Travaux dirigés n°12 (2h15): Structures: Déclaration et utilisation

Objectifs

- Comprendre comment déclarer et manipuler une structure.

Exercice 1: Les étudiants

Les propriétés relatives à un étudiant sont son nom (maximum 30 caractères), son prénom (maximum 20 caractères), son groupe (une structure), sa moyenne générale (un float), et ses notes (maximum 40 notes (float)).
Un groupe est une structure composée de son nom (deux caractères), de sa salle de cours (4 caractères) (à chaque groupe est affecté une salle de cours).

a) Donner les structures *groupe* et *étudiant*.

On suppose que les groupes sont stockés dans un tableau.

a) Écrire en C la fonction *rechercheGr*. Son prototype en algorithmique est **int fonction rechercheGr (→struct groupe tGr[], →int tailleTGr, → char nomGr[]) ;** cette fonction recherche si le groupe de nom *nomGr* existe dans le tableau de groupe *tGr* de taille effective *tailleTGr* et retourne sa position dans le tableau s'il existe et –1 sinon. On suppose que tous les groupes du tableau *TGr* ont un nom unique (différent du nom des autres groupes).

b) Écrire en C la procédure *initEtudiant* qui initialise toutes les champs d'un étudiant :
les valeurs des nom, prénom sont demandées à l'utilisateur.
le nom du groupe est demandé à l'utilisateur tant que le groupe n'existe pas (filtrage).
la moyenne ainsi que toutes les valeurs du tableau de notes sont initialisées à –1.
Le prototype en algorithmique de la procédure *initEtudiant* est :
procédure initEtudiant(←struct etudiant etu,→struct groupe tGr[], →int tailletGr) ;
etu est l'étudiant à initialiser, *tGr* est le tableau des groupes et *tailleGr* est la taille du tableau *tGr*.

c) Écrire en C une procédure *initTabEtudiant* qui initialise un tableau d'étudiants. Son prototype en algorithmique est :
procédure initTabEtudiant(←struct etudiant tabEtu, →int tailleTabEtu, →struct groupe tGr[], →int tailletGr) ; *tabEtu* est le tableau d'étudiants et *tailleTabEtu* la taille effective de ce tableau.

d) Écrire une procédure *ajoutNote* qui permet d'ajouter une note à un certain indice dans le tableau des notes d'un étudiant (la note et l'indice dans le tableau sont en paramètres d'entrée).

e) Écrire en C une fonction qui calcule la moyenne générale d'un étudiant, qui initialise le champ *moyGene* de cet étudiant et qui renvoie cette moyenne (la moyenne est calculée à partir des notes qui sont différentes de -1).

Exercice 2: Les polygones

Un polygone est constitué d'un ensemble de points (par exemple : 5 points pour un pentagone, 8 points pour un octogone, …).
Un polygone peut être constitué de 100 points au maximum.
Ainsi par exemple le polygone constitué des points (1,3) (2,4) (5,3) (6,2) (3,1) est un pentagone.
Écrire les procédures/fonctions en C dont les prototypes en langage algorithmique sont:

int fonction pointEgal(→struct point p1,→struct point p2) qui retourne 1 si les deux points *p1* et *p2* ont les mêmes valeurs 0 sinon.

procédure afficherPoly(→struct polygone pol1) qui affiche les coordonnées des points successifs constituants le polygone *pol1*.

struct polygone fonction initPoly() qui initialise et retourne un polygone en demandant les valeurs successives des points le constituant.

struct polygone fonction concatPoly(→struct polygone pol1,→struct polygone pol2) qui construit un nouveau polygone par concaténation de *pol1* et *pol2* (le premier point de *pol2* devient le point suivant du dernier point de *pol1* dans le nouveau polygone construit).

struct polygone fonction soustrPoly(→struct polygone pol1,→struct polygone pol2) qui construit un nouveau polygone par soustraction de *pol1* dans *pol2*. Cette soustraction consiste à retirer les points de *pol1* qui apparaissent dans *pol2*.

struct polygone fonction interPoly(→struct polygone pol1,→struct polygone pol2) qui construit un nouveau polygone par intersection de *pol1* et de *pol2*. Cette intersection consiste à garder les points commun de *pol1* et de *pol2* dans l'ordre où ils apparaissent dans *pol1*.

procédure deplPoly(↔struct polygone pol1,→int deplX,→int deplY) qui déplace le polygone *pol1* en déplaçant chacun des points qui constitue le polygone de deplX pour l'abscisse et de deplY pour l'ordonnée.