Notation infixe, notation postfixe

Florian Bourse

1 But du projet

Le but du projet est d'implémenter une calculatrice pour évaluer des expressions arithmétiques écrites en notation infixe (les opérateurs binaires se trouvent entre les deux opérandes, notation habituelle) utilisant les opérateurs `+`, `-`, `*`, `/`, `^`, `!` avec la priorité usuelle, et des parenthèses `(...)` pour changer l'ordre d'évaluation des expressions.

Cette évaluation doit se décomposer en 2 phases :

1. Un programme écrit en langage C, qui lit sur l'entrée standard l'expression en notation infixe, et qui la traduit en expression utilisant la notation postfixe, ou notation polonaise inversée, où les opérateurs se trouvent après les opérandes. Cette notation ne requiert pas de parenthèses.
2. Un programme écrit en langage OCaml, qui lit sur l'entrée standard l'expression en notation postfixe, et qui l'évalue et affiche le résultat.

L'enchaînement des deux programmes, en utilisant le pipe `|` permet d'obtenir une calculatrice fonctionnelle.

Une fois le programme terminé, il devient facile d'ajouter des fonctions et opérateurs si on en a envie.

2 Conversion d'une expression de la notation infixe vers la notation postfixe

Les deux principales difficultées pour écrire le programme demandé consistent en la lecture/écriture sur les flux standards (entrée/sortie), et la gestion de l'ordre des opérateurs/nombres à écrire.

2.1 Lecture/écriture d'un caractère

Pour lire sur l'entrée standard, on peut utiliser la fonction `scanf` qui prend en argument une chaîne de caractères de formattage, qui indique ce qu'elle doit chercher. Par exemple, la chaîne de formattage `"B%d"` lit sur l'entrée standard le caractère `B` suivi d'un nombre entier. Ici, nous allons nous restreindre à lire caractère par caractère, en utilisant la chaîne de formattage `"%c"`. Si `char ici` est une variable de type `char` qui a été déclarée, `scanf("%c", &ici)` essaye de lire un caractère, et si elle réussit, modifie la valeur pointée par `&ici`, donc la variable `ici` pour qu'elle prenne la valeur du caractère lu. La fonction `scanf` renvoie la constante `EOF` si la lecture a échoué, sinon le nombre d'arguments qu'elle a réussit à lire et à affecter. Voici un exemple de son utilisation.

Exemple de lecture de caractères sur l'entrée standard — C

```c
char c; int cpt = 0;
while (scanf("%c",&c) != EOF && c != '\n') {
/* L'ordre des opérandes garanti que la seconde partie du && */
/* n'est évaluée que si la partie gauche est vraie */
/* On s'arrête si la lecture a échouée, ou au changement de ligne */
        if (c == 'x') {
                cpt = cpt + 1; // compte le nombre de 'x'
        }
}
printf("%i\n",cpt); // affiche l'entier cpt
```

La commande `printf` permet d'écrire sur la sortie standard, elle prend en premier argument une chaîne de formattage, comme `scanf`. Si la chaîne de formattage contient des arguments spéciaux (par exemple `%i`, ou `%c`), la commande `scanf` prend des arguments supplémentaires correspondant au types qui conviennent et remplace `%i` ou `%d` par un entier, `%c` par un caractère, ou `%s` par une chaîne de caractère par exemple.

Dans l'exemple ci-dessus, `printf("%i\n",cpt)` affiche l'entier `cpt`, suivi d'un retour à la ligne. Car `%i` est remplacé par `cpt`.

2.2 Ordre des opérandes

Pour plus d'information sur la résolution de cette tâche, on peut se renseigner sur l'algorithme Shunting-yard (*i.e., de la gare de triage*) inventé par Edsger Dijkstra (nom que l'on reverra dans le cours).

3 Évaluation d'une expression en notation polonaise inversée

Quelques rappels pour la lecture des données en OCaml.

Exemple de lecture de l'entrée

```
let l = read_line ();;
(* l contient la prochaine ligne sans le '\n' *)
try print_int (int_of_string l * 2) (* affiche le double *)
with | Failure _ -> print_string (String.sub l 0 1)
(* sinon le premier char *)
```

La conversion d'une chaîne de caractère en entier lève l'exception `Failure "int_of_string"` si le chaîne de caractère ne contient pas uniquement un entier, et la lecture `read_line ()` lève l'exception `End_of_file` si il n'y a plus de ligne à lire dans le fichier.

Si le programme lit l'entrée standard sur le clavier, on peut envoyer le signal de fin de fichier (EOF) avec les touches ctrl + D (aussi noté ^D).

4 Rendre son projet

1. Créer un dossier `Calc_Nom_prenom`, contenant les codes sources (.c et .ml), ainsi que tout fichier qui vous semble important pour l'évaluation de votre travail (README.txt pour expliquer comment utiliser le programme, jeux de tests (entrées/sorties attendues), .pdf expliquant les différents choix effectués pour l'implémentation, les prochaines étapes pour améliorer ce programme, la documentation/analyse des différentes fonctions créées).
2. Créer le fichier `Calc_nom_prenom.tar.gz` à l'aide de la commande

   ```
   tar cvfz Calc_nom_prenom.tar.gz Calc_nom_prenom
   ```

3. L'envoyer en pièce jointe d'un courrier électronique respectueux dont l'objet est pertinent à l'adresse

   `florian.bourse(at)ac-lille.fr`