Programação e Métodos Numéricos em Finanças I

Lista de exercícios 3 (data máxima de entrega 24/04/2012)

1. Crie uma classe chamada *Racional* para fazer aritmética com frações. Use variáveis inteiras para representar os dados `private` da classe: o numerador e o denominador. Forneça um construtor default, um construtor com dois parâmetros (numerador e denominador), um construtor com um único parâmetro (uma vez que todo inteiro é um número racional, com denominador unitário) e um construtor de cópia. O construtor deve armazenar a fração em formato reduzido (por exemplo, a fração 2/4 seria armazenada no objeto como 1 no numerador e 2 no denominador). Escreva métodos `public` para cada um dos seguintes itens:

- Método *Add*: Adição de dois números do tipo *Racional*. Retorna um objeto *Racional*, com o resultado armazenado em forma reduzida;
- Método *Sub*: Subtração de dois números do tipo *Racional*. Retorna um objeto *Racional*, com o resultado armazenado em forma reduzida;
- Método *Mult*: Multiplicação de dois números do tipo *Racional*. Retorna um objeto *Racional*, com o resultado armazenado em forma reduzida;
- Método *Div*: Divisão de dois números do tipo *Racional*. Retorna um objeto *Racional*, com o resultado armazenado em forma reduzida;
- Método *Less*: Retorna *true* se a fração atual é menor que a fração passada como argumento do método e *false* caso contrário;
- Imprimir números do tipo *Racional* no formato *a/b*, onde *a* é o numerador e *b* o denominador;
- Imprimir números do tipo *Racional* em formato de ponto flutuante.

Na criação da classe, separe em arquivos distintos a interface da implementação.

2. Crie uma classe em C++ chamada Relogio para armazenar um horário, composto por hora, minuto e segundo. Escreva um programa para testar sua classe.

- Escreva um construtor que inicialize os dados com zero e outro construtor que inicie os dados com valores fornecidos como parâmetros;
- Escreva um construtor de cópia;
- Escreva um método para solicitar a hora, minutos e segundos para o usuário.
- Escreva um método para imprimir a hora no formato `hh:mm:ss`;
- Escreva um método chamado `SetHora`, que deve receber o horário desejado por parâmetro (hora, minuto e segundo);
- Escreva um método chamado `GetHora` para retornar o horário atual, através de 3 variáveis passadas por referência;
- Escreva um método para avançar o horário para o próximo segundo.

Na criação da classe, separe em arquivos distintos a interface da implementação.

3. Crie uma classe abstrata que representa uma opção. Para esta classe:

- Crie membros `protected` que representam as variáveis que afetam o preço de uma opção:
  - Spot
  - Strike
  - Volatilidade do ativo objeto
  - Taxa de juros livre de risco
  - Prazo para o vencimento em anos
- Crie um membro `protected` que representa o tipo da opção: CALL ou PUT (esse membro deve ser uma variável do tipo `enum`);
- Escreva um construtor *default* que inicializa os membros com valores fixos válidos quaisquer;
- Escreva um construtor de cópia;
- Escreva um construtor que receba como parâmetros os valores de cada um dos membros;
- Escreva um destrutor virtual
- Crie um membro `static protected` para contar o número de objetos já criados da classe e implemente a lógica para que ele seja atualizado adequadamente;
- Crie um método `static public` que retorna o número de objetos já criados da classe;
- Escreva métodos `public Get` e `Set` para todos os membros;
- Declare como constantes todos os métodos que assim podem ser declarados, para que objetos constantes possam ser criados adequadamente;
- Escreva um método virtual puro `public` que calcula e retorna o preço da opção com a seguinte definição: `double CalculaPreco()=0;`

Na criação da classe, separe em arquivos distintos a interface da implementação.

4. Crie uma classe derivada da classe do exercício 3 para representar uma opção européia. Para esta classe:

- Escreva um construtor *default* que inicializa os membros com valores fixos válidos quaisquer;
- Escreva um construtor de cópia;
- Escreva um construtor que receba como parâmetros os valores de cada um dos membros (incluindo os membros da classe base);
- Escreva um destrutor virtual
- Implemente o método `CalculaPreco` utilizando o modelo de Black & Scholes;
- Declare como constantes todos os métodos que assim podem ser declarados, para que objetos constantes possam ser criados adequadamente;
- Escreva um programa para testar sua classe.

Na criação da classe, separe em arquivos distintos a interface da implementação.

5. Crie uma classe derivada da classe do exercício 3 para representar uma opção americana. Para esta classe:

- Adicione um membro `protected` que representa o número de passos utilizado na precificação por árvore trinomial;
- Adicione outros dois membros `protected` que representam as árvores do ativo e do derivativo;
- Escreva dois métodos `protected`, um para alocar memória dinamicamente para as árvores e outro para desalocar a memória alocada dinamicamente. Procure alocar a menor quantidade de memória possível;
- Escreva um construtor *default* que inicializa os membros com valores fixos válidos quaisquer;
- Escreva um construtor de cópia;
- Escreva um construtor que receba como parâmetros os valores de cada um dos membros (exceto as árvores, incluindo os membros da classe base);
- Escreva um destrutor virtual
- Adicione métodos `Get` e `Set` para o membro que representa o número de passos. Ao alterar o número de passos, as árvores devem ser desalocadas da memória e na sequência realocadas caso o novo número de passos seja diferente do atual;
- Sobrescreva o método `CalculaPreco` por uma versão que calcula o preço da opção americana utilizando árvore trinomial;
- Declare como constantes todos os métodos que assim podem ser declarados, para que objetos constantes possam ser criados adequadamente;

Na criação da classe, separe em arquivos distintos a interface da implementação.

Em uma árvore trinomial supõe-se que o ativo objeto, em cada passo de tempo, pode aumentar por um fator *u*, permanecer com o mesmo valor ou diminuir por um fator *d*. Os parâmetros da árvore trinomial são:

- T: prazo (em anos) para o vencimento da opção;
- N: número de passos da árvore;
- $dt = \frac{T}{N}$: passo de tempo;
- $u = e^{\sigma\sqrt{\frac{3 \cdot dt}{2}}}$: fator de subida do preço do ativo objeto;
- $d = \frac{1}{u}$: fator de descida do preço do ativo objeto;
- $p_u = \frac{1}{3} + \left(\frac{r - 0.5 \cdot \sigma^2}{\sigma}\right)\sqrt{\frac{dt}{6}}$: probabilidade do preço do ativo objeto subir pelo fator *u* em um passo de tempo;
- $p_m = \frac{1}{3}$: probabilidade do preço do ativo objeto permanecer constante em um passo de tempo;

- $p_d = \frac{1}{3} - \left(\frac{r - 0.5 \cdot \sigma^2}{\sigma}\right)\sqrt{\frac{dt}{6}}$: probabilidade do preço do ativo objeto diminuir pelo fator *d* em um passo de tempo;
- $C_{i,j} = e^{-r \cdot dt} \cdot \left(p_u \cdot C_{i+1,j} + p_m \cdot C_{i+1,j+1} + p_u \cdot C_{i+1,j+2}\right)$: valor da opção no nó do passo de tempo *i* para o j-ésimo estado do preço do ativo (supondo estados em cada passo i são numerados de 0 até 2i+1, ordenados de forma que o preço do ativo seja crescente).