

Segundo Trabalho de Implementação

Programação III 2021/02

Prof. João Paulo A. Almeida

17 de fevereiro de 2022

O objetivo deste trabalho é praticar os conceitos básicos da programação orientada a objetos. Para isso, vocês deverão implementar o primeiro trabalho novamente, mas agora na linguagem C++. O formato dos arquivos de entrada e o formato da saída são os mesmos do primeiro trabalho.

O trabalho deverá ser realizado em duplas (ou individualmente).

Leia atentamente **todo** este documento. Havendo dúvidas ou inconsistências, entre em contato com o professor imediatamente.

O prazo para entrega é dia **16 de março de 2022**.

Condições de Entrega (C++)

A função `main` deve receber, pela linha de comando, os nomes dos arquivos CSV com os dados referentes aos candidatos e partidos para um município e a data da eleição (que será usada para calcular as idades dos candidatos). O nome do executável gerado deve ser `vereadores`. Todos os arquivos fornecidos serão gerados com codificação UTF-8.

Exemplo de comando de execução:

```
./vereadores vitoria-candidatos.csv vitoria-partidos.csv 15/11/2020
```

Inclua o código fonte e o arquivo Makefile (ver abaixo) em um arquivo `.zip` nomeado com `vereadores` seguido dos nomes dos componentes do grupo separados por hífen. (Exemplo: `vereadores-JoseSilva-AntonioVieira.zip`). O mesmo arquivo `.zip` deve ser submetido por cada integrante do grupo no Classroom.

Quando o professor receber seu trabalho, ele executará a seguinte sequência de comandos:

```
unzip <arquivo>.zip
make
./vereadores <caminho_arquivo_candidatos> <caminho_arquivo_partidos> <data>
```

Haverá desconto de nota dos grupos cujos trabalhos não seguirem esse padrão.

Relatório

Esse trabalho **não demandará um relatório** com descrição de implementação e testes. Caso você queira descrever algum *bug* conhecido de sua implementação, faça-o em um arquivo `README.txt` localizado na pasta raiz (na mesma pasta do Makefile).

Avaliação

Serão considerados os critérios abaixo.

- A fração de casos de teste que seu trabalho resolver de forma correta é um fator importante na determinação da nota.
- Organização e clareza de código valem nota. Trabalhos confusos e que violem boas práticas de programação orientada a objetos (discutidas em aula) sofrerão desconto.
- A presença de vazamento de memória implicará em desconto. Utilize o utilitário *valgrind* ao longo do desenvolvimento do trabalho.
- Colaboração indevida (cópia entre diferentes grupos) não será tolerada, e grupos com trabalhos copiados (mesmo que parcialmente) receberão nota mínima no trabalho como um todo.

Observação 1: dado que existem várias versões da linguagem e dos compiladores de C++, fica determinado a versão 11 do C++ e o compilador g++ como versões de referência para a correção dos trabalhos. Além disso, os arquivos de código-fonte devem estar codificados em UTF-8 para evitar erros de compilação.

Observação 2: a opção `-std=c++11` do g++ garante que seu trabalho estará sendo compilado de acordo com a versão 11 do C++.

Um *script* de testes (similar ao do trabalho Java) será disponibilizado.

Makefile

Para que o executável do seu trabalho possa ser gerado de forma automatizada, o arquivo Makefile deve, obrigatoriamente, encontrar-se na raiz da pasta criada. Além disso, ele deve ser feito de forma a responder aos seguintes comandos:

Comando	Resultado esperado
make	O código-fonte deve ser compilado, gerando o executável <code>vereadores</code>
make run	O programa deve ser executado especificando os arquivos <code>candidatos.csv</code> e <code>partidos.csv</code> e a data <code>15/11/2020</code> como parâmetros.
make clean	Todos os arquivos gerados e eventuais arquivos de entrada de dados devem ser excluídos, sobrando somente o conteúdo original do arquivo compactado (ou seja, o código-fonte e o Makefile).

Um arquivo Makefile que satisfaz esses requisitos será disponibilizado, com o seguinte conteúdo (você pode reutilizá-lo como está ou ajustá-lo de acordo com a sua organização de arquivos fonte):

```
#####
# Exemplo de makefile
# Prof. João Paulo A. Almeida
# Programação III
#
# A princípio, você não precisa alterar nada, mas aqui assume-se que o diretório atual
# é o diretório onde estão os códigos fonte (.cpp).
#
# nome do compilador
CPP = g++

# opções de compilação
CFLAGS = -Wall -g
CPPFLAGS = -std=c++11

# define lista de arquivos-fonte, assumindo que estão no diretório atual
FONTES = $(wildcard *.cpp)
```

```

# define lista dos arquivos-objeto usando nomes da lista de arquivos-fonte
OBJETOS = $(FONTES:.cpp=.o)

# nome do arquivo executável
EXECUTAVEL = vereadores

##### alvos
#
# use @ antes de todos os comandos, pois é necessário no script de teste
#

# alvo principal é o executável
all: $(EXECUTAVEL)

# para linkar o executável, precisamos dos arquivos-objetos
$(EXECUTAVEL): $(OBJETOS)
    @$(CPP) -o $$@ $$^

# alvo para cada arquivo-objeto depende do código fonte
# (observação, aqui não estamos capturando as dependências de arquivos .h)
%.o: %.cpp
    @$(CPP) $(CPPFLAGS) -c $(CFLAGS) $$^

# o comando para executar deve especificar sempre como parâmetro divulga.csv
run: $(EXECUTAVEL)
    @./$(EXECUTAVEL) candidatos.csv partidos.csv 15/11/2020

# comando para limpeza
clean:
    @rm *.o vereadores *.csv *.txt

```