Programmation : TP 2

Objectifs : [C, O'Caml] Premiers programmes. Présentation du toplevel O'Caml.

Quelques consignes

On s'efforcera dans ce TP (et dans ceux qui suivront) de respecter les consignes de programmation suivantes :

- saisir tout votre code dans un seul fichier par langage de programmation (`tp2.c` ou `tp2.ml` par exemple), y compris les fonctions testées sous le toplevel O'Caml avec leurs exemples d'appels en commentaire ;
- commenter (art difficile, ne pas paraphraser, le minimum est de dire ce que font les fonctions). Mettre un en-tête (ex : `TP2 Programmation`, et votre nom) ;
- utiliser des identificateurs significatifs (un identificateur court est très rarement une économie) ;
- structurer : définir des fonctions pertinentes, utiliser des variables (C) ou des identificateurs (O'Caml) intermédiaires...
- «rendre» le TP par courrier électronique, via votre agent Mail, à l'adresse : `barnier@recherche.enac.fr`.

[C] Polynôme du second degré

1. Connectez vous, ouvrez une fenêtre shell, placez-vous dans votre répertoire C, et ouvrez un nouveau fichier `tp2.c` sous Xemacs (avec `^x^f`).

   Conseil : *N'oubliez pas de sauvegarder régulièrement (`^x^s`).*

2. Écrire un programme qui calcule et affiche, si elles existent, les racines réélles d'un polynôme du second degré, en déclarant et en initialisant les coefficients du polynôme avec des valeurs de votre choix.

   Indications : *Vous aurez besoin de la fonction* `sqrt` *calculant la racine carrée d'un réel, et définie dans la librairie* `math` *ainsi que de la fonction* `printf` *de la librairie* `stdio`.

   Pour pouvoir utiliser les fonctions de ces librairies, vous devez d'abord écrire les directives suivantes en début de programme :

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

   D'autre part, au moment de la compilation, utilisez l'option `-lm` *qui signale au compilateur qu'il doit faire le lien avec la librairie* `math` *(le lien avec* `stdio` *est implicite) :*

   ```
   gcc -W -Wall -o tp2 tp2.c -lm
   ```

   La fonction `printf` *s'utilise comme suit :*

   ```
   printf("Voici mes résultats : %f et %f\n",x,y);
   ```

   Les signes `%f` *indiquent où doivent être insérées les valeurs de x et y (dans ce cas des flottants) dans la chaîne de caractères. Faire* `man fprintf` *pour plus d'informations.*

3. Modifiez le programme pour pouvoir passer les coefficients en arguments de votre commande.

   Indication : *Pour lui passer un argument, la fonction* `main` *doit être écrite comme suit :*

   ```
   int main(int argc, char **argv) { ... }
   ```

   `argc` *est le nombre d'arguments, et* `argv` *le tableau d'arguments. La valeur du premier argument est alors* `argv[1]`*.C'est une chaîne de caractères, qu'il faudra ici convertir en flottant en utilisant la fonction* `atof` *de la librairie* `stdlib`.

[O'Caml] Polynôme du second degré

1. De la même façon, écrire dans un fichier `tp2.ml` de votre répertoire `Ocaml` un programme O'Caml affichant et calculant les racines d'un polynôme du second degré.
2. Modifiez ce programme de façon à produire un exécutable prenant en arguments les coefficients du polynôme.

   Indication : *Le passage d'arguments se fait en utilisant le tableau d'arguments* `argv` *défini dans le module* `Sys` *de la librairie standard O'Caml. La valeur du premier argument est alors* `Sys.argv.(1)`*.C'est une chaîne de caractères, qu'il faudra convertir en flottant en utilisant la fonction* `float_of_string`*. Vous pouvez consulter la doc O'Caml en ligne avec votre navigateur préféré :* `http://rafale/`

[Ocaml] Le toplevel, évaluation de quelques expressions simples

Jusqu'à présent, nous avons vu comment compiler des programmes simples avec des compilateurs C et O'Caml. En plus de la compilation, O'Caml propose un système interactif, le `toplevel`, qui permet d'évaluer facilement des expressions.

1. Dans une fenêtre shell, lancez le toplevel O'Caml :

   ```
   ocaml
   ```

   Le caractère `#` apparaît : c'est le *prompt* du toplevel O'Caml. A partir de là vous pouvez entrer une expression O'Caml se terminant par `;;` et taper la touche `Return`. À ce moment, le toplevel va :
   – *analyser la syntaxe de votre expression*
   – *la typer*
   – *chercher à l'évaluer (c'est-à-dire à réduire l'expression)*
   Le résultat de ces traitements s'affiche alors à l'écran. Avec le compilateur, il vous faudrait demander explicitement l'affichage du résultat (instruction `printf` par exemple), ce qui est inutile avec le toplevel.
2. Évaluez quelques expressions simples (du cours par exemple), puis plus compliquées. Pour ces dernières, le toplevel n'ayant aucune facilité d'édition de texte, il est fortement conseillé de se placer dans la fenêtre Xemacs et d'y taper vos expressions. Recopiez-les ensuite dans le toplevel (sélectionnez en maintenant l'appui sur le bouton gauche de la souris, placez le pointeur de la souris dans la fenêtre du toplevel, puis cliquez sur le bouton du milieu).

   Note : *Il se peut, par exemple lorsque vous entrez plusieurs expressions mal formatées à la fois, que le toplevel dysfonctionne. En cas de problème, entrez simplement deux point-virgules sous le prompt et tapez la touche* `Return`.
3. Évaluez la fonction calculant les racines d'un polynôme.
4. Comparer les deux expressions suivantes (sans le `#`) :

```
#let s1 =
#  fun r ->
#    let pi = acos (-1.) in
#    pi *. r *. r;;
val s1 : float -> float = <fun>

#let s2 =
#  let pi = acos (-1.) in
#  fun r -> pi *. r *. r;;
val s2 : float -> float = <fun>
```

   Laquelle des deux est plus efficace quand on l'utilise ?
5. Pour quitter le toplevel, vous pouvez soit faire `^d` ou entrer la commande :

   ```
   # quit;;
   ```
6. Terminez votre session.