Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux dirigés 10 : types composées `struct`

```
1  /* Declaration de fonctionnalites supplementaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf() */
4
5  /* Declarations constantes et types utilisateurs */
6
7  struct duree_s
8  {
9      int h; /* heures   */
10     int m; /* minutes  */
11     int s; /* secondes */
12 };
13
14 /* Declarations de fonctions utilisateurs */
15
16 /* Multiplier une duree par un facteur (positif) */
17 struct duree_s multiplier_duree(int facteur, struct duree_s d);
18 /* Normaliser une duree (minutes et secondes seront dans [0, 59]) */
19 struct duree_s normaliser_duree(struct duree_s d);
20 /* Afficher une duree */
21 void afficher_duree(struct duree_s d);
22
23 /* Fonction principale */
24 int main()
25 {
26     /* Declaration et initialisation des variables */
27     struct duree_s duree_unitaire = {1, 32, 14}; /* Duree unitaire */
28     int seances = 100; /* Nombre de seances */
29     struct duree_s duree_totale; /* Duree totale */
30
31     /* Affichage */
32     printf("Duree d'une seance : ");
33     afficher_duree(duree_unitaire);
34     printf("\n");
35
36     duree_totale = multiplier_duree(seances, duree_unitaire);
37
38     /* Affichage */
39     printf("Duree totale des %d seances : ", seances);
40     afficher_duree(duree_totale);
41     printf("\n");
42
43     /* Valeur fonction */
44     return EXIT_SUCCESS;
45 }
46
47 /* Definitions de fonctions utilisateurs */
48 struct duree_s multiplier_duree(int facteur, struct duree_s d)
49 {
50     struct duree_s res; /* resultat */
51     /* Calcul */
52     res.h = d.h * facteur;
53     res.m = d.m * facteur;
54     res.s = d.s * facteur;
55     /* Normalisation */
```

```
56      res = normaliser_duree(res);
57      /* Retour valeur */
58      return res;
59  }
60
61  struct duree_s normaliser_duree(struct duree_s d)
62  {
63      struct duree_s res; /* resultat */
64      /* Normalisation secondes */
65      res.m = d.m + d.s / 60;
66      res.s = d.s % 60;
67      /* Normalisation minutes */
68      res.h = d.h + res.m / 60;
69      res.m = res.m % 60;
70      /* Retour valeur */
71      return res;
72  }
73
74  void afficher_duree(struct duree_s d)
75  {
76      printf("%d heures %d minutes %d secondes", d.h, d.m, d.s);
77  }
```

1 Enregistrements (`struct`)

1. Faire la trace du programme précédent.
2. Modifier le programme précédent (notamment le `struct duree_s`) de manière à ce que les durées intégrent un nombre de jours et que les heures soient dans l'intervalle $[0, 23]$.
3. Écrire une fonction réalisant la somme de deux durées.
4. Changer de `main`. Déclarer un tableau `durees` de cinq durées initialisé avec les durées : 1h30, 3h, 1h30, 3h, 1h30 ; et un tableau `seances` de cinq entiers initialisé avec 12, 12, 12, 2, 1. Le tableau `durees` représente les durées des cours, TD, TP, partiels et prérentrée de l'UE éléments d'informatique et le second tableau le nombre d'occurrences de ces séances durant le semestre. Compléter le `main` pour qu'il calcule le temps total consacrée à l'UE, hors révisions.
5. Comment faire en sorte que les données initiales (durées des séances, nombre de séances) soient renseignées dans un seul et même tableau, sans modifier `struct duree_s` ?

Les `struct` ne sont pas si complexes et point difficiles

6. Proposer un type utilisateur pour les nombres complexes en notation algébrique ($a + \mathrm{i}b$, $a, b \in \mathbb{R}$).
7. Donner une fonction d'addition et une fonction de multiplication entre ces nombres complexes.
8. Proposer un type utilisateur pour les points de l'espace en coordonnées réelles : $\mathbb{R}^3$.
9. Écrire une fonction `est_dans_sphere` prenant en argument un point c, un réel r, et un point p, qui renvoie vrai si le point p appartient à la sphère de centre c et de rayon r, faux sinon.
10. Définir une fonction `distance` qui retourne la distance entre deux points de l'espace passés en arguments. Simplifier la fonction `est_dans_sphere` en faisant appel à cette fonction distance.
11. Proposer un type utilisateur pour représenter une sphère dans l'espace.
12. Définir une fonction `collision_spheres` qui prend en entrée deux sphères et retourne `TRUE` si les deux sphères ont une intersection non vide, `FALSE` sinon.

Université Paris-Nord L1
Institut Galilée Année 2011-2012
Licence 1 - 1er semestre

Travaux pratiques 10 : un struct craquant

1 Le jeu de la tablette de chocolat

Une tablette (ou plaquette) de chocolat est une matrice de carreaux de chocolat que l'ont peut couper selon les lignes ou les colonnes qui séparent les carreaux. La hauteur et la largeur se comptent en nombre de carreaux. On a peint en vert un carreau dans le coin d'une tablette. Deux joueurs coupent tour à tour la tablette, au choix, selon une ligne ou une colonne (pas nécessairement près du bord). L'objectif est de ne pas être le joueur qui se retrouve avec le carreau vert (la tablette de hauteur 1 et de largeur 1).

Nous allons programmer ce jeu en faisant jouer à l'ordinateur le rôle de l'opposant.

1.1 Type utilisateur

Définir un type utilisateur qui servira à représenter la tablette (faire simple !).

1.2 Le tour de jeu et l'arbitre

Le joueur commencera la partie, puis l'opposant (l'ordinateur) jouera et ainsi de suite. À chaque fois l'une des deux dimensions de la tablette sera modifiée, jusqu'à ce qu'il y ait un perdant. Vous adopterez la structure suivante pour le programme :

- Une variable sera utilisée pour déterminer à qui vient le tour de jouer (joueur ou opposant) ;
- Une boucle réalisera le tour de jeu ;
- Une fonction `partie_perdue()` prenant en argument la tablette et renvoyant vrai lorsque celle-ci se réduit au carré vert, servira à construire la condition de sortie de la boucle ;
- Après quoi un affichage annoncera au joueur s'il a gagné ou perdu.
- À l'intérieur de la boucle, selon le tour, il sera fait appel à une fonction `joueur()` ou bien à une fonction `opposant()` (ces fonctions doivent modifier la tablette).

Vous êtes libre du choix de la taille initiale de la tablette.

1.3 Affichage

Réaliser une fonction d'affichage de la tablette qui sera appelée au début de chaque tour de jeu. Vous pouvez dans un premier temps vous contenter d'un affichage numérique de ses dimensions, et par la suite améliorer votre programme avec un affichage plus graphique.

1.4 Les joueurs

Le joueur étant l'humain entre le clavier et l'écran, il faudra lui faire saisir le choix de son coup, en deux temps : couper des colonnes ou couper des lignes ? Combien ? À vous de déterminer la manière de réaliser cette saisie.

Vous pouvez commencer à tester le fonctionnement de votre programme en utilisant la fonction `joueur()` également pour l'opposant.

L'opposant jouera au hasard. Pour cela vous pouvez vous aider d'une fonction `nombre_aleatoire` qui tire au hasard un nombre entre zéro et son argument (un entier positif). Cette fonction utilisera `rand`. N'oubliez pas d'initialiser le générateur de nombres aléatoires dans le `main()` ni d'inclure les bons fichiers d'en-tête (fonctionnalités supplémentaires).

Tester votre programme. Avez-vous prévu les cas où l'un des deux joueurs n'a pas le choix ? Est-ce que les coups invalides sont rendus impossibles ? Vous pouvez réutiliser et améliorer des fonctions vues précédemment (`choix_utilisateur()`, par exemple).

2 Améliorations

2.1 Tablette aléatoire

Dans votre programme la tablette est initialisée au départ. Pouvez-vous utiliser une fonction sans argument, `tablette_aleatoire` qui renvoie une tablette aléatoire ?

2.2 L'opposant parfait

Pouvez-vous trouver la meilleure manière de jouer (cela dépend de la tablette de départ) ? Si oui, faîtes en profiter votre opposant. Vous pouvez également le faire se tromper de temps en temps, en introduisant pour cela une dose de hasard...

2.3 Plusieurs tablettes

On peut utiliser plusieurs tablettes pour ce jeu : chaque joueur choisit sur quelle tablette jouer son tour et le perdant sera celui recevant pour son tour toutes les tablettes à la dimension 1×1.

2.4 Intégration au menu

Comme pour *deviner un nombre*, nous voulons maintenant que *croque tablette* soit un jeu disponible dans notre menu.