

UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG

Atividade Prática - Aula 02

JOÃO MARCOS RIBEIRO TOLENTINO - 2021049536

Introdução

O problema foi fazer 2 programas de duas formas diferentes. Ou seja, 4 programas (2 programas que calcula o fatorial e 2 que calculam o fibonacci). Sendo para cada um duas versões: a iterativa e a recursiva. Como pedido, é empregado um parâmetro de linha de comando que define a tarefa a ser utilizada, nominalmente o fatorial ou o número de Fibonacci e também a versão desejada.

Método

Basicamente, o programa foi desenvolvido em C++, compilado pelo compilador GCC da GNU Compiler Collection. O Computador utilizado tem as seguintes especificações:

Sistema Operacional: Linux Mint 23.1

Versão: 5.6.5

Processador: AMD Ryzen 5 5500U with Radeon Graphics × 6

Memória: 5.6 GB

Funções

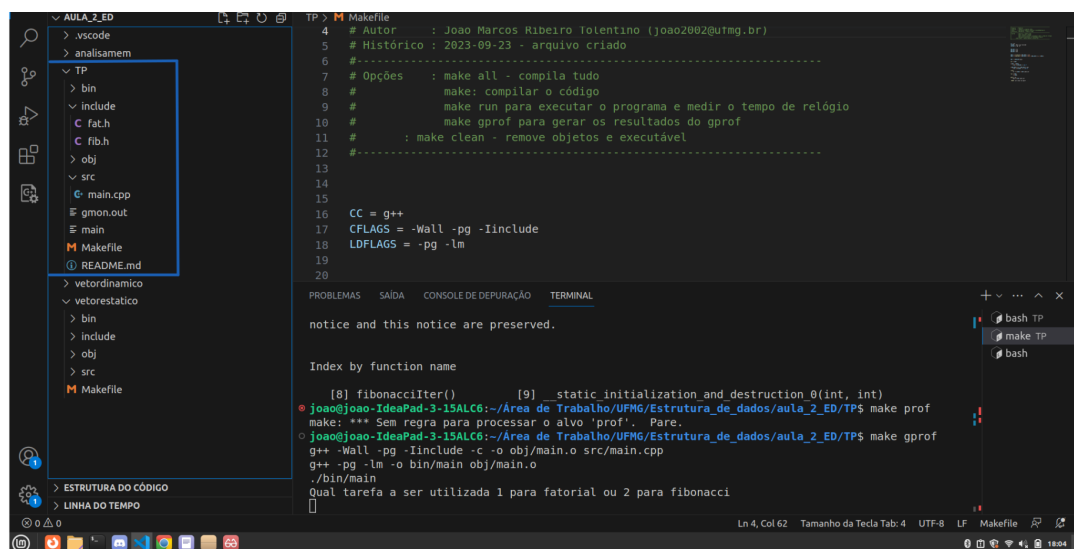
Fatorial (Iterativa): Essa função calcula o fatorial de maneira iterativa. Inicialmente cria-se uma variável que representa o fatorial e após isso usa o while para multiplicar essa variável que a representa de 1 até n que é o resultado final.

Fatorial (recursiva): Essa função calcula o fatorial de um dado número “n” de maneira recursiva. Ela vai verificar se esse valor é 1, caso seja, ele retorna o próprio. No contrário,

fibonacciRec(int n): Esta função calcula o termo n da sequência de Fibonacci de forma recursiva. Caso n seja menor que 3, ela retorna. Caso contrário, a função chama a si mesma para os dois termos anteriores da sequência ($n-1$ e $n-2$) e retorna a soma desses dois valores.

fibonacciIter(): Esta função exibe os termos da sequência de Fibonacci de forma iterativa. Ela começa solicitando ao usuário quantos termos da sequência ele deseja calcular (limitado de 1 a 46 que é especificado na tela como pedido no enunciado da atividade devido a limitações de precisão). Em seguida, utiliza um loop while para calcular e exibir os termos da sequência até o n -ésimo termo especificado. Ela mantém o controle dos termos anteriores (ult e penult) para calcular o próximo termo.

Estrutura do projeto:



The screenshot shows a VS Code editor with a project structure on the left and a Makefile in the center. The project structure includes a 'TP' directory with subdirectories 'bin', 'include', 'src', and 'obj'. The 'src' directory contains 'main.cpp', 'gmon.out', and 'main'. The 'Makefile' is located in the 'TP' directory. The Makefile content is as follows:

```
4 # Autor : Joao Marcos Ribeiro Tolentino (joao2002@utm.br)
5 # Histórico : 2023-09-23 - arquivo criado
6 #-----
7 # Opções : make all - compila tudo
8 #         make: compilar o código
9 #         make run para executar o programa e medir o tempo de relógio
10 #         make gprof para gerar os resultados do gprof
11 #         : make clean - remove objetos e executável
12 #-----
13
14
15
16 CC = g++
17 CFLAGS = -Wall -pg -Iinclude
18 LDFLAGS = -pg -lm
19
20
```

The terminal at the bottom shows the following commands and output:

```
notice and this notice are preserved.

Index by function name

[8] fibonacciIter() [9] __static_initialization_and_destruction_0(int, int)
* joao@joao-IdeaPad-3-15ALC6:~/Área de Trabalho/UFMG/Estrutura_de_dados/aula_2_ED/TP$ make prof
make: *** Sem regra para processar o alvo 'prof'. Pare.
o joao@joao-IdeaPad-3-15ALC6:~/Área de Trabalho/UFMG/Estrutura_de_dados/aula_2_ED/TP$ make gprof
g++ -Wall -pg -Iinclude -c -o obj/main.o src/main.cpp
g++ -pg -lm -o bin/main obj/main.o
./bin/main
Qual tarefa a ser utilizada 1 para fatorial ou 2 para fibonacci
```

Análise de Complexidade

Fatorial (Iterativa): Como essa função calcula o fatorial de n . Sua complexidade é linear, ou seja é representada por $O(n)$. Assim, a função possui sua complexidade definida pelo número passado como parâmetro.

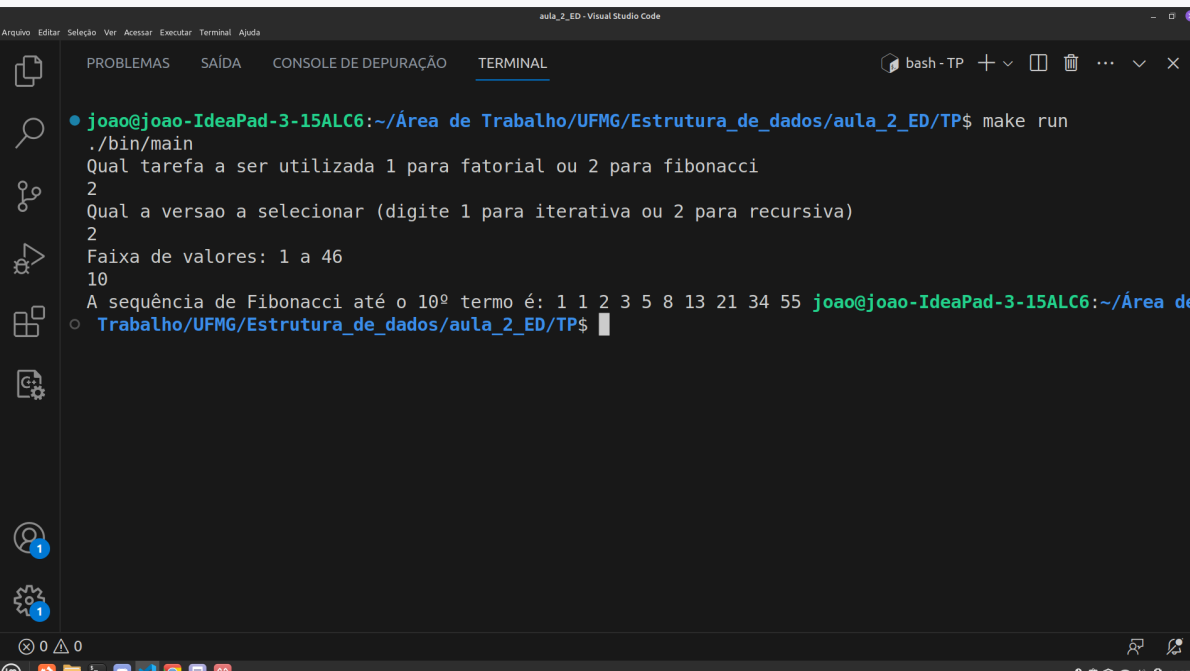
Fatorial (recursiva): Calcula o fatorial de um dado número n . A complexidade de tempo dessa função é exponencial, $O(2^n)$, onde ' n ' é o valor de entrada. Cada chamada

recursiva multiplica o valor atual de 'n' pelo resultado da chamada recursiva anterior. Como resultado, a função realizará um número exponencial de chamadas recursivas (2^n chamadas) para calcular o fatorial. Complexidade exponencial são comuns em algoritmos recursivos ineficientes.

fibonacciRec(int n): A complexidade dessa função é exponencial. Ou seja, $O(2^n)$. No caso, a cada chamada recursiva, a função faz duas chamadas adicionais para a mesma. Assim, a função se torna bastante ineficiente para valores grandes de n.

fibonaccilter: Para gerar os termos de fibonacci temos um loop que itera para cada termo da sequência dada. Por isso, a complexidade da função é $O(n)$, o que quer dizer que ela é eficiente e o tempo de execução aumenta logicamente a depender do valor de "n".

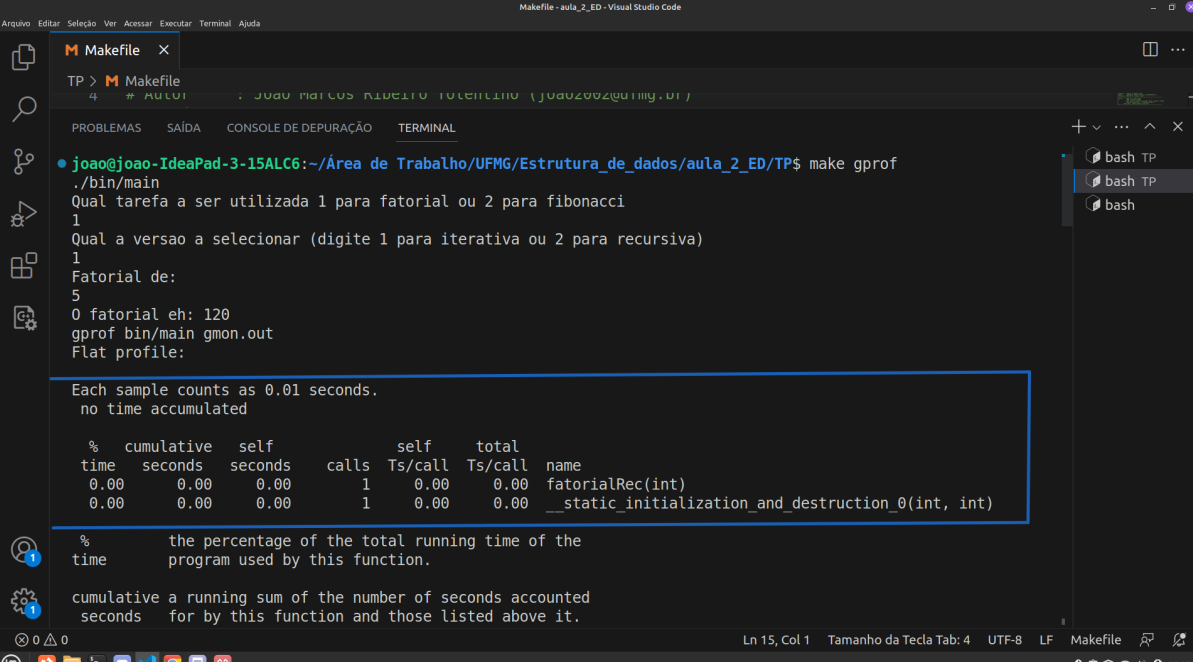
Resultado de exemplo em tela da sequência de fibonacci (observe as opções a escolher e a faixa de valor definida)



```
aula_2_ED - Visual Studio Code
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL
bash - TP + - □ □ □ ... ▾ ✕
• joao@joao-IdeaPad-3-15ALC6:~/Área de Trabalho/UFGM/Estrutura_de_dados/aula_2_ED/TP$ make run
./bin/main
Qual tarefa a ser utilizada 1 para fatorial ou 2 para fibonacci
2
Qual a versao a selecionar (digite 1 para iterativa ou 2 para recursiva)
2
Faixa de valores: 1 a 46
10
A sequência de Fibonacci até o 10º termo é: 1 1 2 3 5 8 13 21 34 55 joao@joao-IdeaPad-3-15ALC6:~/Área de
Trabalho/UFGM/Estrutura_de_dados/aula_2_ED/TP$
```

Análise Experimental

Função FatorialRec (Fatorial recursiva):



The screenshot shows a Visual Studio Code window with a terminal running a program. The user has executed `make gprof`. The program prompts for a task (1 for factorial, 2 for fibonacci) and a version (1 for iterative, 2 for recursive). The user enters 1 for both. The program calculates the factorial of 5, resulting in 120. The gprof output is shown, with a table of execution statistics for the recursive factorial function.

```
TP > Makefile
4 # AUL01 : JOAO MARCOS RIBEIRO TUCENTINO (joao2002@unmg.br)

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL

• joao@joao-IdeaPad-3-15ALC6:~/Área de Trabalho/UFMG/Estrutura_de_dados/aula_2_ED/TP$ make gprof
./bin/main
Qual tarefa a ser utilizada 1 para fatorial ou 2 para fibonacci
1
Qual a versao a selecionar (digite 1 para iterativa ou 2 para recursiva)
1
Fatorial de:
5
0 fatorial eh: 120
gprof bin/main gmon.out
Flat profile:

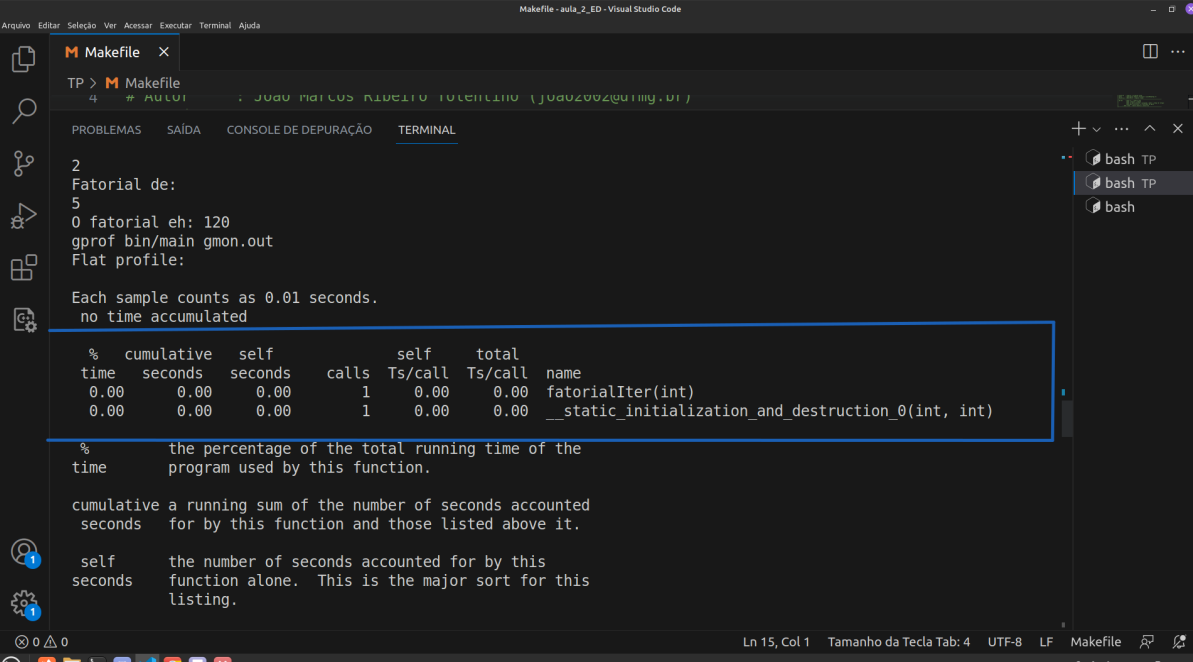
Each sample counts as 0.01 seconds.
no time accumulated

% cumulative self self total
time seconds seconds calls Ts/call Ts/call name
0.00 0.00 0.00 1 0.00 0.00 fatorialRec(int)
0.00 0.00 0.00 1 0.00 0.00 __static_initialization_and_destruction_0(int, int)

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
seconds for by this function and those listed above it.
```

Função FatorialIter (FatorialIterativo):



The screenshot shows a Visual Studio Code window with a terminal running a program. The user has entered 2 for both task and version. The program calculates the factorial of 5, resulting in 120. The gprof output is shown, with a table of execution statistics for the iterative factorial function.

```
TP > Makefile
4 # AUL01 : JOAO MARCOS RIBEIRO TUCENTINO (joao2002@unmg.br)

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL

2
Fatorial de:
5
0 fatorial eh: 120
gprof bin/main gmon.out
Flat profile:

Each sample counts as 0.01 seconds.
no time accumulated

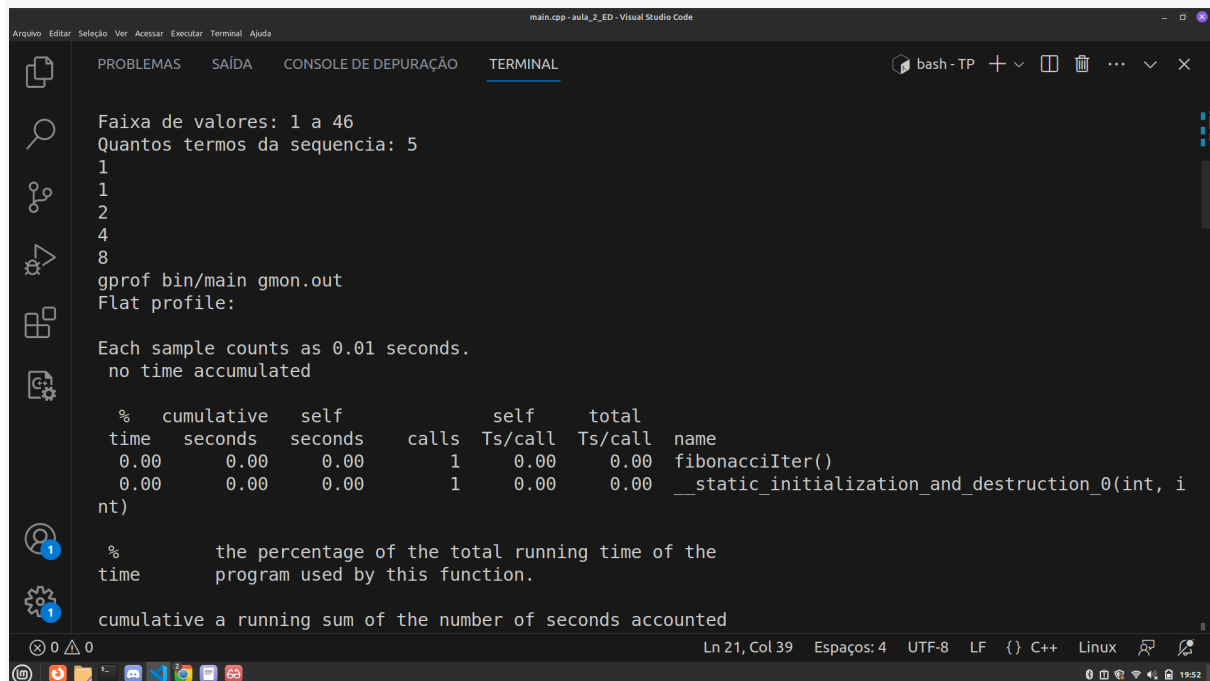
% cumulative self self total
time seconds seconds calls Ts/call Ts/call name
0.00 0.00 0.00 1 0.00 0.00 fatorialIter(int)
0.00 0.00 0.00 1 0.00 0.00 __static_initialization_and_destruction_0(int, int)

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
seconds for by this function and those listed above it.

self the number of seconds accounted for by this
seconds function alone. This is the major sort for this
listing.
```

Função Fibonaccilter (Fibonacci iterativo):



```
main.cpp - aula_2_ED - Visual Studio Code
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
bash - TP + - - - - -

Faixa de valores: 1 a 46
Quantos termos da sequencia: 5
1
1
2
4
8
gprof bin/main gmon.out
Flat profile:

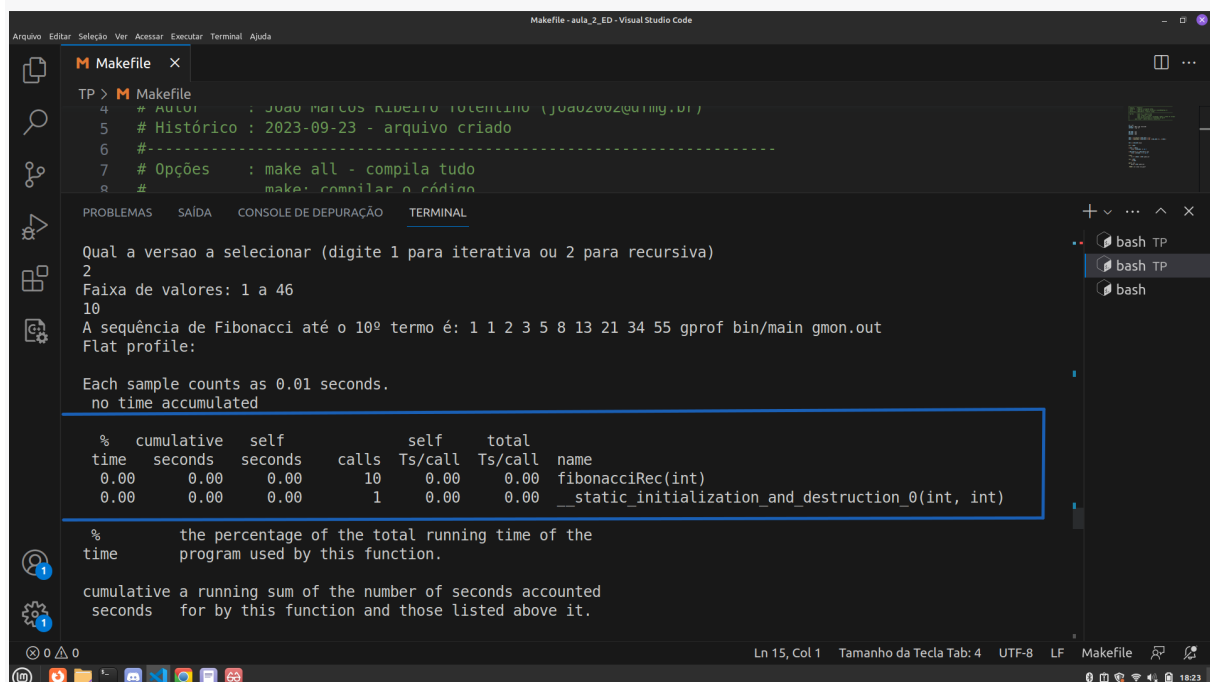
Each sample counts as 0.01 seconds.
no time accumulated

% cumulative self self total
time seconds seconds calls Ts/call Ts/call name
0.00 0.00 0.00 1 0.00 0.00 fibonacciIter()
0.00 0.00 0.00 1 0.00 0.00 __static_initialization_and_destruction_0(int, i
nt)

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
```

função FibonacciREc (Fibonacci recursivo)



```
Makefile - aula_2_ED - Visual Studio Code
Makefile
TP > Makefile
4 # Autor : João Marcos Ribeiro Potentino (joao2002@gmail.com)
5 # Histórico : 2023-09-23 - arquivo criado
6 #
7 # Opções : make all - compila tudo
8 # make: compilar o código

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
bash - TP + - - - - -

Qual a versao a selecionar (digite 1 para iterativa ou 2 para recursiva)
2
Faixa de valores: 1 a 46
10
A sequência de Fibonacci até o 10º termo é: 1 1 2 3 5 8 13 21 34 55
gprof bin/main gmon.out
Flat profile:

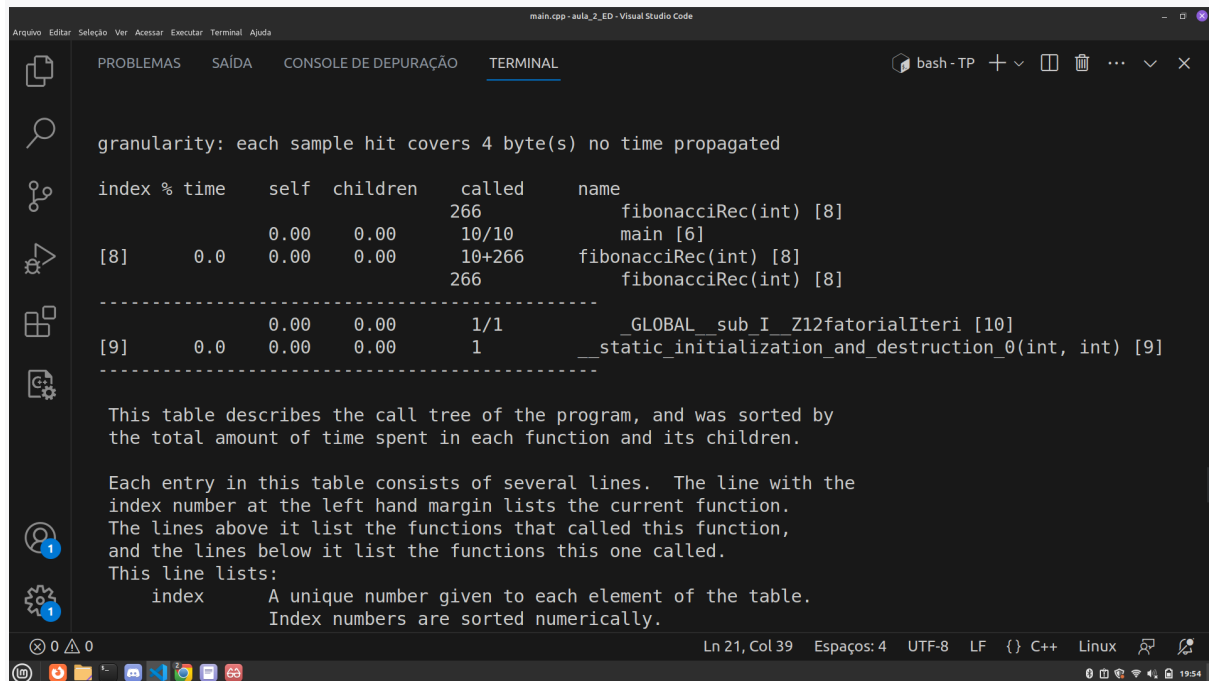
Each sample counts as 0.01 seconds.
no time accumulated

% cumulative self self total
time seconds seconds calls Ts/call Ts/call name
0.00 0.00 0.00 10 0.00 0.00 fibonacciRec(int)
0.00 0.00 0.00 1 0.00 0.00 __static_initialization_and_destruction_0(int, int)

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
seconds for by this function and those listed above it.
```

Chamadas que o gprof identifica (no exemplo: FibonacciRecursivo):



```
main.cpp - aula_2_ED - Visual Studio Code
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
bash - TP  +  -  [ ]  [ ]  ...  -  x

granularity: each sample hit covers 4 byte(s) no time propagated

index % time      self  children    called    name
-----
[8]    0.0        0.00    0.00      10/10      fibonacciRec(int) [8]
[8]    0.0        0.00    0.00     10+266     fibonacciRec(int) [8]
[8]    0.0        0.00    0.00      266       fibonacciRec(int) [8]
-----
[9]    0.0        0.00    0.00      1/1       _GLOBAL_sub_I_Z12fatorialIteri [10]
[9]    0.0        0.00    0.00      1        __static_initialization_and_destruction_0(int, int) [9]
-----

This table describes the call tree of the program, and was sorted by
the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the
index number at the left hand margin lists the current function.
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
This line lists:
    index      A unique number given to each element of the table.
               Index numbers are sorted numerically.
```

Conclusão

Com esse trabalho deu para entender o funcionamento e como se faz a análise de complexidade de um programa. Dando assim uma boa inicializada prática no conteúdo dado em Estrutura de dados. Tive alguns desafios durante a sua implementação - a exemplo da utilização da ferramenta “gprof”. Concluindo, foi construído nessa atividade 4 programas simples e analisados sua eficiência de execução.

Instruções para compilar e executar:

1. Abrir o terminal e entrar na pasta TP

2. Para compilar e rodar o programa:

Comando: `make run`

ou: `g++ -o main main.cpp` e `./main` na pasta correta (conforme estrutura definida acima)

3. Para deletar os arquivos desnecessários:

Comando: `make clean`

