Explication des structures de données utilisées

Structure List

Il s'agit de la structure qui va recueillir la liste des tronçons que l'utilisateur veut, ainsi que la liste des barres dont il dispose. Cette structure de données est adaptée à ses deux utilisations dans la mesure où une fois une barre ou un tronçon utilisés, il faut pouvoir les retirer de la liste de manière peu coûteuse. La List 120, 895, 525 voudrait dire slon le cas : "L'utilisateur a besoin d'un tronçon de 120 mm, un de 895 et un autre de 425 mm", ou bien "L'utilisateur dispose d'une barre de 120 mm, une de 895 et une autre de 425 mm"

Les fonctionalités que propose cette structure de données sont également adaptées à cette utilisation : 1. tri de la liste par orde croissant. 2. Suppression d'un élément par position : supprimer le troisième élément par exemple 3. Suppression d'un élément par valeur : supprimer le premier tronçon qui vaut 125.2 4. Différentes fonctions de test sur la présence d'une valeur dans la liste ou non, sur la longueur de la liste, sur la plus grande valeur, la plus petite etc.

Structure Paire

Il s'agit de l'association d'une longueur de tronçon, et de sa position dans la liste générale des tronçons. La paire (450, 3) désigne un tronçon de 450mm qui se trouve en troisième position de la liste des demandes de l'utilisateur. Il est nécessaire d'associer aux tronçons des index pour les besoins de l'algorithme de combinatoire. En effet, imaginons que l'utilisateur a demandé 4 tronçons et qu'un maximum de 3 tronçons peut tenir dans une barre, le moteur doit méthodiquement créer les combinaisons suivantes 123, 124, 134 et 234. Il a donc besoin d'une certaine manière, de travailler sous forme d'index, d'où la structure de données Paire qui associe un index à chaque tronçon pour pouvoir former des combinaisons.

Problématique soulevée

On a donc mis en évidence que la structure list traitait les tronçons sous forme de liste triée, c'est à dire sans indice, et qu'elle est susceptible d'en supprimer, décallant et embrouillant par là même toute tentative de travail par indice. Pour résoudre ce problème, on introduit un vecteur m_copieTroncons qui contiendra le même contenu que la liste des tronçons. On s'en servira pour travailler par indice sur le vecteur, et la fonction copie se charge de faire le va-et-viens entre l'approche liste et l'approche vecteur.

Structure Combinaison

Il s'agit d'une liste de Paire, qui représente une des combinaisons possibles de tronçons sur une certaine barre, associé d'un rendement de la coupe.

Structure MoteurCalculs

Il s'agit de *la* super structure, très artificielle, d'ailleurs, puisqu'elle regroupe dans ses attributs pas mal de notions différentes telles que la liste des tronçons demandés, la taille des barres, le nombre de barres, une liste de Combinaisons qui rassemble toutes les possibilités avec tous les rendements calculés, une liste de Combinaisons qui contient le résultat final, une certaine exigence de rendement qui permet de ne pas utiliser inutilement de la mémoire pour les rendements beaucoup trop petits. Cette exigence diminuera d'elle-même si aucune des possibilités offertes n'y répond.