INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade II - 3 - Sobrecarga de Operadores

Lista Única

Instruções

Responda às questões abaixo. Pode usar este próprio documento. Questões práticas devem ser anexadas separadamente.

As primeiras 4 questões valem 1. As questões 5 e 6 valem 2, a questão 7 vale 3 e a 8 vale 4.

Questões

1. Para que serve a sobrecarga de operadores?

Serve para dar um novo significado a um determinado operador para que funcione em objetos. Ex: Conta conta1 >= Conta conta2, comparando se uma conta tem mais dinheiro que a outra.

2. Quando podemos sobrecarregar um operador como um método da Classe?

Quando sobrecarregamos (), [], -> ou algum operador de atribuição. Ex: res = SEU_OBJ + OUTRO_OBJ. Ou seja, sempre que o "seu" objeto invocar a operação.

3. Quando devemos sobrecarregar um operador como uma função friend?

Quando o objeto invocador não for o "seu". Ex: res = OUTRO_OBJ + SEU_OBJ.

4. Quando devemos sobrecarregar um operador como uma função global?

Embora seja raro, a regra é que nenhum dos objetos seja "seu". Ex: res = OUTRO_OBJ + ALGUM_OUTRO.

5. (Classe Array) Usando a última versão da Classe Array apresentada em sala (Aula9Ex2-2), sobrecarregue o operador + (soma) de modo a permitir a concatenação de um objeto Array com outro. O operador não deve modificar o objeto invocador, mas criar e retornar um terceiro objeto. Você pode implementar o operador + (soma) como método ou como função friend (ou global). Sobrecarregue

também o operador +=. Dessa vez, o objeto que invoca deve ser modificado, obviamente. O operador += deve ser implementado como método.

6. (Classe IntegerSet *Revisitada*) Crie uma Classe IntegerSet (Conjunto de Inteiros) pela qual cada objeto pode "armazenar um conjunto de inteiros" no intervalo de 0 a 99. Internamente, a Classe deve ter um array de 100 posições, onde cada posição representa o respectivo inteiro. Assim, se o elemento 0 do array estiver setado como 1, significa que o inteiro 0 está presente no conjunto. Se o elemento estiver setado como 0, significa que o inteiro não está no conjunto.

   Forneça 2 construtores. Um construtor-padrão, que não recebe argumentos. Esse deve criar um IntegerSet (conjunto) vazio. E um construtor que recebe um array de inteiros e o tamanho, para inicializar o array interno.

   Sobrecarregue os operadores a seguir, observando atentamente o que se pede:

   a. **<<** : insere um novo inteiro k no conjunto (seta a posição k do array para 1). Uso: objetoIntegerSet << 2;

   b. **>>** : remove um inteiro k do conjunto (seta a posição k do array para 0). Uso: objetoIntegerSet >> 2;

   c. **|** : faz a união de dois IntegerSet (conjuntos) e cria e retorna um terceiro conjunto que representa a união dos conjuntos. Uso: objA = objB | objC;.

   d. **&** : faz a interseção de dois IntegerSet (conjuntos) e cria e retorna um terceiro que representa a intersecção. Uso: objA = objB & objC;.

Forneça uma função friend para **imprimir os inteiros presentes no conjuntos**. Essa função também vai sobrecarregar o operador <<.

7. (Classe Complex) Crie uma Classe chamada Complex para realizar aritmética com números complexos. Os números complexos tem a forma: parteReal + parteImaginária * i, onde i é $\sqrt{-1}$. Sua Classe deve sobrecarregar os operadores +, -, +=, -=, ++, --. Além disso deve permitir o uso de << junto com std::cout. Considere:

   a. + : soma o objeto com outro objeto Complex e cria e retorna um terceiro objeto que representa a soma

   b. += : soma o objeto invocador com outro objeto Complex

   c. - : subtrair o objeto com outro objeto Complex e cria e retorna um terceiro objeto que representa a diferença

   d. -= : subtrai os objetos modificando o invocador

   e. ++ : incrementa a parte real em 1. Implemente a forma pré e pós fixada

   f. -- : decrementa a parte real em . Implemente a forma pré e pós fixada.

8. (Classe HugeInteger *Revisitada*) Crie uma Classe HugeInteger que utiliza um array de 40 elementos (char) para armazenar inteiros de até 40 dígitos. Forneça os seguintes operadores para esta Classe:

    a. = : recebe uma string contendo o inteiro. Deve verificar se realmente está recebendo um número. Não precisa considerar sinal (+ ou -), pois os números quando corretos sempre serão positivos. Caso não seja um número, deve-se considerar 0 (zero);

    b. + : soma objeto com outro HugeInteger e cria e retorna um terceiro objeto

    c. +=: soma o objeto com outro HugeInteger modificando o objeto invocador

    d. ==,!=,>,<,>=,<=: compara o objeto com outro objeto HugeInteger passado como argumento. O retorno deve ser do tipo bool;

    e. <<: imprime o número na saída padrão quando usado junto com std::cout