

UNIVERSIDADE ESTADUAL DE MONTES CLAROS
Centro de Ciências Exatas e Tecnológicas
Curso de Engenharia de Sistemas

Diogo Nascimento Silva
Lucas Franklin Silva

TRABALHO PRÁTICO II
PROGRAMAÇÃO ORIENTADA A OBJETOS

Montes Claros - MG
Outubro / 2013

1 – INTRODUÇÃO

Esse trabalho teve como objetivo a aplicação de conceitos de orientação a objetos aprendidos em aula usando a linguagem C++. Foram utilizadas classes pré-definidas como String, Vector, Ifstream, criação de classes simples com construtores, destrutores, sobrecarga de funções, classes aninhadas, padrões de projeto com o Singleton, estruturas básicas de programação como alocação dinâmica de memória, cast de tipo de variáveis e estruturas de dados como lista. Alguns padrões foram adotados por nós para que a codificação ficasse organizada e legível. O código foi indentado usando o plugin Source Code Formater (AStyle) presente no Code::Blocks.

2 – DESENVOLVIMENTO

2.1 - PADRÕES

Para o desenvolvimento dos códigos foram adotados os seguintes padrões:

- Nomes de variáveis, classes e métodos em inglês, com exceção das já especificadas no trabalho.
- Nomes de classes sempre iniciam com letra maiúscula.
- Padrão lowerCamelCase, variação do CamelCase, para nomes de métodos e variáveis.
- Prefixo get para métodos com a finalidade de imprimir ou retornar atributos de uma determinada classe.
- Prefixo set para métodos com a finalidade de setar atributos de uma determinada classe.
- Funções que não modificam o objeto são constantes.

2.2 – COMENTÁRIOS

Foram adotados, para variáveis e métodos, nomes que realmente reflitam pra que eles servem ou o que fazem. Existem poucos comentários no código, colocamos apenas onde foi realmente necessário explicar o que determinada linha de código estava fazendo, o nome das variáveis e métodos fala por si próprio.

2.3 – TESTES E IMPLEMENTAÇÃO

Abaixo serão listados alguns resultados e discussões importantes que tivemos em cada questão:

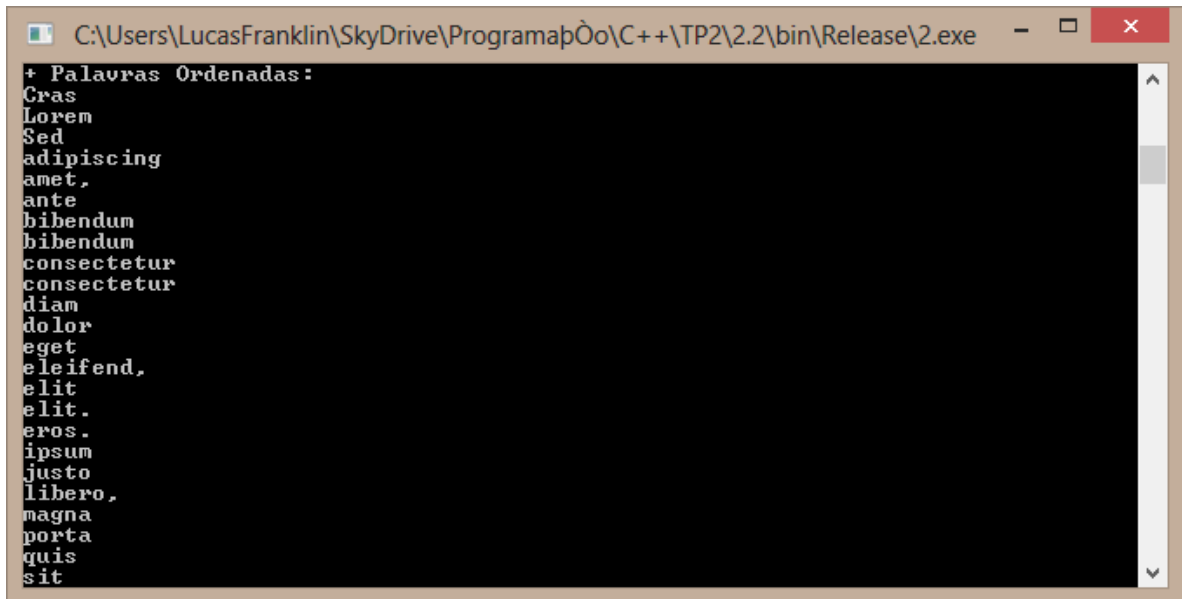
2.3.1 – PROBLEMA 2.1

O problema 2.1 propõe a implementação de um algoritmo de criptografia o rot 13. Esse algoritmo “gira” cada caractere 13 posições no alfabeto. Assim ‘a’ é substituído por ‘n’ e ‘x’ é trocado por ‘k’.

Optamos por implementar essa questão usando um FOR que percorre toda a linha do arquivo de texto lido e, se a letra corrente estiver entre A e M, será somado 13 à variável, o que nos dará a letra “simétrica” na segunda metade do alfabeto e, com um cast, a gravamos em arquivo. Caso esteja entre N e Z, subtraímos 13. Essa solução foi bem simples e transparente, qualquer pessoa que fosse dar manutenção nesse código entenderia rapidamente do que se trata. Usamos métodos da classe String para manipular o nome e a extensão do arquivo. A função rot13Decrypt() simplesmente chama a função rot13Encrypt(), passando uma flag para que a extensão do arquivo seja mudada.

Arquivo usado para o teste: test.txt

2.3.2 – PROBLEMA 2.2

A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\LucasFranklin\SkyDrive\Programapão\C++\TP2\2.2\bin\Release\2.exe. The window contains the following text:

```
+ Palavras Ordenadas:  
Cras  
Lorem  
Sed  
adipiscing  
amet,  
ante  
bibendum  
bibendum  
consectetur  
consectetur  
diam  
dolor  
eget  
eleifend,  
elit  
elit.  
eros.  
ipsum  
justo  
libero,  
magna  
porta  
quis  
sit
```

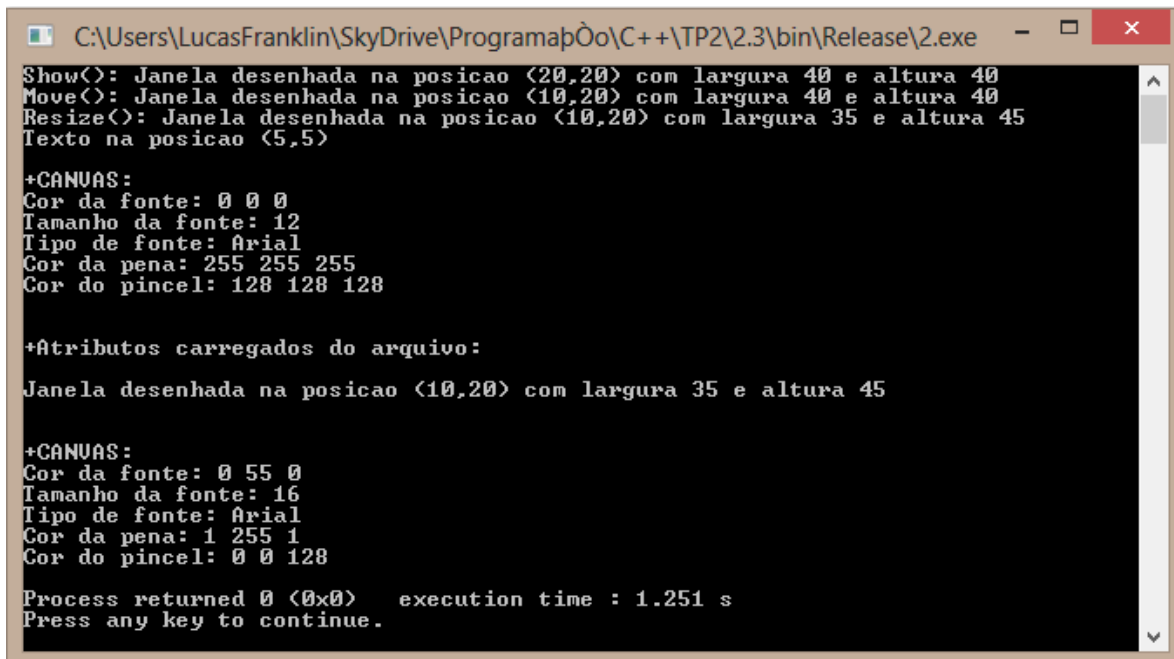
Figura1: Teste-P2.2

O problema 2.2, propõe a criação de um programa que peça ao usuário o nome de um arquivo, abra este arquivo e armazene as strings lidas em um vetor de strings. Logo após, deve-se ordenar as strings e imprimi-las novamente.

Como podemos perceber pela *figura1*, o algoritmo de `sort()` padrão do C++ ordena de forma que Strings com letras maiúsculas apareçam primeiro, depois Strings com letras minúsculas.

Arquivo usado para o teste: test.txt

2.3.3 – PROBLEMA 2.3



```
C:\Users\LucasFranklin\SkyDrive\Programapão\C++\TP2\2.3\bin\Release\2.exe
Show(): Janela desenhada na posicao <20,20> com largura 40 e altura 40
Move(): Janela desenhada na posicao <10,20> com largura 40 e altura 40
Resize(): Janela desenhada na posicao <10,20> com largura 35 e altura 45
Texto na posicao <5,5>
+CANVAS:
Cor da fonte: 0 0 0
Tamanho da fonte: 12
Tipo de fonte: Arial
Cor da pena: 255 255 255
Cor do pincel: 128 128 128

+Atributos carregados do arquivo:
Janela desenhada na posicao <10,20> com largura 35 e altura 45

+CANVAS:
Cor da fonte: 0 55 0
Tamanho da fonte: 16
Tipo de fonte: Arial
Cor da pena: 1 255 1
Cor do pincel: 0 0 128

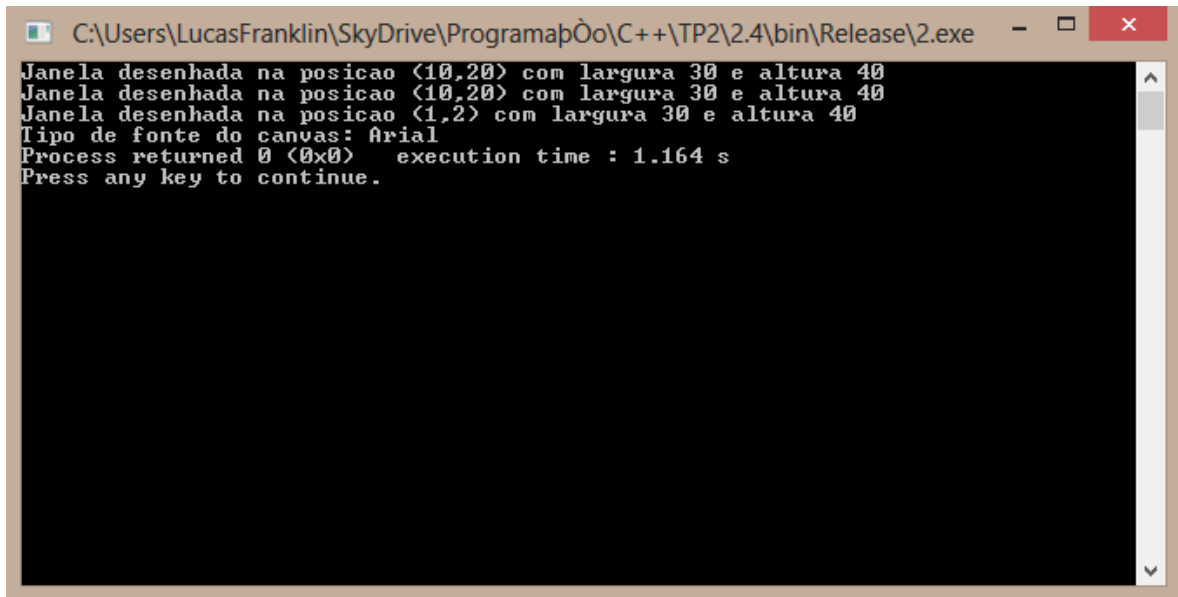
Process returned 0 (0x0)   execution time : 1.251 s
Press any key to continue.
```

Figura2: Teste-P2.3

O problema 2.3 propõe a implementação da classe CWindow visto em sala, a classe Canvas aninhada com CWindow e a classe COLOREF como COLOR.

A classe Canvas é uma classe aninhada com CWindow, sendo assim, apenas CWindow pode instanciá-la e acessar seus métodos e atributos públicos, por isso desenvolvemos nela, métodos que pudessem acessar métodos em Canvas, que por sua vez retornam os valores dos atributos da mesma.

2.3.4 – PROBLEMA 2.4



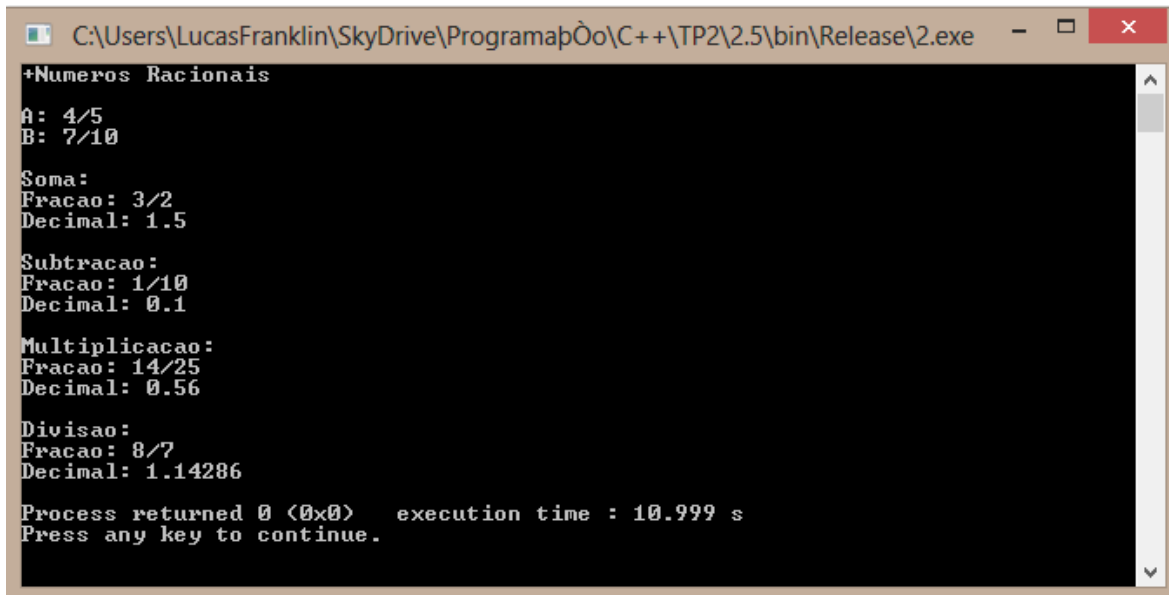
```
C:\Users\LucasFranklin\SkyDrive\Programapão\C++\TP2\2.4\bin\Release\2.exe
Janela desenhada na posicao <10,20> com largura 30 e altura 40
Janela desenhada na posicao <10,20> com largura 30 e altura 40
Janela desenhada na posicao <1,2> com largura 30 e altura 40
Tipo de fonte do canvas: Arial
Process returned 0 (0x0) execution time : 1.164 s
Press any key to continue.
```

Figura3: Teste-P2.4

A questão 2.4 propôs modificarmos a classe CWindow da questão 2.2 e transformá-la em um Singleton, criando uma nova classe (CWindowSingleton) que só deve ter uma instância.

Para desenvolver o Singleton usamos um ponteiro para a instância da classe, que é global e setado com NULL na primeira vez que é executado, ele é privado e do tipo CWindow. No programa de testes mostramos que podemos instanciar a classe e acessar os métodos sem precisar declarar um objeto próprio para isso. Mostramos também que o objeto não pode ser instanciado mais de uma vez, um ponteiro para a instância é criado para acessarmos os métodos.

2.3.5 – PROBLEMA 2.5



```
C:\Users\LucasFranklin\SkyDrive\Programas\Oo\C++\TP2\2.5\bin\Release\2.exe
+Numeros Racionais
A: 4/5
B: 7/10

Soma:
Fracao: 3/2
Decimal: 1.5

Subtracao:
Fracao: 1/10
Decimal: 0.1

Multiplicacao:
Fracao: 14/25
Decimal: 0.56

Divisao:
Fracao: 8/7
Decimal: 1.14286

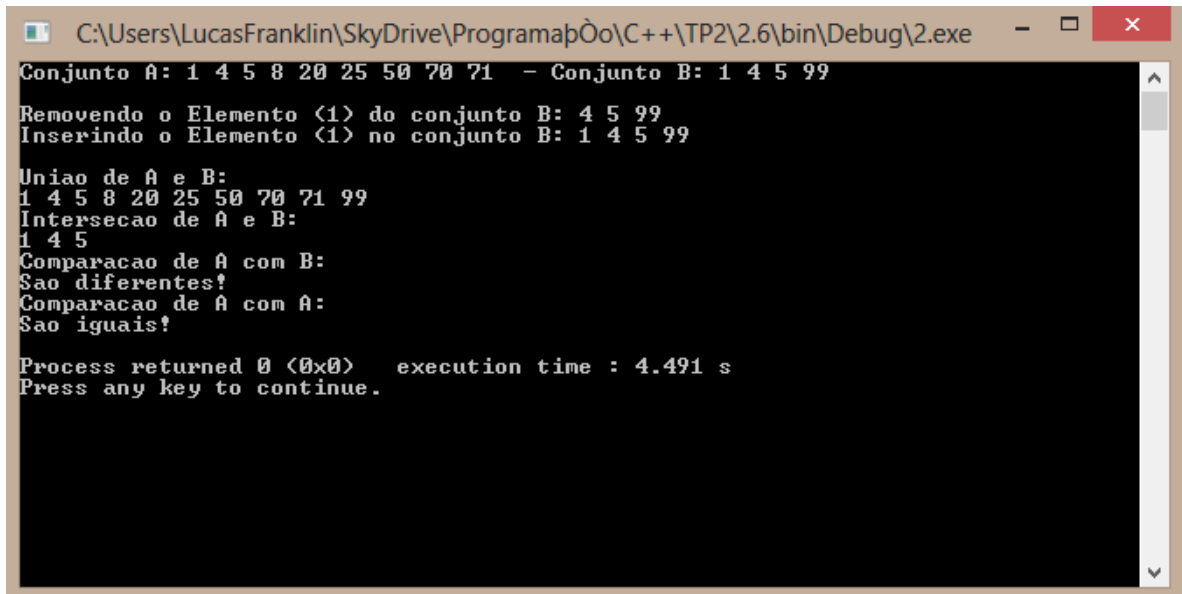
Process returned 0 (0x0)   execution time : 10.999 s
Press any key to continue.
```

Figura4: Teste-P2.5

O problema 2.5 propôs a criação de uma classe (Racional) que realize operações aritméticas com frações. Essa classe deve ter numerador e denominador como atributos privados e deve possuir um construtor que permita a inicialização de um objeto com valores default quando declarado. O programa deve permitir a adição, subtração, multiplicação e divisão de números racionais, além de permitir imprimi-los na forma a/b (numerador/denominador) e em ponto flutuante.

Como a adição e subtração de números racionais seguem o mesmo princípio, criamos apenas um método que faz esse tipo de operação, recebendo uma flag que determina o que será feito: adição ou subtração. Foram criados métodos de soma e subtração que apenas invocam o método de operação passando a flag correspondente. O mesmo foi feito para a multiplicação e divisão: como o princípio é o mesmo, apenas um método de multiplicação foi criado. Os métodos de multiplicação e divisão apenas invocam esse método e, no caso da divisão, passamos o denominador no lugar do numerador e vice-versa. Caso seja passado um denominador inválido para o construtor, uma mensagem de erro é retornada e é adotado o valor padrão 1. As operações são realizadas de forma que um objeto recebe o resultado da operação de outros dois.

2.3.6 – PROBLEMA 2.6

A screenshot of a Windows command prompt window titled "C:\Users\LucasFranklin\SkyDrive\Programap\Oo\C++\TP2\2.6\bin\Debug\2.exe". The window displays the output of a C++ program. The text is as follows:

```
Conjunto A: 1 4 5 8 20 25 50 70 71 - Conjunto B: 1 4 5 99
Removendo o Elemento <1> do conjunto B: 4 5 99
Inserindo o Elemento <1> no conjunto B: 1 4 5 99

Uniao de A e B:
1 4 5 8 20 25 50 70 71 99
Intersecao de A e B:
1 4 5
Comparacao de A com B:
Sao diferentes!
Comparacao de A com A:
Sao iguais!

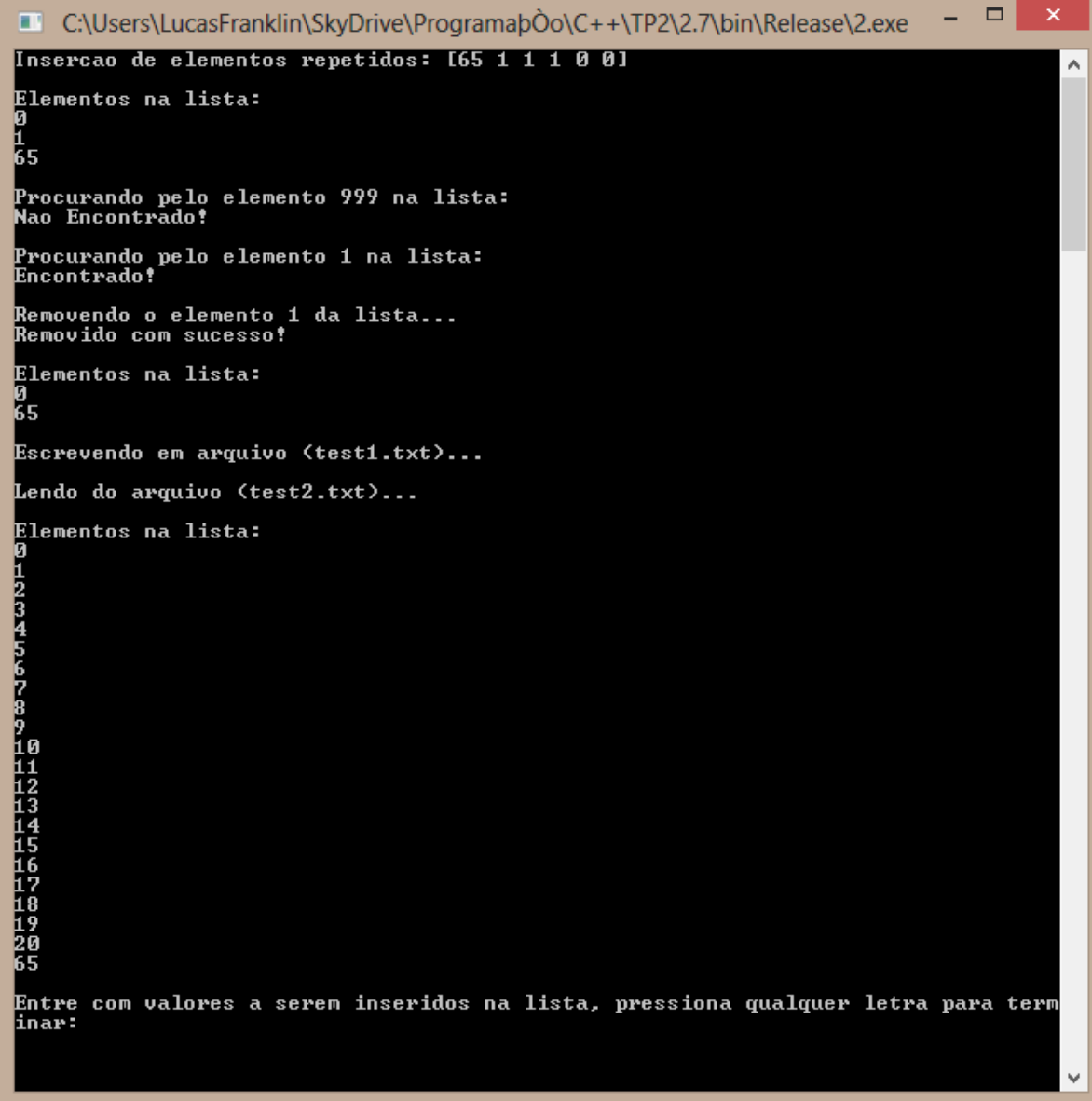
Process returned 0 (0x0)   execution time : 4.491 s
Press any key to continue.
```

Figura5: Teste-P2.6

No problema 2.6 foi proposto criar uma classe (IntegerSet), em que cada objeto pode armazenar inteiros de 0 a 100. O conjunto é representado por um array de bool, onde o elemento $a[i]$ é true se o inteiro i estiver no conjunto e false se não estiver. Um construtor deve inicializar o conjunto com todas as posições iguais a false e o programa deve permitir a união, interseção e impressão de conjuntos, bem como remover ou inserir elementos, inclusive lidos de arquivos, e comparar se dois conjuntos são iguais.

As operações de conjuntos nessa questão funcionam da seguinte maneira: um objeto recebe a união ou interseção de outros dois objetos. Para a comparação de arrays, usamos a função `equal()` do C++, ela recebe como parâmetro a posição inicial e final da primeira sequência e a posição inicial da segunda sequência, retornando verdadeiro se todos os elementos em ambas as cadeias forem iguais. Esse método da biblioteca padrão do C++ usa conceitos de iterators e templates que ainda não foram estudados, mas como é bastante simples, achamos válido aplicá-lo aqui.

2.3.7 – PROBLEMA 2.7

A screenshot of a Windows command prompt window titled "C:\Users\LucasFranklin\SkyDrive\Programapão\C++\TP2\2.7\bin\Release\2.exe". The window displays the output of a C++ program. The text is as follows:

```
Insercao de elementos repetidos: [65 1 1 1 0 0]
Elementos na lista:
0
1
65
Procurando pelo elemento 999 na lista:
Nao Encontrado!
Procurando pelo elemento 1 na lista:
Encontrado!
Removendo o elemento 1 da lista...
Removido com sucesso!
Elementos na lista:
0
65
Escrevendo em arquivo <test1.txt>...
Lendo do arquivo <test2.txt>...
Elementos na lista:
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
65
Entre com valores a serem inseridos na lista, pressiona qualquer letra para terminar:
```

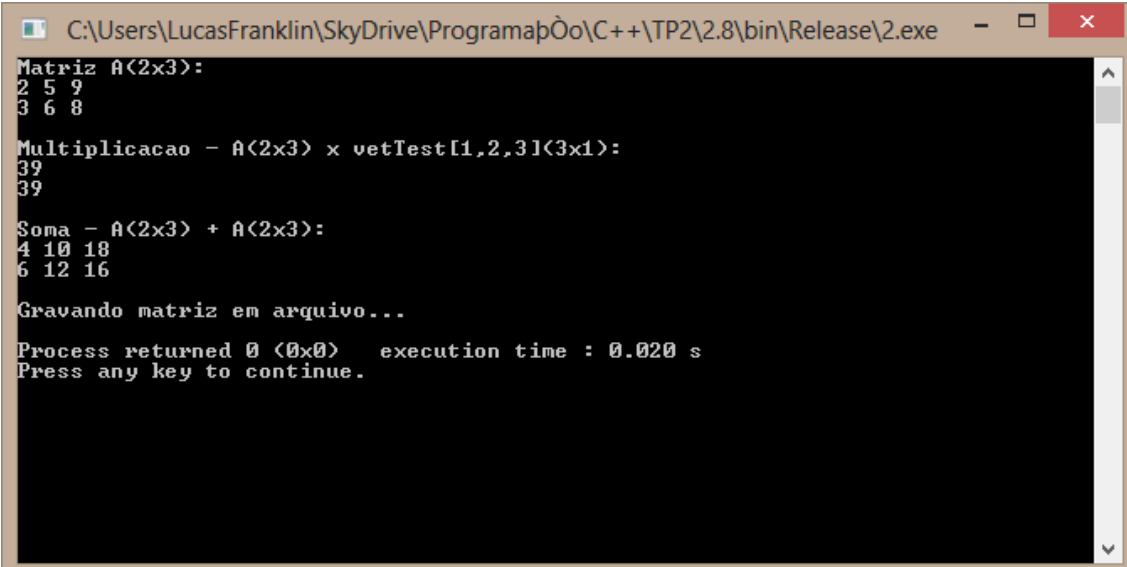
Figura6: Teste-P2.7

Para a questão 2.7, criar uma classe para uma lista encadeada de inteiros, que contenha um construtor que crie lista vazia e permita inserir novo elemento ordenadamente, buscar e remover um inteiro, caso esteja presente na lista e desalocar todos os nós no momento de sua destruição. O programa deve permitir, também, gravar elementos em uma ostream e ler elementos de uma ifstream.

Para impedir elementos repetidos na lista, usamos a busca binária na mesma, antes da inserção verificamos se já existe tal elemento. Para a exclusão de elementos usamos também a busca, ela procura o elemento e nos retorna sua posição, assim o

método responsável por remover posições é chamado. Essa lista é baseada na do professor Renê Veloso, que estudamos na disciplina de Algoritmos e Estruturas de Dados II. Usamos os tipos ostream e istream para o fluxo de entrada e saída, esses tipos podem receber por referencia qualquer objeto de entrada ou saída, no nosso caso foi o cout, cin, ofstream e ifstream. Para destruir a lista, primeiramente removemos todos os elementos e depois desalocamos os objetos Begin, End e Next, ao final do destrutor a função para impressão de elementos na tela é invocada para mostrar que não existe mais nenhum elemento na lista.

2.3.8 – PROBLEMA 2.8



```
C:\Users\LucasFranklin\SkyDrive\Programapão\C++\TP2\2.8\bin\Release\2.exe
Matriz A(2x3):
2 5 9
3 6 8
Multiplicacao - A(2x3) x vetTest[1,2,3](3x1):
39
39
Soma - A(2x3) + A(2x3):
4 10 18
6 12 16
Gravando matriz em arquivo...
Process returned 0 (0x0) execution time : 0.020 s
Press any key to continue.
```

Figura7: Teste-P2.8

No problema 2.8 foi proposto que criássemos uma matriz de double com ponteiros duplos, que pudesse ser alocada dinamicamente e destruída no fim do programa.

As operações com as matrizes funcionam da mesma maneira que nos exemplos que desenvolvemos que envolvem operações: um objeto recebe a soma de outras duas matrizes e, no caso da multiplicação, um objeto recebe a multiplicação de uma matriz com um vetor. O programa aceita que sejam criadas matrizes nulas, ou seja, que não têm nem linhas nem colunas. Esse tipo de matriz pode ser usada para receber os valores das operações entre duas matrizes. Antes de realizar qualquer operação, é verificado se a matriz do objeto que irá receber esse resultado já foi alocada, se sim, ela é desalocada

e realocada com as dimensões necessárias, seguindo as regras de operações com matrizes. Ao terminar o programa, o destrutor desaloca toda a matriz.

3 - CONCLUSÃO

Esse primeiro trabalho aplicando C++ na prática foi bem desafiador, principalmente em relação aos erros na hora da compilação. Mesmo sendo a linguagem C++ muito próxima da linguagem C, os problemas que nós enfrentamos foram bem diferentes, tivemos que reler muito o código para decifrar o que determinado erro estava tentando nos dizer. A cada questão que avançávamos, novos erros apareciam, mas no final conseguimos entender todos que ocorreram.

Com esse trabalho aprendemos muito mais da linguagem C++, aplicando o que já sabíamos na teoria, o que é um pouco diferente, e através das discussões entre a dupla e, ocasionalmente, com Walmir, integrante de outra equipe. Esse trabalho foi extremamente relevante para iniciarmos, verdadeiramente, a programação orientada a objetos e será muito importante para os próximos trabalhos e para o curso como um todo.

4 – BIBLIOGRAFIA

Cplusplus.com. [Online] [Citado em: 30 de 09 de 2013.] <http://www.cplusplus.com/>.

Stack Overflow. [Online] [Citado em: 30 de 09 de 2013.] <http://stackoverflow.com>.