NF01 - Automne 2005

Examen Final - 2 heures

Documents Interdits

ATTENTION !
Utilisez quatre copies séparées, une par problème

Problème n°1 (5 points) : Procédures, fonctions et tableaux

L'objectif est de calculer la surface de polygones fermés. Un polygone est codé sous la forme d'une suite de points formant les segments consécutifs des polygones.

1) Nous commencerons par mettre au point une fonction qui calcule la surface d'un trapèze.
Soit deux points A et B de coordonnées respectives (x1,y1) et (x2,y2), nous définissons la surface du trapèze sous le segment par la formule suivante : Surface=(x2-x1)*(y1+y2)/2. Cette formule ne devra pas être modifiée.

Q1 (0,5 pt) : écrire la fonction qui calcule la surface selon la formule ci-dessus, et dont l'en-tête est : function surf_trapez(x1,y1,x2,y2 :real) :real ;

Q2 (0.5 pt) : Calculer la surface S1 du trapèze donné par les points A=(x1=1, y1=1), et B=(x2=2,y2=2). Soit S1 :=surf_trapez(1,1,2,2);

Q3 (0.5 pt) : Calculer la surface S2 du trapèze donné par les points B=(x1=2,y1=2) et C=(x2=1,y2=3). Soit S2 :=surf_trapez(2,2,1,3); Cette surface est négative, c'est normal bien que cela semble absurde de prime abord.

Q4 (0.5 pt) : Calculer S=S1+S2. Au signe près est-ce que S correspond à la surface du triangle (A,B,C) ? Dessinez les points, et hachurez les surfaces S1 et S2.

2) Un polygone est défini sous la forme d'un tableau de n lignes et de deux colonnes. Voici le début du programme pascal :

```
Program calcule_surface_polygone;
Const MAX_NB_POINTS=32;
Type polygone= array[1..MAX_NB_POINTS,1..2] of real ;
var
   poly : polygone;
   nb_points_utiles : integer ;
```

Une ligne i correspond aux coordonnées du point numéro i. Pour une ligne i, poly[i,1] donne la coordonnée sur l'axe des x et poly[i,2] donne la coordonnée sur l'axe des y.

Deux points consécutifs d'indice i et i+1 dans le tableau définissent le segment entre les points i et i+1.

Le dernier segment qui ferme le polygone est **implicitement** défini entre le point numéro nb_points et le point numéro 1. On remarquera que les déclarations ci-dessus limitent le nombre de segments à 32.

La variable nb_points_utiles correspondra au nombre de points réellement utiles présents dans le tableau. On supposera que nb_points_utiles est inférieur ou égal à MAX_NB_POINTS.

Q5 (1,5 pts) : écrire la fonction qui calcule la surface d'un polygone en parcourant consécutivement ses segments. L'en-tête sera obligatoirement :
function surf_polygone(pol : polygone, nbpt : integer) :real ;
Le paramètre pol contiendra le polygone et nbpt le nombre de points du polygone.

Q6 (0,5 pt) : Appliquer cette fonction avec nb_points_utiles=4 et le polygone suivant :

1	1
2	2
1	3
0	2

Au signe près vous avez la bonne réponse si votre fonction est correcte.

Q7 (1 pt) : Vous avez pu remarquer que le sens de parcours des segments influe sur le signe du résultat, c'est-à-dire le signe de la surface. Si les segments sont définis dans le sens trigonométrique en parcourant le polygone alors la surface calculée est négative. S'ils sont définis dans le sens des aiguilles d'une montre alors la surface sera positive.
Ecrire une procédure reverse qui inverse le sens de parcours. L'en tête en sera obligatoirement :
Procedure rev_pol(var pol_reverse: polygone; pol: polygone; nbpt:integer);

Pol_reverse contiendra le polygone résultat, pol le polygone d'origine et nbpt le nombre de points du polygone d'origine.

Problème n°2 (5 points) : Fichiers et enregistrements

Le but de cet exercice est de proposer un « Gestionnaire de carnet d'adresses ». Chaque entrée dans ce gestionnaire est donnée sous la forme d'une chaîne de caractères qui contient les informations suivantes : nom, prénom, téléphone, e-mail. Lors de la saisie d'un contact les différentes informations sont séparées par des virgules. Tous les contacts sont sauvegardés dans un fichier « texte » (<u>un contact par ligne</u>).

Votre programme doit comporter les fonctionnalités suivantes :
- création d'un fichier (type texte) « Carnet d'adresse »
- saisie d'un contact
- consultation du fichier
- recherche d'un contact
- suppression d'un contact
- affichage des contacts à l'écran sous la forme suivante :

NOM	**PRENOM**	**TELEPHONE**	**E-MAIL**
Dupont	Gérard	09870923	gdupont@utc.fr

Problème n°3 (5 points) : Récursivité

1 Écrire une fonction récursive ‘add(x,y)’ qui fasse l'addition de 2 nombres entiers naturels x et y. Vous ne pouvez utiliser les opérateurs ‘+’ et ‘-‘ que si l’un des deux opérandes est 0 ou 1.

2 Écrire ensuite une autre fonction récursive qui multiplie deux nombres entiers naturels n et m. Vous devez utiliser la première fonction ‘add(x,y)’ définie précédemment.

3 Un palindrome est une séquence S de lettres dont les lectures de gauche à droite puis de droite à gauche donnent le même résultat, par exemple : "sas", "elle",etc . Supposons que S soit représentée par un tableau de n caractères. Ecrire en Pascal une fonction récursive renvoyant la valeur 1 si la séquence S est palindromique et 0 dans le cas contraire.

Problème n°4 (5 points) : Cours hors poly et TD

Q1 (1 pt) La moyenne olympique d'un ensemble de nombres est obtenue en faisant la moyenne des nombres qui restent, une fois supprimés le plus grand et le plus petit. Ecrire un programme **en C** qui lit une série de nombres et en affiche la moyenne olympique. L’algorithme a déjà été vu en TD.

Q2 (2 pts) Ajout et suppression d’éléments dans une PILE et dans une FILE. Pensez à gérer les débordements par défaut (structure vide) et excès (structure pleine) dans la pile et la file.

0.5 pt : Ecrire en Pascal la procédure PUSHPILE permettant d’ajouter un élément dans une pile

0.5 pt : Ecrire en Pascal la procédure POPPILE utilisée pour retirer un élément d’une pile

0.5 pt : Ecrire en Pascal la procédure PUSHFILE qui ajoute un élément dans une file

0.5 pt : Ecrire en Pascal la procédure POPFILE servant à retirer un élément d’une file. Ces piles et files seront implantées par contiguïté, sous forme de tableaux.

Q3 (2 pts) Nous souhaitons gérer une liste de personnes (nom, prénom, age, téléphone) implantée par chaînage (avec des pointeurs). La liste sera triée par ordre alphabétique sur le nom. Le début de la liste chaînée sera indiqué par le pointeur TETE.

1 pt : Ecrire en Pascal une procédure AJOUTE permettant d’insérer une nouvelle personne après la cellule pointée par le pointeur PTR, passé en paramètre.

1 pt : Ecrire ensuite la procédure EFFACE utilisée pour supprimer un élément placé après une cellule pointée par le pointeur PTR, passé en paramètre.