Universidade de Brasília
Departamento de Ciência da Computação
Disciplina: Métodos de Programação
Código da Disciplina: 201600

Métodos de Programação - 201600

Trabalho 2

O objetivo deste trabalho é utilizar o desenvolvimento orientado a testes (TDD) para resolver o problema de conversão de algarismos romanos para algarismos arábicos. O número em algarismos romanos poderá ser no máximo 3000. A função deverá pegar uma string de tamanho até 30 caracteres e retornar um inteiro correspondente ao número romano. Deverá retornar -1 se o número romano for inválido. Todas as strings correspondentes a números romanos válidos devem retornar o valor correto (ex. XXX deve retornar 30). Todas as strings correspondentes a números romanos inválidos devem retornar -1 (ex. XXXX, VV, VX, etc. devem retornar -1)

O desenvolvimento deverá ser feito passo a passo seguindo a metodologia TDD. A cada passo deve-se pensar qual é o objetivo do teste e o significado de passar ou não no teste.
Os resultados podem ser comparados com os números em:
http://numeracaoromana.babuo.com/numeros-romanos-de-1-a-3000
Ou outra fonte.

1) O programa deverá ser dividido em módulos e desenvolvido em C ou C++. Deverá haver um arquivo romano.c (ou .cpp) e um arquivo romano.h (ou .hpp). Deverá haver também um arquivo testa_romano.c (ou .cpp) cujo objetivo é testar o funcionamento da biblioteca de números romanos. Sendo que usar arquivos .cpp e .hpp pode facilitar o uso dos frameworks de teste.

Faça um Makefile utilizando o exemplo de makefile 5 dado em:

(http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/)
Não se esqueça de criar os diretórios correspondentes.

O programa e o módulo devem ser depurados utilizando o GDB.
(http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Debug.html)
(https://www.cs.umd.edu/~srhuang/teaching/cmsc212/gdb-tutorial-handout.pdf)

2) Utilize o padrão de codificação dado em: https://google.github.io/styleguide/cppguide.html quando ele não entrar em conflito com esta especificação. O código dever ser claro e bem comentado. O código deve ser verificado se está de acordo com o estilo usando o cpplint (https://github.com/cpplint/cpplint).

Utilize o cpplint desde o início da codificação pois é mais fácil adaptar o código no início.

3) Faça um documento dizendo quais testes você fez a cada passo e o que passar neste teste significa.

4) O desenvolvimento deverá ser feito utilizando um destes frameworks de teste:

4.1) gtest (https://code.google.com/p/googletest/)
4.2) catch (https://github.com/philsquared/Catch/blob/master/docs/tutorial.md)

5) Deverá ser entregue o histórico do desenvolvimento orientado a testes feito através do git (mandando o diretório local) ou do github (https://github.com/)
No caso de fazer local usando o git, mandar o diretório ".git" junto. Este diretório normalmente contém o histórico das versões. É importante verificar se foi enviado junto.

6) Faça a análise estática do programa utilizando o cppcheck, corrigindo os erros apontados pela ferramenta.

Utilize cppcheck --enable=warning .
para verificar os avisos nos arquivos no diretório corrente (.)

Utilize o cppcheck sempre e desde o início da codificação pois é mais fácil eliminar os problemas logo quando eles aparecem. Devem ser corrigidos apenas problemas no código feito e não em bibliotecas utilizadas (ex. gtest, catch)

7) Utilizar o Valgrind (*valgrind.org/*) para detectar problemas relacionados a memória.
Use valgrind --leak-check=full ./nome_executavel

Você pode inclusive adicionar as verificações dentro do makefile.

Devem ser enviados para a tarefa no aprender3.unb.br um arquivo zip onde estão compactados todos os diretórios e arquivos necessários. O documento deve estar na raiz do diretório. Todos os arquivos devem ser enviados compactados em um único arquivo (.zip) e deve ser no formato matricula_primeiro_nome ex: 06_12345_Jose.zip. Deve conter também um arquivo leiame.txt que diga como o programa deve ser compilado.

Lembrar de mandar o link para o github contendo o histórico do desenvolvimento ou o diretório ".git" contendo o histórico do desenvolvimento.

Data de entrega:

30/ 3 /21

Pela tarefa na página da disciplina no aprender3.unb.br