLORESTAL

2019

Sumário

[Sumário 2](#_gjdgxs)

[Introdução 3](#_30j0zll)

[Tipos de Artes 7](#_3znysh7)

[Desenvolvimento 11](#_2et92p0)

[Conclusão 14](#_tyjcwt)

# Introdução

O trabalho apresentado a seguir entrega dum algoritmos capaz de resolver o problema da pirâmide de números (Tarefa A).

Inicialmente para se executar o programa da Tarefa A, é necessário realizar a compilação do código fonte em C. Para isso, em algum terminal Linux execute o “*makefile”* da seguinte forma:

Para compilar:

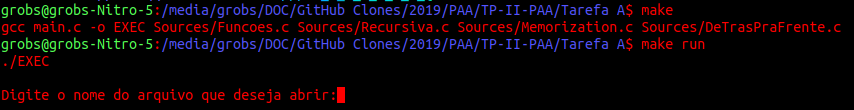
$ make

E para executar:

$ make run

A figura abaixo exemplifica o processo de compilação e execução pelo Linux:

Figura 1

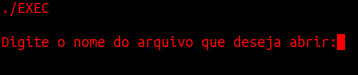


Fonte: Terminal Linux

O algoritmo foi testado apenas em sistema operacional baseado em Linux, a execução em Windows pode não ser satisfatória.

O programa inicialmente exibirá um menu com as seguintes opções:

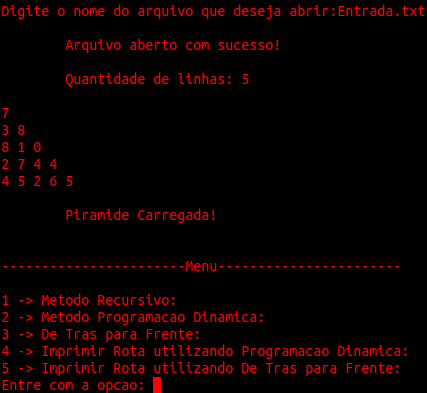
Figura 2



Fonte: Terminal Linux

Primeiramente, entre com o nome do arquivo de entrada desejado, em seguida, caso a abertura seja realizada com sucesso, a pirâmide lida é impressa na tela e um menu com 5 opções é exibido. Dentre as possibilidades de opções estão: 1 - Resultado da maior soma utilizando recursividade; 2 - Resultado da maior soma utilizando programação dinâmica; 3 - Resultado da maior soma utilizando algoritmo de trás para frente; 4 - Imprimir rota utilizando como base programação dinâmica; 5 - Imprimir rota utilizando como base algoritmo de trás para frente. A figura a seguir ilustra bem a situação descrita:

Figura 3



Fonte: Terminal Linux

Após a escolha desejada e execução da mesma, o programa ira sempre finalizar em seguida. Caso deseje executar mais de uma operação em um mesmo arquivo de entrada, será necessário reexecutar o programa.

# Comparação de tempos da Tarefa A

Os gráficos a seguir servem de comparação entre o desempenho dos algoritmos utilizados. Os mesmos relacionam tempo X tamanho da entrada (em linhas da pirâmide):

É notável a enorme diferença de desempenho entre o algoritmo recursivo com os demais com entradas maiores que 25 linhas. Para as pequenas entradas, nenhuma diferença impactante é sentida. Dentre os 3, o de melhor desempenho é o algoritmo de trás para frente.

# Modo DEBUG

O programa conta com a opção de utilização em modo “debug” que (quando ativado) exibe na tela o tempo de execução de cada algoritmo escolhido, para utiliza-lo, o valor de “DEBUG” no arquivo “main.c” deve ser definido para o 1, caso seja definido como 0, o programa ira rodar sem exibir o tempo de execução de cada algoritmo.

A medição do uso de CPU de cada algoritmo é realizado através da chamada do seguinte comando no momento da execução:

$ command time -v ./EXEC

Assim sendo, é necessário ter o programa “time” devidamente instalado no sistema operacional Linux no qual a medição será realizada. O uso de CPU será impresso da seguinte forma:

A medição do uso de memória é obtido através do uso da ferramenta “Valgrind”, que assim como o programa “time” é chamado no momento da execução do programa. Para executar a ferramenta e obter o uso de memória execute:

$ valgrind ./EXEC

Para facilitar o uso da ferramenta “Valgrind”, o comando para a respectiva chamada foi adicionado ao “makefile”. Para realizar a chamada da medição de memória utilizando o “makefile” digite o seguinte comando:

$ make Mem

Em ambos os comandos, o seguinte resultado será impresso na tela:

# Desenvolvimento

# Conclusão

Sem dúvidas, o desenvolvimento de uma ideia abstrata para o devido cumprimento do comportamento desejado do programa requisitado pela descrição do trabalho foi a maior empecilho em comparação com os demais. A decisão de se trabalhar com uma matriz de caracteres serviu como uma espécie de “gatilho” mental para fazer com que os demais problemas que surgissem em seguida fossem facilmente contornados. Portanto, após longos testes de execução e verificações no código fonte, o programa se encontra executando da maneira desejada. E por último, mas não menos importante, é notável a suma importância com relação a aprendizagem e aperfeiçoamento de conceitos em programação, não ficando limitado apenas a *sintax* da linguagem, mas abrangendo também ao pensamento lógico que possibilita a correta programação.

Agradecimentos ao professor Daniel Mendes pela oportunidade de realização do trabalho e dúvidas sanadas.

Todo o desenvolvimento e distribuição do trabalho encontra-se hospedado na seguinte página do [GitHub](https://github.com/Globson/Gerador-de-Arte-TP_PAA).