Exercício 27

Defina uma classe Instituição na qual trabalham pessoas. Defina as operações de adição, procura e remoção de um Aluno e Professor à instituição.

ColecaoHibrida.h

```
#pragma once
#include<set>
template<class T>
class less_pointers{
public:
    bool operator()(const T &left, const T &right) const{
         return (*left < *right);
    }
};
template<class K>
class ColecaoHibrida: public std::set<K, less_pointers<K>>{
public:
    bool insert(const K &c);
    K find(const K &c);
    int size() const;
    void erase(const K &);
    //void clear();
    //bool empty() const;
    //iterator begin();
    //iterator end();
};
```

ColecaoHibrida.h (continuação)

```
template<class K>
bool ColecaoHibrida<K>::insert(const K &c){
   pair<typename set<K, less_pointers<K>>::iterator, bool> r;
   r=set<K, less_pointers<K>>::insert(c);
   return(r.second);
}
template<class K>
K ColecaoHibrida<K>::find(const K &c){
   K r=0;
   typename set<K, less_pointers<K>>::iterator i;
   i=set<K, less_pointers<K>>::find(c);
   if(i!=set<K, less_pointers<K>>::end()) r=*i;
   return(r);
}
template<class K>
int ColecaoHibrida<K>::size() const{
   return((int)set<K, less_pointers<K>>::size());
}
template<class K>
void ColecaoHibrida<K>::erase(const K &c ){
   set<K, less_pointers<K>>::erase(c);
}
```

classe Pessoa

```
class Pessoa{
   string nome;
public:
   Pessoa(const string &n):nome(n){}

   virtual void Print(){ cout << nome << endl; }

   virtual bool operator<(const Pessoa &p) const{
       return nome < p.nome;
   }
};
```

classe Aluno

```
class Aluno: public Pessoa{
   int no_mec;
public:
   Aluno(const string &n, int no): Pessoa(n) {no_mec=no;}

   void Print(){
       Pessoa::Print();
       cout << no_mec << endl;
   }
};
```

classe Professor

```
class Professor: public Pessoa{
   string categoria;
public:
   Professor(const string &n, const string &c)
       :Pessoa(n), categoria(c){}

   void Print(){
       Pessoa::Print();
       cout << categoria << endl;
   }
};
```

classe Instituicao

```
#include"ColecaoHibrida.h"

class Instituicao{
   ColecaoHibrida<Pessoa*> pessoas;
public:

  bool addAluno(const string &nome, int num){
       Pessoa *a=new Aluno(nome, num);
       return pessoas.insert(a);
  }

  bool addProfessor(const string &nome, const string &cat){
       Pessoa *p=new Professor(nome, cat);
       return pessoas.insert(p);
  }
  ...
```

classe Instituicao (continuação)

```
  Pessoa *findPessoa(const string &nome){
        Aluno p(nome, 0); //ou Professor p(nome, ""); ou Pessoa p(nome);
        return pessoas.find(&p);
  }

  bool remPessoa(const string &nome){
        Pessoa *p=findPessoa(nome);
        if(p!=NULL){
          pessoas.erase(p);
          delete p;
          return true;
        }else return false;
  }

  void PrintPessoas() const{
        cout<<"todas as Pessoas da Instituicao:\n";
        ColecaoHibrida<Pessoa*>::iterator i;
        for(i=pessoas.begin(); i!=pessoas.end(); i++) (**i).Print();
  }
};
```

main

```
...

void main(){
  Instituicao estig;
  estig.addAluno("Luis", 5555);
  estig.addProfessor("Antonio", "Prof. Adjunto");
  estig.addAluno("Miquelino", 6666);
  estig.PrintPessoas();

  cout<<"----------------\n";
  cout<<"Sera que ha algum Antonio?\n";
  Pessoa *p=estig.findPessoa("Antonio");
  if(p!=NULL) p->Print();
  else cout<<"Nao\n";
  cout<<"----------------\n";

  estig.remPessoa("Luis");
  estig.PrintPessoas();
}
```

```
todas as Pessoas da Instituicao:
Antonio
Prof. Adjunto
Luis
5555
Miquelino
6666
```

```
----------------
Sera que ha algum Antonio?
Antonio
Prof. Adjunto
----------------
```

```
todas as Pessoas da Instituicao:
Antonio
Prof. Adjunto
Miquelino
6666
```