Modalités du CC1

- L'étudiant dispose du temps d'autonomie supervisé en classe par l'enseignant (pendant lequel celui-ci se tient disponible pour reformuler une incompréhension, donner des pistes de résolution en cas de blocage, guider l'étudiant à la résolution d'éventuelles problématiques).

- S'il le souhaite, l'étudiant a la possibilité d'avancer individuellement et sans assistance sa production sur le temps non dédié au cours.

- Il est conseillé de maîtriser les exercices d'entraînement (à rendre) jusqu'à la section 3 (expressions) pour réussir les exercices proposés.

- Chaque exercice doit être représenté par un code source rendu dans une archive `.zip` sous forme de fichiers sources en **Langage C** en extension `.c` (qui compilent). Soyez rigoureux sur votre rendu.

- Le code à produire par l'étudiant pour réussir les exercices s'attend à la restitution des notions étudiées jusqu'à la section 3 (un code plus avancé est fourni : exemple du mini-projet).

- L'évaluation vérifiera et valorisera des éléments de code, d'efficacité et d'adéquation de la solution proposée (entrées, sorties, logique conditionnelle, validité d'opérations, inclusions et autres éléments pertinents liés au langage C et à la logique du code proposé).

- L'utilisation d'outils ou moyens se substituant à la connaissance ou aux compétences de l'étudiant sera pénalisée en cas de détection par l'enseignant (niveau de l'étudiant dissimulé et donc non évaluable : 0).

- Le rendu d'un travail par groupe est autorisé uniquement selon les modalités exprimées dans le support de cours (perte de 4 points par participant supplémentaire, note du CC1 lourdement pénalisée en cas doute sur un rendu potentiellement malhonnête ou non déclaré).

- Le devoir est à rendre sur **MyGES** au plus tard le **Dimanche 22 Octobre** 2023 à 23h59.

Bon courage !

Assiduité (4 points)

Votre rendu doit contenir vos traces de recherche / solutions pour les exercices d'entraînement suivants :

- Section 2 (variables) : exercices 4 à 11 (2 points).
- Section 3 (expressions) : exercices 12 à 17 (2 points).

Attendus :

- Traces de recherche et de compréhension.
- Proposition d'une solution personnelle valide.

Compréhension (4 points)

Alice s'essaie à essayer de fabriquer des messages codés. Pour le moment, Alice essaie de coder un caractère et le décoder pour vérifier que sa méthode fonctionne. Son code est le suivant :

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  unsigned char c = 0;
  int k;
  printf("Entrez une clé : ");
  scanf("%d", &k);
  printf("Entrez un caractère : ");
  scanf("%c", &c);
  int a = k % 100;
  int b = k / 100;
  c = (c + a) * b;
  printf("code : %u\n", c);
  c = (c / b) - a;
  printf("decode = %c\n", c);
  exit(EXIT_SUCCESS);
}
```

Alice rencontre les problématiques suivantes :

- Sa clé 42 ne fonctionne pas. Pourquoi ? Des clés pourraient être plus adaptées ?
- La saisie du caractère ne semble pas fonctionner. Pourquoi ? Pourriez-vous le corriger ?
- Le décodage de la valeur obtenue semble incorrect. Pourquoi ? Dans quelles conditions ceci pourrait fonctionner ? Proposez une solution.

Attendus :

- Identifier et expliquer des problèmes d'opérations mathématiques.
- Identifier et expliquer des problèmes de saisie.
- Identifier et expliquer des problèmes d'opérations en mémoire.
- Proposition d'une solution personnelle valide.

Implémentation (4 points)

Bob a remarqué que les entiers et flottants peuvent par exemple se stocker sur 4 octets. Il aimerait voir si ceci peut être utilisé pour envoyer des messages codés. Pour ceci, il aimerait un programme qui prend un entier, le transforme en flottant en gardant la même représentation en mémoire (il ne veut pas de `3` deviendrait `3.f`). Puis, il faudrait vérifier que le flottant permette de retrouver l'entier saisi.

Bob en a parlé à Charlie qui a réalisé un programme valide. Pour pouvoir communiquer des entiers avec Charlie via des flottants, le programme que vous donnerez à Bob doit proposer des sorties cohérentes avec les suivantes (valeur int et valeur float sont saisies par l'utilisateur) :

```
valeur int : 42
5.88545355e-44
valeur float : 5.88545355e-44
42
```

```
valeur int : 1000000000
4.72378731e-03
valeur float : 4.72378731e-03
1000000000
```

Attendus :

- Interprétation de la représentation d'un int comme un float.
- Interprétation de la représentation d'un float comme un int.
- Fonctionnement pour les exemples donnés.
- Utilisation astucieuse des outils vus sur les entrées-sorties.

Performances (4 points)

Charlie a entendu parler de la suite de Fibonacci. A priori, cette suite pourrait se calculer par la formule mathématique suivante :

$$\mathcal{F}_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Charlie vous informe qu'avec uniquement la base du Langage C, si c'est codé correctement, la méthode informatique est précise jusqu'à $\mathcal{F}_{90}$ et qu'ensuite ceci resterait une approximation pertinente.

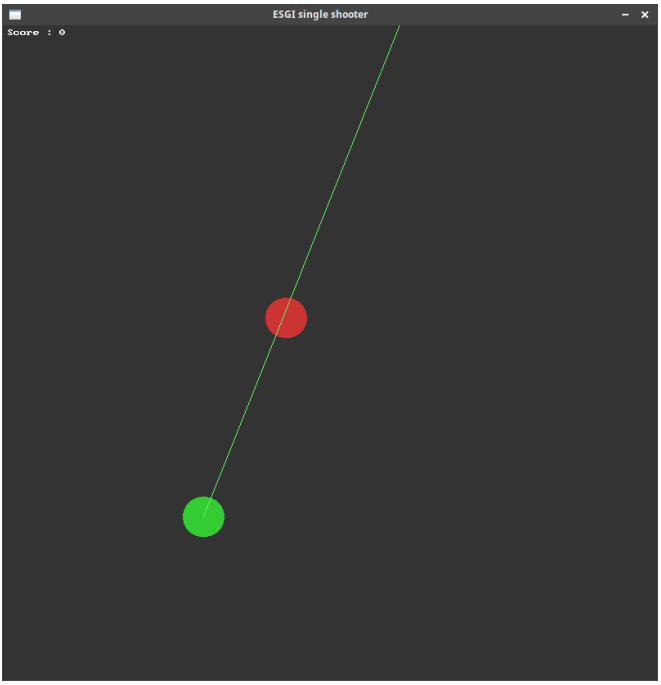
Attendus :

- Votre méthode calcule la suite de Fibonacci par la méthode donnée.
- Votre méthode calcule exactement $\mathcal{F}_{90}$.
- Votre méthode calcule une approximation de $\mathcal{F}_{20000}$.
- Votre méthode calcule la valeur de la suite demandée sous la milliseconde.

Mini-projet (4 points)

Oscar est fatigué de coder dans un terminal et trouverait plus fun de commencer à créer des jeux sous interface graphique. Il a commencé à étudier la partie sur SDL dans le cours et vous a fait un squelette d'un mini-jeu de tir. Le but est de coder des fonctionnalités qui permettent à une personnage de se déplacer et tirer sur un adversaire en mouvement.

Il aimerait que le jeu ressemble au moins à l'image suivante :



Il vous a fait le code graphique et aimerait que vous complétiez les `TODO` avec des instructions (voir l'initialisation de SDL, section 15 dans le cours). Il vous indique que pour que le jeu soit encore meilleur, il peut être intéressant de regarder la section 4 (conditions).

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```c
#include <time.h>
#include <SDL/SDL.h>
#include <SDL/SDL_gfxPrimitives.h>

/* Paramètres de la fenêtre : */
const int largeur = 800;
const int hauteur = 800;
const char * titre = "ESGI single shooter";

/* Coordonnées du joueur : */
int px, py;
/* Ligne de tir du joueur : */
int slpx, slpy;
/* Vitesse du joueur : */
int pv;

/* Coordonnées de l'adversaire : */
int ax, ay;
/* Vitesse de l'adversaire : */
int av;

void joueur_se_dirige_vers(int tx, int ty) {
  /* TODO : déplacer le joueur (px, py) vers la cible (tx, ty) */
  /* par pas de (pv)                                           */
}

void adversaire_se_dirige_vers(int tx, int ty) {
  /* TODO : déplacer l'adversaire (ax, ay) vers la cible       */
  /* (tx, ty) par pas de (av)                                  */
}

void joueur_regarde(int tx, int ty) {
  /* TODO : orienter la ligne de tir (slpx, slpy) vers         */
  /* (tx, ty)                                                  */
}

int distance_tir(int x, int y) {
  /* TODO : calculer la distance de la position (x, y) à la    */
  /* ligne de tir et remplacer 0x7fffffff par cette valeur     */
  return 0x7fffffff;
}

void place_adversaire() {
```

```c
  /* TODO : placer l'adversaire (ax, ay) aléatoirement en      */
  /* bordure de la fenêtre (largeur, hauteur)                  */
}

int distance_joueur(int x, int y) {
  /* TODO : calculer la distance d'une position (x, y) au      */
  /* joueur (px, py)                                           */
  return 0x7fffffff;
}

void affichage(SDL_Surface * ecran) {
  /* Remplissage de l'écran par un gris foncé uniforme : */
  SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 51, 51, 51));
  /* Affichage du joueur : */
  filledCircleRGBA(ecran, px, py, 25, 51, 204, 51, 255);
  /* Affichage de l'adversaire : */
  filledCircleRGBA(ecran, ax, ay, 25, 204, 51, 51, 255);

  lineRGBA(ecran, px, py, slpx, slpy, 102, 255, 102, 255);
}

int main() {
  srand(time(NULL));
  /* Création d'une fenêtre SDL : */
  if(SDL_Init(SDL_INIT_EVERYTHING) != 0) {
    fprintf(stderr, "Error in SDL_Init : %s\n", SDL_GetError());
    exit(EXIT_FAILURE);
  }
  SDL_Surface * ecran = NULL;
  if((ecran = SDL_SetVideoMode(largeur, hauteur, 32, SDL_HWSURFACE |
  ↪  SDL_DOUBLEBUF)) == NULL) {
    fprintf(stderr, "Error in SDL_SetVideoMode : %s\n",
    ↪  SDL_GetError());
    SDL_Quit();
    exit(EXIT_FAILURE);
  }
  SDL_WM_SetCaption(titre, NULL);

  /* Placement du joueur au centre de la fenêtre : */
  px = ecran->w / 2;
  py = ecran->h / 2;
  pv = 10;
```

```
ax = ecran->w / 2;
ay = 0;
av = 5;

int active = 1;
int grab = 0;
SDL_Event event;
int last_mouse_x = px;
int last_mouse_y = py;

int score = 0;
char display[100];

while(active) {

  affichage(ecran);
  sprintf(display, "Score : %d", score);
  stringRGBA(ecran, 5, 5, display, 255, 255, 255, 255);
  SDL_Flip(ecran);

  while(SDL_PollEvent(&event)) {

    switch(event.type) {
      /* Utilisateur clique sur la croix de la fenêtre : */
      case SDL_QUIT : {
        active = 0;
      } break;

      /* Utilisateur enfonce une touche du clavier : */
      case SDL_KEYDOWN : {
        switch(event.key.keysym.sym) {
          /* Touche Echap : */
          case SDLK_ESCAPE : {
            active = 0;
          } break;
        }
      } break;

      /* Utilisateur commence le clic : */
      case SDL_MOUSEBUTTONDOWN : {
        switch(event.button.button) {
          case SDL_BUTTON_LEFT : {
            grab = 1;
```

```
            last_mouse_x = event.button.x;
            last_mouse_y = event.button.y;
          } break;

          case SDL_BUTTON_RIGHT : {
            if(abs(distance_tir(ax, ay)) < 25) {
              ++score;
              ++pv;
              ++av;
              place_adversaire();
            }
          } break;
        }
      } break;

      /* Utilisateur relâche le clic : */
      case SDL_MOUSEBUTTONUP : {
        switch(event.button.button) {
          case SDL_BUTTON_LEFT : {
            grab = 0;
          } break;
        }
      } break;

      /* Utilisateur bouge la souris : */
      case SDL_MOUSEMOTION : {
        last_mouse_x = event.motion.x;
        last_mouse_y = event.motion.y;
      } break;
    }
  }
  if(grab) {
    joueur_se_dirige_vers(last_mouse_x, last_mouse_y);
  }
  joueur_regarde(last_mouse_x, last_mouse_y);
  adversaire_se_dirige_vers(px, py);
  if(distance_joueur(ax, ay) < 25) {
    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 204, 51,
    ↪  51));
    filledCircleRGBA(ecran, px, py, 25, 0, 0, 0, 255);
  stringRGBA(ecran, 5, 5, display, 255, 255, 255, 255);
    SDL_Flip(ecran);
    SDL_Delay(1000);
```

```
      active = 0;
    }
    SDL_Delay(1000 / 60);
  }

  SDL_FreeSurface(ecran);
  SDL_Quit();
  exit(EXIT_SUCCESS);
}
```

Attendus :

- Déplacement du joueur à la souris.
- Déplacement de l'adversaire vers le joueur.
- Ligne de tir orientée vers la souris.
- Distance à la ligne de tir correcte pour élimination d'un adversaire.
- Placement aléatoire de l'adversaire.
- Élimination du joueur par sa distance à l'adversaire.