Langage C
TD7

Itheri Yahiaoui && Stéphane Cormier

Exercice 1 : « Les structures »

1. Qu'est-ce qui ne va pas avec les déclarations suivantes ?
   - struct point (double x, y)
   - struct point { double x, double y };
   - struct point { double x; double y }
   - struct point { double x; double y; };
   - struct point { double x; double y; struct point *p ;};
   - struct point { double x; double y; struct point p ;};
   - struct point { double x; double y; }

   - typedef struct { double x; double y } Point;
   - typedef struct { double x; double y ; Point *p ;} Point;
   - typedef { double x; double y; } Point;
   - typedef struct { double x; double y; };
   - typedef struct { double x; double y; } Point;

2. Quelles sont les différences entre ces programmes, compléter le dernier afin d'obtenir le même fonctionnement.

#include <stdio.h>
 	
struct point
{ double x;
double y;
};
 	
int main(void)
{
struct point test;
test.x = .25;
test.y = .75;
 	
printf("[%f %f]\n",
test.x, test.y);
return 0;
}
	#include <stdio.h>
	
	typedef struct
	{ double x;
	double y;
	} Point;
	
	int main(void)
	{
	Point test;
	test.x = .25;
	test.y = .75;
	
	printf("[%f %f]\n",
	test.x, test.y);
	return 0;
	}
		#include <stdio.h>
		
		typedef struct Point
		{ double x;
		double y;
		} Point;
		
		int main(void)
		{
		Point test = {.25, .75};
		
		printf("[%f %f]\n", test.x,
		test.y);
		return 0;
		}
			#include <stdio.h>
			
			typedef struct
			{ double x;
			double y;
			} Point;
			
			void
			POINTshow(Point) ;
			
			int main(void)
			{
			Point test = {.25, .75};
			
			POINTshow(test);
			return 0;
			}

3. Ajouter une fonction POINTdist() qui calcule la distance euclidienne entre deux points.

4. Définir une fonction POINTeq() qui retourne 1 si les deux points sont "égaux" ; et 0 sinon. Avec des valeurs en virgule flottante, il n'est pas très logique de tester l'égalité exacte ; vérifiez plutôt si la distance entre les points est inférieure à 0,000001.

5. Définir un type Rect pour les rectangles parallèles aux axes dans un système de coordonnées cartésiennes. Représenter un rectangle par ses extrémités inférieure gauche et supérieure droite en utilisant le type de point ci-dessus.

6. Ecrire une fonction RECTarea() qui calcule la surface d'un rectangle.

7. Ecrire une fonction qui retourne 1 si un point donné se trouve à l'intérieur d'un rectangle, 0 sinon.

8. Ecrire une fonction qui retourne 1 si le premier rectangle est complètement contenu dans le second rectangle, et 0 sinon.

Exercice 2 : « Les structures et les pointeurs »

1. Donner la trace des deux programmes suivants :

```
struct
  {
          int x, y;
} s[ ] = { 10, 20, 15, 25, 8, 75, 6, 2 };

int main(void)
{ int *i ;
  i = s ;
  printf("%d",*( i + 3 ));
printf("%d",s[i[7]].x);
  printf("%d",s[ (s + 2)->y / 3[i]].y);
  printf("%d",i[i[1]-i[2]]);
printf("%d",i[s[3].y]);
  printf("%d",( s + 1 )->x + 5);
  printf("%d",*( 1 +i )**( i + 4 ) / *i);
  printf("%d",( *(s + *( i + 1) / *i ) ).x + 2);
  printf("%d",++i[i[6]]);
  return 0;
}
```

```
#include <stdio.h>

struct produit {
  char *nom;
  float prix;
  struct produit *p;
};

struct menu {
   char * menu_nom;
   float menu_prix;
   struct produit *p;
```

```
};

int main(void)
 {
    int i;
   float  somme_prix=0;
   struct menu s[]={{"Menu A", 0, NULL},{"Menu B", 0, NULL},{"Menu C", 0, NULL},
                    {"Menu D", 0, NULL}};
   struct produit burger[] = {{"Burger", 4.5, NULL},{"Cheese burger", 5.5, NULL},
                              {"Chicken burger", 6.0, NULL}};
   struct produit boisson = {"Boisson", 1.75, NULL};
   struct produit accompagnement = {"Frites", 2.5, NULL};
   struct produit dessert = {"Ice cream", 3.75, NULL};
   struct produit *pt;

  // Menu B: Cheese burger, Boisson, Frites
  s[1].p = &burger[1];
  burger[1].p = &boisson;
  boisson.p = &accompagnement;

  printf("\n********** %s **********\n", s[1].menu_nom);
  for(pt = s[1].p; pt != NULL; pt = pt->p){
    somme_prix += pt->prix;
    printf("%17s: %5.2f euros\n", pt->nom, pt->prix);
  }
  printf("----------------------------\n");
  printf("Somme de tous les produits : %5.2f euros\n", somme_prix);
  s[1].menu_prix = somme_prix*0.9/10*10;
  printf("       Prix menu : %5.2f euros\n", s[1].menu_prix);

  return (0);
}
```

2. Modifier le code précédent, tel que les tableaux « burger, boisson, accompagnement, dessert » deviennent des tableaux dynamiques dont les tailles et les contenus sont définis par l'utilisateur. Pour le tableau « menu », les noms sont du style « Menu ***lettre***», la ***lettre*** prend une lettre de l'alphabet de « A à Z » selon l'indice du menu dans le tableau. La composition de chaque menu est décidée par l'utilisateur via des menus de sélection que vous lui afficherez.

Exercice 3 : « Les fichiers »

1. Que fait le code suivant :

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

int main()
{
    char phrase[500];
    FILE *fptr;
    char fnom[20]="Exemple.txt";

     printf("\n\n Création du fichier (Exemple.txt) et saisie du texte :\n");
     printf("-------------------------------------------------\n");
     fptr=fopen(fnom,"w");
     if (fptr==NULL)
       {
```

```
            printf(" Erreur d'ouverture du fichier!");
            exit(1);
          }
    printf(" Merci de saisir une phrase à stocker dans le fichier: ");
    gets(phrase);                    // ou fgets(phrase, sizeof phrase,stdin);   ou scanf("%[^\n]s",phrase);
    fputs(phrase,fptr);              // fprintf(fptr,"%s",phrase);
    fclose(fptr);
    printf("\n Le fichier %s a été bien créé...!!\n\n",fnom);
    return 0;
}
```

2. Modifier ce programme en écrivant une fonction qui permet de stocker N phrases dans un fichier dont le nom est saisi par l'utilisateur.

3. Ecrire une fonction qui permet de lire un fichier, d'afficher son contenu « ligne après ligne» et de compter sa taille en nombre de lignes.

4. Ecrire une fonction qui lit un fichier, caractère par caractère, en comptant le nombre total de caractères présents dans le fichier.

5. Ecrire une fonction qui permet de stocker les lignes du fichier dans un tableau dynamique à deux dimensions.