



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?

Resposta: A sobrecarga tem como objetivo dar um novo significado para os operadores para que eles passem a funcionar com objetos.

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

Resposta: Podemos usar quando o seu objeto é quem faz a chamada do operador.

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

Resposta: Pode ser usado quando um objeto que não for seu invocar o operador com um outro objeto seu.

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

Resposta: Deve ser usado quando nenhum dos dois objetos que estão se relacionando com o operador não for seu.

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.

Resposta: [main.cpp - Questão 5 - Replit](#)

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:

- `<<` : insere um novo inteiro k no conjunto (seta a posição k do array para 1).  
Uso: `objetoIntegerSet << 2;`
- `>>` : remove um inteiro k do conjunto (seta a posição k do array para 0). Uso: `objetoIntegerSet >> 2;`
- `|` : faz a união de dois IntegerSet (conjuntos) e cria e retorna um terceiro conjunto que representa a união dos conjuntos. Uso: `objA = objB | objC;`
- `&` : faz a interseção de dois IntegerSet (conjuntos) e cria e retorna um terceiro que representa a intersecção. Uso: `objA = objB & objC;`

Resposta: [main.cpp - Questão 6-2 - Replit](#)

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:
- `+` : soma o objeto com outro objeto Complex e cria e retorna um terceiro objeto que representa a soma
  - `+=` : soma o objeto invocador com outro objeto Complex
  - `-` : subtrair o objeto com outro objeto Complex e cria e retorna um terceiro objeto que representa a diferença
  - `-=` : subtrai os objetos modificando o invocador
  - `++` : incrementa a parte real em 1. Implemente a forma pré e pós fixada
  - `--` : decrementa a parte real em 1. Implemente a forma pré e pós fixada.

Resposta: [main.cpp - Questão 7-2 - Replit](#)

8. (Classe HugelInteger *Revisitada*) Crie uma Classe HugelInteger 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 HugelInteger e cria e retorna um terceiro objeto
  - c. +=: soma o objeto com outro HugelInteger modificando o objeto invocador
  - d. ==, !=, >, <, >=, <=: compara o objeto com outro objeto HugelInteger passado como argumento. O retorno deve ser do tipo bool;
  - e. <<: imprime o número na saída padrão quando usado junto com std::cout