Instituto de Matemática e Estatística da USP

MAC0216 - Técnicas de Programação I - 2s2022

EP3
Entrega até 8:00 de 14/11/2022
(INDIVIDUAL)

Prof. Daniel Macêdo Batista

1 Problema

Bibliotecas são muito úteis para evitar a reescrita de trechos de código que já foram escritos no passado para resolver um dado problema. Isso acelera o processo de escrita de novos programas além de deixar o código bem organizado, já que cada biblioteca estaria resolvendo uma parte bem definida de todo o problema e mudanças em uma biblioteca não afetariam outras bibliotecas.

Este EP vai explorar a implementação, o teste e a documentação de bibliotecas estáticas e dinâmicas em C, além da implementação de um pequeno bash script responsável por compilar as bibliotecas, compilar um programa de teste e gerar a documentação.

Implementação de bibliotecas é mais uma importante Técnica de Programação pois agiliza o desenvolvimento de sistemas computacionais que se baseiam em conjuntos de funções frequentemente necessárias para o domínio de aplicação de uma dada organização.

2 Requisitos

2.1 Biblioteca estática

A biblioteca estática, de nome `propriedadesnumericas`, deverá implementar as seguintes funções:

- `int ehPar(long long)`: devolve 1 se o parâmetro for par ou 0 se o parâmetro for ímpar;
- `int ehPrimo(long long)`: devolve 1 se o parâmetro for um número primo ou 0 se não for um número primo;
- `int ehQuadradoPerfeito(long long)`: devolve 1 se o parâmetro for um quadrado perfeito ou 0 se o parâmetro não for um quadrado perfeito;

- `int ehCuboPerfeito(long long)`: devolve 1 se o parâmetro for um cubo perfeito ou 0 se o parâmetro não for um cubo perfeito;
- `int ehFibonacci(long long)`: devolve 1 se o parâmetro for um número de Fibonacci ou 0 se o parâmetro não for um número de Fibonacci;
- `int ehFatorial(long long)`: devolve 1 se o parâmetro for um número fatorial ou 0 se o parâmetro não for um número fatorial;

A biblioteca precisará ser implementada em dois arquivos. Um contendo a interface e outro contendo a implementação propriamente dita. Os únicos `.h` externos que poderão ser usados são `<stdio.h>` e `<stdlib.h>`. As bibliotecas não poderão depender de nenhum arquivo externo para os cálculos necessários e também não poderão conter nenhuma tabela pré-calculada de valores. Todos os cálculos precisam ser realizados em tempo de execução.

2.2 Biblioteca dinâmica

A biblioteca dinâmica, de nome `vetoraleatorio`, deverá implementar apenas uma função:

- `long long * criaVetorAleatorio(int semente, int tamanho)`: devolve um vetor de `tamanho` valores inteiros do tipo `long long`, com o gerador de números aleatórios sendo inicializado com o valor de `semente`.

A biblioteca precisará ser implementada em dois arquivos. Um contendo a interface e outro contendo a implementação propriamente dita. Os únicos `.h` externos que poderão ser usados são `<stdio.h>` e `<stdlib.h>`. Recomenda-se fortemente o uso das funções `rand` e `srand`, que podem ser compreendidas com a leitura da manpage 3 da função `rand` (Um `man 3 rand` exibe essa manpage).

2.3 Documentação

Os comentários nos códigos-fonte explicando como usar as funções das bibliotecas precisarão ser suficientes para que o programa `doxygen`[1] possa gerar uma documentação das bibliotecas em arquivos html e demais formatos necessários para que a documentação possa ser lida em um navegador web.

2.4 Código de teste

Para conferir que as funções da biblioteca estática possuem o comportamento esperado, um programa, implementado em um arquivo chamado `testa.c`, deverá ser desenvolvido. Esse programa precisará criar um (ou mais) vetor utilizando a função `criaVetorAleatorio` da biblioteca dinâmica e realizar testes que você considere suficientes para atestar que as funções estão corretas com os números desse vetor. Esse programa de teste não deverá receber nenhum parâmetro na linha de comando. Em todas as chamadas às funções das bibliotecas, considere que os parâmetros estão dentro da faixa de valores esperados para uma variável do tipo `long long`.

[1] `https://www.doxygen.nl/`

2.5 Código em bash

A compilação das bibliotecas e do programa de testes deverá ser realizada por um bash script nomeado como `compila.sh`. Caso ocorra algum erro nas compilações, o script deverá retornar `1` . Caso contrário deverá retornar `0`. O bash script também precisará executar o `doxygen` para gerar a documentação das bibliotecas. De forma similar à compilação dos programas, se a execução do `doxygen` gerar algum erro, o script deverá retornar `1` e, em caso contrário, retornar `0`. O bash script não deve receber nenhum parâmetro ao ser invocado no shell.

2.6 Relatório

O desempenho de todas as funções das duas bibliotecas deverá ser medido em termos do tempo necessário para cada função executar. Esse tempo poderá ser medido com a função `gettimeofday`[2] que possui interface no header `sys/time.h>` diretamente no código de teste. Caso ela não seja suficiente para retornar os tempos (se todos os valores forem zero), você pode utilizar outra função com maior precisão. Nesse caso, informe no LEIAME do arquivo qual foi a função utilizada. Cada função deverá ter o seu tempo medido dez vezes e deverá ser informado o tempo médio, o tempo máximo e o tempo mínimo de suas execuções preenchendo uma tabela como aquela apresentada em Tabela 1.

Tabela 1: Desempenho das funções em termos do tempo de execução

Função	**Média**	**Mínimo**	**Máximo**
`ehPar`	Valor médio	Menor valor	Maior valor
`ehPrimo`	Valor médio	Menor valor	Maior valor
`ehQuadradoPerfeito`	Valor médio	Menor valor	Maior valor
`ehCuboPerfeito`	Valor médio	Menor valor	Maior valor
`ehFibonacci`	Valor médio	Menor valor	Maior valor
`ehFatorial`	Valor médio	Menor valor	Maior valor
`criaVetorAleatorio`	Valor médio	Menor valor	Maior vaor

Além das tabelas, o seu relatório precisa ter um cabeçalho com seu nome e NUSP e uma conclusão explicando se os resultados obtidos em termos de tempo de execução fazem sentido comparando os tempos das funções entre eles.

2.7 LEIAME

Junto com os programas e com o relatório você terá que entregar também um arquivo LEIAME em texto puro (Ele pode ser nomeado apenas como LEIAME ou pode ser LEIAME.txt). Crie o seu arquivo com no mínimo estas cinco seções dentro dele:

```
AUTOR:
<Seu nome, NUSP e endereço de e-mail>

DESCRIÇÃO:
<Explique o que os programas fazem (Não diga apenas que ele é o EP3 da
disciplina tal, explique o que de fato os programas fazem)>
```

[2](Leia a manpage da função para entender como chamá-la)

```
COMO EXECUTAR:
<Informe como compilar as bibliotecas e como usá-las. Você pode informar que
basta executar o bash script mas nesse caso o bash script precisa estar com
bons comentários.>

TESTES:
<Forneça explicações sobre o testa.c>

DEPENDÊNCIAS:
<Informe o que é necessário para rodar o programa. Informações como o
SO onde você executou e sabe que ele funciona são importantes de serem
colocadas aqui.>
```

3 Entrega

Você deverá entregar um arquivo tarball comprimido (.tar.gz) contendo os seguintes itens:

- Interface e implementação da biblioteca estática;
- Interface e implementação da biblioteca dinâmica;
- Implementação do `testa.c`;
- Implementação do `compila.sh`;
- 1 arquivo LEIAME (pode ser LEIAME.txt) em texto puro;
- 1 arquivo .pdf com o relatório.

Todos esses arquivos devem estar presentes. Caso algum arquivo esteja faltando, o EP não será corrigido e a nota dele será zero.

O desempacotamento do tarball deve produzir um diretório contendo os itens. O nome do diretório deve ser ep3-seu_nome. Por exemplo: ep3-joao_dos_santos.

A entrega do tarball deve ser feita no e-Disciplinas.

O EP deve ser feito individualmente.

Obs.1: Serão descontados 2,0 pontos de EPs com tarballs que não estejam nomeados como solicitado ou que não criem o diretório com o nome correto após serem descompactados. Confirme que o seu tarball está correto, descompactando ele no shell (não confie em interfaces gráficas na hora de verificar o seu tarball pois alguns gerenciadores de arquivos criam o diretório automaticamente mesmo quando esse diretório não existe).

Obs.2: A depender da qualidade do conteúdo que for entregue, o EP pode ser considerado como não entregue, implicando em MF=0,0. Isso acontecerá também se for enviado um tarball corrompido, códigos fonte vazios ou códigos fonte que resolvam um problema completamente diferente do que foi solicitado.

Obs.3: O prazo de entrega expira às 8:00:00 do dia 14/11/2022.

4 Avaliação

80% da nota será dada pela implementação, 10% pelo LEIAME e 10% pelo relatório. Os critérios detalhados da correção serão disponibilizados apenas quando as notas forem liberadas.