Université de Versailles – Saint-Quentin-en-Yvelines 2019-2020
Licence 1

IN 100 – TD

Version du 19 septembre 2019

Franck Quessette – Franck.Quessette@uvsq.fr
Sandrine Vial – Sandrine.Vial@uvsq.fr

Table des matières

1 TD 1: Introduction, déclaration de variable, affectation, opérateurs, utilisation de fonctions graphiques

Prise en main

Pour pouvoir faire les TDs, il vous faut :

1. Un ordinateur, soit emprunté à la BU, soit le votre avec la machine virtuelle installée.
2. Ce poly qui est sur e-campus dans l'espace IN100.
3. Éventuellement la doc de l'environnement graphique qui est aussi sur e-campus.
4. Vous devez utiliser la machine virtuelle "Lubuntu Linux"ou "Debian Linux".

Créer un fichier pour chaque exercice et nommez le fichier en utilisant le numéro d'exercice, par exemple `exo17.c` pour l'exercice numéro 17.

À la fin de la séance, si vous utilisez un ordinateur emprunté à la BU vous devez :

1. Sauvegarder ce que vous avez fait sur une clé USB (non fournie).
2. Éteindre la machine virtuelle en appuyant sur le bouton ON/OFF de l'ordinateur, ce qui vous ramène au menu de choix des machines virtuelles.
3. Éteindre l'ordinateur en appuyant sur le bouton ON/OFF de l'ordinateur.

Exercices

Le but de cet ensemble d'exercices est de se familiariser avec l'environnement de programmation et d'afficher des objets simples.

Exercice 0 *Vocabulaire*

Rappel sur le vocabulaire :

— Environnement : Dossier, Fichier, Éditeur, Compiler, Exécuter.

— Programmation : Variable, Type, Instruction, Affectation, Appel de Fonction.

Exercice 1 *Premier Programme*

Dans l'éditeur geany, le code contenu dans le fichier `exo1.c` est :

Attention sur la première ligne il peut y avoir `#include "graphics.h"` ce qui est incorrecte, rajouter `uvsq` devant `graphics`.

```
#include "uvsqgraphics.h"

int main()
{
/* Déclarer les variables ci-dessous */
POINT p1;
POINT p2;

/* Initialisation de la fenêtre graphique */
init_graphics(900,600);

/* Tapez votre code ci-dessous */
p1.x = 100; p1.y =  90;
p2.x = 200; p2.y = 300;
draw_fill_rectangle(p1, p2, bleu);

p1.x =  10; p1.y = 10;
p2.x = 400; p2.y = 500;
draw_line(p1,p2,rouge);
```

```
p1.x = 300; p1.y = 350;
draw_circle(p1,100, magenta);
/* Fin de votre code */


/* Permet d'attendre un appui sur Echap avant de fermer la fenêtre graphique */
wait_escape();
exit(0);
}
```

1. Cliquer sur Compiler.
2. Cliquer sur Executer.
3. Comprendre le lien entre le code et l'affichage.

Exercice 2 *Programmes*
En utilisant les fonctions `draw_line`, `draw_rectangle`, `draw_fill_rectangle`, `draw_circle` ou `draw_fill_circle` dessiner les dessins suivants :

1. Une ligne horizontale et 3 cercles : un centré sur l'extrémité gauche de la ligne, le deuxième centré au milieu de la ligne, le troisième centré sur l'extrémité droite de la ligne.
2. 5 lignes horizontales les unes au dessus des autres espacées de 100 points (de couleurs différentes).
3. Un carré de côté 100 points dont le coin en bas à gauche est aux coordonnées $(20, 20)$ à l'aide de la fonction `draw_rectangle`.
4. Un carré de côté 100 points dont le coin en bas à gauche est aux coordonnées $(200, 200)$ dont les quatre côtés sont de couleurs différentes.
5. Une croix (style $\times$) formée de deux segments, insérée dans un carré de côté 50 et dont le centre est en $(120, 170)$.
6. Dessiner un hexagone et son cercle circonscrit.

Exercice 3 *Quadrillage*

1. Faites une fenêtre graphique de 600 par 400.
2. Dessinez un quadrillage avec les lignes espacées de 100 (`draw_line`). Utilisez le copier-coller de l'éditeur.
3. Faites une fenêtre graphique de 400 par 400.
4. Dessiner un plateau de jeu d'échecs (8×8).

Exercice 4 *Gestion de la souris*
Utiliser la fonction `wait_clic` qui renvoie les coordonnées du point qui a été cliqué par la souris :

```
POINT p;
p = wait_clic();
```

Notez bien que quand le programme exécute la fonction `wait_clic` il attend que l'utilisateur clique avant de continuer.

1. Attendre un clic et dessiner un cercle de rayon 100 là où l'utilisateur a cliqué.
2. Répéter 10 fois.
3. Même question, mais effacer le cercle précédent.

Exercice 5 *Affichage d'un entier à l'écran*

1. Déclarer une variable de type entier.
2. Affecter lui une valeur (celle que vous voulez).
3. Utiliser `write_int (int n)` pour écrire la valeur de cet entier à l'écran.
4. Quel est le comportement si on n'affecte pas de valeur à l'entier avant de l'afficher ?

Rappel de Cours

Dans la suite on utilise la notion d'`Expression Arithmétique` et de son évaluation.
Une expression arithmétique peut être simplement une constante : `27` ou une variable `a`, ou contenir des opérateurs `+`, `-`, `<`, `==`, `=` ou encore une combinaison de tout celà : `a - 2*(b+4)`.
L'évaluation d'une Expression Arithmétique est sa valeur qui est la valeur de la constante pour une constante, le contenu de la variable pour une variable et le résultat de l'opération pour les opérateurs.

Exercice 6 *Affectation*

1. Déclarer deux variables `a` et `b`.
2. Donner une valeur à `a`.
3. Recopier la valeur de `a` dans `b`.
4. Ajouter `2` à `a`, que vaut `b` ?

Exercice 7 *La division entière*
Faites un programme qui déclare deux variables `a` et `b` et qui les affiche après avoir fait les affectations suivantes.

1. `a = (10/3)*3;`
2. `b = (10*3)/3;`

Exercice 8 *La division entière*
Faites un programme qui déclare quatre variables `a`, `b`, `c` et `d`. Donner des valeurs à `a` et `b` puis calculer :

1. `c = a/b;`
2. `d = a%b;`

Que valent `c` et `d`. Comment recalculer `a` à partir de `b`, `c` et `d`.

Dans la suite, les énoncés des exercices ne précisent pas qu'il faut déclarer les variables mais bien sûr il faut le faire.

Exercice 9 *Associativité de la soustraction*
Faites un programme qui affecte des valeurs à trois variables `a`, `b` et `c` et qui calcule et affiche les variables `d`, `e` et `f`.

1. `d = a-b-c;`
2. `e = (a-b)-c;`
3. `f = a-(b-c);`

Que constatez vous ?

Exercice 10 *Associativité de la division entière*
Faites un programme qui affecte des valeurs à trois variables `a`, `b` et `c` et qui calcule et affiche les variables `d`, `e` et `f`.

1. `d = a/b/c;`
2. `e = (a/b)/c;`
3. `f = a/(b/c);`

Que constatez vous ?

Exercice 11 *Calcul avec du graphique*
Déclarer deux variables de type `POINT` et fixer leur ordonnée à 100. Demander à l'utilisateur les valeurs des abscisses.

1. Tracer deux cercles remplis de rayon 5 centrés sur ces points.
2. Tracer une ligne entre ces deux points.
3. Calculer un troisième point tel que les trois points forment un triangle isocèle.

Exercice 12 *Calcul avec du graphique 2*
Demander à l'utilisateur de cliquer pour obtenir un point. Dessiner un cercle de rayon 100 centré sur le point. Dessiner un hexagone inscrit dans le cercle.

2 TD 2: Embranchement (le `if`), introduction aux booléens

Le but de cet ensemble d'exercices est de maîtriser la programmation des embranchements : `if`.

Rappel de Cours

Syntaxe

Le code du `if` se présente sous la forme :

```
Instructions Avant
if (Expression Arithmétique)
   {
   Bloc d'Instruction 1
   }
   else {
        Bloc d'Instruction 2
        }
Instructions Après
```

Principe

1. Si la valeur de l'`Expression Arithmétique` est différente de zéro alors c'est le `Bloc d'Instruction 1` qui est exécuté.
2. Si l'`Expression Arithmétique` vaut zéro c'est le `Bloc d'Instruction 2` qui est exécuté.

Exemples

Il y a donc deux exécution possibles :

```
Instructions Avant
Expression Arithmétique
Bloc d'Instruction 1
Instructions Après
```

ou bien :

```
Instructions Avant
Expression Arithmétique
Bloc d'Instruction 2
Instructions Après
```

Dans toute la suite, pour les cercles à afficher, vous utiliserez toujours un rayon de 50.

Exercice 13 *Clic 1*

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre un clic de l'utilisateur. Si le clic est à gauche de la ligne afficher un cercle bleu et s'il est à droite afficher un cercle rouge.
   Dans les deux cas, le cercle devra être centré là où l'utilisateur a cliqué.

Exercice 14 *Clic 2*

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre un clic de l'utilisateur. Si l'utilisateur clique à droite de la ligne, afficher un cercle bleu à la position en miroir par rapport à la ligne centrale verticale et vice versa (si le clic est à gauche, afficher le cercle à droite).

Exercice 15 *Double clic*

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre deux clics. Si les deux clics sont chacun d'un côté de la ligne verticale afficher une ligne rouge reliant les deux points où l'utilisateur a cliqué. Si les deux clics sont du même côté de la ligne verticale afficher une ligne bleu reliant ces deux points.

Exercice 16 *Triple clic*

1. Afficher une ligne verticale en blanc qui sépare en l'écran en deux parties égales.
2. Attendre trois clics. Si les trois clics sont du même côté de la ligne afficher un triangle reliant les 3 points, sinon, ne rien afficher.

Exercice 17 *Tiens un OU*

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Afficher une ligne horizontale en blanc qui sépare l'écran en deux parties égales.
3. Attendre un clic. Si le clic est en haut à gauche ou en bas à droite afficher un cercle bleu. Si le clic est dans les deux autres zones, afficher un cercle rouge.

Exercice 18 *Encore un OU*

1. Afficher deux traits verticaux qui séparent l'écran en trois zones de même taille.
2. Attendre un clic.
3. Si le clic est dans la zone de gauche ou de droite, afficher un cercle bleu.
4. Si le clic est dans la zone du milieu, afficher un cercle blanc.

Écrivez ce programme avec un seul test.

Exercice 19 *Trois zones*

1. Afficher deux traits verticaux qui séparent l'écran en trois zones de même taille.
2. Attendre un clic.
3. Si le clic est dans la zone de gauche, afficher un cercle bleu.
4. Si le clic est dans la zone du milieu, afficher un cercle blanc.
5. Si le clic est dans la zone de droite, afficher un cercle rouge.

Écrivez ce programme avec trois tests non imbriqués.
Écrivez ce programme avec deux tests imbriqués.

Exercice 20 *Parité*

1. Attendre un clic de l'utilisateur.
2. Afficher un cercle dont la couleur dépend de la parité des coordonnées cliquées :

	abscisse paire	abscisse impaire
ordonnée paire	`rouge`	`bleu`
ordonnée impaire	`jaune`	`vert`

   Utilisez le modulo (`%`) pour tester la parité.
3. Écrire le programme avec 3 tests.

3 TD 3: Boucles (le `while`, le `for`)

Le but de cet ensemble d'exercices est de maîtriser la programmation de la boucle : `while`.

Rappel de Cours

Syntaxe

Le code du `while` se présente sous la forme :

```
Instructions Avant
while (Expression Arithmétique)
   {
   Bloc d'Instruction
   }
Instructions Après
```

Principe

1. Tant que la valeur de l'`Expression Arithmétique` est différente de zéro le `Bloc d'Instruction` est exécuté.
2. L'exécution alterne donc entre l'évaluation de l'`Expression Arithmétique` et l'exécution du `Bloc d'Instruction`.
3. Dès que la valeur de l'`Expression Arithmétique` est égale zéro, ce sont les `Instructions Après` qui sont exécutées.
4. Le `Bloc d'Instructions` est exécuté au minimum zéro fois et potentiellement une infinité de fois.
5. L'`expression Arithmétique` est exécuté au minimum une fois et potentiellement une infinité de fois.

Exemples

On ne passe pas dans la boucle

```
Instructions Avant
Expression Arithmétique
Instructions Après
```

On passe une fois dans la boucle

```
Instructions Avant
Expression Arithmétique
Bloc d'Instruction
Expression Arithmétique
Instructions Après
```

On passe trois fois dans la boucle

```
Instructions Avant
Expression Arithmétique
Bloc d'Instruction
Expression Arithmétique
Bloc d'Instruction
Expression Arithmétique
Bloc d'Instruction
Expression Arithmétique
Instructions Après
```

Exercice 21 *Quadrillage 1*

1. Afficher des lignes verticales en blanc tous les 100 avec la plus à gauche qui est à l'abscisse 0. Vous ne devrez utiliser qu'une seule fois la fonction `draw_line`.

Exercice 22 *Quadrillage 2*

1. Afficher des lignes verticales en blanc tous les 100 avec la plus à gauche qui est à l'abscisse 0.
2. Afficher des lignes horizontales en blanc tous les 100 avec la plus basse qui est à l'ordonnée 0.

Exercice 23 *Quadrillage 3*
Reprendre l'exercice précédent et faire les lignes non pas tous es

1. Reprendre l'exercice précédent et faire les lignes non pas tous les 100 mais tous les 50.
2. Faites maintenant tous les 20.
3. déclarer une variable `delta` et faites les lignes tous les `delta`.

Exercice 24 *Cercles concentriques*

1. Déclarer une variables `taille` et intialiser là à 200.
2. Déclarer une fenêtre graphique carrée de taille `taille`×`taille`.
3. Dessiner 10 cercles concentriques centrés sur la fenêtre dont le plus petit est de rayon 10 et le plus grand rentre exactement dans la fenêtre.

Exercice 25 *Allumé-éteint 1*

1. Dessiner un carré au centre de l'écran (taille au choix).
2. À chaque clic, le carré se remplit et se vide alternativement.
3. Au bout de 10 clics, le programme se termine.

Exercice 26 *Allumé-éteint 2*
Même programme que l'exercice précédent mais le programme se termine quand le clic est en dehors du carré.

4 TD 4: Boucles et embranchements imbriqués

Le but de cet ensemble d'exercices est de maîtriser les programmes dans lesquels ont met des `if` *dans des* `while` *ou des* `for`*.*

Exercice 27 *Quadrillage 1*

1. Afficher des lignes verticales tous les 100 avec la plus à gauche qui est à l'abscisse 100. Vous ne devrez utiliser qu'une seule fois la fonction `draw_line`. Les lignes doivent être alternativement rouge et bleues.

Exercice 28 *5 clics*

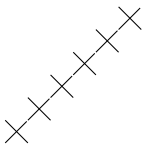
1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Faire une boucle qui répète cinq fois attendre un clic de l'utilisateur, puis si le clic est dans la partie gauche afficher un cercle rouge centré sur le point cliqué si le clic est dans la partie droite afficher un cercle bleu centrée sur le point cliqué.

Exercice 29 *Bouton OFF*

1. Dessiner un carré rempli de coté 10 en bas à gauche de la fenêtre graphique.
2. le programme attend des clis.
3. À chaque clic dans la fenêtre afficher un cercle centré sur le point cliqué.
4. Si le clic est dans le carré en bas à gauche, terminer le programme.

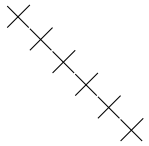
Exercice 30 *Diagonale (1)*

1. Initialiser une fenêtre carrée de taille 400×400.
2. Dessiner une diagonale de croix comme sur le dessin ci-dessous :



Exercice 31 *Diagonale (2)*

1. Initialiser une fenêtre carrée de taille 400×400.
2. Dessiner une diagonale de croix comme sur le dessin ci-dessous :



3. Modifier votre programme pour que les 2 diagonales s'affichent en même temps.

Le but des exercices suivants est d'afficher des dégradés de gris puis de couleur. La fonction :
`COULEUR couleur_RGB(int r, int g, int b)`
prend en argument trois entiers dans l'intervalle [0..256[et renvoie la couleur correspondante.

Exercice 32 *Dégradé de gris*
Ecrire un programme qui initialise une fenêtre de taille 512×512. Le programme doit afficher une suite de rectangles de largeur 2, de hauteur la hauteur de la fenêtre. Le i-ème rectangle aura pour couleur $(r, g, b) = (i, i, i)$. Le coin en bas à gauche du i-ème rectangle a pour coordonnées $(2 \times i, 0)$.

Exercice 33 *Dégradé de gris*
Modifier le programme précédent pour que le dégradé ne soit plus horizontal mais vertical.

Exercice 34 *Dégradé de rouge*
Modifier votre programme pour faire un dégradé de rouge plutôt que de gris.

Exercice 35 *Dégradé 2D*
Ecrire un programme qui initialise une fenêtre de taille 512×512. Le programme doit afficher des carrés de largeur 2 et de hauteur 2. Le carré dont le coin en bas à gauche est de coordonnées $(2 \times i, 2 \times j)$ aura pour couleur $(r, g, b) = (i, 0, j)$.

5 TD 5: Khôlle #1

Principe

1. La khôlle se déroule dans le créneau horaire habituel et dans la salle habituelle de la séance de TD.
2. Chaque étudiant a une heure de convocation.
3. Quand l'étudiant arrive, il tire au hasard un sujet.
4. Il a 40 minutes devant l'ordinateur pour programmer ce qui est demandé sur le sujet.
5. Tous les documents : livres, notes de cours, notes de TD, programmes déjà faits en TD,... sont autorisés.
6. Au bout de 40 minutes, l'enseignant examine le code et l'exécution de ce dernier et note l'étudiant sur 5.

6 TD 6: Les Tableaux

Le but de ce TD est de se familiariser avec la première structure de données non élémentaire que sont les tableaux.

Exercice 36 *Tableau 1*
Cliquer 10 fois et après chaque clic afficher un cercle rouge centré sur le point cliqué, de rayon 10. Une fois les 10 cercles affichés, attendre un clic, après ce clic réafficher les 10 cercles en bleu.

Exercice 37 *Tableau 2*
Cliquer 10 fois et après chaque clic afficher un cercle rouge centré sur le point cliqué, de rayon 10. Une fois les

10 cercles affichés, cliquer 10 fois et à chaque clic effacer les cercles dans l'ordre inverse d'affichage.

Exercice 38 *Tableau 3*
Cliquer 10 fois et après chaque clic afficher un cercle rouge centré sur le point cliqué, de rayon 10. Une fois les 10 cercles affichés, attendre de nouveau des clics et à chaque clic effacer le ou les cercles qui contiennent le point cliqué. Arrêter quand tous les cercles sont effacés.

Exercice 39 *Affichage d'un tableau*

Le code ci-dessous permet de remplir un tableau de 20 cases avec des valeurs aléatoires dans l'intervalle [0..100[:

```
int T[20];
int i;
for (i=0 ; i<20 ; i=i+1) { T[i] = alea_int(100); }
```

Remplir un tableau de taille 20 avec des nombres aléatoires puis pour chaque case i du tableau, afficher un rectangle rempli bleu dont les coordonnées sont :
— pour le point en bas à gauche : $(100 + 20 \times i, 50)$,
— pour le point en haut à droite : $(100 + 20 \times i + 19, 50 + 3 \times T[i])$.

Exercice 40 *Recherche de la plus petite valeur*

1. Afficher le tableau comme dans l'exercice précédent.
2. Afficher en rouge le rectangle qui correspond à la plus petite valeur.
3. Afficher en vert le rectangle qui correspond à la plus grande valeur.

7 TD 7: Procédures

Le but de ce TD est de se familiariser avec les procédures qui sont des fonctions qui prennent éventuellement des arguments et qui ne renvoient pas de valeur.

Exercice 41 *Triangle*
Écrire une fonction :
`void dessine_triangle(POINT p1, POINT p2, POINT p3)` qui dessine le triangle formé par les 3 points passés en argument, le trait entre `p1` et `p2` est bleu, celui entre `p2` et `p3` blanc et celui entre `p3` et `p1` rouge.

Tester cette fonction par différents appels dans la fonction `main`, en particulier, écrivez le code suivant dans le `main`.

```
POINT p1, p2, p3;
p1 = wait_clic();
p2 = wait_clic();
p3 = wait_clic();
dessine_triangle(p1, p2, p3);
```

On considère maintenant le code suivant :

```
POINT p1, p2, p3;
p1.x =   0; p1.y =   0;
p2.x = 300; p2.y = 600;
p3.x = 600; p3.y =   0;
dessine_triangle(p1, p2, p3);
```

Que se passe-t-il si on change l'appel de la fonction par :

```
dessine_triangle(p2, p3, p1);
```

Exercice 42 *Croix*
Écrire une fonction :
`void dessine_croix(POINT centre, int largeur, COULEUR c)` qui dessine la croix centrée sur le point `centre` et dont le carré englobant a un coté de taille `largeur`.

Tester cette fonction par différents appels dans la fonction `main`.

Exercice 43 *Mickey*
Écrire une fonction :
`void dessine_mickey(POINT centre, int rayon, COULEUR c)` qui dessine un Mickey (sans les effets de reflets).

Le rayon passé en argument est le rayon du grand cercle.

Tester cette fonction par différents appels dans la fonction `main`.

Exercice 44 *Cercle de couleur*
Écrire une fonction `void dessine_cercle_couleur(int rayon)` qui :

- attend un premier clic ;
- dessine un cercle plein, de couleur rouge centré au point du clic et de rayon `rayon` ;
- à chaque clic suivant :
  - si le clic est à l'intérieur du cercle change la couleur avec la séquence rouge, bleu, blanc, rouge, ...
  - si le clic est à l'extérieur du cercle, la fonction se termine.

Tester cette fonction par différents appels dans la fonction `main`.

Exercice 45 *Quadrillage*
Écrire une fonction `void dessine_line_H(COULEUR c)` qui dessine des lignes horizontales sur tout l'écran espacées de 100.

Écrire une fonction `void dessine_line_V(COULEUR c)` qui dessine des lignes verticales sur tout l'écran espacées de 100.

Écrire une fonction `void dessine_quadrillage(COULEUR c)` qui dessine un quadrillage dont les lignes horizontales sont espacées de 100 et les lignes verticales sont espacées également de 100.

Tester ces fonctions par différents appels dans la fonction `main`.

Exercices difficiles

Exercice 46 *Choix couleur*
Faites les fonctions suivantes :
`void dessine_couleurs()` qui dessine 4 carrés de couleurs différentes en bas à droite de l'écran.
`COULEUR choix_couleur(POINT p)` si le point est dans un des quatre carrés, la fonction renvoie la couleur de ce carré, sinon, la fonction renvoie `noir`.
`void dessine_cercle()`
Cette fonction a une variable `COULEUR courante;` qui stocke la couleur courante.
La fonction attend un clic. Si le clic est dans un des 4 carrés, la valeur de courante est modifiée en prenant la valeur du carré cliqué. Si le clic est en dehors du carré, la fonction dessine un cercle de la couleur courante.

Au bout de 20 cercles dessinés, le programme se termine.

Exercice 47 *Ellipse*
Écrire une fonction :
`void dessine_ellipse(POINT f1, POINT f2, int dist , COULEUR c);` qui dessine une ellipse dont les deux foyers sont `f1` et `f2` et dont la somme des distances aux foyers est `dist`.
http://fr.wikipedia.org/wiki/Ellipse_(mathématiques)

Vous utiliserez la fonction :
`void draw_pixel(COULEUR color)` qui affiche un pixel.

Tester cette fonction par différents appels dans la fonction `main`.

Exercice 48 *Quadrilatère*
Écrire une fonction :
`void dessine_quadri(POINT p1, POINT p2, POINT p3, POINT p4, COULEUR c)` qui dessine un quadrilatère dont les 4 sommets sont passés en arguments et dont les lignes ne se croisent pas.

Tester cette fonction par différents appels dans la fonction `main`.

8 TD 8: Fonctions

Exercice 49 *En quatre morceaux*
Déclarer trois variables globales :
`int haut_ou_bas;`
`int gauche_ou_droite;`
`int OU_CA;`

Écrire une fonction `void qui_dit_ou_c_est(POINT p)` qui met dans la variable `haut_ou_bas` la valeur 0 si le point `p` est dans la moitié inférieure de l'écran et 1 sinon. qui met dans la variable `gauche_ou_droite` la valeur 0 si le point `p` est dans la moitié gauche de l'écran et 1 sinon.

Écrire une fonction :
`void calcul_OU_CA ()`
qui met dans la variable `OU_CA` la valeur 0 si en bas à gauche
qui met dans la variable `OU_CA` la valeur 1 si en bas à droite
qui met dans la variable `OU_CA` la valeur 2 si en haut à gauche
qui met dans la variable `OU_CA` la valeur 3 si en haut à droite

Écrire une fonction `void dessine_cercle_couleur(POINT centre)` centré sur le point `centre`. La couleur est bleu, rouge, vert ou jaune selon que `OU_CA` vaut 0, 1, 2 ou 3.

Tester ces fonctions en utilisant le main ci-dessous :

```
int main()
{
int i;
init_graphics(400,400);

for (i=0 ; i<20 ; i++)
    {
    // tapez votre code ici



   }
}
```

Exercice 50 *Morpion*
Le morpion se joue sur un tableau de 3×3 cases. Vous pouvez utiliser une fenêtre graphique de 300×300

Déclarer la variable globale : `int a_qui_de_jouer;`

Écrire une fonction `void quadrillage()` qui dessine les lignes nécessaires pour séparer la fenêtre en neuf cases.

Écrire une fonction `void dessine_action(POINT p)`

1. qui calcule dans une variable locale `centre` le centre correspondant à la case à laquelle le point `p` appartient.
2. qui ensuite dessine une croix centrée sur le point `centre` si `a_qui_de_jouer` vaut 0 ou qui dessine un cercle centré sur le point `centre` si `a_qui_de_jouer` vaut 1

Tester ces fonctions en utilisant le main ci-dessous.

```
int main()
{
int i;
init_graphics(300,300);
```

```
a_qui_de_jouer = 0;
for (i=0 ; i<9 ; i++)
    {
    // tapez votre code ici



   a_qui_de_jouer =  1 - a_qui_de_jouer;
   }
}
```

Exercice 51 *Cercle qui bouge*
Déclarer deux variables globales :
`POINT centre, centre_precedent;`

Écrire une fonction `void efface_affiche()` qui affiche un cercle rempli noir de rayon 30 centré sur le point `centre_precedent` et qui affiche un cercle rempli bleu de rayon 30 centré sur le point `centre`.

Écrire une fonction `void plus_ou_moins(POINT p)` qui recopie la variable `centre` dans la variable `centre_precedent` et qui ajoute 2 dans le champ `y` de centre si le point `p` est au dessus de centre et qui retire 2 dans le champ `y` de centre si le point `p` est en dessous de centre.

Tester ces fonctions en utilisant le main ci-dessous.

```
int main()
{
int i;
init_graphics(400,400);

centre_precedent.x = 0;
centre_precedent.y = 0;

centre.x = 200;
centre.y = 200;

for (i=0 ; i<20 ; i++)
    {
    // tapez votre code ici



   }
}
```

9 TD 9: Méthodologie de programmation

Le but de cet ensemble d'exercices est d'être capable à partir d'un énoncé en français de trouver la structure du code qui permet de répondre à la question posée.

Exercice 52 *Droite, gauche, ...*
Ce programme se termine au bout de 20 clics

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre un clic de l'utilisateur.
3. Si le clic est dans la partie gauche, afficher un cercle rouge centré sur le point cliqué et recommencer à attendre un clic.
4. Si le clic est dans la partie droite, afficher un cercle bleu centré sur le point cliqué et recommencer à attendre un clic.

Exercice 53 *Droite, gauche, droite, gauche, ...*
Ce programme se termine au bout de 20 clics

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre un clic de l'utilisateur.
3. Si le clic est dans la partie gauche, afficher un cercle rouge centré sur le point cliqué et recommencer à attendre un clic
4. Si le clic est dans la partie droite et s'il y a au moins un cercle rouge affiché, alors afficher un cercle bleu centré sur le point cliqué et recommencer à attendre un clic.
5. Si le clic est dans la partie droite et s'il n'y a aucun cercle rouge affiché, alors ne rien afficher et recommencer à attendre un clic.

Exercice 54 *Droite, gauche, droite, gauche, droite, gauche, ...*
Ce programme se termine au bout de 20 clics

1. Afficher une ligne verticale en blanc qui sépare l'écran en deux parties égales.
2. Attendre un clic de l'utilisateur.
3. Si le clic est dans la partie gauche, afficher un cercle rouge centré sur le point cliqué et recommencer à attendre un clic.
4. Si le clic est dans la partie droite et si le clic précédent était également dans la partie droite, alors afficher un cercle bleu centré sur le point cliqué et recommencer à attendre un clic.
5. Si le clic est dans la partie droite et si le clic précédent était dans la partie gauche, alors ne rien afficher et recommencer à attendre un clic.

Exercice 55 *Bleu, blanc, rouge*

1. Afficher deux lignes verticales en blanc qui séparent l'écran en 3 parties égales.
2. La première fois que l'utilisateur clique dans la partie gauche, coloriez celle-ci en bleu, la deuxième fois l'effacer et ainsi de suite.
3. La première fois que l'utilisateur clique dans la partie du milieu, coloriez celle-ci en blanc, la deuxième fois l'effacer et ainsi de suite.
4. La première fois que l'utilisateur clique dans la partie droite, coloriez celle-ci en rouge, la deuxième fois l'effacer et ainsi de suite.
5. L'utilisateur devra pouvoir cliquer dans n'importe quelle partie qui devra alors se colorier ou s'effacer suivant l'état dans lequel elle est.
6. Arrêter le programme au bout de 20 clics.

Exercice 56 *Échiquier*

1. Représenter un échiquier (cases noires et blanches) de 8 cases par 8 cases.

Le code ne doit comporter aucun appel à la fonction `draw_line()` et qu'un seul appel à la fonction `draw_fill_rectangle()`.

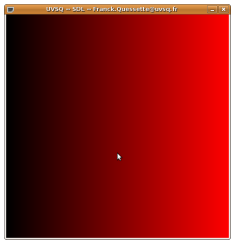
2. À faire avec une seule boucle.
3. À faire avec deux boucles imbriquées (une pour les lignes et une pour les cases de chaque ligne).

Exercice 57 *Déplacement de pièces d'échec*
Sur l'échiquier de l'exercice précédent :

1. Représenter un disque plein en rouge dans la case en bas à gauche : ce disque représente une tour dans le jeu des échecs. Tant que l'utilisateur clique dans une case où la tour peut se rendre, effacer le pion de sa case initiale et afficher le dans la case cliquée (le jeu s'arrête lorsque l'utilisateur clique dans une case non atteignable par la tour).
   Vous trouverez les règles de déplacement dans cet article :
   http://fr.wikipedia.org/wiki/Regles_du_jeu_d'echecs
2. Remplaçer la tour par une reine.
3. Remplacer la tour par un cavalier.

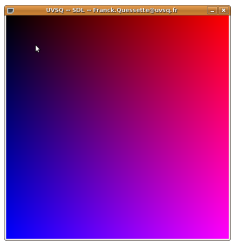
Exercice 58 *Dégradé 1*
La fonction `COULEUR couleur_RGB(int r, int g, int b)` génère une couleur, lui donnant en argument les valeurs des trois composante `red`, `green` et `blue` qui doivent être dans l'intervalle [0..255].

1. Initialiser la fenêtre graphique avec comme taille (512×512).
2. Afficher un dégradé de rouge (faites varier la composante `red` et fixez les composante `green` et `blue` à 0).
3. L'affichage du dégradé se fera en utilisant la fonction `draw_fill_rectangle()`.



Exercice 59 *Dégradé 2*

1. Initialiser la fenêtre graphique avec comme taille (512×512).
2. Afficher un dégradé de rouge et de bleu (faites varier les composantes `red` et `blue` et fixez la composante `green` à 0).
3. L'affichage du dégradé se fera en utilisant la fonction `draw_fill_rectangle()`.



10 TD 10: Jeu Noir et Blanc

Le but de ce TD est d'être capable de concevoir un ensemble de fonctions permettant de faire un jeu élémentaire.

Ce jeu se joue seul sur un plateau. Le plateau est découpé en cases (10×10). Sur chaque case est posé un jeton qui a une face blanche et une face noire.

Au départ, un jeton est posé sur chaque case. La face au dessus étant au hasard blanche ou noire.

À chaque tour de jeu, le joueur choisit un jeton. Le jeton choisi et ses voisins horizontaux et verticaux sont retournés. Les cases du coin ont 2 voisins, celles du bord 3 et les autres 4.

Le but est de mettre tous les jetons avec la même couleur au dessus.

Étape 1

Exercice 60 *Le plateau de jeu*
Donner une variable globale permettant de stocker le plateau de jeu.

Exercice 61 *Initialisation du plateau*
Écrire une fonction `void initialise_plateau()` qui met les jetons au hasard sur le plateau. Utiliser un appel à la fonction `alea_int(2)` qui renvoie aléatoirement 0 ou 1.

Exercice 62 *Affichage du plateau*
Écrire une fonction `void affiche_plateau()` qui affiche le plateau de jeu.

Exercice 63 *Premier test*
Écrire dans la fonction `main` un code qui vous permet de tester les fonctions précédentes.

Étape 2

Exercice 64 *Modification du plateau*
Écrire une fonction `void modifie_plateau(POINT p)` qui prend en argument un point, calcule la case correspondante du plateau et modifie le plateau de jeu suivant les règles.

Exercice 65 *Deuxième test*
Écrire dans la fonction `main` un code qui vous permet de tester la fonction précédente.

Étape 3

Exercice 66 *Test de fin*
Écrire une fonction `int encore()` qui renvoie zéro si tous les jetons sont de la même couleur et différent de zéro sinon.

Exercice 67 *Troisième test*
Écrire dans la fonction `main` un code qui vous permet de tester la fonction précédente.

Étape 4

Exercice 68 *Le jeu*
Écrire une fonction `void jeu()` qui initialise le tableau et fait jouer le joueur jusqu'à ce qu'il ait gagné ou bien

joué 100 coups.

Exercice 69 *Quatrième test*
Écrire dans la fonction `main` un code qui vous permet de tester la fonction précédente.

Exercice 70 *Questions théoriques*

1. Peut-on gagner depuis n'importe quelle situation de départ ?
2. Comment générer une situation de départ qui est gagnante à coup sûr ?
3. Peut-on calculer le nombre minimal de coupd à jouer pour gagner ?

11 TD 11: Algorithmique sur les Tableaux

Exercice 71 *Tri 1*

1. Afficher le tableau comme dans l'exercice **39** du TD 6.
2. Rechercher une case contenant la plus petite valeur stockée dans le tableau.
3. Échanger le contenu de cette case avec celui de la première case.
4. Effacer tout l'écran : utilisez la fonction `fill_screen (noir)`.
5. Afficher de nouveau le tableau en affichant la première valeur en rouge et les autres en bleu.

Exercice 72 *Tri*
En vous inspirant de l'exercice précédent, écrire un programme qui trie le tableau en ordre croissant en l'affichant après chaque échange de valeur.

Exercice 73 *Tableau à une dimension*

1. Remplir un tableau de points au hasard. Le tableau sera de taille 20 (utiliser la fonction `int alea_int(int N)`).
2. Remplir un second tableau de `int` au hasard. Ce tableau sera également de taille 20.
3. Afficher des cercles centrés sur les points du premier tableau et dont les rayons sont dans le deuxième tableau.

Exercice 74 *Convergence*
Dans cet exercice les rayons seront tous de 20.

1. Déclarer un tableau permettant de stocker des points. Ce tableau est de taille 20.
2. Cliquer 20 fois. À chaque clic remplir le tableau avec les points cliquées et afficher un cercle centré sur le point.
3. Cliquer de nouveau 20 fois. À chaque clic supprimer le cercle le plus éloigné du centre de l'écran et le remplacer par celui centré sur le nouveau clic. Si le nouveau clic est plus éloigné du centre que les 20 déjà stockés, ne rien faire.

Exercice 75 *Grossir et maigrir*

1. Déclarer deux tableaux permettant de stocker des points et des entiers. Ces tableaux sont de taille 20.
2. Cliquer 20 fois. À chaque clic remplir le tableau des points avec les points cliquées, au départ tous les rayons sont de 20.
3. Cliquer de nouveau. À chaque clic, pour tous les cercles qui contiennent le point cliqué augmenter leur rayon de 5 pour les autres, le diminuer de 5. Les cercles dont le rayon passe à 0 ne devront plus jamais apparaitre.
4. Le programme se termine quand il n'y a plus aucun cercle.

12 TD 12: Khôlle #2

Principe

1. La khôlle se déroule dans le créneau horaire habituel et dans la salle habituelle de la séance de TD.
2. Chaque étudiant a une heure de convocation.
3. Quand l'étudiant arrive, il tire au hasard un sujet.
4. Il a 40 minutes devant l'ordinateur pour programmer ce qui est demandé sur le sujet.
5. Tous les documents : livres, notes de cours, notes de TD, programmes déjà faits en TD,... sont autorisés
6. Au bout de 40 minutes, l'enseignant regarde le code et l'exécution de ce dernier et note l'étudiant sur 5.

A Programmation Jeu Démineur

Le but de ce TD est de programmer un jeu de démineur

Les règles du démineur ne sont pas rapellées ici.

Certains paramètres sont fixés (variables globales)
— Taille du plateau de jeu : 20×20.
— Nombre de mines : 20.

Le joueur à deux modes de clic sur le plateau de jeu :
— Mode découverte : quand l'utilisateur clique sur une case, celle-ci est découverte et si c'est une bombre, il a perdu.
— Mode déduction : quand l'utilisateur clique sur une case non découverte la case reste non découverte mais s'affiche en avec une croix rouge. Si l'utilisateur reclique dessus elle redevient une case non découverte normale.
— En mode découverte, un clic sur une case rouge, ne la découvre pas
— En bas de la fenêtre s'affiche deux informations :
  — Combien de cases non découvertes sont affichées avec une croix rouge.
  — Un carré indiquant si on est en mode découverte ou en mode déduction. Un clic sur ce carré fait passer d'un mode à l'autre.

Ci-dessous des suggestions d'affichage :
— Case non découverte grise (avec ou sans croix rouge)
— Case découverte qui n'est pas une bombe : fond blanc avec chiffre en noir dessus
— Case découverte qui est une bombe : complétement rouge
— Carré indiquant le mode : carré gris pour le mode découverte carré gris avec croix rouge pour le mode déduction
— Le nombre de cases avec une croix rouge s'affiche en rouge à coté du carré indiquant le mode.

Exercice 76
Donner le programme principal qui permet de créer un plateau de jeu et à un joueur de jouer jusqu'à ce qu'il ait gagné ou perdu.
Ne pas se soucier de comment est stocké le jeu.
Compiler ce programme avec les fonctions non faites, (juste les entêtes et un corps vide).

Exercice 77 *Stockage le jeu*
Répondez à ces questions :

1. Pour chaque case du plateau de jeu, quelles informations ont besoin d'être stockées.
2. Comment coder chacune de ces informations.
3. Proposer un stockage utilisant qu'un seul tableau en variable globale.
4. Proposer un stockage avec 3 tableaux en variable globale.
5. Comment stocker le mode ?

Écrire une fonction affichage pour le cas des 3 tableaux.
Compiler et tester.

Exercice 78 *Au début*
Écrire la fonction qui initialise au hasard le plateau de jeu.
Compiler et tester.

Exercice 79 *Où suis-je 1 ?*
Écrire une fonction qui prend en argument un point et qui renvoie 1 si le point est dans le carré de mode et 0 sinon.

Compiler et tester.

Exercice 80 *Où suis-je 2 ?*
Écrire une fonction qui renvoie le numéro de la ligne du plateau qui correspond au point cliqué. Cette fonction renvoie -1 si le point cliqué correspond à une ligne hors du tableau.

Idem avec les colonnes.

Compiler et tester.

Exercice 81 *Continuer à jouer*
Écrire une fonction qui renvoie -1 si lme joueur a perdu, 1 s'il a gagné et 0 s'il a ni gagné ni perdu.

Compiler et tester.

Exercice 82 *C'est perdu*
Si le joueur a perdu, afficher toutes les cases où il y a des mines.

Compiler et tester.

Exercice 83 *Tout mettre ensemble*
Mettez tout ensemble et ça marche !

B Programmation Jeu Tic-Tac-Toe

Le but de ce TD est de se familiariser avec la notion d'algorithme et la mise en œuvre d'un alghorithme par de la programmation.

Dans tout ce TD, onconsidère que l'ordinateur joue les croix et que l'humain joue les cercle. De plus, on considère que l'ordinateur commence.

Exercice 84 *Le Tic Tac Toe*

1. Combien de premiers coups différents y a-t-il ?
2. Combien de deuxième coups différents y a-t-il pour chaque premier coup ?
3. Combien d'alignements possible y a-t-il ?
4. Combien y a-t-il de parties possibles différentes au Tic Tac Toe ?
5. Combien d'alignements possible y a-t-il ?

Exercice 85 *Jouer bêtement au Tic Tac Toe*
Le but est de faire jouer l'ordinateur à la place d'un des deux joueurs.

1. Proposer une manière de jouer de l'ordinateur basée sur la tableau du TD précédent qui grantie que l'ordinateur joue dans une case vide.
2. Ecrire une fonction `ordinateur_joue` qui prend en argument le tableau et renvoie le numéro de la case du tableau dans lequel il joue.
3. Connaissant cette méthode de jeu de l'ordinateur, donner une méthode pour le battre systématiquement.

Exercice 86 *Mieux jouer au Tic Tac Toe*

1. Ecrire une fonction qui cherche s'il y a deux symboles de l'ordinateur déjà alignés et qui renvoie le numéro de la troisième case si elle est jouable et -1 sinon.
2. Ecrire une fonction qui cherche s'il y a deux symboles du joueur humain déjà alignés et qui renvoie le numéro de la troisième case si elle est jouable et -1 sinon.
3. Ecrire une fonction `ordinateur_joue` qui prend en argument le tableau et renvoie le numéro de la case du tableau dans lequel il joue en utilisant les deux fonctions précédentes.

Exercice 87 *Bien jouer au Tic Tac Toe*

1. Quelle est la meilleure case à jouer quand on commence ?
2. Combien y a-til de configurations de jeu différentes pour le deuxième coup du joueur qui commence ?
3. Ecrire une fonction `ordinateur_joue` qui prend en argument le tableau et renvoie le numéro de la case du tableau dans lequel il joue le meilleur premier coup et le meilleur deuxième coup.
4. Prouver que l'ordinateur ne perd jamais quand il commence.

Exercice 88 *En plus*

1. Développer la stratégie de l'ordinateur quand il joue en deuxième.
2. Ecrire le programme complet qui demande au joueur humain qui commence et qui fait jouer l'ordinateur optimalement.

C Priorité et associativité des opérateurs en `C`

Du plus prioritaire au moins prioritaire :

Catégorie d'opérateurs	Symbole	Arité	Associativité
fonction, tableau, membre de structure, pointeur sur un membre de structure	`() [] . ->`	2	G ⇒ D
opérateurs unaires	`+ - ++ -- ! ~ * & sizeof (type)`	1	D ⇒ G
multiplication, division, modulo	`* / %`	2	G ⇒ D
addition, soustraction	`+ -`	2	G ⇒ D
opérateurs binaires de décalage	`<< >>`	2	G ⇒ D
opérateurs relationnels	`< <= > >=`	2	G ⇒ D
opérateurs de comparaison	`== !=`	2	G ⇒ D
et binaire	`&`	2	G ⇒ D
ou exclusif binaire	`^`	2	G ⇒ D
ou binaire	`|`	2	G ⇒ D
et logique	`&&`	2	G ⇒ D
ou logique	`||`	2	G ⇒ D
opérateur conditionnel	`?:`	3	D ⇒ G
opérateurs d'affectation	`= += -= *= /= %= &= ^= |= <<= >>=`	2	D ⇒ G
opérateur virgule	`,`	2	G ⇒ D

D Annexe : types, variables constantes et fonctions disponibles

D.1 Types, Variables, Constantes

Types

```
typedef struct point int x,y; POINT;
typedef Uint32 COULEUR;
typedef int BOOL;
```

Variables

La largeur et la hauteur de la fenêtre graphique :

```
int WIDTH;
int HEIGHT;
```

Ces deux variables sont initialisées lors de l'appel à `init_graphics()`.

Constantes

Déplacement minimal lorsque l'on utilise les flèches :

```
#define MINDEP 1
```

Les constantes de couleur :

```
#define noir    0x000000
#define gris    0x777777
#define blanc   0xffffff
#define rouge   0xff0000
#define vert    0x00ff00
#define bleu    0x0000ff
#define jaune   0x00ffff
#define cyan    0xffff00
#define magenta 0xff00ff
```

Les constantes booléennes :

```
#define TRUE 1
#define True 1
#define true 1
#define FALSE 0
#define False 0
#define false 0
```

D.2 Affichage

Initialisation de la fenêtre graphique

```
void init_graphics(int W, int H);
```

Affichage automatique ou manuel

Sur les ordinateurs lent, il vaut mieux ne pas afficher les objets un par un, mais les afficher quand c'est nécessaire. c'est aussi utilme pour avoir un affichage plus fluide quand on fait des animations.

On a donc deux modes d'affichage, automatique ou non. Quand l'affichage automatique est activé, chaque dessin d'objet l'affiche automatiquement. Quand il n'est pas automatique c'est l'appel à la fonction `affiche_all();` qui affiche les objets.

Pour basculer de l'affichage automatique au non automatique, il y a deux fonctions :

```
void affiche_auto_on();
void affiche_auto_off();
```

Quand on est en mode non automatique l'affichage se fait lorsque l'on appelle la fonction :

```
void affiche_all();
```

Par défaut on est en mode automatique.

Création de couleur

`COULEUR couleur_RGB(int r, int g, int b);` prend en argument les 3 composantes rouge (`r`), vert (`g`) et bleue (`b`) qui doivent être comprises dans l'intervalle : [0..255].

D.3 Gestion d'événements clavier ou souris

Gestion des flêches

`POINT get_arrow();`

Si depuis le dernier appel à `get_arrow()`, il y a eu G appuis sur la flèche gauche, D appuis sur la flèche droite H appuis sur la flèche haut et B appuis sur la flèche bas.

Le point renvoyé vaudra en `x` $D - B$ et en `y` $H - B$.

Cette instruction est non bloquante, c'est à dire que si aucune flèche n'a été appuyée les champs x et y vaudront 0.

Gestion des déplacements la souris

`POINT get_mouse();` renvoie le déplacement de souris avec la même sémantique que `get_arrow()`. Cette instruction est non bloquante : si la souris n'a pas bougé les champs x et y vaudront 0.

Gestion des clics de la souris

`POINT wait_clic();` attend que l'utilisateur clique avec le bouton gauche de la souris et renvoie les coordonnées du point cliqué. Cette instruction est bloquante.

`POINT wait_clic_GMD(char *button);` attend que l'utilisateur clique et renvoie dans `button` le bouton cliqué :

— `*button` vaut 'G' (pour Gauche) après un clic sur le boutton gauche,
— `*button` vaut 'M' (pour milieu) après un clic sur le boutton du milieu,
— `*button` vaut 'D' (pour Droit) après un clic sur le boutton droit.

Cette instruction est bloquante.

Fin de programme

`POINT wait_escape();` attend que l'on tape Echap et termine le programme.

D.4 Dessin d'objets

`void fill_screen(COULEUR color);` remplit tout l'écran.

`void pixel(POINT p, COULEUR color);` dessine un pixel.

`void draw_line(POINT p1, POINT p2, COULEUR color);` dessine un segment.

`void draw_rectangle(POINT p1, POINT p2, COULEUR color);` dessine un rectangle non rempli. Les deux points sont deux sommets quelconques non adjacents du rectangle

`void draw_fill_rectangle(POINT p1, POINT p2, COULEUR color);` dessine un rectangle rempli Les deux points sont deux sommets quelconques non adjacents du rectangle

`void draw_circle(POINT centre, int rayon, COULEUR color);` dessine un cercle non rempli.

`void draw_fill_circle(POINT centre, int rayon, COULEUR color);` dessine un cercle rempli.

```
void draw_circle_HD(POINT centre, int rayon, COULEUR color);
void draw_circle_BD(POINT centre, int rayon, COULEUR color);
void draw_circle_HG(POINT centre, int rayon, COULEUR color);
void draw_circle_BG(POINT centre, int rayon, COULEUR color);
```

dessinent des quarts de cercle.

`void draw_fill_ellipse(POINT F1, POINT F2, int r, COULEUR color);` dessine une ellipse remplie.

`void draw_triangle(POINT p1, POINT p2, POINT p3, COULEUR color);` dessine un triangle non rempli.

`void draw_fill_triangle(POINT p1, POINT p2, POINT p3, COULEUR color);` dessine un triangle rempli.

D.5 Écriture de texte

L'affichage de texte n'est pas en standard dans SDL. Il faut donc que la librairie `SDL_ttf` soit installée.

La configuration fournie teste (grâce au `Makefile`) si `SDL_ttf` est installée ou pas. Si elle est installée, l'affichage se fait dans la fenêtre graphique sinon il se fait dans la fenêtre shell.

`void aff_pol(char *a_ecrire, int taille, POINT p, COULEUR C);` affiche du texte avec

- le texte est passé dans l'argument `a_ecrire`
- la police est celle définie par la constante `POLICE_NAME` dans `graphics.c`
- la taille est passée en argument
- l'argument p de type POINT est le point en haut à gauche à partir duquel le texte s'affiche
- la COULEUR C passée en argument est la couleur d'affichage

`void aff_int(int n, int taille, POINT p, COULEUR C);` affiche un entier. Même sémantique que `aff_pol()`.

Les fonctions suivantes affichent dans la fenêtre graphique comme dans une fenêtre shell. Commence en haut et se termine en bas :

`void write_text(char *a_ecrire);` écrit la chaîne de caractère passée en argument.

`void write_int(int n);` écrit l'entier passé en argument.

`void write_bool(BOOL b);` écrit le booléen passé en argument.

`void writeln();` renvoie à la ligne.

D.6 Lecture d'entier

`int lire_entier_clavier();` renvoie l'entier tapé au clavier. Cette fonction est bloquante

D.7 Gestion du temps

Chronomètre élémentaire

Ce chronomètre est précis à la microseconde.

`void chrono_start();` déclenche le chrono. Le remet à zéro s'il était déjà lancé.

`float chrono_val();` renvoie la valeur du chrono et ne l'arrête pas.

Attendre

`void attendre(int millisecondes);` attend le nombre de millisecondes passé en argument.

L'heure

`int heure();` envoie l'heure de l'heure courante.

`int minute();` renvoie le nombre de minutes de l'heure courante

`int seconde();` renvoie le nombre de secondes de l'heure courante.

D.8 Valeur aléatoires

`float alea_float();` renvoie un `float` dans l'intervalle $[0; 1[$.

`int alea_int(int N);` renvoie un `int` dans l'intervalle $[0..N[$ soit N valeurs différentes de 0 à $N-1$.

D.9 Divers

`int distance(POINT P1, POINT P2);` renvoie la distance entre deux points.