TP n°1

Exercice n°1

Ecrire en C++ un programme qui affiche les entiers impairs non divisibles par 3.

Exercice n°2

Demander à l'utilisateur de rentrer un texte puis comptez le nombre de lettres, de chiffres ou d'autres caractères. Afficher le résultat à l'écran.

Exercice n°3

Ecrivez un programme qui permet à l'aide du triangle de Pascal d'afficher le développement de $(x+y)^n$. On utilisera la notation $\mathrm{x\^{}i}$ pour symboliser x^i. On se servira pour le calcul du triangle de pascal de la classe $\mathrm{std::vector}$.

Pour aller plus loin : essayez de minimiser la place mémoire occupée par le programme

Exercice n°4

Générez une suite aléatoire d'entiers qu'on triera en les insérant au fur et à mesure dans une structure représentant un arbre de trie binaire dont un nœud contient un entier et deux pointeurs uniques sur deux nœuds suivant.

Afficher ensuite les entiers générés stockés dans l'arbre dans l'ordre croissant.

Pour aller plus loin : Regarder la bibliothèque random de la bibliothèque standard pour générer une suite aléatoire suivant une loi uniforme puis gaussienne.

Exercice n°5

Créer une liste de nombre allant de 2 à N (N défini par l'utilisateur) puis utiliser le crible d'Eratosthène pour ne conserver dans la liste que les nombres premiers.

Afficher la liste à l'écran.

Exercices avancés

Exercice n°6

Le but de cet exercice est de créer un quadtree afin d'accélérer la recherche dans un nuage de point du point le plus proche d'une coordonnée donnée.

1. On définit un point comme une paire de réel (on utilisera à cet usage $\mathrm{std::pair}$ proposé par la bibliothèque standard)
2. On génère aléatoirement un tableau de pairs de réels compris entre -100 et 100 en abscisse et ordonnée. Ce tableau représentera un nuage de points.
3. On crée ensuite un quadtree à l'aide de l'algorithme suivant :
    a. Un calcul une boîte englobant le nuage de points
    b. On subdivise en quatre parties égales (en découpant en deux dans chaque direction) la boîte englobante
    c. On trie par sous-boîte via un tableau d'indices les points du nuage
    d. Pour chaque sous-boîte on recommence à l'étape b jusqu'à n'avoir dans les dernières sous-boîtes qu'une dizaine de points.

Ecrire ensuite une fonction permettant de retrouver à l'aide du quadtree le point le plus proche d'une coordonnée donnée.

Ecrire une fonction n'utilisant pas le quadtree permettant de trouver le point le plus proche d'une coordonnée donnée (à l'aide d'une méthode « brute »)

Comparer le temps pris par les deux fonctions pour trouver une coordonnée rentrée par l'utilisateur.