Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES
Estrutura de Dados I
Prof. Oberlan Romão

Tipo Abstrato de Dados

Atenção: Ao terminar, não se esqueça de enviar as soluções no AVA.

Exercícios

1. Um TAD Racional pode ser definida como mostrado no código (incompleto) do arquivo `ex1.c`. Note que o código não foi separado entre interface (`Racional.h`), implementação (`Racional.c`) e cliente (`main.c`).

   a) Compile e execute o programa para ver o resultado.

   b) Implemente as funções incompletas;

   c) Descomente a linha que contém o `imprimeRacional(z)`. O que aconteceu? Por quê?

   d) Modifique a função `imprimeRacional` para que a mesma só imprima se o ponteiro passado já tenha sido inicializado/alocado. Alguma outra função precisa dessa verificação? Se sim, modifique-as.

   e) Descomente as outras linhas, compile e execute o programa para testá-lo.

   f) Crie um `Racional` com valor 9/6 para testar a função `comparaRacional` com o `Racional y` (que é igual) e `x` (que não é igual).

```
Racional *r = inicializaRacional(9, 6);
if (comparaRacional(r, y))
    printf("São iguais\n");
else
    printf("São diferentes\n");
...
```

   g) Ao fazermos `z = somaRI(z, 3);` teremos um vazamento de memória. Por quê? O que podemos fazer para resolver esse problema?

2. Separe o TAD Racional em interface (`Racional.h`), implementação (`Racional.c`) e cliente (`main.c`).

   a) Compile o código para gerar cada objeto (arquivo .o). Corrija os possíveis erros.

```
gcc -c Racional.c
gcc -c main.c
gcc main.o Racional.o -o Racional.exe
```

b) Faça um arquivo `Makefile` para auxiliar na compilação e linkagem. Se não tiver o MAKE instalado, use o Repl.it ou CS50 IDE. Veja alguns exemplos:

- https://repl.it/@oberlan/ED1-Aula-Teorica-TADPonto-Makefile#Makefile
- https://repl.it/@oberlan/ED1-Aula-Teorica-TADCirculo-Makefile#Makefile
- GitHub da Disciplina

3. Utilizando o TAD Racional do exercício anterior, faça um programa para calcular o valor do seguinte somatório, dado o número n de termos:

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots + \frac{1}{2^n}$$

Até que valor de n o seu programa consegue um resultado válido?

Use o Valgrind para ter certeza que toda memória alocada foi desalocada.

4. Faça uma função `simplifica` para o TAD Racional para simplificar o racional, por exemplo, $\frac{4}{8}$ pode ser simplificado e representado por $\frac{1}{2}$. Para isso, divida o numerador e o denominador pelo seu Máximo Divisor Comum (MDC). O valor do MDC pode ser calculado recursivamente por:

$$MDC(x, y) = \begin{cases} x, & \text{se } y = 0 \\ mdc(y, x\%y), & \text{caso contrário} \end{cases}$$

a) A função `simplifica` pode ser usada após algumas operações (`somaRR`, `somaRI`, `setNum`, `setDen`, `multiplicaRR`, `multiplicaRI`, etc.). Assim o usuário do TAD nunca precisa chamar a função `simplifica`, e o valor do Racional estará sempre na sua forma mínima. Faça isso e coloque a função `simplifica` (e a função `mdc`) apenas no arquivo `Racional.c` (ou seja, não coloque o protótipo dessas duas funções na interface), assim outros arquivos não terão acesso a essas funções.

Sugestão: a função `simplifica` deve receber apenas um ponteiro para um Racional. Nela calcule o MDC do numerador e denominador do Racional. Em seguida, divida o numerador e o denominador pelo MDC.

b) Refaça o exercícios 3 utilizando esse TAD, que sempre guarda o resultado simplificado. Note que agora você pode obter resultados mais precisos para valores maiores de n.