Classe Abstrata Poligono

- Atributos: lados (vetor de elementos do tipo double)
- Construtor único (que inicializa os lados do polígono) e Destrutor
- Método getLado(int) que retorna o tamanho do *i-ésimo* lado
- Método getPerimetro() que retorna o perímetro (soma dos lados)
- Método getNroLados() que retorna o número (quantidade) de lados
- Método abstrato double getArea() a ser implementado pelas subclasses
- Sobrecarga do operador <<
- Método void imprime() – que imprime número de lados, perímetro e área do polígono (utilize o operador << implementado anteriormente)
- Método estático static bool comparaArea(Poligono* p1, Poligono* p2) – Comparação de 2 polígonos: retorna true se p1 é "menor" que p2 e false, caso contrário. Deve-se levar em consideração primeiramente a área. Se mesma área compara-se pelo perímetro.
- Método estático static bool comparaNroLados(Poligono* p1, Poligono* p2) – Comparação de 2 polígonos: retorna true se p1 é "menor" que p2 e false, caso contrário. Deve-se levar em consideração primeiramente o número de lados. Se mesmo número de lados compara-se pelo perímetro.

```
#ifndef POLIGONO_H
#define POLIGONO_H

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

class Poligono {
public:
    Poligono(vector<double>&);
    virtual ~Poligono();
    double getLado(int) const;
    double getPerimetro() const;
    virtual double getArea() const = 0;
    virtual void imprime() const ;
    friend ostream& operator<<(ostream&, const Poligono&);

    static bool comparaArea(const Poligono* p1, const Poligono* p2);
    static bool comparaNroLados(const Poligono* p1, const Poligono* p2);
private:
    vector<double>& lados;
};

#endif /* POLIGONO_H */
```

Classe Triângulo (**subclasse de** Poligono

- Método double getArea() – que retorna a área do triângulo.
  $A = \sqrt{S * (S - lado_1) * (S - lado_2) * (S - lado_3)}$, onde $S = \frac{Perimetro}{2}$

```
#ifndef TRIANGULO_H
#define TRIANGULO_H

#include "Poligono.h"

class Triangulo : public Poligono {
public:
    Triangulo(vector<double>&);
    virtual ~Triangulo();
    virtual double getArea() const;
private:
};

#endif /* TRIANGULO_H */
```

Classe Retangulo (**subclasse de** Poligono)

- Método double getArea() – que retorna a área do triângulo.
  $A = lado_1 * lado_2$

```
#ifndef RETANGULO_H
#define RETANGULO_H

#include "Poligono.h"

class Retangulo :public Poligono {
public:
    Retangulo(vector<double>&);
    virtual ~Retangulo();
    virtual double getArea() const;
private:
};

#endif /* RETANGULO_H */
```

Programa Principal

Crie o arquivo main.cpp com o seguinte conteúdo:

```
#include <iostream>
#include <algorithm> // std::sort
#include <vector>    // std::vector
#include "Poligono.h"
#include "Triangulo.h"
#include "Retangulo.h"

int main() {

    vector<Poligono *> poligonos;

    vector<double> v1{3, 4, 5};
    poligonos.push_back(new Triangulo(v1));

    vector<double> v2{3, 4, 3, 4};
    poligonos.push_back(new Retangulo(v2));

    vector<double> v3{3, 3, 3};
    poligonos.push_back(new Triangulo(v3));

    vector<double> v4{2, 3, 2, 3};
    poligonos.push_back(new Retangulo(v4));

    cout << "poligonos:" << endl;

    for (unsigned long int i = 0; i < poligonos.size(); i++) {
        cout << *poligonos[i] << endl;
    }

    cout << endl << "poligonos (ordenado pela Area):" << endl;

    sort(poligonos.begin(), poligonos.end(), Poligono::comparaArea);

    for (unsigned long int i = 0; i < poligonos.size(); i++) {
        poligonos[i]->imprime();
    }

    cout << endl << "poligonos (ordenado pelo Nro Lados):" << endl;

    sort(poligonos.begin(), poligonos.end(), Poligono::comparaNroLados);

    for (unsigned long int i = 0; i < poligonos.size(); i++) {
        poligonos[i]->imprime();
    }

    return 0;
}
```