ODLU MIAGE 1 :
Structures de contrôle

Janvier 2005

1 Construction de nos fonctions d'entrées-sorties

Exercice 0.— Les fonctions putchar et getchar.

La fonction `int putchar (int c);` prend en entrée un entier représentant le code ASCII d'en caractère et affiche ce caractère sur la sortie standard.

La fonction `int getchar (void);` récupère un caractère dans l'entrée standard et le retourne.

L'usage de ces fonctions nécessite l'ajout de la directive `#include <stdio.h>` à votre code.

Construisez un programme qui permette à l'utilisateur de saisir une lettre minuscule au clavier et affiche la majuscule correspondante à l'écran (ne vous occupez pas de la gestion d'erreur).

Même exercice mais la lettre est supposée être stockée dans un fichier et la majuscule doit être stockée dans un autre (utilisez les redirections du shell).

Exercice 1.— Saisie d'entier avec getchar.

Construire un programme dont la fonction principale permette de saisir un entier stocké dans l'entrée standard (sous forme d'une chaîne de caractères terminée par un retour chariot) et qui le retourne. Stocker votre chaîne de caractères en utilisant une commande interne du shell. Puis, en utilisant une variable prédéfinie du shell, vérifiez que votre programme marche correctement (attention, la variable shell $? est codée non signée sur un octet).

Exercice 2.— Affichage d'entier avec putchar.

Construire un programme dont la fonction principale permette d'afficher un entier machine stocké dans une variable.

2 Pretty printer

Cet exercice est tiré des notes de Ph. Marquet.

Le but de ce TP est de développer un *filtre* en langage C. On appelle filtre un programme qui lit un texte sur l'entrée standard (stdin) et qui sort un texte sur la sortie standard (stdout), avec éventuellement quelques modifications. Le plus simple des filtres est la commande Unix cat qui lit stdin et écrit le même texte sur stdout.

Le filtre développé durant ce TP est appelé *pretty-printer* (on nommera le fichier source pp.c et l'exécutable pp). Il permet de mettre en forme un fichier texte contenant un programme C. On se limitera à une version simplifiée qui s'occupe uniquement de l'indentation et des commentaires.

2.1 Spécification de la commande pp

Indentation À chaque accolade ouvrante, on passera à la ligne suivante et on incrémentera l'indentation courante (par défaut, on considérera qu'une indentation vaut 4 blancs). À chaque accolade fermante, on ira aussi à la ligne après avoir décrémenté l'indentation. Tout début effectif de ligne se fera au niveau de l'indentation courante (attention à la lecture de blancs ou de tabulations '`\t`' en début de ligne).

Commentaires On placera les commentaires en début de ligne, au niveau de l'indentation courante. On se limitera à un commentaire par ligne. Quand une fin de ligne ('`\n`') apparaît dans un commentaire, on fermera ce commentaire et on en ouvrira un second sur la ligne suivante.

Erreur En cas d'erreur (commentaire non fermé ou texte mal *accoladé*), on sortira un message d'erreur sur stderr, tout en continuant le formatage. En fin de formatage, on pourra afficher un message d'avertissement si les nombres d'accolades ouvrantes et fermantes ne semblent pas correspondre. L'exécution de pp se terminera alors sur un échec (EXIT_FAILURE).

Attention

- une accolade dans un commentaire doit être ignorée.
- aucune modification ne doit être faite sur une ligne commençant par une directive de cpp (#define, #include... ou d'autres lignes commençant par '#').
- aucune modification ne doit être faite à l'intérieur des chaînes de caractères littérales ("blabla"). On pourra, dans un premier temps, considérer qu'il n'y a pas de guillemets dans une chaîne littérale.

Exemple À partir du fichier file.c suivant :

```
#include <stdio.h>
        /* Ce programme C ne fait pas grand chose */

void main() {
   int n;
     char c;

       c = getchar(); /* on lit un caractere */ /* sur stdin */

if (c==' ') { n++;putchar(c);}
     else /* sinon,
             on ne fait rien */
      { ;}
}
```

la ligne de commande

```
% pp < file.c > file-i.c
```

va créer un fichier file-i.c qui contiendra :

```
#include <stdio.h>
/* Ce programme C ne fait pas grand chose */

void main()
{
    int n;
    char c;

    c = getchar();
```

```
    /* on lit un caractere */
    /* sur stdin */

    if (c==' ')
    {
        n++;putchar(c);
    }
    else
    /* sinon, */
    /* on ne fait rien */
    {
        ;
    }
}
```

2.2 Codage d'un automate

Pour faciliter la conception, on peut coder le programme pretty-printer par un automate. Par exemple, l'automate utilisé pour supprimer les espaces ou les tabulations en début de chaque ligne comporte deux états et les transitions suivantes entre ces deux états :

Exercice 0.— Construction d'un programme à partir d'un automate.

Inspirez vous de cet automate pour écrire un filtre qui supprime les espaces ou les tabulations en début de chaque ligne.

2.3 Le travail restant à faire

Construisez l'automate codant nos règles de styles et implantez le.