DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Programação de Computadores (2013-2014)

Décima folha de Problemas

Matéria abordada: funções (estruturas; passagem de parâmetros por valor e por referência; ponteiros.)
Bibliografia: Capítulos 12 e 16 de [Oua03].
Nota: Da folha 6 em diante é proibido o uso de variáveis globais!

1. Os problemas que se seguem têm um tema comum: operações com fracções. Assim sugere-se:

   - Coloque no ficheiro `frac.h` a definição da `struct frac`, e dos protótipos de todas as funções que implementar neste problema ou nos seguintes desta folha.
   - Coloque no ficheiro `frac.cpp` todas as definições das funções que implementar neste problema ou nos seguintes desta folha.
   - Coloque no ficheiro `racional.cpp` o programa principal que lhe permite testar as funções à medida que as vai implementando.

   Considere a seguinte representação para um número racional (positivo ou negativo):

   ```
   struct frac {
     int num, den; // numerador e denominador
   };
   ```

   (a) Escreva uma função que dados dois inteiros (numerador e denominador) devolve um objecto do tipo `struct frac`.

   Comece por escrever o protótipo dessa função. Designe-a por **fraccao**.

   (b) Escreva uma função que dado um objecto do tipo `struct frac` apresenta o seu valor na forma numerador/denominador. Exemplo: `struct frac a = {1,4};` seria apresentado como `1/4`. Comece por escrever o protótipo dessa função (designe-a por **mostra**).

   (c) Escreva uma função que dado dois objectos do tipo `struct frac` retorna a sua soma (designe-a por **soma**).

   (d) Escreva uma função que dada uma tabela de objectos `struct frac`, e o número de elementos dessa tabela, calcula a sua soma. Faça duas implementações:

   i. a função devolve a fracção soma;

   ii. a fracção soma é um parâmetro de saída da função.

   (e) Em operações com fracções recorda-se que é útil escrever uma fracção na sua forma irredutível:

   $$\frac{10}{20} = \frac{1}{2}$$

   i. A forma irredutível de uma fracção a/b pode ser obtida com base no seguinte algoritmo (em pseudo código), assumindo que $a > b$ $(a, b > 0)$:

Entrada(s): a, b
1: dividendo ← a
2: divisor ← b
3: **Enquanto** resto(dividendo/divisor) diferente de 0 **Faz**
4: c ← resto(dividendo/divisor)
5: dividendo ← divisor
6: divisor ← c
7: **Fim Enquanto**
8: a ← a/divisor
9: b ← b/divisor
10: **retorna** a,b **{Através dos parâmetros}**

Implemente a função. Considere o protótipo:

```
void irred(int &a, int &b);
```

ii. A função **irred** supõe que os seus argumentos verificam $a > b$ $(a, b > 0)$. Escreva uma função **irredutivel** que não tenha essas restricções. Considere o protótipo:

```
// Reduz a fraccao em entrada}
void irredutivel(struct frac & entrada);
```

iii. Escreva um função, que dada uma tabela de elementos do tipo `struct frac` e o seu número de elementos, substitui cada elemento da tabela pela fracção irredutível correspondente.

iv. Implemente uma função, designada **fracigual**, que dadas duas fracções do tipo `struct frac`, verifica se são ou não iguais.

v. Note que se a/b é uma dada fracção e a'/b' é a correspondente fracção irredutível então o máximo divisor comum entre a e b é dado por $\mathbf{mdc}\,(a, b) = a/a' = b/b'$. Com base nesta propriedade implemente uma função que permita calcular o máximo divisor comum de dois números.

(f) Ao dividir fracções que não estão reduzidas, $\frac{a}{b} : \frac{c}{d} = \frac{ad}{bc}$ pode ocorrer transbordo aritmético, que seria evitado se as funções fossem reduzidas antes de serem multiplicadas.

Escreva uma função, **divisao**, a qual procede à redução das fracções (usando **irredutivel**) antes de as dividir. Considere o seguinte protótipo:

```
struct frac divisao(const struct frac &a, const struct frac &b);
```

para diminuir a quantidade de informação que é passada na pilha.

(g) Escreva um programa principal que lhe permita ir testando as funções à medida que as vai construindo.

2. Em operações com fracções recorda-se que é útil calcular o mínimo múltiplo comum para reduzir fracções ao mesmo denominador (o menor possível) antes de as subtrair:

$$\frac{1}{6} - \frac{1}{10} = \frac{5}{30} - \frac{3}{30} = \frac{5-3}{30} = \frac{2}{30}$$

(a) O mínimo múltiplo comum é dado por:

$$\text{mmc}(a, b) = \frac{a \times b}{\text{mdc}(a, b)} = \frac{a}{\text{mdc}(a, b)} \times b$$

Escreva uma função, **mmc** que dada a função **mdc** construída no exercício anterior calcula o mínimo múltiplo comum de dois inteiros.

(b) Escreva uma função, **subtraccao**, que dadas duas fracções como objectos `struct frac` retorna a sua diferença sob a forma de fracção reduzida.

No exemplo anterior, o resultado final depois de reduzido:

$$\frac{2}{30} = \frac{1}{15}.$$

Utilize as funções **irredutivel** e **mmc**.

(c) Escreva um programa principal em que ilustre a chamada às funções construídas.

3. Com base nas funções já desenvolvidas nas dois exercícios anteriores é fácil construir uma calculadora de números racionais sob a forma de fracções:

(a) Implemente a função **adicao** com base na função **subtraccao**.

(b) Implemente a função **produto** com base na função **divisao**.

(c) Implemente a função **caluladora** que dadas duas fracções e o símbolo da operação desejada (*, +, -, /) calcula e devolve a fracção que resulta da operação indicada.

TPC (a) Escreva uma função, de protótipo `int fracmp(const frac & fa, const frac & fb);` que devolve -1 se a fracção `fa` é menor que a fracção `fb`, 0 se são iguais e +1 se a fracção `fa` é maior que a fracção `fb`.

(b) Implemente uma função que dada uma tabela de elementos do tipo `struct frac`, e o número de elementos dessa tabela, devolve a fração de menor valor. Utilize como função auxiliar a função `fracmp` que implementou na alínea anterior.

(c) Escreva um program principal que lhe permita testar a correção das funções implementadas.

Referências

[Oua03] S. Oualline. *Practical C++ Programming.* O'Reilly, 3rd edition, 2003.