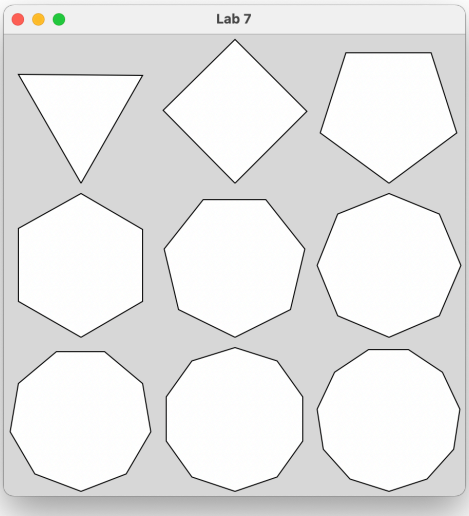
INFO-F-202
Langages de programmation 2
Université libre de Bruxelles
Laboratoire 7

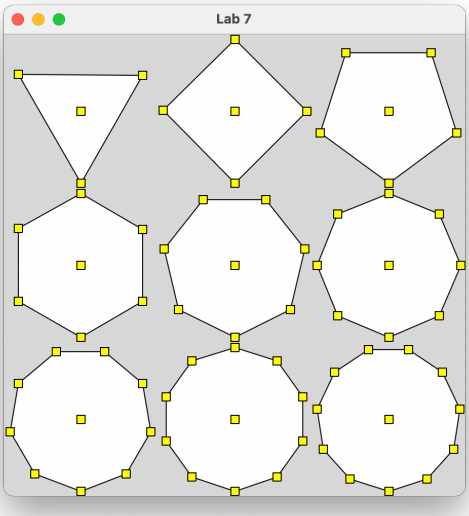
Justin Dallant John Iacono
Yannick Molinghen Alexis Reynouard

1 But

Itérateurs !

2 Sommaire





Exécutez le code. Cela devrait vous donner l'image de gauche.

Décommentez le code dans `Canvas::draw`. Ajoutez du code à `Polygon` pour que cela vous donne l'image de droite.

N'ajoutez aucun code à `Canvas::draw` autre que décommenter. Ne modifiez pas `Polygon::draw`. N'ajoutez du code qu'à `Polygon`,

3 Code fourni

On vous donne une classe `Polygon` :

```
class Polygon {
  const vector<Point> vertexes;
  Fl_Color fillColor, frameColor;
  Point center;
public:
  Polygon(const vector<Point> &vertexes,
          Point center,
          Fl_Color frameColor = FL_BLACK,
          Fl_Color fillColor = FL_WHITE);
  void draw() const;
};
```

Vous devrez ajouter à cette classe mais vous n'aurez pas besoin de modifier le code déjà là-bas. Le constructeur prend un vecteur de points, un centre et des couleurs, et a d'une méthode de dessin. Cette classe peut vous être utile dans votre projet.

Dans la classe Canvas, il existe une variable d'instance `shapes` qui est un vecteur de polygones.

Le constructeur de la classe `Canvas` ajoute neuf polgons à `shapes`. La méthode `draw` de `Canvas` dessine chaque polygon dans `shapes` :

```
void Canvas::draw() {
  for (auto &s: shapes) {
    s.draw();
}
```

4 Tâches

Votre devez décommenter le code dans `Canvas::draw` qui donne :

```
void Canvas::draw() {
  const int r=4;
  for (auto &s: shapes) {
    s.draw();
    for (const Point &p:s)
      Polygon{{{p.x+r,p.y+r},{p.x-r,p.y+r},{p.x-r,p.y-r},{p.x+r,p.y-r}},p,
          FL_BLACK,FL_YELLOW}.draw();
  }
}
```

Observez qu'il y a deux boucles for :

```
  for (auto &s: shapes) {
    ....
    for (const Point &p:s)
```

La première boucle est sur s, qui est un vecteur de polygones.

La deuxième boucle fait une boucle sur le polygone qui donne des `Points` et dessine un petit carré jaune sur chacun. Mais que signifie boucler sur un polygone ? Vous devriez le savoir depuis la leçon sur l'itérateur.

Vous devrez créer une classe d'itérateur, appelez-la `Polygon::Iterator` (une classe interne appelée `Iterator` à l'intérieur de la classe `Polygon`). Cette classe aura besoin d'un constructeur, `operator ++`, `operator *`, `operator ->` et `operator ==`. Vous devrez ajouter des méthodes `begin` et `end` à la classe `Polygon` qui renvoient des instances de votre classe `Iterator`.

Dans un premier temps, vous pouvez essayer de le faire fonctionner sans le point central.

Après avoir fait dessiner le point central aussi sans changer la boucle dans `Canvas::draw`.

5 Iterator

Vous auriez dû regarder la vidéo sur les itérateurs. Si vous l'avez fait, vous devriez comprendre ce que vous devez faire pour que la boucle for sur le polygone fonctionne.

Voici un très bref reprise.

```
for (auto &x:A)
```

est un raccourci pour

```
for (auto &x=begin(A);x!=end(A);++x)
```

`begin(A)` appelle juste `A.begin()` et `end(A)` appelle `A.end()`. Donc ce qui précède est le même que :

```
for (auto &x=A.begin();x!=A.end();++x)
```

Quel est le type de `x` ? `x` a le type renvoyé par les méthodes `begin` et `end` sur `A`, et ce type est appelé un itérateur. L'itérateur permet logiquement de parcourir tout le contenu de `A`, et possède une implémentation interne qui lui permet de le faire.

Notez qu'un itérateur n'est qu'une terminologie pour toute classe que vous écrivez et qui peut être utilisée pour faire fonctionner une boucle for. Il est distinct du type de `A`, et est généralement une classe interne du type de `A` appelée `Iterator`. Il doit prendre en charge les opérations suivantes :

- `++x`.Déplace logiquement l'itérateur vers l'élément suivant dans `A`.
- `*x` (et `x->`) Obtenez l'élément dans `A` vers lequel `x` pointe logiquement.
- `x==y` (et `x!=y`) `x` et `y` font-ils référence au même élément dans `A` ?

De plus, quel que soit le type de `A`, il doit avoir les méthodes `begin` et `end` qui renvoient des itérateurs. La méthode `begin` doit renvoyer un itérateur vers le premier élément, et la méthode `end` doit renvoyer un itérateur vers le dernier élément.

Veuillez regarder l'exemple de code pour l'itérateur pour la classe de pile de la leçon 18 pour un exemple élaboré.