Aula Prática 2 – Roteiro

Objetivos:

- Série de Fibonacci

Versão Inicial: 02/05/2023

Prazo: 05/05/2023 – 8:00

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 (incluindo 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 que 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 como o valor padrão para a ferramenta de compilação e de *linkedição*.

  Para a definição da ferramenta desejada, deverá ser utilizada uma macro no *FreeBSD* e um argumento com o valor desejado no *Linux*. As duas macros utilizadas deverão ser ***GCC*** e ***CLANG*** (definidas usando a opção de linha de comando **-D** do comando ***make***). O argumento, identificado por ***cc***, deverá ser igual a ***GCC*** ou a ***CLANG***.
- Independente da ferramenta utilizada para a compilação, as opções de compilação poderão ser redefinidas no instante da execução do comando ***make*** (mantendo-se a exibição de todas as mensagens de advertência, definida pelo valor ***-Wall***). O valor padrão para estas opções deverá ser ***-Wall -ansi***.

  Estas opções poderão ser redefinidas 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 *Linux* deverá ser definido o argumento ***dialeto*** com um dos seguintes valores ***ANSI***, ***C89***, ***C90***, ***C99*** ou ***C11***.
- Os arquivos de dependências deverão incluir a macro ***DIALECT*** contendo o dialeto a ser utilizado na compilação do código. Esta macro será inicialmente igual a ***ansi*** e poderá ser alterada para ***c89***, ***c90***, ***c99*** ou ***c11*** de acordo com o esquema definido acima.
- Os arquivos de dependências deverão incluir também a macro ***STANDARD*** contendo a opção de linha de comando correspondente ao dialeto selecionado. Se, por exemplo, o dialeto selecionado for o ***ANSI***, esta macro deverá ser igual a ***-ansi***. Por outro lado, se o dialeto for uma das outras quatro opções, esta macro deverá ser igual a ***-std=CXX***, onde ***XX*** deverá ser substituído pelo número correspondente (se o dialeto for igual a ***C89***, *XX* deverá ser igual a ***89***, se o dialeto for igual a ***C90***, ***XX*** deverá igual a ***90*** e assim por diante).
- A *linkedição* deverá utilizar a opção ***-Wall***.
- 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 em questão será igual a zero.
- Seguem alguns exemplos (todos devem funcionar):
  - ***make*** - compila/*linkedita* (tanto no *FreeBSD*, quanto no *Linux*) com a ferramenta e dialeto padrões, ou seja, ***gcc*** e **ANSI** respectivamente.
  - ***make clean-all all***
  - ***make clean-all aula01***
  - ***make clean aula0101***
  - ***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 *Linux*).
  - ***make cc=CLANG*** - compila/*linkedita* usando o ***clang*** e o dialeto **ANSI** (somente *Linux*).
  - ***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 dialero=C99*** - compila/*linkedita* usando o ***clang*** e o dialeto **C99** (somente *Linux*).
  - ***make cc=GCC dialeto=C90*** - compila/*linkedita* usando o ***gcc*** e o dialeto **ANSI** (somente *Linux*).

- Inclua, no início de todos os arquivos solicitados (código-fonte e arquivos de dependências), os seguintes comentários (**sem caracteres especiais**):

```
Universidade Federal do Rio de Janeiro
Escola Politecnica
Departamento de Eletronica e de Computacao
EEL270 - Computacao II - Turma 2023/1
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, o seguinte comentário:

```
$RCSfile$
```

A série de Fibonacci é dada por:

Número	0	1	2	3	4	5	6	7	...
Valor	0	1	1	2	3	5	8	13	...

F(n) = n se n ≤ 1

F(n) = F(n-1) + F(n-2) se n > 1

1. Crie o arquivo ***aula0201.h*** contendo a definição 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*** deverá ser igual a ***AULA0201*** e deverá ser definida como uma *string* igual a "@(#)aula0201.h $Revision$".

   ```
   unsigned long long
   CalcularTermoSerieFibonacci (unsigned short);
   ```

2. Crie o arquivo ***aula0201a.c*** contendo o código-fonte da função definida no item anterior. Esta função deverá ser implementada utilizando-se recursividade e não poderá utilizar nenhuma função de nenhuma biblioteca disponível no *FreeBSD* e no *Linux*.

3. 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 o limite superior para a exibição dos valores da série de *Fibonacci*. O programa deverá exibir os valores dos termos da série de *Fibonacci* desde o elemento 0 até o elemento limite definido (inclusive). Se, por exemplo, o limite for igual a 5 a saída deverá ser exatamente igual a:

   F (0) = 0
   F (1) = 1
   F (2) = 1
   F (3) = 2
   F (4) = 3
   F (5) = 5

   Formato da saída: letra F maiúscula, espaço, abre parênteses, número do termo, fecha parênteses, espaço, igual, espaço, valor do termo.

   Esta saída deverá ser exibida utilizando-se fundo branco e caracteres pretos. Utilize o arquivo ***aula01.h*** com as definições das cores.

   Se o valor de um termo da série de Fibonacci (termo ***k***, onde ***k*** é menor ou igual ao limite) for superior ao valor máximo que pode ser armazenado em uma variável do tipo ***unsigned long long***, 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: F (k) ultrapassa o limite superior permitido para o tipo ***unsigned long long***.

Esta mensagem deverá ser exibida utilizando-se fundo branco e caracteres vermelhos. Todas as mensagens de erro deverão utilizar este padrão.

4. 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 o objetivo ***aula02*** (com as declarações necessárias para gerar todos os binários definidos pela macro ***AULA02*** – por enquanto ***aula0202a***) e o objetivo ***aula0202a*** (com as declarações necessárias para gerar o binário de mesmo nome).
5. Crie e teste as 20 versões do binário ***aula0202a***.
6. Submeta os arquivos ***aula0201.h***, ***aula0201a.c***, ***aula0202.c***, ***BSDmakefile*** e ***GNUmakefile*** ao sistema de controle de versão.
7. Recupere uma cópia de leitura dos arquivos contendo o código-fonte e uma cópia de escrita dos arquivos de dependências.
8. Crie o arquivo ***aula0201b.c*** contendo o código-fonte da função definida no item 1. Esta função deverá ser implementada utilizando-se a estrutura de controle *do...while* e não poderá utilizar nenhuma função de nenhuma biblioteca disponível no *FreeBSD* e no *Linux*.
9. Altere, nos arquivos de dependências, a macro ***AULA02*** - incluindo o binário ***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 binário ***aula0202b***. Inclua também o objetivo ***aula0202b*** com as declarações necessárias.
10. Crie e teste as 20 versões do binário ***aula0202b***.
11. Submeta os arquivos ***aula0201b.c***, ***BSDmakefile*** e ***GNUmakefile*** ao sistema de controle de versão.
12. Recupere uma cópia de leitura do arquivo contendo o código-fonte e uma cópia de escrita dos arquivos de dependências.
13. Crie o arquivo ***aula0201c.c*** contendo o código-fonte da função definida no item 1. Esta função deverá ser implementada utilizando-se a estrutura de controle for e não poderá utilizar nenhuma função de nenhuma biblioteca disponível no *FreeBSD* e no *Linux*.
14. Altere, no arquivo de dependências, a macro ***AULA02*** - incluindo o binário ***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 binário ***aula0202c***. Inclua também o objetivo ***aula0202c*** com as declarações necessárias.
15. Crie e teste as 20 versões do binário ***aula0202c***.

16. Submeta os arquivos "**aula0201c.c**", ***BSDmakefile*** e ***GNUmakefile*** ao sistema de controle de versão.
17. Recupere uma cópia de leitura do arquivo contendo o código-fonte e uma cópia de escrita dos arquivos de dependências.
18. Crie o arquivo ***aula0201d.c*** contendo o código-fonte da função definida no item 1. Esta função deverá ser implementada utilizando-se a estrutura de controle *while* e não poderá utilizar nenhuma função de nenhuma biblioteca disponível no *FreeBSD* e no *Linux*.
19. Altere, no arquivo de dependências, a macro ***AULA02*** - incluindo o binário ***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 binário ***aula0202d***. Inclua também o objetivo ***aula0202d*** com as declarações necessárias.
20. Crie e teste as 20 versões do binário ***aula0202d***.
21. Submeta os arquivos ***aula0201d.c***, ***BSDmakefile*** e ***GNUmakefile*** ao sistema de controle de versão.
22. Recupere uma cópia de leitura do arquivo contendo o código-fonte e uma cópia de escrita dos arquivos de dependências.
23. Inclua, nos arquivos de dependências, as macros:
    - ***LIBMATEMATICARECURSAOOBJS***,
    - ***LIBMATEMATICADOWHILEOBJS***,
    - ***LIBMATEMATICAFOROBJS*** e
    - ***LIBMATEMATICAWHILEOBJS***

    Estas macros devem corresponder respectivamente aos arquivos ***aula0201a.o***, ***aula0201b.o***, ***aula0201c.o*** e ***aula0201d.o***.
24. Inclua também as macros:
    - ***LIBMATEMATICARECURSAO***,
    - ***LIBMATEMATICADOWHILE***,
    - ***LIBMATEMATICAFOR*** e
    - ***LIBMATEMATICAWHILE***

    Estas macros devem corresponder 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 no exemplo a seguir:

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

Este comando cria o arquivo ***libcomputacao.a*** (biblioteca estática ***computacao***) a partir do código dos três arquivos contendo código objeto. Os arquivos contendo o código objeto não poderão conter nenhuma instância da função *main*. Verifique se as quatro bibliotecas são geradas corretamente em todos os contextos de compilação/*linkedição* (sistemas operacionais x compiladores x dialetos).

25. Submeta os arquivos de dependências ao sistema de controle de versão.
26. Recupere uma cópia de escrita dos arquivos de dependências.
27. Limpe o diretório (*make clean-all*).
28. Arquivos que devem ser disponíveis ao final da aula:

    Subdiretório "~/private/EEL270/2022-2/Aulas-Praticas"
    - *aula0201.h*
    - *aula0201a.c*
    - *aula0201b.c*
    - *aula0201c.c*
    - *aula0201d.c*
    - *aula0202.c*
    - *BSDmakefile*
    - *GNUmakefile*

Além dos correspondentes gerados pela ferramenta de controle de versão (localizados no subdiretório ***RCS***) e dos arquivos gerados na aula prática 1.