Trabalho prático de CI1001
tp2
Departamento de Informática/UFPR

1 Sobre a entrega do trabalho

São requisitos para atribuição de notas a este trabalho:

- Uso de um arquivo `makefile` para facilitar a compilação. Os professores executarão `make` e deverão obter o arquivo executável funcional com a sua solução. Este executável, cujo nome deverá ser `tp2`, deverá estar no subdiretório `tp2`;
- Se não compilar, o trabalho vale zero. Haverá desconto por cada *warning*;
- Arquivo de entrega:
  - Deve estar no formato TAR comprimido (`.tgz`);
  - O arquivo `tgz` deve ser criado considerando-se que existe um diretório com o nome do trabalho. Por exemplo, este trabalho é o `tp2`;
  - Então seu arquivo `tgz` deve ser criado assim:
    - Estando no diretório `tp2`, faça:
    - `cd ..`
    - `tar zcvf tp2.tgz tp2`
  - Desta maneira, quando os professores abrirem o arquivo `tgz` (com o comando `tar zxvf tp2.tgz`) terão garantidamente o diretório correto da entrega para poderem fazer a correção semi-automática.
  - O que colocar no arquivo `tgz`? Todos os arquivos que são necessários para a compilação. Por isso, se você usa arquivos além dos especificados, coloque-os também. Mas minimamente ele deve conter todos os arquivos `.c`, `.h` e o `makefile`;
  - Os professores testarão seus programas em uma máquina do departamento de informática (por exemplo, cpu1), por isso, antes de entregar seu trabalho faça um teste em máquinas do DINF para garantir que tudo funcione bem.

2 Objetivos

Este trabalho tem como objetivo modificar o Tipo Abstrato de Dados (TAD) para números racionais feito no trabalho anterior para que algumas funções recebam parâmetros por endereços. O motivo da alteração é para que os protótipos das funções fiquem mais elegantes, pois tendo a opção de modificar o valor de uma variável recebida como parâmetro por endereço, podemos utilizar o retorno da função para retornar um código de erro.

3 O trabalho

Você deve adaptar a sua implementação do arquivo `racional.c` conforme o novo arquivo `racional.h` fornecido.

Você deve baixar o arquivo `tp2.tgz` anexo a este enunciado e abri-lo para poder fazer o trabalho, pois irá precisar de todos os arquivos ali contidos:

- `racional.h`: arquivo (*read only*) de *header* com todos os protótipos das funções para manipular números racionais;
- `racional.c`: esqueleto de arquivo, a completar;
- `makefile`: sugestão de um makefile que você pode usar.
  É sua responsabilidade fazer as adaptações necessárias neste arquivo sugerido.
- `tp2.c`: esqueleto de arquivo, a completar.
- `entrada*.txt`: arquivos de teste com dados de entrada.

O arquivo `.h` não pode ser alterado. Na correção, os professores usarão o arquivo `.h` original.

- Use boas práticas de programação, como indentação, bons nomes para variáveis, comentários no código, bibliotecas, *defines*... Um trabalho que não tenha sido implementado com boas práticas vale zero.
- Quaisquer dúvidas com relação a este enunciado devem ser solucionadas via email para `prog1prof@inf.ufpr.br`, pois assim todos os professores receberão os questionamentos. Na dúvida, não tome decisões sobre a especificação, pergunte!
- Não envie mensagens pelo Moodle, os professores nem sempre estão logados na plataforma. As respostas serão mais rápidas se as mensagens vierem no e-mail acima.
- Dúvidas podem e devem ser resolvidas durante as aulas.

4 Seu programa

No arquivo `racional.h` foi definida uma interface para o tipo abstrato de dados *racional* que usa a mesma estrutura para os números racionais usada no trabalho anterior. Você deve implementar o arquivo `racional.c` conforme especificado no arquivo `racional.h` fornecido. Seu programa principal deve incluir o *header* `racional.h` e deve ter uma função `main` que implemente corretamente em C o seguinte pseudo-código:

```
defina um vetor para até 100 números racionais
```

```
leia um valor n tal que 0 < n < 100

preencha o vetor com n números racionais lidos da entrada
(leia o numerador e o denominador de cada racional)

imprima "VETOR = " e o conteúdo do vetor lido

elimine deste vetor os elementos inválidos
imprima "VETOR = " e o conteúdo do vetor resultante

ordene este vetor
imprima "VETOR = " e o conteúdo do vetor resultante

calcule a soma de todos os elementos do vetor
imprima "SOMA = " e a soma calculada acima
nova linha

retorne 0
```

Imprima os elementos do vetor em uma única linha, usando um único espaço em branco para separar cada elemento. Ao final do vetor mude de linha.

5 Exemplos de entrada e saída

Considerando que o usuário digitou a seguinte entrada (arquivo `entrada1.txt`):

```
15
-1 3
5 0
2 7
1 9
9 0
8 -5
-7 0
0 8
6 11
7 -17
1 0
12 36
-5 1
4 9
5 0
```

A saída correspondente deve ser:

```
VETOR = -1/3 NaN 2/7 1/9 NaN -8/5 NaN 0 6/11 -7/17 NaN 1/3 -5 4/9 NaN
VETOR = -1/3 4/9 2/7 1/9 -5 -8/5 1/3 0 6/11 -7/17
VETOR = -5 -8/5 -7/17 -1/3 0 1/9 2/7 1/3 4/9 6/11
SOMA = -331343/58905
```

Observações:

- Os números inválidos foram impressos como `NaN`, que significa *Not a Number*. Essa é uma notação padrão usada em muitas linguagens de programação para imprimir números com valor indefinido ou inválido. Você deve modificar sua função `imprime_r` para usar essa notação.
- A segunda linha da saída pode variar em função do algoritmo usado para remover os números inválidos do vetor, mas as demais linhas devem ser idênticas às mostradas acima.
- Os arquivos `entrada*.txt` fornecidos contêm os dados de entrada destes testes. Para testar seu programa usando os dados contidos no arquivo `entrada1.txt`, por exemplo, você pode digitar `./tp2 < entrada1.txt`. Essa operação se chama *redireção de entrada* e será estudada mais tarde.

Para a entrada de dados contida no arquivo `entrada2.txt`, a saída deve ser:

```
VETOR = NaN NaN
VETOR =
VETOR =
SOMA = 0
```

Para a entrada de dados contida no arquivo `entrada3.txt`, a saída deve ser esta (não se preocupe com as quebras de linha no meio da impressão de cada vetor):

```
VETOR = 19/3 16/3 2/9 11/16 -9 -7/19 15/16 1/13 -7/18 13/20 11/9 16/15
-17/15 2/7 7/4 5/6 -9/10 NaN 1/12 3/2 -17/12 8/3 5/3 14/13 -9 2/9 1/9
18/19 -13/10 -7 -11/17 17/20 -1/14 1/6 NaN -12/7 0 4/7 -2/7 -7/15 1/11
11/2 8/3 19/11 19/2 2/7 8/19 -1 -1 -3/11
VETOR = 19/3 16/3 2/9 11/16 -9 -7/19 15/16 1/13 -7/18 13/20 11/9 16/15
-17/15 2/7 7/4 5/6 -9/10 -3/11 1/12 3/2 -17/12 8/3 5/3 14/13 -9 2/9 1/9
18/19 -13/10 -7 -11/17 17/20 -1/14 1/6 -1 -12/7 0 4/7 -2/7 -7/15 1/11
11/2 8/3 19/11 19/2 2/7 8/19 -1
VETOR = -9 -9 -7 -12/7 -17/12 -13/10 -17/15 -1 -1 -9/10 -11/17 -7/15
-7/18 -7/19 -2/7 -3/11 -1/14 0 1/13 1/12 1/11 1/9 1/6 2/9 2/9 2/7 2/7
8/19 4/7 13/20 11/16 5/6 17/20 15/16 18/19 16/15 14/13 11/9 3/2 5/3
19/11 7/4 8/3 8/3 16/3 11/2 19/3 19/2
SOMA = 82626407/6126120
```

6 O que entregar

Entregue um único arquivo `tp2.tgz` que contenha por sua vez os seguintes arquivos:

- `racional.h`: o mesmo arquivo fornecido, não o modifique;
- `racional.c`: sua implementação das funções definidas em `racional.h`;
- `tp2.c`: contém a função `main` que usa os racionais;
- `makefile`

Atenção: Não modifique em nenhuma hipótese o arquivo `racional.h`. Na correção, os professores usarão o arquivo originalmente fornecido.

Bom trabalho!