CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF. JUDSON SANTIAGO

LABORATÓRIO 10

FUNÇÕES AMIGAS

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE ACOMPANHAR PARA REVISAR OS CONCEITOS IMPORTANTES

1. Amplie a classe `Tempo` mostrada abaixo para que seja possível:

   - Somar números inteiros com objetos da classe `tempo`
   - Exibir objetos da classe `Tempo` usando cout
   - Ler objetos da classe `Tempo` usando cin

```
class Tempo
{
private:
    int horas;
    int minutos;

public:
    Tempo(int h = 0, int m = 0);

    Tempo operator+(const Tempo & t) const;
    Tempo operator+(int num) const;
};
```

Teste a classe com o seguinte programa:

```
int main()
{
    Tempo ida, volta, total;

    cout << "Tempo de ida:\n";
    cin >> ida;
    cout << "Tempo de volta:\n";
    cin >> volta;

    total = 5 + ida + volta;  // 5 horas de passeio

    cout << "Tempo da viagem:\n" << total << endl;
};
```

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Considerando a classe `Tempo` desenvolvida na questão anterior, contendo agora funções amigas para leitura com cin e exibição com cout, utilize os métodos de leitura e exibição para ler e gravar arquivos.

```
class Tempo
{
private:
    int horas;
    int minutos;

public:
    Tempo(int h = 0, int m = 0);

    Tempo operator+(const Tempo & t) const;
    Tempo operator+(int num) const;
    friend Tempo operator+(int num, const Tempo & t);

    friend istream & operator>>(istream & is, Tempo & t);
    friend ostream & operator<<(ostream & os, const Tempo & t);
};
```

Teste a classe com o seguinte programa:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream fin { "viagem.txt" };
    if (!fin.is_open())
    {
       cout << "Arquivo viagem.txt não localizado\n";
       return EXIT_FAILURE;
    }

    Tempo ida, volta;
    fin >> ida;
    fin >> volta;
    fin.close();

    Tempo total = 5 + ida + volta;  // 5 horas de passeio

    ofstream fout { "passeio.txt" };

    fout << "Ida: " << ida << endl;
    fout << "Volta: " << volta << endl;
    fout << "Tempo Total: " << total << endl;

    fout.close();
    return 0;
};
```

EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. Construa uma classe para representar uma cor no formato RGB. Uma cor pode ser descrita por três valores inteiros na faixa 0-255. Cada valor representa uma intensidade de cor vermelha (Red), verde (Green) e azul (Blue). Estas três cores, quando combinadas, podem gerar qualquer cor do espectro visível ao olho humano.

   Sobrescreva o operador de multiplicação para permitir multiplicar cores:

   $(R_x,G_x,B_x) * (R_y,G_y,B_y) = (R_x*R_y/255,\ G_x*G_y/255,\ B_x*B_y/255)$

   Forneça funções amigas para ler cores com cin e escrever com cout.

   A exibição de uma cor não deve mostrar nada na tela, deve apenas enviar um código de escape ANSI alterando a cor padrão do terminal para que as próximas exibições usem a cor escolhida. A cor do terminal é modificada enviando para o `cout` um código:

```
cout << "\x1b[38;2;R;G;Bm";  // R, G e B são números de 0-255
cout << "TEXTO";             // TEXTO na cor RGB
cout << "\x1b[0m";           // retorna para a cor padrão
```

   O terminal precisa ter suporte a códigos de escape ANSI TrueColor para que o código acima funcione. A maioria dos terminais modernos no Linux, Windows e MacOS possuem suporte. Teste a classe com um programa:

```
int main()
{
    Cor azul { 0, 163, 215 };
    Cor laranja {255, 170, 0 };

    cout << "Entre com o código de uma cor:\n";
    Cor cor;
    cin >> cor;

    Cor verde = azul * laranja;

    cout << verde << "VERDE" << endl;
    cout << cor << "SUA COR" << endl;

    // volta para a cor padrão
    cout << normal;
};
```

   Exemplo de saída:

```
Entre com o código de uma cor:
145 39 44
VERDE
SUA COR
```

2. Modifique a classe Packet, criada no `Lab09` – Sobrecarga de Operadores, para que seus dados possam ser exibidos com `cout` e lidos com `cin`. Faça com que o código abaixo funcione:

```
int main()
{
    cout << "Entre com os dados do pacote:\n";
    Packet packet;
    cin >> packet;

    cout << "Enviando pacote...\n";
    packet.send();

    cout << "Conteúdo do pacote:\n";
    cout << packet << endl;
}
```

A exibição do pacote deve mostrar os dados separados nas suas partes constituintes (campo `part` da união `Data`).

```
union Data
{
    struct {
        short x;
        short y;
        short z;
        short w;
    } part;

    long long all;
};
```

Exemplo de saída do programa:

```
Entre com os dados do pacote:
9 4 3 7
Enviando pacote...
1970337722138633
Conteúdo do pacote:
9 4 3 7
```

3. Não é uma boa ideia modificar a forma como os operadores são usados na linguagem, mas apenas por diversão, tente sobrescrever o `operator>>` de modo que o programa abaixo exiba o objeto `passeio` na tela.

```
int main()
{
    Tempo passeio {3, 15};
    passeio >> cout;
}
```

Faça uma versão usando uma função membro e outra usando uma função amiga não-membro.