UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO

LABORATÓRIO DE PROGRAMAÇÃO – CK0215 2023.1 T01A

TRABALHO 4: DICIONÁRIO VIA ÁRVORE AVL (VERSÃO 1: 2023-06-24)

PROFESSOR: PABLO MAYCKON SILVA FARIAS

1 Resumo

Você deverá implementar em C++ um dicionário baseado em árvore AVL, conforme a interface mencionada a seguir. A sua implementação será acoplada a um programa do professor, que realizará uma bateria de testes automatizados sobre as operações de dicionário e sobre a estrutura da árvore; caso seja detectado algum problema, o professor o informará para que seja feita uma correção (desde que dentro do prazo de submissão). A implementação será entregue ao professor através de tarefa no SIGAA.

Atenção: Colabore para o bom andamento da disciplina e do aprendizado de todos, não copiando códigos de quaisquer fontes nem compartilhando seu próprio código, nem mesmo em versão preliminar. Em caso de dificuldade sua ou com algum colega, fale com o professor.

2 Interface do Dicionário

A sua implementação deve seguir a interface presente no arquivo `interface_trabalho_4.hpp` disponibilizado no SIGAA. Você deve realizar a implementação sem modificar a interface, para que funcione o acoplamento com o código do professor que realizará testes automatizados sobre o seu dicionário e sobre a árvore AVL.

O uso de um dicionário com a interface em questão deve ser imediato dadas as explicações feitas nas aulas da disciplina e no próprio arquivo da interface, mas de qualquer forma segue um exemplo ilustrando um uso básico do dicionário:

```
// Este arquivo deve ser compilável através de um comando como:
//
//   g++ -Wall -Wextra -std=c++17 -pedantic teste.cpp

#include <exception>
#include <iostream>
using namespace std;

// Ao fazer a implementação, renomeie o arquivo para "solucao.hpp".
#include "solucao.hpp"

int main ()
  {
   try
     {
      DicioAVL<int,char> D;
```

```
      DicioAVL<int,char>::Iterador its[10], it;
      int i;

      for (i = 48; i < 58; ++i)
        {
         it = D.inserir(i, (char) i);  if (it == D.end()) return 2;

         its[i-48] = it;
        }

      for (it = D.begin(); it != D.end(); ++it)
        {
         cout << "O código de '" << it.valor()
              << "' é " << it.chave() << '\n';
        }

      for (i = 48; i < 58; ++i)
        {
         it = D.buscar(i);  if (it != its[i-48]) return 2;

         D.remover(it);
        }

      cout << "Executou o teste básico conforme esperado.\n";
     }
   catch (const exception &e)
     {
      cerr << "Exceção: " << e.what() << '\n';  return 1;
     }
  } // main
```

Observe que o programa acima não garante que a sua implementação está executando inteiramente conforme esperado. Em particular, observe que o programa não percorre a árvore do dicionário; isso será feito pelo programa de testes do professor.

3 Compilação e Submissão da Solução

A sua implementação deve ser escrita em C++17 padrão. Você deverá completar o arquivo da interface, renomeando-o para `solucao.hpp` e incluindo sua identificação no início do arquivo:

```
// Nome:      ...
// Matrícula: ...
```

Eventuais comentários sobre a sua solução do trabalho podem também ser feitos no arquivo (nesse caso, colocá-las no início pode ser uma boa ideia).

O professor incluirá o seu código num programa de testes, a ser compilado através de uma linha como:

```
g++ -Wall -Wextra -std=c++17 -pedantic  main.cpp
```

A compilação do seu código deve ocorrer sem erros e preferencialmente também sem avisos. Observe que opções de compilação como `-Wall -Wextra -std=c++17 -pedantic` nos ajudam

a encontrar erros de programação. Procure utilizá-las ao realizar o trabalho.

Atente ao fato de que a sua solução para o trabalho consiste apenas na implementação do dicionário `DicioAVL`, e não em um programa inteiro. Naturalmente, você precisará testar o seu dicionário antes de submetê-lo, e portanto deverá escrever uma função `main` para fazer os testes, como ilustrado pelo programa da seção anterior; entretanto, para evitar conflito com o código do professor, você não deve enviar a sua função `main` no arquivo da solução do trabalho, ou então deve comentar tudo o que não seja a implementação do dicionário.

Você deve submeter sua solução para o trabalho através da tarefa cadastrada pelo professor no SIGAA. O professor tentará testar a sua solução e dar um retorno até o dia útil seguinte, de forma que você possa tentar corrigir sua implementação e ressubmetê-la, desde que dentro do prazo.

Importante: Não deixe para submeter a sua solução apenas no fim do prazo, pois, se isso acontecer, é possível que não haja tempo para o professor lhe dar um retorno e você corrigir a sua implementação.

Em caso de dúvida, contate o professor rapidamente.

– Que você tenha uma prática rica em aprendizados. Bom trabalho! –