

**TP7**  
**Programmation C L2.1**  
**Comptage des mots dans un fichier**

---

Dans cette séance de travaux pratiques, nous utiliserons des structures modélisant divers objets géométriques. On utilise la librairie graphique MLV comme au TP 5 (il est conseillé de réutiliser le makefile du TP5).

La fenêtre sera toujours de même taille, on définira donc deux valeurs constantes LONGUEUR et LARGEUR en début de fichier. On pourra par exemple travailler avec une fenêtre de taille 640\*480.

Dans le TP nous travaillerons avec des droites, par soucis de simplicité, nous ne considérerons pas de droites verticales.

---

### 1. Structure point

- Écrire une structure `Point` permettant de modéliser un point à coordonnées entières dans le plan.
- Écrire une fonction `void sasiePoint(Point * p)` attendant un clic de l'utilisateur pour ensuite affecter à l'adresse du point donné en paramètre les coordonnées du clic.

### 2. Structure droite

- Écrire une structure `Droite` modélisant une droite d'équation  $y = ax + b$ .
- On souhaite créer une fonction construisant une droite à partir de deux points, pour cela écrire la fonction `int construireDroite(Point a, Point b, Droite * d)` stockant dans la droite `d` l'équation de la droite passant par les deux points. Le retour de la fonction indiquera si tout c'est bien passé ou si la droite était verticale.
- Écrire une fonction `void dessineDroite(Droite d)` dessinant dans la fenêtre la droite donnée en paramètre.
- En utilisant les questions précédentes, on souhaite écrire un programme demandant à l'utilisateur de cliquer en deux points de la fenêtre et traçant la droite passant par ces deux points. Si les deux points forment une droite verticale, le programme se terminera en marquant un message d'erreur dans le terminal.
- Écrire une fonction `int intersectionDroite(Droite alpha, Droite beta, Point * p)` calculant le point d'intersestion des deux droites. On utilisera le retour de la fonction pour différencier suivant les trois cas suivant :
  - tout c'est bien passé (on a stocké les coordonnées du point à l'adresse `p`);
  - Le point n'était pas dans la fenêtre (on a quand même stoqué les coordonnées du point à l'adresse `p`);
  - Les droites sont parallèles.

### 3. Structure polygône

- Définir une structure `Polygone` possédant deux champs : un pour stocker sa taille et l'autre pour stocker l'ensemble de ses points.  
(On supposera qu'un polygône a toujours au plus `TAILLE_MAX = 20` sommets)
- En utilisant la fonction `MLV_draw_filled_polygon`, écrire une fonction `void affichePolygone(Polygone p)` permettant d'afficher `p`.
- On souhaite que l'utilisateur puisse construire un polygône en cliquant dans la fenêtre à la position de ses sommets. Étant donné que le programme ne sait pas combien de sommets aura le polygône, on utilisera la fonction `MLV_wait_keyboard_or_mouse` qui renvoie le `MLV_Event` : `MLV_MOUSE_BUTTON` si l'utilisateur a cliqué avec la souris ou `MLV_KEY` si l'utilisateur a appuyé sur une touche du clavier. On ajoutera donc le point au polygône si l'utilisateur clique et on arrêtera la saisie s'il appuie sur une touche quelconque du clavier ou si le nombre maximum de sommet est atteint. Attention de respecter que la taille maximum d'un polygône est égale à `TAILLE_MAX`.

- Écrire une fonction `int effaceSommet(Polygone* p, int i)` qui modifie `p` en retirant son  $i$ -ième sommet si possible. On avisera qu'on ne peut retirer le  $i$ -ième sommet si  $i$  est plus grand que le nombre de sommets et qu'on ne peut pas retirer de sommet si le polygône a moins de trois sommets.
- Écrire un programme demandant à l'utilisateur de cliquer dans la fenêtre pour définir un polygône. Le programme devra dans un second temps effacer à chaque seconde un sommet aléatoire du polygône jusqu'à ce qu'il ne reste plus que trois sommets.  
(On utilisera la fonction `void MLV_clear_window(MLV_Color color)` pour ne pas garder l'affichage de l'ancien polygône)