TD1 Rappels de programmation en langage C

Exercice 1 – « Hello, world »

Pour créer un fichier source en langage C, on utilise un éditeur de texte ou un EDI (environnement de développement intégré). Le fichier doit avoir l'extension « .c » (et non « .C » ou « .cpp » comme en C++).

Pour compiler les fichiers sources (par exemple *exo.c*) et créer le fichier exécutable correspondant (par exemple *exo*), il faut ouvrir une console, se placer dans le répertoire dans lequel le fichier est enregistré (commande Unix *cd*) et utiliser la commande *gcc* :

```
$ gcc exo.c -o exo -Wall
```

- L'option -o permet de spécifier un nom autre que *a.out* à l'exécutable final
- L'option -Wall (all warnings) demande à *gcc* d'afficher l'intégralité des avertissements de compilation et d'édition de liens

On peut alors lancer l'exécution du programme en utilisant la commande :

```
$ ./exo
```

Travail à faire. Saisir le programme C ci-dessous, le compiler et l'exécuter.

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

int main()
{
  printf("hello, world\n");
  exit(0);
}
```

Cet exemple célèbre a été utilisé pour la première fois en 1978 par Brian Kernighan, co-auteur avec Dennis Ritchie du premier livre sur le langage C : The C Programming Language. Plus tard, B. Kernighan expliquera que cette phrase provient d'un dessin animé qu'il avait vu, dans lequel un poussin sortait de son œuf en criant « Hello, world ».

Exercice 2 – Les sorties en langage C

Le langage C ne dispose pas du flot de sortie *cout* bien connu en C++. À sa place, on utilise une fonction spécifique : *printf*.

Exemple. L'instruction

```
printf("La moyenne des %d entiers est %f.\n", i, moyenne);
```

affiche sur la sortie standard (l'écran) la chaîne de caractères comprise entre les guillemets (appelée **format**), mais où :

- %d a été remplacé par la valeur de la variable entière *i*
- %f a été remplacé par la valeur de la variable réelle *moyenne*
- « \n » n'a pas été affiché mais un retour à la ligne a été effectué

%d et %f sont appelés des **codes**. Ils définissent la manière dont une variable d'un certain type doit être affichée. Le ***n*-ième** code est destiné au ***n*-ième** paramètre après la chaîne de format.

Le caractère spécial « \n » désigne un retour à la ligne.

Les principaux codes sont :

- %*d* : afficher un entier
- %*f* : afficher un flottant simple précision (type *float*)
- %*lf* : afficher un flottant double précision (type *double*).
- %*c* : afficher un caractère (type *char*)
- %*s* : afficher une chaîne de caractères (terminée par le caractère de fin de chaîne « \0 »

Travail à faire. Récupérer le programme incorrect suivant *pi.c*, présent sur l'ENT.

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

int main()
{
  double pi = 3.14;

  printf("La valeur de pi est %lf\n");
  exit(0);
}
```

Compiler ce programme avec, puis sans l'option -Wall de *gcc*. Que remarque-t-on ? Conclure sur le caractère indispensable de cette option -Wall.

Corriger le programme pour qu'il fonctionne correctement.

Exercice 3 – Les pointeurs

Soit le programme incomplet suivant *pointeurs.c*, présent sur l'ENT :

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

int main()
{
    double valeur;     // déclaration d'un réel double précision
    double *pv;        // déclaration d'un pointeur sur réel double préc.

    valeur = 37.2;
    pv = &valeur;

    printf("valeur = %f\n", valeur);
    printf("*pv = %f\n", *pv);
    printf("&valeur = %p\n", &valeur);
    printf("pv = %p\n", pv);

    exit(0);
}
```

Compiler le programme et l'exécuter pour répondre aux questions suivantes. On rappelle que le code *%p* permet d'afficher une adresse mémoire sous forme hexadécimale.

1- Compléter le tableau suivant et comprendre.

Quantité	Valeur
valeur	37.2
&valeur	
pv	
*pv	

2- Que vaut **pv* + 1 ? Vérifier à l'aide du programme.

Exercice 4 – Le retour des pointeurs

Rappel. Pour connaître la taille en octets de la zone mémoire allouée à une variable, il existe l'opérateur *sizeof*. Par exemple, la place mémoire occupée par l'entier *num* est donnée par *sizeof(num)* ou encore *sizeof(int)*.

Travail à faire. Soit le programme incomplet suivant *pointeurs2.c*, présent sur l'ENT :

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

int main()
{
    double valeur = 65.21;
    double *pv;

    int val = 7, *ptr;      // ptr est un pointeur sur entier

    pv = &valeur;
    ptr = &val;

    printf("taille valeur : %d octets\n", sizeof(valeur));
    printf("taille pv  : %d octets\n", sizeof(pv));

    printf("taille val : %d octets\n", sizeof(val));
    printf("taille ptr : %d octets\n", sizeof(ptr));

    exit(0);
}
```

1- Le compiler, l'exécuter et compléter le tableau suivant.

Variable	Taille mémoire (octets)
valeur	
pv	
val	
ptr	

2- Alors que les tailles de *valeur* et *val* sont différentes, pourquoi sont-elles identiques pour *pv* et *ptr* ?

Exercice 5 – Les entrées en langage C

Le langage C ne dispose pas du flot d'entrée *cin* du C++. À sa place, on utilise une fonction spécifique : *scanf*.

Exemple. L'instruction

```
scanf("%d", &val);
```

réalise la saisie clavier d'un entier (code %d) qu'elle stocke dans la variable *val*.

Les codes %d, %f, etc. sont les même que ceux de la fonction *printf*.

La fonction *scanf* devant modifier le paramètre réel (*val* dans l'exemple ci-dessus), il est donc nécessaire d'opérer un **passage par adresse**, d'où le « & » devant *val*.

Travail à faire. Rédiger un programme qui demande à l'utilisateur le prix d'une baguette de pain et un nombre de baguettes, et qui affiche le prix total des baguettes sous la forme indiquée par l'exemple ci-dessous.

```
$ ./bag
  Saisir le prix d'une baguette
  0.8
  Saisir le nombre de baguettes
  3
  3 baguettes à 0.800000 euros coutent 2.400000 euros
$
```

Exercice 6 – Passage de paramètres à une fonction

Soit le programme suivant *parametre.c*, à récupérer sur l'ENT.

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

void Incremente(int val)        // val : paramètre formel
{
   val = val + 1;
}

int main()
{
    int x = 148;

    Incremente(x);               // x : paramètre réel

    printf("x vaut %d\n", x);    // x devrait valoir 149

    exit(0);
}
```

1. Lancer un navigateur web. Aller sur le site http://pythontutor.com.
2. Cliquer sur l'hyperlien « *Visualize your code and get live help now* ». Ce site permet l'exécution d'un programme pas à pas, en visualisant l'évolution des variables, y compris des pointeurs.
3. Dans la liste déroulante « *Write code in* », sélectionner le langage C.
4. Copier-coller le programme ci-dessus dans la zone d'édition.
5. Pour démarrer l'exécution, cliquer sur « *Visualize execution* ». Cliquer sur « *Forward* » pour passer à l'exécution suivante.
6. Ce programme fonctionne-t-il correctement ? Remédier au problème. Remarquer l'action des pointeurs. Recopier les corrections sur le programme ci-dessus.

Exercice 7 – Schtroumpf de deux tableaux

1- Ecrire une fonction qui calcule et retourne le schtroumpf des deux tableaux. Ce schtroumpf s'obtient en multipliant chaque élément du tableau 1 par chaque élément du tableau 2, et en additionnant le tout.

Par exemple, si on a :

- Tableau 1 :

4	8	7	12

- Tableau 2 :

3	6

Le schtroumpf vaut :

$$3 \times 4 + 3 \times 8 + 3 \times 7 + 3 \times 12 + 6 \times 4 + 6 \times 8 + 6 \times 7 + 6 \times 12 = 279$$

2- Ecrire un programme principal *main* qui fait appel à la fonction rédigée en 1-

Pour éviter de réaliser une saisie clavier, on pourra initialiser les tableaux dans le programme à l'aide de la syntaxe suivante :

```c
int T1[] = {4,8,7,12};
```