Dans ce premier TP, consacré à l'expérimentation avec le langage C, nous mettons de côté pour l'instant les problématiques de compilation et exécution du code et nous nous concentrons sur le problème d'écrire des programmes qui soient à la fois *syntaxiquement corrects*, *corrects* et *bien commentés*.
Afin de faire cette partie d'expérimentation du langage, nous allons programmer dans un *notebook JupyterHub*. Sur moodle vous trouverez un tutoriel décrivant l'utilisation de JupyterHub.

Exercice 1 : Entrées, sorties

Afin de pouvoir interagir avec l'exécution d'un programme, un programmeur doit être en mesure de modifier la valeur d'une variable d'un programme qui s'exécute, mais aussi de faire imprimer par le programme s'exécutant le contenu d'une variable. Le langage C fournit pour cela deux fonctions prédéfinies :

— `printf`
— `scanf`

`printf` La fonction `printf` s'utilise de la manière suivante : son premier argument est une chaîne de caractère (entre ") contenant des *motifs de remplacement* de la forme `%_` où `_` est une lettre dépendant du type de la valeur à imprimer. Les arguments suivants de l'appel de fonction sont les valeurs à utiliser pour remplacer les motifs de remplacement lors de l'impression. Les motifs de remplacement pour les types déjà vus en classes sont les suivants : `%d` pour un entier et `%f` pour un nombre flottant.

— L'exécution de l'instruction

```c
printf("exemple numéro %d", 0);
```

imprime à l'écran :

```
exemple numéro 0
```

— L'exécution de l'instruction

```c
printf("un %d-ième ex.\n<-noter cela et le float: %f\n    +.", 1 + 1, 3.5);
```

imprime à l'écran :

```
un 2-ième ex.
<-noter cela et le float: 3.500000
    +.
```

`scanf` La fonction `scanf` effectue l'opération inverse : lorsqu'elle est exécutée le programme s'arrête et attend une entrée utilisateur. Le premier argument de la fonction `scanf` est une chaîne de caractères contenant des motifs de remplacement. Les arguments suivant sont des variables précédées d'un `&`.

— L'exécution de l'instruction

```c
scanf("%d", &a);
```

attend une entrée utilisateur, puis stocke l'entier lu dans la variable `a`.

Q. 1 Dans une cellule du notebook (on ne le précisera plus dans la suite) écrire une instruction C, permettant l'impression à l'écran de la chaîne de caractère `"Hello, world!"`.

Q. 2 Donner une séquence d'instructions, demandant un entier à l'utilisateur, puis un second, puis imprimant la somme des deux entiers, précédée de la mention `la somme des entiers est :`.

Q. 3 On suppose l'environnement courant du programme défini sur 3 variables $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, donner une instruction permettant d'imprimer le contenu complet de l'environnement de la manière suivante :

```
Environnement résultat:
(
  a -> 1,
  b -> 4,
  c -> 8
)
```

Q. 4 On suppose l'environnement courant du programme défini sur 3 variables $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, donner une instruction demandant à l'utilisateur de produire 3 entiers qui sont ensuite stockés dans les variables a, b et c.

Exercice 2 : Séquences d'instructions comme modificateurs d'environnements

Dans tout cet exercice, on utilisera l'instruction d'affichage de la question 3 (à une légère variation près) pour tester que les environnements résultats sont les bons. On utilisera l'instruction de lecture de la question 4 (à une légère variation près) pour générer les environnements initiaux.

Q. 1 Donner une séquence d'instructions, effectuant la transformation d'un environnement ρ définie sur $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ en un environnement ρ' tel que

$$\begin{aligned}\rho'(\mathsf{a}) &= 0\\ \rho'(\mathsf{b}) &= \rho(\mathsf{b})\\ \rho'(\mathsf{c}) &= \rho(\mathsf{a})\rho(\mathsf{b})\end{aligned}$$

Q. 2 Donner une séquence d'instructions, effectuant la transformation d'un environnement ρ définie sur $\{\mathsf{a}, \mathsf{b}\}$ en un environnement ρ' tel que

$$\begin{aligned}\rho'(\mathsf{a}) &= \rho(\mathsf{a}) + \rho(\mathsf{b})\\ \rho'(\mathsf{b}) &= \rho(\mathsf{a}) - \rho(\mathsf{b})\end{aligned}$$

Q. 3 Donner une séquence d'instructions, effectuant la transformation d'un environnement ρ définie sur $\{\mathsf{a}, \mathsf{b}\}$ en un environnement ρ' tel que

$$\begin{aligned}\rho'(\mathsf{a}) &= |\rho(\mathsf{b})|\\ \rho'(\mathsf{b}) &= \rho(\mathsf{b})\end{aligned}$$

Q. 4 Donner une séquence d'instructions, effectuant la transformation d'un environnement défini sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ en un environnement dont les valeurs de `a`, `b` sont les valeurs originales et `c` contient le 4-ème terme de la suite définie par $\begin{cases} u_0 & = \rho(\mathtt{c}) \\ u_{n+1} & = au_n^2 + bu_n \end{cases}$ où $\rho(\mathtt{c})$ désigne la valeur initiale `c`.

Exercice 3 : Expérimentation avec les valeurs de vérité

Nous avons vu que la syntaxe de l'alternative était le suivant :

```
if (expr) {
    inst1;
} else {
    inst2;
}
```

et avons énoncé que le comportement de cette instruction était la suivante :
- l'expression `expr` est évaluée dans l'environnement courant produisant une valeur v ;
- Si la valeur v "code pour vrai" alors l'instruction `inst1` est exécutée ;
- Si la valeur v "code pour faux" alors l'instruction `inst2` est exécutée.

L'objectif de cet exercice est d'expérimenter ce que le langage C considère comme une valeur de vérité "vraie" et une "fausse".

Q. 1 Proposer une méthode permettant d'observer laquelle des deux instructions `inst1` ou `inst2` est exécutée.

Q. 2 Utiliser la méthode de la question précédente pour établir expérimentalement ce que le langage C considère comme une valeur "vraie" et une "fausse". On s'efforcera de tester sur plusieurs valeurs, résultantes de l'évaluation de différentes expressions, ayant différents types.

Interlude : un programme C complet

Dans les cellules du notebook jupyter, il est possible d'écrire non seulement des séquences d'instructions simples comme nous l'avons fait jusqu'ici. Mais aussi un programme C complet c'est à dire un fichier contenant :
- les inclusions des librairies (ensemble de fonctions prédéfinies, telles que `sqrt`, `printf`, ...) utilisées dans le fichier ;
- les définitions des fonctions du programmeur ;
- la définition d'une fonction de nom `main`, qui est le point d'entrée du programme.

L'exécution du programme commence alors par l'exécution du corps de la fonction `main`.
Nous utiliserons souvent la fonction `main` pour faire des tests de correction des fonctions que nous venons de définir. Afin de faire de tels tests nous utiliserons la fonction prédéfinie `assert` prenant en argument une expression. Lorsque la fonction `assert` s'exécute, si la valeur de l'expression est vraie alors le programme continue de s'exécuter, si elle est fausse l'exécution s'arrête et signale à l'utilisateur qu'un des tests a échoué.

Q. 3 Sur la page moodle du TP, récupérer le fichier `triangle.c` que vous ajouterez à votre notebook. Repérer dans ce programme les parties mentionnées ci-avant. Exécuter le programme et vérifier que que les impressions se font dans l'ordre attendu. Ajouter un test faux et vérifier que l'exécution est bien arrêtée par le `assert` ainsi ajouté.

Exercice 4 : Surface d'un triangle

On s'intéresse au calcul de la surface d'un triangle.

Q. 1 Donner une fonction `float surface_triangle(int a, int b, int c);` permettant le calcul, à partir des longueurs des trois côtés d'un triangle : a, b et c de la surface du triangle. Ce calcul devra être fait au moyen de la formule de Héron :

$$\text{Soit } p = \frac{a+b+c}{2}$$
$$\text{Alors } A = \sqrt{p(p-a)(p-b)(p-c)}$$

où A désigne l'aire du triangle. Compléter la documentation de votre fonction pour préciser que cette formule n'est valide que lorsque a, b et c sont des longueurs formant bien un triangle.

Q. 2 Dans une fonction `main` ajouter des tests de votre fonction `surface_triangle` au moyen de la fonction `assert`.

Exercice 5 : Et, non, ou, implique, ...

De la même manière que la manipulation des entiers nécessite les opérations classiques que sont l'addition, la soustraction, la multiplication, ..., la manipulation des valeurs vrai et faux va nécessiter l'utilisation des opérations `et`, `ou`, `non`, ... Si l'on désigne par $\top$ la valeur vrai, et par $\bot$ la valeur faux, le comportement de ces opérations est défini (rappelé ?) ci-dessous :

et	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

ou	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

implique	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$

	non
$\top$	$\bot$
$\bot$	$\top$

xor	$\top$	$\bot$
$\top$	$\bot$	$\top$
$\bot$	$\top$	$\bot$

Ces tableaux sont à lire de la manière suivante : $\texttt{et}(\top, \top) = \top$ et $\texttt{et}(\top, \bot) = \bot$. Le langage C fournit des opérateurs permettant la manipulation des valeurs vrais et faux, que nous nommerons dans la suite, valeurs *booléenne* : `et` est calculé par l'opérateur `&&` qui s'utilise de manière infixe[♣], `ou` par `||` et `non` par `!`.

Q. 1 Redéfinir les fonctions `int et(int x, int y);`, `int ou(int x, int y);` et `int non(int x, int y);` au moyen de l'alternative, sans utiliser `&&`, `||` ni `!`.

Q. 2 Définir les fonctions `int implique(int x, int y);` et `int xor(int x, int y);` en utilisant les fonctions précédemment définies.

Q. 3 Comparer le comportement de `0 && (1/0)` et de `et(0, (1/0))`. Expliquer la différence.

Exercice 6 : Polynômes

Q. 1 Définir une fonction `float affine(float a, float b, float x)` calculant la valeur $ax + b$.

Q. 2 **Déduire** de la fonction précédente une fonction `float pol3(float a, float b, float c, float d, float x)` calculant la valeur $ax^3+bx^2+cx+dd$ n'utilisant ni addition ni multiplication.

♣. entre ses arguments : `1 && 0`

Exercice 7 : Alternatives

Q. 1 Un cinéma pratique les tarifs suivants :

— séance débutant avant midi : 6.50 ;
— normal : 11.20 ;
— senior (plus de 65 ans) et (uniquement du lundi au vendredi) : 10.20 ;
— moins de 14 ans : 4.50 ;
— moins de 18 ans : 6.00

Donner une fonction permettant le calcul du tarif à payer à partir de l'heure `heure` (un entier) , de l'âge `age` (un entier) et du jour `jour` (un entier). Si plusieurs tarifs peuvent s'appliquer, c'est le plus bas qui est choisi.

Q. 2 Une année est bissextile si elle est divisible par 4 mais pas par 100 ou si elle est divisible par 400. Définir une fonction permettant de tester si une année est bissextile.

Q. 3 Étant donné trois entiers x, y, z tels que $x < y < z$, on appelle y la médiane stricte des trois entiers x, y, z. Ainsi la médiane stricte de $3, 1, 2$ est 2, la médiane stricte de $7, 10, 1$ est 7, la médiane stricte de $5, 4, 1$ est 4. La médiane stricte de 3 entiers n'existe pas toujours, en effet le triplet $2, 2, 1$ n'a pas de médiane stricte puisque 2 n'est pas strictement inférieur à 2. Donner une fonction calculant la médiane stricte de 3 entiers `a`, `b` et `c`, si celle-ci est définie, et `-1` sinon.

Q. 4 Dans le domaine des réseaux de neurones une fonction importante est la fonction ReLU (rectification linéaire) dont la définition est la suivante : $f(a, b, x) = \max(ax+b, 0)$. Cette fonction vaut donc 0 si $ax + b$ est négatif et $ax + b$ sinon. Donner un programme calculant le ReLU des variables `a`, `b` et `x`.

Q. 5 Un spectacle propose un tarif enfant et un tarif adulte, chacun en nombre limité. Les enfants ont le droit de rentrer sur un tarif enfant ou sur un tarif adulte, les adultes n'ont le droit de rentrer que sur un tarif adulte. Donner une fonction permettant de calculer si un groupe peut acheter assez de places, à partir :

— du nombre e d'enfants dans le groupe ;
— du nombre a d'adultes dans le groupe ;
— du nombre te de tarifs enfants encore disponible ;
— du nombre ta de tarifs adultes encore disponible.