Université Paris-Nord L1
Institut Galilée Année 2008-2009
Licence 1 - 1er semestre

Travaux pratiques 11 : un struct craquant

Correction. Note aux chagé de TP : j'utiliserai ce jeu pour évoquer la notion d'invariant de boucle pour le dernier cours, la stratégie gagnante (toujours rendre la tablette carrée) sera explicitée à ce moment là (ne pas trop les aider à la trouver !).

Il est important que les étudiants apprennent à manipuler les struct. On déploiera donc des trésors de pédagogie pour les forcer à représenter une tablette par un struct plutôt que par deux d'entiers (les étudiants qui ne sont pas sensibles aux tablettes aux noisettes, au chocolat noir ou au lait, devront donc être convaincus par d'autres moyens : essayer l'allégé, le bio, le culturisme).

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 !).

Correction.

```
struct tablette_s
{
  int l; /* largeur (nombre de colonnes) */
  int c; /* hauteur (nombre de lignes) */
}
```

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 devrez utiliser la fonction `nombre_aleatoire()` qui tire au hasard un nombre entre zéro et son argument (un entier positif). Cette fonction a été écrite dans un TP précédent, pour retrouver dans quel fichier vous pouvez utiliser dans le terminal la commande `grep nombre_aleatoire ~/TP*/*.c` qui cherchera toutes les occurences du texte `nombre_aleatoire` dans tous les fichiers d'extension `.c` de vos répertoires dont le nom commence par TP. 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.

Correction.

```
1   #include <stdlib.h>
2   #include <stdio.h>
3   #include <time.h>
4
5   /* Declarations de types et fonctions utilisateur */
6   #define TRUE 1
7   #define FALSE 0
8
9   #define JOUEUR 1
10  #define OPPOSANT 0
11
12  struct tablette_s
13  {
14      int l; /* largeur */
15      int h; /* hauteur */
16  };
17
18  /* Declaration de fonction utilisateur */
19  int nombre_aleatoire(int n);
20  int choix_utilisateur(int a, int b);
21  void afficher_tablette(struct tablette_s choco);
22  int partie_perdue(struct tablette_s choco);
23  struct tablette_s joueur(struct tablette_s choco);
24  struct tablette_s opposant(struct tablette_s choco);
25
26
27  int main()
28  {
29      struct tablette_s choco = {8, 5}; /* miam */
30      int tour = JOUEUR;
31
32      /* Initialisation */
33      srand(time(NULL)); /* à ne faire qu'une fois */
34
35      /* Tour de jeu */
36      while (!partie_perdue(choco))
37      {
38          afficher_tablette(choco);
39          if (tour == JOUEUR) /* Au joueur de jouer */
40          {
41              choco = joueur(choco);
42              tour = OPPOSANT;
43          }
44          else /* A l'opposant de jouer */
45          {
46              printf("Opposant joue\n");
47              choco = opposant(choco);
48              tour = JOUEUR;
49          }
```

```
50      }
51      /* La partie est finie, le gagnant est ... */
52      if (tour == JOUEUR) /* L'opposant gagne */
53      {
54          printf("Perdu !\n");
55      }
56      else /* le joueur gagne */
57      {
58          printf("Arghh, vous avez gagné\n");
59      }
60
61      return EXIT_SUCCESS;
62  }
63
64
65
66  int nombre_aleatoire(int n)
67  {
68      /* tirage du nombre secret */
69      return rand() % (n + 1); /* entre 0 et n inclus */
70  }
71
72  void afficher_tablette(struct tablette_s choco)
73  {
74      int i;
75      int j;
76
77      for (i = 0; i < choco.h; i = i + 1)
78      {
79          for (j = 0; j < choco.l; j = j + 1)
80          {
81              if ( (i == choco.h - 1) &&  (j == 0) )
82              {
83                  printf("X ");
84              }
85              else
86              {
87                  printf("M ");
88              }
89          }
90          printf("\n");
91      }
92  }
93
94  int partie_perdue(struct tablette_s choco)
95  {
96      return (choco.l == 1)
97          && (choco.h == 1);
98  }
99
100 struct tablette_s joueur(struct tablette_s choco)
101 {
102     char c = ' '; /* colonne ou ligne */
103     int n; /* croque croque */
104
105     /* Choix entre colonnes et lignes */
106     if ( (choco.l > 1) && (choco.h > 1) )
```

```
107       {
108           while ( ! ((c == 'c') || (c == 'C') || (c == 'l') || (c == 'L')) )
109           {
110               printf("Croquer des _C_olonnes ou des _L_ignes ? (entrer C ou L) : ");
111               scanf(" %c", &c);
112           }
113       }
114       if (choco.l == 1)
115       {
116           c = 'l';
117       }
118       if (choco.h == 1)
119       {
120           c = 'c';
121       }
122       /* Croquer */
123       if ( (c == 'c') || (c == 'C') ) /* croquer des colonnes */
124       {
125           printf("Combien de colonnes voulez vous croquer [1, %d] : ", choco.l - 1);
126           n = choix_utilisateur(1, choco.l - 1);
127           choco.l = choco.l - n;
128       }
129       if ( (c == 'l') || (c == 'L') ) /* croquer des lignes */
130       {
131           printf("Combien de lignes voulez vous croquer [1, %d] : ", choco.h - 1);
132           n = choix_utilisateur(1, choco.h - 1);
133           choco.h = choco.h - n;
134       }
135       return choco;
136   }
137
138   struct tablette_s opposant(struct tablette_s choco)
139   {
140       int cote; /* 1 ligne, 0 colonne */
141
142       /* Cet opposant choisit au hasard le côté */
143       if ((choco.h > 1) && (choco.l > 1))
144       {
145           cote = nombre_aleatoire(1);
146       }
147       if (choco.h == 1)
148       {
149           cote = 1;
150       }
151       if (choco.l == 1)
152       {
153           cote = 0;
154       }
155
156       /* Mais il joue intelligement son coup ensuite, lorsque c'est possible */
157       if (cote == 1) /* croquer ligne */
158       {
159           if (choco.h > choco.l)
160           { /* on peut rendre une tablette carrée */
161               choco.h = choco.l;
162           }
163           else
```

```
164          { /* sinon on joue au hasard */
165              choco.h = nombre_aleatoire(choco.h - 2) + 1;
166          }
167      }
168      if (cote == 0) /* croquer colonne */
169      {
170          if (choco.l > choco.h)
171          { /* on peut rendre une tablette carrée */
172              choco.l = choco.h;
173          }
174          else
175          { /* sinon on joue au hasard */
176              choco.l = nombre_aleatoire(choco.l - 2) + 2;
177          }
178      }
179      return choco;
180  }
181
182
183  int choix_utilisateur(int a, int b)
184  {
185     int compteur = 5; /* compteur du nombre d'essais */
186     int choix; /* choix de l'utilisateur */
187
188     scanf("%d", &choix);
189     while ((compteur > 0) &&  ((choix < a) || (choix > b)))
190     {
191         printf("Le nombre doit etre entre %d et %d (inclus) : ", a, b);
192         scanf("%d", &choix);
193         compteur = compteur - 1;
194     }
195     if (compteur == 0)
196     {
197         choix = a;
198     }
199     return choix;
200  }
```