TP 5 : Programmation modulaire

IUT CAEN – Département Informatique

Module M1103

Exercice 1

Dans cet exercice nous allons nous intéresser à la compilation conditionnelle. Nous allons voir comment la compilation conditionnelle peut remédier aux problèmes de dépendances cycliques entre modules. Pour cela, vous allez reprendre l'exercice du jeu des cailloux.

1. Créez les fichiers suivants: `joueur.h`, `joueur.c`, `machine.h`, `machine.c`, `gest_partie.h`, `gest_partie.c` et `principal.c`.
   - Dans `joueur.h` sera définit le prototype de la fonction `utilisateur_joue` qui demande le nombre de cailloux et quel joueur commence le premier. Le corps de cette fonction sera réalisé dans `joueur.c`.
   - Dans `machine.h` sera définit le prototype de la fonction `machine_joue` qui permet à la machine de prendre un nombre de cailloux et qui met à jour le nombre de cailloux restants. Le corps de cette fonction sera réalisé dans `machine.c`.
   - Dans `gest_partie.h` seront définis les prototypes des fonctions `initialiser` et `change_t -our_jeu`. Les corps de ces fonctions seront rśalisés dans `gest_partie.c`
   - Le fichier `joueur.h` doit dépendre du fichier `machine.h` et réciproquement.
   - Le fichier `principal.c` devra dépendre des fichiers `joueur.h`, `machine.h`, `gest_partie.h` et implémentera la boucle principale du jeu des cailloux.
2. Écrivez la Makefile qui permet de compiler l'ensemble de ces fichiers. Que constatez-vous ?
3. Remédiez à ce problème en utilisant la compilation conditionnelle.

Exercice 2

Le but de l'exercice est d'implémenter une structure de type vecteur ainsi que différentes fonctions permettant de réaliser diverses opérations sur les vecteurs.

1. Définissez dans un fichier nommé `vecteur.h` un type vecteur composé de trois rééls, ainsi que les prototypes des fonctions `init_vecteur` qui initialise un vecteur, et `coord_1`, `coord_2`, `coord_3` qui renvoient respectivement la première, deuxième et troisième coordonnée du vecteur qui leur est passé en paramètre. Les coprs de ces fonctions seront réalisés dans le fichier `vecteur.c`.
2. Définissez dans un fichier nommé `norme.h` le prototype de la fonction `norme` prenant en argumant un vecteur et renvoyant sa norme. Le corps de cette fonction sera réalisé dans le fichier `norme.c`.
3. Définissez dans un fichier nommé `produits.h` les prototypes des fonctions `produit_scalaire` et `produit_vectoriel` permettant de calculer respectivement le produit scalaire et vectoriel. de deux vecteurs passés en argument. Les corps de ces fonctions seront réalisés dans le fichier `produits.c`.
4. Écrivez dans un fichier nommé `principal.c` le programme principal qui, à partir de deux vecteurs, calcule la norme de leur produit vectoriel.
5. Écrivez le Makefile qui permet de compiler l'ensemble de ces fichiers.

Rappels

Notons par $\overrightarrow{v} = (v_1,v_2,v_3)$ et $\overrightarrow{u} = (u_1,u_2,u_3)$ les coordonnées respectives des vecteurs $\overrightarrow{v}$ et $\overrightarrow{u}$.

- Norme : $|| \overrightarrow{u} || = \sqrt{u_1^2 + u_2^2 + u_3^2}$
- Produit scalaire: $\overrightarrow{u} \times \overrightarrow{v} = \frac{1}{2}(||\overrightarrow{u}||^2 + ||\overrightarrow{v}||^2 - ||\overrightarrow{v} - \overrightarrow{u}||^2) = u_1 * v_1 + u_2 * v_2 + u_3 * v_3$
- Produit vectoriel: $\overrightarrow{u} \wedge \overrightarrow{v} = (u_2 * v_3 - u_3 * v_2, u_3 * v_1 - u_1 * v3)$