

Représentation de polygones par listes chaînées
Programmation C L2.1
Listes de points

Nous allons travailler dans une fenêtre de taille 512 par 512 pixels et avec des listes de points . Nous allons utiliser la structure suivante, classique pour les points. Dans le but de stocker des polygones de taille variables, nous allons utiliser une liste chaînée de points. Ces structures sont rassemblées dans un fichier **polygone.h**.

L'intégralité du code produit de ce TP devra former un unique programme, dont vous veillerez à ce que l'utilisation soit agréable.

```
#define TAILLE_X    512
#define TAILLE_Y    512
typedef struct {
    int x;
    int y;
} Point ;

typedef struct cellulePoint {
    Point p;
    struct cellulePoint *suivant;
} CellulePoint ,*ListePoint ;
```

1. Dessiner un polygone pas à pas.

- (a) Écrivez un programme qui déclare une **ListePoint** vide, crée une fenêtre, l'affiche, puis la détruit (après avoir attendu un temps raisonnable, ou un clic de souris). À chaque question suivante, vous n'oublierez pas de mettre à jour le programme principal pour qu'il appelle (et donc teste) les fonctions réalisées.
- (b) Écrivez une fonction **ListePoint allouerCellule(Point pval)** qui alloue l'espace mémoire nécessaire pour une nouvelle cellule d'une liste de points et retourne l'adresse de cette nouvelle cellule après avoir affecté le point **pval** au champ **p** et **NULL** au champ **suivant**. Si il n'y a plus de place disponible, la fonction renvoie la valeur **NULL**.
- (c) Écrivez une fonction **int insererEnTete(ListePoint* liste, Point val)** qui insère le point **val** au début de la liste de points *** liste**. La fonction renvoie 0 en cas de problème et 1 sinon.
- (d) Écrivez une fonction **int EntrerPolygone(ListePoint* lst)** qui à chaque clic de l'utilisateur ajoute le point entré à une liste de points, affiche la ligne entre le point entré et le point précédent, ceci jusqu'à ce que l'utilisateur appuie sur une touche du clavier. Le programme renvoie 1 si la liste de points a été correctement créée.
- (e) Écrivez une fonction **void libererListe(ListePoint* liste)** qui libère tout l'espace mémoire occupé par la liste ***liste**. La liste devra être **NULL** après l'appel

2. Découper un polygone.

- (a) Écrivez une fonction **ListePoint extraireListeBasGauche(ListePoint* liste, Point p)** qui extrait de la liste des points contenus dans **liste** ceux qui sont situés en bas et à gauche du point **p** et en forme une nouvelle liste.

- (b) De manière similaire, écrivez les fonctions `ListePoint extraireListeBasDroite(ListePoint* liste, Point p)`, `ListePoint* extraireListeHautGauche(ListePoint* liste, Point p)` et `ListePoint* extraireListeHautDroite(ListePoint* liste, Point p)`.
- (c) Modifiez votre programme précédent pour qu'après avoir rentré son polygone, l'utilisateur rentre un point à la souris et que les 4 ensembles des points situés de chacun des quatre rectangles dessinent des polygones de quatre couleurs différentes.

3. Recoller deux polygones.

- (a) Écrivez une fonction `void concatenerListes(ListePoint* origine, ListePoints *ajout)` qui concatène la liste `ajout` à la fin de `origine`. la liste `*ajout` sera vide après l'appel
- (b) Modifiez votre programme précédent pour qu'après un nouveau clic de souris, il y ait maintenant deux polygones : un au-dessus du point cliqué à la question précédente, et un au-dessous.

4. Tri, le retour.

- (a) Écrivez une fonction `void deplacerEnTete(ListePoint * liste, Point p)` qui déplace la première occurrence du point `p` en tête de la liste `*liste`.
- (b) Écrivez une fonction `Point plusPres(ListePoint* liste, Point p)` qui renvoie le point le plus près de `p` présent dans `liste`.
- (c) Écrivez une fonction `void deplacerPlusPresEnTete(ListePoint* liste, Point p)` qui déplace en tête de `liste` le point le plus près de `p`.
- (d) Écrivez une fonction qui place en tête le point le plus près de l'origine, puis en deuxième le point le plus près du premier, en troisième le point le plus près du deuxième, etc.
- (e) Modifiez votre programme pour trier les listes en fonction de cet ordre avant chaque affichage.