

Laboratoire 31, génération de clés de cryptage

• Buts

- Implanter une classe permettant de manipuler des entiers signés de taille arbitraire

• Travail à réaliser

- Dans un premier temps, compléter au besoin la classe `Uint`, programmée au laboratoire 30, pour qu'elle puisse faire fonctionner le programme de cryptage—décryptage `rsa.cpp`. Indications :
 - L'opérateur de transformation de type `Uint` \rightarrow `uint64_t` peut éventuellement être déclaré `explicit` pour continuer à bénéficier des conversions implicites opérées avec un constructeur, dans des expressions comme `variable_Uint + 1`.
 - Pour activer le bit `badbit` d'un flux d'entrée `is`, par exemple lorsqu'on n'arrive pas à lire correctement un `Uint` avec l'opérateur surchargé `>>`, on peut appeler la méthode `is.clear(std::ios::badbit | is.rdstate())`.
- Créer une classe `Sint` permettant la manipulation d'entiers signés de taille arbitraire. Indications :
 - Cette classe peut comprendre comme données membre un `Uint` ainsi qu'un signe.
 - Elle peut être déclarée amie dans `Uint`, pour pouvoir bénéficier de ses opérateurs.
 - Penser à surcharger l'opérateur unaire `-`, notamment pour réutiliser efficacement certains opérateurs de la classe `Uint`.
 - Il n'y a que quelques opérateurs qui prennent plus d'une ligne de code ; le plus long à programmer nécessite moins de 10 lignes.
- Utiliser cette classe dans un programme, à rendre également, générant des clés de cryptage publique et privée dont la taille est spécifiée par l'utilisateur.

• Délai

- Mardi 18 janvier 2022, 8h15 ; le laboratoire donnera lieu à une évaluation.