# Projeto de Programação I

## 1. Introdução

Código Morse é um sistema de representação de letras, algarismos e sinais de pontuação através de um sinal codificado enviado de modo intermitente. Foi desenvolvido por Samuel Morse em 1835, criador do telégrafo elétrico, dispositivo que utiliza correntes elétricas para controlar eletroímãs que atuam na emissão e na recepção de sinais.[1]

## 2. Objetivo

O objetivo deste trabalho criar um programa que faça a tradução de um texto em português para código morse e código morse para texto.

O programa terá duas funcionalidade principais:

1. Deve fazer com que o usuário escreva uma mensagem e seja gerado um arquivo o texto com a mensagem em formado morse.
2. Receber um arquivo em código morse, realizar a tradução e mostrar na tela a tradução para o usuário.

## 3. Tarefas

As tarefas a serem realizados neste exercício de programação consistem em três partes: (1) *implementação*, na qual você deverá projetar e implementar o tradutor morse; (2) *experimentação*, na qual você deverá realizar sucessivas execuções do algoritmo, e na qual você deverá realizar sucessivas execuções de tradução com diferentes textos para morse e morse para texto; (3) *relato*, na qual você deverá elaborar um relatório descrevendo as atividades que foram realizadas e realização de testes.

### 3.1. Implementação

O código deverá ser devidamente comentado e anotado para dar suporte à geração automática de documentação no formato de páginas Web (HTML) utilizando a

1 - https://pt.wikipedia.org/wiki/C%C3%B3digo_Morse

ferramenta Doxygen. Para maiores informações, você poderá acessar a página do Doxygen na Internet (http://www.doxygen.org/).

O código do projeto deve seguir a configuração de pastas e arquivos:

1. /bin – código executável
2. /src – código fonte
3. /docs – documentação
4. makefile
5. `README` – arquivo contendo informações sobre: configuração, compilação e execução

### 3.2. Experimentação

A fim de permitir uma avaliação consistente, você deverá executar seu experimento pelo menos seis vezes com informações diferentes: I - três textos em português, com diferentes tamanhos, mostrando em seguida seu resultado e; II - três em morse e mostrando em seguida seu resultado. Executar o código usando as ferramentas de *debug* e gerenciamento de memória mostrando que não ocorreu nenhum problema na execução.

### 3.3. Relato

Uma vez realizadas as tarefas de implementação e experimentação, você deverá elaborar um relatório técnico contendo, no mínimo, as seguintes seções:

1. **Introdução –** Explicar o propósito do relatório, bem como o link para o repositório do projeto.
2. **Detalhes de Implementação –** Descrever como foi feita a sua implementação em termos de arquivos, funções, etc. e como o programa funciona de uma maneira geral. Explicar no também os pontos da avaliação , sessão 7.
3. **Resultados –** Apresentar os testes realizados bem como os resultados obtidos.

## 4. Orientações gerais

Você deverá atentar para as seguintes observações gerais no desenvolvimento deste trabalho:

1. Apesar da completa compatibilidade entre as linguagens de programação C e C++, seu código fonte não deverá conter recursos da linguagem C nem ser resultante de mescla entre as duas linguagens, o que é uma má prática de programação. Dessa forma, deverão ser utilizados estritamente recursos da linguagem C++.

2. Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois elas podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.
3. Aplique boas práticas de programação. Codifique o programa de maneira legível e documente-o adequadamente na forma de comentários. Como anteriormente instruído, o código fonte deverá ser anotado para dar suporte à geração automática de documentação utilizando o Doxygen (http://www.stack.nl/~dimitri/doxygen/).
4. Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.
5. Para melhor organização do seu código, implemente diferentes funções e faça a separação da implementação do programa entre arquivos cabeçalho (`.h`) e corpo (`.cpp`).

## 5. Autoria e política de colaboração

O trabalho poderá ser feito individualmente ou em equipe composta por no máximo dois estudantes, sendo que, neste último caso, é importante, dentro do possível, dividir as tarefas igualmente entre os integrantes da equipe. A fim de estimular o desenvolvimento colaborativo, uma sugestão de divisão de tarefas é que cada membro da equipe esteja responsável pela implementação dos arquivos cabeçalho e corpo de das funcionalidade definidas pelo grupo, sendo os códigos fonte feitos individualmente em um repositório Git local e, posteriormente, unificados em um repositório Git remoto. Cada equipe terá um repositório individual no Gitlab do IMD-UFRN (http://projetos.imd.ufrn.br/), no qual deverão ser disponibilizados todos os arquivos referentes ao programa implementado, conforme definido na seção Implementação. O endereço do repositório deverá ser enviado na tarefa do sigaa.

A atividade de cada integrante da equipe deverá ser registrada através de *commits* sobre o repositório Git, que será examinado pelo professor durante a avaliação do trabalho. Além disso, a critério do professor, qualquer equipe pode ser convocada para uma entrevista cujo objetivo é confirmar a autoria do trabalho desenvolvido e determinar a contribuição real de cada integrante. Durante a entrevista, cada membro da equipe deverá ser capaz de explicar, com desenvoltura, qualquer parte do trabalho, mesmo que esta tenha sido desenvolvida por outro membro da equipe. Portanto, é possível que ocorra, após a entrevista e/ou exame das atividades registradas, redução da nota geral do trabalho ou ajustes nas notas individuais com vistas a refletir a verdadeira contribuição de cada membro da equipe.

O trabalho em cooperação entre estudantes da turma é estimulado, sendo aceitável a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de outras equipes, o que pode

caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outras equipes ou da Internet serão sumariamente rejeitados e receberão nota zero.

## 6. Entrega

Todos os códigos fonte referentes à implementação do trabalho deverão ser disponibilizados sem erros de compilação e devidamente testados e documentados através do repositório Git da equipe no Gitlab do IMD-UFRN (http://projetos.imd.ufrn.br/). Além disso, você deverá submeter, até o horário da aula do dia 13 de abril de 2017, um único arquivo compactado através da opção *Tarefas* na Turma Virtual do SIGAA contendo: (1) os mesmos arquivos de código fonte disponíveis no repositório Git; (2) a documentação do projeto na forma de páginas HTML, geradas automaticamente com a ferramenta Doxygen, e; (3) o relatório escrito, preferencialmente em formato PDF. É importante destacar que serão avaliados única e exclusivamente os arquivos submetidos via SIGAA.

## 7. Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (1) utilização correta dos conteúdos vistos nas aulas presenciais da disciplina; (2) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (3) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (4) qualidade do relatório técnico produzido. O trabalho possuirá nota máxima de 5,00 (cinco) pontos, distribuídos de acordo com a seguinte composição:

| **Item avaliado** | **Nota máxima** |
|---|---|
| Modularização adequada | 0,25 |
| Documentação do código | 0,30 |
| Criação do Makefile | 0,20 |
| Leitura do arquivos | 0,25 |
| Escrita de arquivos | 0,25 |
| Tradução texto para morse | 1,00 |
| Tradução morse para texto | 1,00 |
| Uso correto de controle de versão com Git | 0,50 |
| Uso consistente de alocação dinâmica de memória | 0,25 |
| Aplicação adequada de sobrecarga de funções e ponteiros para funções | 0,50 |
| Qualidade do relatório escrito | 0,50 |
| **Total** | **5** |

Por sua vez, o não cumprimento de algum dos critérios anteriormente especificados poderá resultar nos seguintes decréscimos, calculados sobre a nota total obtida até então:

| **Falta** | **Decréscimo** |
|---|---|
| Programa apresenta erros de compilação, não executa ou apresenta saída incorreta | –70% |
| Falta de comentários no código fonte e/ou de documentação gerada com Doxygen | –10% |
| Implementação na linguagem C ou resultante de mistura entre as linguagens C e C++ | –30% |
| Programa compila com mensagens de aviso (*warnings*) | –50% |
| Plagiarismo | –100% |

*Figura 1: Here go!*