TP 6 VARI 1

Informations techniques PC Suse :

(a) Pour démarrer une session : utilisateur **licencep** et mot de passe **7002n***. Vous trouverez : une `icône lézard`  en haut à droite pour accéder au menu.

(b) Pour démarrer *Processing* : clic sur `l'icône lézard` en haut à droit → Dévelopemment → Processing.

(c) Pour démarrer un *terminal* : `l'icône lézard` → Terminal → Konsole.

(d) Pour ouvrir un gestionaire/navigateur de fichiers : `l'icône lézard` → Système → Dolphin, ou cliquer sur «Dossier Personnel» en haut à gauche.

(e) Pour modifier un fichier, clic droit sur le fichier → Ouvrir avec Kate

1 Commandes Linux et un petit programme C

Démarrer une console/terminal en suivant les instructions au point (c) ci-dessus.
Démarrer un gestionnaire de fichiers en suivant les instructions au point (d) ci-dessus.
Vous allez vous servir à la fois du terminal et du gestionnaire graphique de fichiers.

Exercice 1 Taper une commande dans le terminal pour créer un dossier où vous allez déposer tous vos fichiers à utiliser aujourd'hui. Indication : utiliser `mkdir` (un acronyme pour l'anglais `make directory`), n'hésitez pas à regarder les exemples à la fin du Cours 3-4 sur le systèmes sur le site :
`cedric.cnam.fr/~porumbed/vari1`

Exercice 2 Taper une commande `cd` (un acronyme pour `change directory`) pour se placer dans le dossier crée au premier exercice.

Exercice 3 Taper `touch monPremierProgC.c` pour faire un fichier vide qu'on va bientôt modifier pour écrire un programme `C` de quelques lignes.

Exercice 4 Aller dans le gestionnaire de fichiers et chercher le dossier que vous avez crée à l'exercice 1. Cliquer sur ce dossier pour trouver `monPremierProgC.c` à l'intérieur. Clic droit sur `monPremierProgC.c` et choisir `Ouvrir avec` → `Kate`.

Exercice 5 Écrire le programme suivant à l'intérieur de la fenêtre de l'éditeur `Kate` et sauvegarder.

```
void main(){
    int maFortune = 100;
    if(maFortune>100000)
        printf("Je suis riche");
    else
        printf("Pas si riche");
}
```

Exercice 6 Utiliser le terminal pour taper
`gcc -w monPremierProgC.c -o exec`

Cette commande permet de compiler le programme C, ignorez les "warning". Lancer le programme `exec` via la commande :
`./exec`

Exercice 7 Modifier le programme (et re-compiler) pour le faire afficher `''Je suis riche''`.

Exercice 8 La commande ci-dessous permet de chercher tous les fichiers dont le nom commence avec `gimp` dans le dossier `/usr/`. Modifier cette commande pour chercher tous les fichiers dont le nom commence avec `javac`.
`find /usr -name "gimp*"`

Exercice 9 Taper la commande `pwd` pour connaitre/afficher le dossier courant. Essayer de mémoriser cette commande utile, c'est un acronyme pour `print working directory`.

2 Processing

Exercice 1 Traduire en `processing` le programme de l'exercice 5 plus haut, mais initialiser `maFortune` à une valeur aléatoire entre 1 et 200000.

Exercice 2 Soit le code ci-après. Corriger deux erreurs de compilation et une erreur de logique !

```
int note1;note2;          //déclaration de variables
note1=19;                 //affectation de valeur
note2=14;
note3=14;
int min=note3;            //on commence le calcul de la note minimale
if(min>note2)
    min=note2;
if(min>note3)
    min=note3;
if(min<10)
    println("échec: vous avez au moins une note inférieure à 10");
else
    println("succès: vous avez validé toutes les UEs avec des notes >=10");
int note3;
```

Exercice 3 Modifier le exercice précédent pour utiliser un tableau `notes` avec trois cases. La déclaration et l'initialisation du tableau sont données ci-après, n'hésitez pas à regarder les exemples du cours 4, disponible en ligne.

```
int[] notes = new int[3];
notes[0] = 19;
notes[1] = 14;
notes[2] = 14;
```

Exercice 4 Remplir (au début) le code ci-après pour afficher un cercle rouge centré au pixel de coordonnées (50,50).

```
.....//à remplir
tab[3]=40;
fill(250,0,0);
ellipse(tab[0],tab[1],tab[2],tab[3]);
```

Exercice 5 La fonction `quad(...)` permet de tracer un quadrilatère. Tester par exemple un programme qui contient une seule ligne : `quad(0,0,100,100,50,90,10,50);`. Remplir le programme ci après pour faire un nouveau quadrilatère, avec les sommets positionnés comme vous le souhaitez.

```
int x[] ....//coordonnées x à remplir
int y[] ....//coordonnées y à remplir
fill(250,0,0);
quad(x[0],y[0],x[1],y[1],
     x[2],y[2],x[3],y[3]);
```

Exercice 6 Utiliser `fill(...)` et `stroke()` pour faire le contour du quadrilatère en rouge et le fond en bleu.

Exercice 7 Utiliser deux appels à la fonction `line(...)` pour tracer les diagonales du quadrilatère.

Exercice 8 Soit deux tableaux x et y de 6 cases initialisés au début du programme, comme dans le code ci-dessous. Tracer un hexagone rempli avec les sommets/coins aux coordonnées indiquées par x et y, c. à. d., le premier sommet est placé à $(x[0], y[0])$, le 2ème à $(x[1], y[1])$, le 3ème à $(x[2], y[2])$. etc.
Indication : utiliser des appels à la fonction `quad(...)`.

```
size(500,500);
int x[] = {100, 200, 400, 500, 400, 200};
int y[] = {273, 100, 100, 275, 446, 446};
```

Exercice 9 On considère une somme d'argent `fortune` déposée sur cinq ans sur un livret rémunéré avec des intérêts. Les taux d'intérêts sur les cinq ans sont donnés dans un tableau d'entiers `taux` avec 5 cases. Par exemple, si `taux`[0]=5 alors le taux d'intérêt sur la première année est de 5%=0.05. Calculer la somme finale générée à la fin des 5 années. Les intérêts sont **capitalisés**, c.à.d., à la fin de chaque année, **les intérêts générés** sont **ajoutés au capital** pour produire de nouveaux intérêts.
Note : Vous pourriez avoir besoin de transformer les entiers en réels. Pour cela on utilise un `cast` : si x est une variable entière et y est réelle, alors `y=(float)x/100` réalise la conversion (ou le `cast`) entier→réel avant de faire le calcul. Si on ne faisait pas ce `cast`, l'affectation `y=x`/100 serait évaluée à 0 pour tout $x < 100$.