

CPP – CR : Template

Robin DELEPINE – Romain POURIAS

SEC 2024 : 08-12-2022

TP : Template

OBJECTIF :

Manipuler des classes, de travailler sur la surcharge d'opérateurs et de mettre en œuvre le polymorphisme et utilisant des Template.

PARTIE 1 : Code Point.h

```
1  #pragma once
2  #include <iostream>
3  #include <vector>
4  using namespace std;
5
6  //-----
7  // declaration de la classe
8  //-----
9
10 template<typename T>
11 class point
12 {
13 protected:
14     T x = 0;
15     T y = 0;
16 public :
17     point();
18     point(T x, T y);
19     void translater(T a, T b);
20     point(point const & pt);
21     T getX();
22     T getY();
23     void setX(T a);
24     void setY(T b);
25 };
26
27 //-----
28 // declaration des fonctions de la classe
29 //-----
30
31 template<typename T>
32 inline point<T>::point()
33 {
34     this->x = 0;
35     this->y = 0;
36 }
37
38 template<typename T>
39 inline point<T>::point(T x, T y)
40 {
41     this->x = x;
42     this->y = y;
43 }
44
```

```
1  #pragma once
2  #include <iostream>
3  #include <vector>
4  using namespace std;
5
6  //-----
7  // declaration de la classe
8  //-----
9
10 template<typename T>
11 class point
12 {
13 protected:
14     T x = 0;
15     T y = 0;
16 public :
17     point();
18     point(T x, T y);
19     void translater(T a, T b);
20     point(point const & pt);
21     T getX();
22     T getY();
23     void setX(T a);
24     void setY(T b);
25 };
26
27 //-----
28 // declaration des fonctions de la classe
29 //-----
30
31 template<typename T>
32 inline point<T>::point()
33 {
34     this->x = 0;
35     this->y = 0;
36 }
37
38 template<typename T>
39 inline point<T>::point(T x, T y)
40 {
41     this->x = x;
42     this->y = y;
43 }
44
```

PARTIE 2 : Forme.h

```

1  #pragma once
2  #include "point.h"
3  template<typename S>
4  class forme
5  {
6  protected:
7      vector<point<S>> vectorPt;
8  public:
9      forme();
10     forme(point<S> pt);
11     vector<point<S>> getVecteurPoint();
12     void setVecteurPoint(vector<point<S>> vector);
13     void addPoint(point<S> pt);
14
15     virtual S perimetre() = 0;
16     virtual S surface() = 0;
17 };
18
19 template<typename S>
20 inline forme<S>::forme()
21 {
22 }
23
24 template<typename S>
25 inline forme<S>::forme(point<S> pt)
26 {
27     this->addPoint(pt);
28 }
29
30
31

```

```

32 template<typename S>
33 inline vector<point<S>> forme<S>::getVecteurPoint()
34 {
35     return this->vectorPt;
36 }
37
38 template<typename S>
39 void forme<S>::setVecteurPoint(vector<point<S>> vector)
40 {
41     this->vectorPt = vector;
42 }
43
44 template<typename S>
45 inline void forme<S>::addPoint(point<S> pt)
46 {
47     this->vectorPt.push_back(pt);
48 }
49
50 template<typename S>
51 inline ostream& operator<<(ostream& s, forme<S> f)
52 {
53     for (int i = 0; i < f.getVecteurPoint().size(); i++)
54     {
55         s << "pt "<< i << " : " << f.getVecteurPoint()[i] << "\n";
56     }
57     return s;
58 }
59

```

PARTIE 3 : rectangle.h

```

1  #pragma once
2  #include "forme.h"
3  template<typename R>
4  class rectangle : public forme<R>
5  {
6  protected :
7      R hauteur;
8      R largeur;
9  public:
10     rectangle();
11     rectangle(point<R> pt);
12     R perimetre();
13     R surface();
14     R getLargeur();
15     R getHauteur();
16
17     void setLargeur(R l);
18     void setHauteur(R h);
19 };
20
21 template<typename R>
22 inline rectangle<R>::rectangle()
23 {
24 }
25
26 template<typename R>
27 inline rectangle<R>::rectangle(point<R> pt):forme<R>(pt)
28 {
29 }
30
31 template<typename R>
32 inline R rectangle<R>::perimetre()
33 {
34     return 2 * this->hauteur + 2 * this->largeur;
35 }
36
37 template<typename R>
38 inline R rectangle<R>::surface()
39 {
40     return this->hauteur * this->largeur;
41 }
42
43 template<typename R>
44 inline R rectangle<R>::getLargeur()
45 {
46     return this->largeur;
47 }
48

```

```

49 template<typename R>
50 inline R rectangle<R>::getHauteur()
51 {
52     return this->hauteur;
53 }
54
55 template<typename R>
56 inline void rectangle<R>::setLargeur(R l)
57 {
58     this->largeur = l;
59 }
60
61 template<typename R>
62 inline void rectangle<R>::setHauteur(R h)
63 {
64     this->hauteur = h;
65 }
66
67 template<typename R>
68 inline ostream& operator<<(ostream& s, rectangle<R> r)
69 {
70     for (int i = 0; i < r.getVecteurPoint().size(); i++)
71     {
72         s << "pt " << i << " : " << r.getVecteurPoint()[i] << "\n";
73     }
74     s << "largeur " << r.getLargeur() << "\n";
75     s << "hauteur " << r.getHauteur() << "\n";
76     return s;
77 }
78

```