Laboratoire 32, analyse empirique de la complexité algorithmique

- **But**
  - Sensibilisation à la complexité algorithmique
  - Dans le laboratoire 31, on peut constater que l'effort de calcul pour générer des nombres premiers augmente considérablement avec leur longueur.
  - On peut donc légitimement se demander s'il faudra plutôt un jour ou une année de calcul pour générer un nombre premier à 2000 chiffres.
  - L'analyse du temps de calcul est compliquée, car il est difficile de prévoir combien de nombre aléatoire il faudra générer avant de tomber sur un nombre premier.

Travail à réaliser

- Mesurer le temps mis par votre programme pour générer une dizaine ou une vingtaine de nombres premiers à $n = 20$ chiffres.
  - La mesure du temps de calcul peut se faire en important la bibliothèque `<chrono>`
  - Le code suivant permet ensuite de mesurer le temps :

```
using namespace chrono;
high_resolution_clock::time_point t1 = high_resolution_clock::now();
   // ici: le code dont on veut mesurer le temps
duration<double> time_span =
  duration_cast<duration<double>>(high_resolution_clock::now() - t1);
cout << time_span.count() << endl;
```

- Recommencer plusieurs fois la mesure du temps en multipliant chaque fois n par 1.1
- À l'aide d'une feuille de calcul, créer un diagramme donnant l'effort de calcul en fonction de n. Avec des échelles logarithmiques pour n et pour le temps, un algorithme polynomial est une droite, asymptotiquement.
- La pente de cette droite donne le degré du polynôme. On peut l'estimer dans une feuille de calcul en insérant une « courbe de tendance » de type « puissance » et en demandant d'en afficher l'équation.
- Connaissant l'équation, il est facile d'estimer le temps pour générer un nombre premier à 2000 chiffres.

- **À rendre :** Une feuille de calcul avec le diagramme comportant l'équation de la fonction estimant le temps de calcul.