Números racionais - alocação dinâmica

Este projeto foi inicialmente proposto pelo prof. Marcos Castilho para a disciplina de Programação 1 (CI1001).



Este trabalho visa modificar o módulo de números racionais desenvolvido anteriormente para usar alocação dinâmica de memória.

Atividade

Você deve implementar um programa que manipule números racionais, que são números da forma $^{a}/_{b}$, onde *a* e *b* são números inteiros. Nesta versão, os números racionais devem ser alocados dinamicamente.

Você deve baixar este arquivo e abri-lo para poder fazer o trabalho, pois irá precisar de todos os arquivos ali contidos:

- `racional.h`: arquivo de cabeçalho com os protótipos das funções (**não deve ser alterado**).
- `racional.c`: arquivo que implementa as operações sobre números racionais ("esqueleto" a completar).
- `tp3.c`: código que usa a biblioteca de racionais ("esqueleto" a completar).
- `makefile`: arquivo do utilitário "make" para compilar seu código.
- `testes/*`: dados de entrada e de saída para testes.
- `testa.sh`: script de teste.

No arquivo `racional.h` foi definida uma estrutura (*struct*) para o tipo abstrato de dados *racional* e os protótipos das funções que permitem manipular essa estrutura. Você deve implementar essas funções no arquivo `racional.c`.

Este enunciado foi produzido pelo prof. Castilho para este projeto.

Programa principal

O programa principal (`tp3.c`) contém a função `main`. Ele deve incluir o *header* `racional.h` e implementar corretamente o seguinte pseudocódigo:

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

aloque dinamicamente um vetor com n ponteiros para números racionais

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

imprima "VETOR = " e os racionais apontados pelo vetor
```

```
elimine do vetor os racionais inválidos
imprima "VETOR = " e o vetor resultante

ordene o vetor em ordem crescente
imprima "VETOR = " e os racionais apontados pelo vetor

calcule a soma dos racionais apontados pelo vetor
imprima "SOMA = " e a soma calculada acima

libere os racionais apontados pelo vetor
imprima "VETOR = " e os racionais apontados pelo vetor

libere o vetor de ponteiros
libere o espaço utilizado para fazer o cálculo da soma

retorne 0
```

Exemplos de funcionamento

O diretório `testes/` fornecido contém um conjunto de entradas e suas respectivas saídas esperadas. Considerando que o usuário informou como entrada o conteúdo de `testes/entrada_1.txt`:

```
6
2 2
6 3
1 0
4 1
1 0
1 0
```

A saída correspondente deve ser esta (conteúdo de `testes/saida_1.txt`):

```
VETOR = 1 2 NaN 4 NaN NaN
VETOR = 1 2 4
VETOR = 1 2 4
SOMA = 7
VETOR = NULL NULL NULL
```

Script de teste

Disponibilizamos um *script shell* para testar seu programa mais facilmente. Neste script fazemos uso de pipes combinado com o comando `diff`, que faz a comparação das saídas geradas pelo programa com as saídas esperadas.

O uso do *script* é simples:

1. `./testa.sh 1` : executa o programa e compara cada saída gerada com a saída esperada.
2. `./testa.sh 2` : analisa a execução do programa usando a ferramenta Valgrind.

No **teste 1**, seu programa é executado para cada uma das entradas fornecidas no diretório `testes/`. Se a saída do script for vazia para uma entrada é porque a saída gerada é igual à saída esperada, ou seja, seu programa tratou corretamente aquela entrada. Caso contrário, haverá uma saída neste formato:

```
3c3
```

```
< VETOR = 1 2 4
---
> VETOR = 1 2 3
```

Onde "<" indica uma linha no primeiro arquivo (saída gerada) e ">" a linha correspondente no segundo arquivo (saída esperada).

No **teste 2**, a ferramenta Valgrind executa seu programa e analisa as alocações, liberações e acessos à memória feitos por ele. Ela pode encontrar acessos inválidos, áreas não liberadas, uso de variáveis não inicializadas e outros problemas relacionados à memória. Caso seu programa esteja correto em relação ao uso da memória, o Valgrind deverá gerar estas mensagens:

```
All heap blocks were freed -- no leaks are possible.
...
ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Caso contrário, seu programa tem vazamentos e as mensagens de erro deverão indicar a causa. Outras mensagens que o Valgrind pode gerar são do tipo:

```
Invalid read of size 8
Invalid write of size 8
Conditional jump or move depends on uninitialised value(s)
...
```

Elas significam que seu programa, embora possa ter passado eventualmente pelo teste 1, ainda tem erros graves de acesso à memória e pode apresentar comportamento instável (pode falhar às vezes).



Caso seu programa apresente erros de memória ou comportamento instável no teste 1, você pode usar o teste 2 para obter mais informações sobre erros, usando o Valgrind.

Entregáveis

Entregue um único arquivo `tp3.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`
- `tp3.c`: contém a função `main` que usa os racionais
- `makefile`

From:
https://wiki.inf.ufpr.br/maziero/ - **Prof. Carlos Maziero**

Permanent link:
https://wiki.inf.ufpr.br/maziero/doku.php?id=c:numeros_racionais_-_malloc

Last update: **2024/10/07 15:26**