IUT Robert Schuman
Institut universitaire de technologie
Département **informatique**
Université de Strasbourg

M11 : TP 1

2019/20

Pense-bête

Structure d'un fichier en C

Ci-après, un exemple de contenu du fichier `exemple1.c` en `C` (dont la syntaxe est proche du `C#`) :

```c
#include <stdio.h>

int somme(int x, int y);

int main()
{
    int a, b, s;

    // Lecture du premier nombre
    printf("Entrez un nombre entier : ");
    scanf("%d", &a);

    // Lecture du deuxieme nombre
    printf("Entrez un autre nombre entier : ");
    scanf("%d", &b);

    /*
    Calcul de la somme a l'aide d'un appel de fonction
    (c'est stupide mais c'est un exemple a vocation pedagogique)
    */
    s = somme(a, b);
```

```
22
23      // Affichage du resultat avec retour a la ligne
24      printf("%d plus %d = %d\n", a, b, s);
25      printf("Le programme se termine.\n");
26
27      return 0;
28  }
29
30  /*
31  Fonction envoyant la somme des deux entiers
32  donnes en argument
33  */
34  int somme(int a, int b) {
35      return a + b;
36  }
```

Exemple 1

On observe la structure suivante qu'on utilisera systématiquement :

1) inclusion des fichiers d'en-tête ;
2) déclaration des fonctions ;
3) fonction `main` ;
4) définition des fonctions déclarées.

De manière plus détaillée :

ligne 1 : inclusion du fichier d'en-tête `stdio.h` (pour STandarD Input/Output), permettant notamment d'utiliser les fonctions `printf` et `scanf` pour l'affichage et la récupération de valeurs au clavier ;
ligne 3 : déclaration de la fonction somme à l'aide de son prototype : elle retourne un int à partir de deux arguments `int` ;
ligne 5 : la fonction main est le point d'entrée du programme ;
ligne 7 : définition des variables du bloc ;
ligne 9, 13, 17-20, ... : commentaires ;
ligne 10 : affichage d'une chaîne de caractères (sans retour à la ligne) ;
ligne 11 : lecture de l'entrée clavier sous format entier (`%d`) et stockage dans la variable a (`&a`) ;
ligne 24 : affichage d'une chaîne de caractères avec spécificateur de format (ici `%d` pour afficher les entiers `a`, `b` et `s`) ;
ligne 27 : on indique qu'on quitte le programme sans erreur ;
ligne 34 : définition de la fonction somme.

Format

Les commandes `printf` et `scanf` peuvent utiliser des écritures formatées. Voici une liste des formats :

Symbole	Type	Format
%c	char	affiche un caractère à partir de son code
%s	char*	affiche une chaîne de caractères
%d	int, short, (char)	affiche un entier signé
%ld	long	affiche un entier signé long
%u	unsigned int, unsigned short, (char)	affiche un entier non signé
%ud	unsigned long	affiche un entier non signé long
%f	float, double	affiche un nombre à virgule flottante

Compilation : un seul fichier

On compilera nos fichiers en ligne de commande.

```
gcc -Wall -o nom_fichier_sortie nom_fichier_source.c
```

gcc : la commande appelant le compilateur (`man gcc` pour plus de 22000 lignes de manuel).
-Wall : option permettant d'afficher un maximum d'alertes de compilation (le but étant de n'en avoir aucune lors de la compilation). Option très importante car le langage C est très permissif.
-o : suivi de nom_fichier_sortie qui est le nom que vous allez donner au fichier généré par la commande (ici un fichier exécutable).

Code ASCII et char

Le type `char` est utilisé pour le code ASCII (American Standard Code for Information Interchange) d'un caractère.
Le type char est donc considéré comme un entier et il y a deux manières d'affecter une valeur à une variable de type `char` (noter les simples quotes) :

```
char c1 = 'Z';
char c2 = 90;
```

D'après le tableau ci-après, les deux variables ont même valeur car le code ASCII du caractère `Z` est 90 :

code	caractère	code	caractère	code	caractère	code	caractère	code	caractère
0	\0	9	\t	10	\n	32	espace	33	!
34	"	35	#	36	$	37	%	38	&
39	’	40	(	41	)	42	*	43	+
44	,	45	-	46	.	47	/	48	0
49	1	50	2	51	3	52	4	53	5
54	6	55	7	56	8	57	9	58	:
59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D
69	E	70	F	71	G	72	H	73	I
74	J	75	K	76	L	77	M	78	N
79	O	80	P	81	Q	82	R	83	S
84	T	85	U	86	V	87	W	88	X
89	Y	90	Z	91	[	92	\	93	]
94	^	95	_	96	‘	97	a	98	b
99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l
109	m	110	n	111	o	112	p	113	q
114	r	115	s	116	t	117	u	118	v
119	w	120	x	121	y	122	z	123	{
124	|	125	}	126	~				

Exercices

Exercice 1 : Affichage sous différents formats
Créer un nouveau fichier dédié à cet exercice qui sera nommé `format.c`.

1) Afficher une variable de type `int` et de valeur -15 au format entier, entier non signé et flottant.
2) Afficher une variable de type `float` et de valeur 15.1 au format entier, entier non signé et flottant.
3) À la compilation, on verra apparaître des warnings (c’est très mal). En effet, il est demandé d’afficher des variables sous un format *a priori* incompatible avec le type. Qu’en déduire ?

Exercice 2 : Les char sont des entiers !
Les variables de type `char` sont des entiers codés sur 1 octet (8 bits).
Lorsqu’on affiche un `char` sous format `%c`, on obtient le caractère de code ASCII correspondant.
Créer un nouveau fichier dédié à cet exercice qui sera nommé `char.c`.
Deviner puis tester l’affichage en expliquant le résultat :

1) Affichons le char ’a’ :

```
printf("%d\n", 'a');
printf("%c\n", 'a');
```

2) Deviner, sans coder, ce qui est affiché par :

```
printf("%d\n", var);
printf("%c\n", var);
```

lorsqu'on assigne à `var` les valeurs :

`49`, `'1' + 1`, `'1' + '1'`, `'1' * 1` et `'1' * '1'`.

Pour le dernier : penser à midi + 123 heures.
Coder, tester et comparer à votre idée !
Si tout se passe bien, il y aura un warning à la compilation qu'on ignorera (c'est encore très mal) pour les besoins du TP. Cela dit, pourquoi ce warning ?

Exercice 3 : Fonction et condition

Le but ici est simplement d'écrire une fonction avec une instruction conditionnelle (comme en `C#`), de l'utiliser dans le `main` et de demander une valeur à l'utilisateur.

1) Créer un fichier `abs.c` sur le modèle de l'exemple donné dans le pense-bête en déclarant la fonction de prototype :

```
unsigned int val_abs(int a);
```

2) Coder la fonction `val_abs` qui renvoie la valeur absolue de a.
3) Terminer d'écrire le code dans la fonction `main` dans le but de demander une valeur entière à l'utilisateur et d'afficher sa valeur absolue.

Exercice 4 : Pour les braves

Si vous en êtes arrivé là, c'est que vous avez besoin d'en faire plus :

1) Afficher les lettres de l'alphabet les unes après les autres à l'aide d'une boucle.
2) Générer une table ASCII.