Rapport Travaux Dirigé 1
Programmation Embarqué

CHAPUS Louka, AMOURA Adil

1.

```
struct Vin
{
   char nom[50];                //chaîne de caractères
   char type[20];               //chaîne de caractères
   int annee;                   //entier pour l'année
   int nbBouteilles;            //entier aussi
};
```

On définit la structure Vin comme demandé dans le sujet, avec 4 attributs, le nom du vin, son type, son année et le nombre de bouteilles que l'on possède. On prend à chaque fois le bon type de données, soit char soit int.

2.

```
struct Cave
{
   struct Vin vins[50];    //tableau de 50 vin
   int compteur;           //entier pour comptabiliser le nombre de vin différents dans la cave
};
```

De même on définit la structure Cave comme un tableau de 50 vins ainsi qu'un compteur pour savoir le nombre de vin différents dans la cave.

3.

```
void saisie_vin(struct Vin *vin)
{
   printf("Nom de la bouteille : ");
   scanf("%s", vin->nom);                    //saisie du nom
   printf("Type de la bouteille : ");
   scanf("%s", vin->type);                   //saisie du type
   printf("Année de la bouteille : ");
   scanf("%d", &vin->annee);                 //saisie de l'année
   printf("Nombre de la bouteille : ");
   scanf("%d", &vin->nbBouteilles);          //saisie du nombre de bouteilles
   printf("\n");                             //retour à la ligne pour un affichage plus clair
}
```

Pour saisir les vins on fait un affichage pour les 4 attributs à rentrer. On a un pointeur vers une structure de type Vin pour pouvoir changer les valeurs des différents attributs sans avoir à renvoyer de données. Puis pour la saisie des informations on utilise le 'scanf'.

4.

```
void saisie_vin_cave(struct Cave *cave)
{
   if (cave->compteur < 50) {                                    //on impose la condition pour ne pas dépasser la capacité du tableau
      saisie_vin(&(cave->vins[cave->compteur]));  //on saisi un nouveau vin
      cave->compteur++;                                          //on incrémente le compteur de 1
   } else {
      printf("La cave est pleine, impossible d'ajouter plus de vins.\n");    //message d'erreur
   }
}
```

De même pour saisir un vin de la cave on passe un pointeur vers la cave en paramètre, puis on regarde le nombre de vin déjà présent dans la cave, pour être sûr de ne pas dépasser les limites du

tableau (50 vins différents). Enfin si la cave est pleine on l'indique à l'utilisateur via un message.

5.

```
void afficher_cave(struct Cave *cave)
{
   printf("Il y a %d vin(s) différent(s) dans la cave actuellement.\n", cave->compteur);
   for(int k=0; k<cave->compteur; k++)                    //parcours le tableau des vins
   {
      printf("Dans la case numéro %d, il y a le vin suivant :\n", k);                    //affichage des informations sur chaque vins
      printf("    Nom : %s\n",cave->vins[k].nom);
      printf("    Type : %s\n",cave->vins[k].type);
      printf("    Année : %d\n",cave->vins[k].annee);
      printf("    Quantités : %d\n",cave->vins[k].nbBouteilles);
   }
   printf("\n");                                          //retour à la ligne
}
```

On passe un pointeur vers la cave en paramètre de la fonction, puis on fait un affichage du nombre de vin présent dans la cave et des informations de chaque vin.

6.

```
void afficher_cave_anne(struct Cave *cave, int annee)
{
   printf("Les vins disponibles dans la cave de l'année %d sont les suivant :\n",annee);
   for(int k=0; k<cave->compteur; k++)              //on parcours tous les vins
   {
      if(cave->vins[k].annee == annee)              //si l'année est la même, on affiche les informations du vin
      {
         printf("Dans la case numéro %d, il y a le vin suivant :\n", k);
         printf("    Nom : %s\n",cave->vins[k].nom);
         printf("    Type : %s\n",cave->vins[k].type);
         printf("    Année : %d\n",cave->vins[k].annee);
         printf("    Quantités : %d\n",cave->vins[k].nbBouteilles);
      }
   }
   printf("\n");                                    //retour à la ligne
}
```

Pour afficher un vin en fonction de son année, il suffit de parcourir le tableau de tous les vins et de comparer leurs années avec l'année fournie en paramètre de la fonction. Et si le deux coïncide alors on affiche les informations du vin.

7.

```
void supprimer_vin(struct Cave *cave, int num)
{
   if (num >= 0 && num < cave->compteur)        //vérifie l'indice demandé
           {
      for (int i = num; i < cave->compteur - 1; i++) {
         cave->vins[i] = cave->vins[i + 1];              //décale de chaque éléments du tableau
      }
      cave->compteur--;                                  //décrémente le compteur
   }
}
```

Pour supprimer un vin il suffit de décaler chaque éléments d'un vers la gauche à partir de l'indice demandé et de décrémenter la compteur d'un car on a supprimé un vin de la cave. Et encore une fois, on a un pointeur vers une cave en entrée de la fonction pour pouvoir directement modifier la cave.

8.

```
void boire(struct Cave *cave)
{
   int quantite = 0;  //initialisation des quantités
   int nbCase = 0;                  //initialisation du numéro de la case

   afficher_cave(cave);

   printf("Saisir le numéro de la case du vin à boire : ");
   scanf("%d", &nbCase);
   printf("Saisir le nombre de bouteilles à boire : ");
   scanf("%d", &quantite);

   if(quantite > 0 || nbCase < 0 || nbCase > cave->compteur)          // vérifie la validité des données rentré
   {
      if(quantite > cave->vins[nbCase].nbBouteilles)                                  //si on arrive à 0 bouteilles ,ou moins
      {
         printf("Toutes les bouteilles ont été bues !\n");
         supprimer_vin(cave,nbCase);
      }
      else
      {
         printf("La quantité de bouteille restante à été mis à jour.\n");
         cave->vins[nbCase].nbBouteilles = cave->vins[nbCase].nbBouteilles - quantite;
      }
   }
   else{
      printf("Erreur dans la saisie de la case ou des quantités !\n");
   }
   printf("\n");
}
```

La fonction boire permet de boire un nombre de bouteilles, on affiche d'abord la cave pour permettre à l'utilisateur de savoir les vins présents. Puis on demande à l'utilisateur de saisir la quantité de bouteille à consommer ainsi que le numéro de la case. Ensuite, on fait des tests sur les quantités

saisies et en fonction du cas soit on supprime le vin (si la quantité est inférieure ou égale à 0), soit on met à jours la quantité de bouteilles disponible.