Aula Prática 6 – Roteiro

Objetivos:

- Geração e validação de números de carteira de identidade.

Versão Inicial: 13/06/2023

Prazo: 19/06/2023 – 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:

```
$RCSfile$
```

O número de um RG (carteira de identidade) gerado pelo *Instituto Felix Pacheco (RJ)* tem a seguinte configuração: XX.XXX.XXX-X. O último dígito (após o hífen) é o dígito verificador do RG.

RG = X1 X2. X3 X4 X5. X6 X7 X8- X9

O dígito verificador é gerado a partir dos dígitos anteriores. Para cada dígito verificador, deverá ser calculada a soma dos produtos dos dígitos anteriores pelos respectivos pesos. Este resultado (soma) deverá ser dividido por 11. O dígito verificador será igual à diferença entre 11 e o resto da divisão anterior. Se a diferença for igual a 11 (11 menos resto 0), o dígito verificador será igual a 0. Se a diferença for igual a 10 (11 menor resto 1), o dígito verificador será igual a X.

Os pesos dos dígitos para o cálculo do dígito verificador são (da esquerda para a direita) 2, 3, 4, 5, 6, 7, 8 e 9.

1. Crie o arquivo ***aula0601.h*** contendo o protótipo da função *GerarDigitosVerificadoresRg*. Este arquivo deverá conter também as macros e os tipos necessários para a implementação desta função. A macro referente à combinação ***ifndef*** e ***define***, como por exemplo ***_AULA0601_***, deverá ser definida como uma *string* valendo: "***@(#)aula0601.h $Revision$***".

   O protótipo da função é definido a seguir:

   ```
   tipoErros
   GerarDigitosVerificadoresRg (byte [ ] /* entrada/saída */);
   ```

   A função *GerarDígitosVerificadoresRg* deverá receber, no primeiro argumento, os 8 primeiros dígitos de um RG e deverá devolver o dígito verificador correspondente (na nona posição do vetor recebido).

2. Crie o arquivo ***aula0601.c*** contendo a implementação da função solicitada no item anterior.

3. Crie o arquivo ***aula0602a.c*** contendo a implementação de um programa de testes para a função *GerarDigitosVerificadoresRg*. Este programa deverá receber os 8 primeiros dígitos do RG desejado através de 8 argumentos da linha de comando (CLI). O programa deverá exibir o RG completo no formato "XX.XXX.XXX-X" (sem as aspas e precedido por RG: - após os caractere dois pontos deverá ser incluído um caractere de espaço). Todos os tratamentos de erros necessários e que não puderem ser realizados na função deverão ser implementados neste programa.

   Exemplo: ./aula0602a 5 6 8 4 3 5 3 9

   RG: 56.843.539-4

4. Inclua, nos arquivos de dependências, as macros ***AULA0602AOBJS*** e ***AULA06***. Altere o valor da macro **EXECS**, de forma que inclua o valor da macro **AULA06**. Inclua também os rótulos ***aula06*** e ***aula0602a*** com os comandos correspondentes.

5. Gere e teste as 20 versões do executável ***aula0602a***.

6. Submeta os arquivos ***aula0601.h***, ***aula0601.c***, ***aula0602a.c*** e ****makefile*** ao sistema de controle de versão.

7. Recupere uma cópia de leitura do arquivo ***aula0602a.c*** e uma cópia de escrita dos arquivos ***aula0601.h***, ***aula0601.c*** e "****makefile***".

8. Crie o arquivo aula***0602b.c*** contendo a implementação de um programa de testes para a função *GerarDigitosVerificadoresRg*. Este programa deverá receber os 8 primeiros dígitos do RG desejado através de um único argumento da interface de linha de comando (CLI) no formato "XX.XXX.XXX" (sem as aspas). O programa deverá exibir o RG completo no formato "XX.XXX.XXX-X" (sem as aspas e precedido por RG: - após o caractere dois pontos deverá ser incluído um caractere de espaço). Todos os tratamentos de erros necessários e que não puderem ser realizados na função *GerarDigitosVerificadoresRg* deverão ser implementados neste programa.

Exemplo:

./aula0602b 56.843.539

RG: 56.843.539-4 9.

9. Inclua, nos arquivos de dependências, a macro ***AULA0602BOBJS*** e o rótulo ***aula0602b*** com os comandos correspondentes. Altere o valor da macro ***AULA06***, incluindo o binário correspondente.
10. Gere e teste as 20 versões do executável ***aula0602b***.
11. Submeta os arquivos ***aula0602b.c*** e ****makefile*** ao sistema de controle de versão.
12. Recupere uma cópia de leitura do arquivo aula***0602b.c*** e uma cópia de escrita dos arquivos "****makefile***".
13. Inclua, no arquivo ***aula0601.h***, o protótipo da função ***ValidarRg*** e a definição dos tipos necessários.

```
tipoErros
ValidarRg (byte [ ] /* entrada */);
```

A função *ValidarRg* deverá receber os 9 dígitos de um RG (como um vetor de bytes) e deverá retornar ***ok*** se os dígitos verificadores forem válidos ou o código de erro correspondente. Esta função deverá utilizar a função *GerarDigitosVerificadoresRg* na sua implementação.

14. Inclua, no arquivo ***aula0601.c***, a implementação da função *ValidarRg*.
15. Crie o arquivo ***aula0603a.c*** contendo a implementação de um programa de testes para a função ***ValidarRg***. Este programa deverá receber os 9 dígitos do RG desejado através de 9 argumentos da interface de linha de comando (CLI) e deverá exibir o RG em questão no formato "XX.XXX.XXX-X" (sem as aspas), indicando se o mesmo é válido ou inválido. Todos os tratamentos de erros necessários e que não possam ser realizados na função *ValidarRg* deverão ser implementados neste programa.

Exemplos:

./aula0603a 5 6 8 4 3 5 3 9 4

RG: 56.843.539-4 – valido.

./aula0603a 5 6 8 4 3 5 3 9 7

RG: 56.843.539-7 – invalido.

16. Inclua, nos arquivos de dependências, a macro ***AULA0603AOBJS*** e o rótulo ***aula0603a*** com os comandos correspondentes. Altere o valor da macro ***AULA06***, incluindo o binário correspondente.
17. Gere e teste as 20 versões do executável ***aula0603a***.
18. Submeta os arquivos ***aula0601.h***, ***aula0601.c***, ***aula0603a.c*** e "****makefile***" ao sistema de controle de versão.

19. Recupere uma cópia de leitura do arquivo ***aula0603a.c*** e uma cópia de escrita dos arquivos ***aula0601.h***, ***aula0601.c*** e "****makefile***".

20. Crie o arquivo aula***0603b.c*** contendo a implementação de um programa de testes para a função *ValidarRg*. Este programa deverá receber os 9 dígitos do RG desejado através de um único argumento da linha de comando (CLI) no formato "XX.XXX.XXX-X" (sem as aspas) e deverá exibir o RG em questão no formato "XX.XXX.XXX-X" (sem as aspas), indicando se o mesmo é válido ou inválido. Todos os tratamentos de erros necessários e que não puderem ser realizados na função *ValidarRg* deverão ser implementados neste programa. Esta função deverá utilizar a função *GerarDigitosVerificadoresRg* na sua implementação.

    Exemplos:

    ./aula0603a 56.843.539-4

    RG: 56.843.539-4 – valido.

    ./aula0603a 56.843.539-7

    RG: 56.843.539-7 – invalido.

21. Inclua, nos arquivos de dependências, a macro ***AULA0603BOBJS*** e o rótulo ***aula0603b*** com os comandos correspondentes. Altere o valor da macro ***AULA06***, incluindo o binário correspondente.

22. Gere e teste as 20 versões do executável ***aula0603b***.

23. Submeta os arquivos ***aula0603b.c*** e "****makefile***" ao sistema de controle de versão.

24. Recupere uma cópia de leitura do arquivo ***aula0603b.c*** e uma cópia de escrita dos arquivos "****makefile***".

25. Repita todos os ítens anteriores trocando o vetor de bytes por uma *string* de acordo com os protótipos abaixo:

    ```
    tipoErros
    GerarDigitosVerificadoresRg (char * /* entrada */, char * /* saida */);
    ```

    Esta função deverá receber, através do primeiro argumento, uma *string* contendo os 8 primeiros dígitos de um possível RG (incluindo pontos) e deverá devolver, através do último argumento, o dígito verificador correspondente.

    ```
    tipoErros
    ValidarRg (char *);
    ```

    A função *ValidarRg* deverá receber uma *string* no formato "XX.XXX.XXX-X" (sem as aspas) e deverá retornar ***ok*** quando esta *string* representar um RG válido. Por outro lado, deverá retornar o código de erro correspondente.

    Não se esqueça de renumerar corretamente os arquivos, ou seja:

    aula0601.h será renomeado para ***aula0604.h***.

    aula0601.c será renomeado para ***aula0604.c***.

aula0602a.c será renomeado para ***aula0605a.c***.

aula0602b.c será renomeado para ***aula0605b.c***.

aula0603a.c será renomeado para ***aula0606a.c***.

aula0603b.c será renomeado para ***aula0606b.c***.

Não se esqueça de incluir as macros e rótulos necessários nos arquivos de dependências e de submeter os arquivos ao sistema RCS.

26. Limpe o diretório (make clean-all)
27. Arquivos que devem ser disponíveis ao final da aula:

    Subdiretório "***~/private/EEL270/2023-1/Aulas-Praticas***"
    - ***aula0601.h***
    - ***aula0601.c***
    - ***aula0602a.c***
    - ***aula0602b.c***
    - ***aula0603a.c***
    - ***aula0603b.c***
    - ***aula0604.h***
    - ***aula0604.c***
    - ***aula0605a.c***
    - ***aula0605b.c***
    - ***aula0606a.c***
    - ***aula0606b.c***
    - ***BSDmakefile***
    - ***GNUmakefile***

Além dos arquivos gerados pela ferramenta de controle de versão (localizados no subdiretório ***RCS***) e dos arquivos gerados nas aulas práticas anteriores.