Aula Prática 2 - Roteiro

03/08/2021 - Roteiro referente à aula prática 2 - Série de Fibonacci

Versões:

- **03/08/2021 - Versão inicial**

Prazo: 05/08/2021 - 18:00

Valor: 10,0 - Peso: 1

Observações:

- Leia este enunciado com **MUITA** atenção até o final antes de iniciar o trabalho.
- Os arquivos solicitados deverão estar disponíveis nos diretórios correspondentes (**Aulas-Praticas** e ***RCS***) até o prazo estipulado acima. Cuidado com os nomes dos diretórios e dos arquivos. Deverão ser exatamente os definidos neste roteiro (maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- **As tarefas deverão ser executadas na ordem solicitada neste roteiro.**
- Os arquivos de dependências deverão possibilitar que a compilação e a *linkedição* sejam executadas utilizando-se tanto o *gcc*, quanto o *clang*. A seleção da ferramenta utilizada deverá ser realizada no momento da execução do comando *make*. O gcc deverá ser considerado o valor padrão para a compilação e para a *linkedição*.

  Para a definição da ferramenta desejada deverá ser utilizada uma macro (no *FreeBSD*) ou um argumento com o valor desejado (no CentOS). As duas macros utilizadas deverão ser *GCC* e *CLANG* (definidas usando a opção -D). O argumento, identificado por *cc,* deverá ser igual a *GCC* ou *CLANG*.
- Independente da ferramenta utilizada para a compilação, o *flag* de compilação deverá ser definido no instante da execução do comando *make*. O valor padrão para este *flag* deverá ser "*-Wall -ansi*" (sem as aspas).

  Durante a execução do comando *make* poderão ser definidos outros valores para este *flag* (mantendo a opção de exibir todas as mensagens de advertência) através de macros ou através de argumentos (de forma semelhante àquela utilizada para definir o compilador/*linkeditor*). No FreeBSD deverão ser definidas as macros ANSI, C89, C90, C99 e C11, enquanto que no CentOS deverá ser definido o argumento *dialeto* com os valores ANSI, C89, C90, C99 ou C11.
- Crie uma macro, DIALETO, contendo o dialeto a ser utilizado na compilação do código. Esta macro será inicialmente igual a "ansi" e poderá ser alterada para "c89", "c99" ou "c11" de acordo com o esquema definido acima.
- O *flag* de *linkedição* deverá ser igual a "-Wall" (sem as aspas).
- Cuidado com os nomes das macros e dos rótulos. Deverão ser exatamente os definidos neste roteiro (maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- Todos os rótulos solicitados no roteiro são obrigatórios. Durante a correção, caso não seja possível alcançar os objetivos (binários e/ou bibliotecas e limpezas de código) solicitados, a nota correspondente ao item/aula questão será zero.
- Seguem alguns exemplos:

  make - compila/linkedita (tanto no FreeBSD, quanto no CentOS) com a ferramenta e dialeto padrões, ou seja, gcc e ANSI respectivamente.
  make -DGCC - compila/linkedita usando o *gcc* e o dialeto ANSI (somente FreeBSD).
  make -DCLANG - compila/linkedita usando o *clang* e o dialeto ANSI (somente FreeBSD).
  make cc=GCC - compila/linkedita usando o *gcc* e o dialeto ANSI (somente CentOS).
  make cc=CLANG - compila/linkedita usando o *clang* e o dialeto ANSI (somente CentOS).

make -DCLANG -DC89 - compila/linkedita usando o *clang* e o dialeto C89 (somente FreeBSD).
make -DCLANG -DC11 - compila/linkedita usando o *clang* e o dialeto C11 (somente FreeBSD).
make cc=CLANG dialeto=C99 - compila/linkedita usando o *clang* e o dialeto C99 (somente CentOS).

- Inclua, no início de todos os arquivos solicitados, os seguintes comentários:

```
Universidade Federal do Rio de Janeiro
Escola Politecnica
Departamento de Eletronica e de Computacao
EEL270 – Computacao II – Turma 2021/1
Prof. Marcelo Luiz Drumond Lanza
Autor: <nome completo>
Descricao: <descrição sucinta dos objetivos do programa>

$Author$
$Date$
$Log$
```

- Inclua, no final de todos os arquivos solicitados, os seguintes comentários:

```
$RCSfile$
```

A série de Fibonacci é dada por:

Número do Termo	0	1	2	3	4	5	6	7	8	9	. . .
Valor do Termo	0	1	1	2	3	5	8	13	21	34	. . .

$$F(n) = n \text{ se } n \leq 1$$
$$F(n) = F(n-1) + F(n-2) \text{ se } n > 1$$

1. Inclua, nos dois arquivos de dependências, as declarações necessárias para que o dialeto ANSI seja usado na compilação quando a ANSI ou a macro C90 for definida no FreeBSD ou quando o valor do argumento dialeto for igual a ANSI ou C90 no Linux.
2. Crie o arquivo "*aula0201.h*" contendo a definição dos tipos *us* (correspondendo a *unsigned short*) e *ull* (correspondendo a *unsigned long long*) e do protótipo da função *CalcularTermoSerieFibonacci*. Esta função deverá receber um inteiro não negativo (número do termo desejado) e deverá retornar o valor deste termo. A macro referente à combinação *ifndef* e *define*, como por exemplo _AULA0201_, deverá ser definida como uma *string* valendo:

   "@(#)aula0201.h $Revision$"

   ull
   CalcularTermoSerieFibonacci (us);

3. Crie o arquivo "*aula0201a.c*" contendo o código fonte da função *CalcularTermoSerieFibonacci* implementada utilizando-se recursividade. A implementação desta função não poderá utilizar nenhuma função de nenhuma biblioteca.
4. Crie o arquivo "*aula0202.c*" contendo o código fonte de um programa de testes para a função criada na questão anterior. Este programa deverá receber, através de um argumento da CLI e utilizando a função *strtoul*, um inteiro não negativo representando ***n*** (o limite superior para a exibição dos valores da série de *Fibonacci*). O programa deverá exibir os valores dos termos da série de *Fibonacci* deste o elemento ***o*** até o elemento ***n***. Se, por exemplo, n for igual a 5 a saída deverá ser:

   F (0) = 0
   F (1) = 1
   F (2) = 1
   F (3) = 2
   F (4) = 3

F (5) = 5

Se o valor de um termo da série de Fibinacci (termo ***k***, onde ***k*** é menor ou igual a ***n***) for superior ao valor máximo que pode ser armazenado em uma variável do tipo ull, o programa de testes deverá exibir o valor dos termos da série, variando do elemento ***0*** até o elemento ***k-1***, seguidos da mensagem de erro correspondente.

5. Inclua, nos arquivos de dependências, as macros *AULA02* - correspondendo ao executável *aula0202a* (resultado da combinação entre a função implementada utilizando-se recursividade e o programa de testes) e *AULA0202AOBJS* - correspondendo aos arquivos objetos necessários para gerar o executável *aula0202a*. Altere o valor da macro *EXECS*, de forma que inclua o valor da macro *AULA02*. Inclua também os objetivos *aula02* e *aula0202a* com as declarações necessárias.
6. Crie e teste as 16 versões do executável *aula0202a*.
7. Submeta os arquivos *aula0201.h*, *aula0201a.c*, *aula0202.c* e **makefile* ao sistema de controle de versão.
8. Recupere uma cópia de leitura dos arquivos *aula0201.h*, *aula0201a.c* e *aula0202.c* e uma cópia de escrita dos arquivos **makefile*.
9. Crie o arquivo "*aula0201b.c*" contendo o código fonte da função *CalcularTermoSerieFibonacci* implementada utilizando-se a estrutura de controle "*do ... while*". A implementação desta função não poderá utilizar nenhuma função de nenhuma biblioteca.
10. Altere, nos arquivos de dependências, a macro *AULA02* - incluindo o executável *aula0202b* (resultado da combinação entre a função implementada utilizando-se a estrutura de controle *do ... while* e o programa de testes). Inclua a macro *AULA0202BOBJS* - correspondendo aos arquivos necessários para gerar o executável *aula0202b*. Inclua também o objetivo *aula0202b* com as declarações necessárias.
11. Crie e teste as 16 versões do executável *aula0202b*.
12. Submeta os arquivos *aula0201b.c* e **makefile* ao sistema de controle de versão.
13. Recupere uma cópia de leitura do arquivo *aula0201b.c* e uma cópia de escrita dos arquivos **makefile*.
14. Crie o arquivo "*aula0201c.c*" contendo o código fonte da função *CalcularTermoSerieFibonacci* implementada utilizando-se a estrutura de controle "*for*". A implementação desta função não poderá utilizar nenhuma função de nenhuma biblioteca.
15. Altere, no arquivo de dependências, a macro *AULA02* - incluindo o executável *aula0202c* (resultado da combinação entre a função implementada utilizando-se a estrutura de controle *for* e o programa de testes). Inclua a macro *AULA0202COBJS* - correspondendo aos arquivos necessários para gerar o executável *aula0202c*. Inclua também o objetivo *aula0202c* com as declarações necessárias.
16. Crie e teste as 16 versões do executável *aula0202c*.
17. Submeta os arquivos *aula0201c.c* e **makefile* ao sistema de controle de versão.
18. Recupere uma cópia de leitura do arquivo *aula0201c.c* e uma cópia de escrita dos arquivos **makefile*.
19. Crie o arquivo "*aula0201d.c*" contendo o código fonte da função *CalcularTermoSerieFibonacci* implementada utilizando-se a estrutura de controle "*while*". A implementação desta função não poderá utilizar nenhuma função de nenhuma biblioteca.
20. Altere, no arquivo de dependências, a macro *AULA02* - incluindo o executável *aula0202d* (resultado da combinação entre a função implementada utilizando-se a estrutura de controle *while* e o programa de testes). Inclua a macro *AULA0202DOBJS* - correspondendo aos arquivos necessários para gerar o executável *aula0202d*. Inclua também o objetivo *aula0202d* com as declarações necessárias.
21. Crie e teste as 16 versões do executável *aula0202d*.
22. Submeta os arquivos *aula0201d.c* e **makefile* ao sistema de controle de versão.
23. Recupere uma cópia de leitura do arquivo *aula0201d.c* e uma cópia de escrita dos arquivos **makefile*.
24. Inclua, nos arquivos de dependências, as macros *LIBMATEMATICARECURSAOOBJS*, *LIBMATEMATICADOWHILEOBJS*, *LIBMATEMATICAFOROBJS* e *LIBMATEMATICAWHILEOBJS* correspondendo respectivamente aos arquivos "*aula0201a.o*", "*aula0201b.o*", "*aula0201c.o*" e "*aula0201d.o*". Inclua também as macros *LIBMATEMATICARECURSAO*, *LIBMATEMATICADOWHILE*, *LIBMATEMATICAFOR* e *LIBMATEMATICAWHILE* correspondendo respectivamente aos arquivos "*libmatematicarecursao.a*", "*libmatematicadowhile.a*", "*libmatematicafor.a*" e "*libmatematicawhile.a*". O valor da macro LIBS deverá ser atualizado de forma que inclua o

valor destas últimas quatro macros. Para cada uma destas quatro bibliotecas (estáticas), inclua o objetivo correspondente (por exemplo, libmatematicarecursao.a), com a(s) dependência(s) e comando(s) necessários para atingir o objetivo em questão.

Observação:
O comando ar deverá ser utilizado para criar bibliotecas estáticas, como mostrado abaixo.

ar -r -c libcomputacao.a arquivo1.o arquivo2.o arquivo3.o

Cria o arquivo libcomputacao.a (biblioteca estática "computacao") a partir do código dos 3 arquivos contendo código objeto.

Os arquivos contendo código objeto não podem conter nenhuma instância da função ***main***.

25. Verifique se as quatro bibliotecas são geradas corretamente em todos os contextos de compilação/linkedição (sistemas operacionais x compiladores x dialetos).
26. Submeta os arquivos *makefile ao sistema de controle de versão.
27. Recupere uma cópia de escrita dos arquivos *makefile.
28. Limpe o diretório (*make clean-all*).