Dúvida comum: Como sobrecarregar o operador de inserção nas subclasses em caso de herança?

Esta é uma dúvida comum. Então vamos a uma breve explicação e uma proposta de solução.

Como o operador de inserção (<<) é implementado fora da classe (por isso o uso do friend), a seu método sobrecarregado não pode ser virtual (ou virtual puro). Diante disto, para permitir a sobrecarga nas subclasses, é necessário:

NA CLASSE Base:

1. Criar um método virtual puro de impressão em ostream

```
virtual std::ostream& print(std::ostream&) const = 0;
```

2. Implementar a sobrecarga do operador de inserção, que deve chamar o método de impressão

```
friend std::ostream& operator << (std::ostream& os, const Base& b) {
   return print(os); // o método print será polimórfico
}
```

NAS CLASSES Derivadas:

3. É necessário apenas implementar a sobrecarga do método para impressão, declarado como virtual na classe Base.

```
std::ostream& print(std::ostream& os) const {
        return os << "Prefixo: " << m_prefixo << " - Codigo: " <<  m_codigo << endl;
}
```

Com isso, qualquer objeto que derive de Base irá responder ao operador de inserção, que deverá chamar o método de impressão de acordo com a subclasse a qual pertence.

O código exemplo completo ficaria:

```
#include <iostream>
#include <string>

using namespace std;

class Base
{
private:
        virtual std::ostream& print(std::ostream&) const = 0;
protected:
```

```
        int m_codigo;
public:
        Base(int vCodigo);
        virtual void imprime()=0;

        friend std::ostream& operator << (std::ostream& os, const Base& b) {
        return b.print(os);
   }
};

Base::Base(int vCodigo):
        m_codigo(vCodigo){}

class TipoA: public Base
{
private:
   string m_prefixo;
   std::ostream& print(std::ostream& os) const {
      return os << "Prefixo: " << m_prefixo << " - Codigo: " <<  m_codigo << endl;
   }
public:
   TipoA(int vCodigo, string vPrefixo);
   void imprime();
};

TipoA::TipoA(int vCodigo, string vPrefixo):
  Base(vCodigo),m_prefixo(vPrefixo){}

void
TipoA::imprime()
{
   cout << m_prefixo << m_codigo << endl;
}

class TipoB: public Base
{
private:
   string m_sufixo;
   std::ostream& print(std::ostream& os) const {
      return os << "Sufixo: " << m_sufixo << " - Codigo: " <<  m_codigo << endl;
   }
public:
   TipoB(int vCodigo, string vSufixo);
   void imprime();
};
TipoB::TipoB(int vCodigo, string vSufixo):
   Base(vCodigo),m_sufixo(vSufixo){}
```

```
void
TipoB::imprime()
{
   cout << m_sufixo << m_codigo << endl;
}

int main(int argc, char const *argv[])
{
   TipoA a(123,"tipoA");
   TipoB b(456,"tipoB");
   cout << a << endl;
   cout << b << endl;
   return 0;
}
```