Licence d'informatique
Module de C/Unix

Travaux pratiques de programmation en C : le pretty-printer

Nouvelle révision, novembre 2000

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`, la commande `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.

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 considèrera qu'une indentation vaut 3 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'`) apparait 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.

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.

Test Vous pourrez tester votre pretty-printer par la suite de commandes Unix :

```
cc -o pp pp.c
pp < pp.c > pp2.c
cc -o pp2 pp2.c
pp2 < pp2.c > pp3.c
diff pp2.c pp3.c
```

Id: tpcprettyp.tex,v 1.7 2001/11/22 17:32:04 marquet Exp

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 > mon_beau_file.c
```

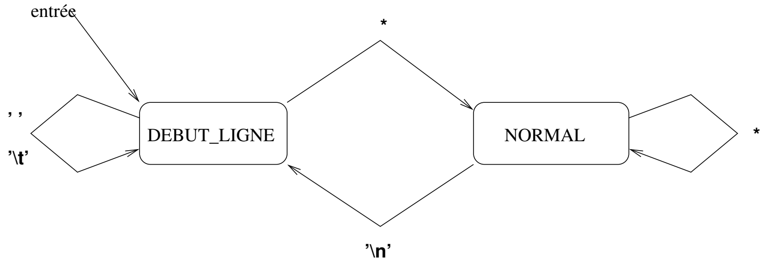
va créer un fichier `mon_beau_file.c` qui contiendra :

```
#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 */
   {
      ;
   }
}
```

Un automate

On peut coder le programme pretty-printer par un automate.

Pour ce faire, vous pouvez vous inspirez du programme suivant : on écrit un filtre qui supprime les espaces ou les tabulations en début de chaque ligne. L'automate utilisé comporte deux états et des transitions entre ces deux états :




```
#include <stdio.h>

main()
{
  int c;
  enum {ETAT_DBT_LIGNE, ETAT_NORMAL } etat = ETAT_DBT_LIGNE;

  while ((c=getchar()) != EOF) {
    switch (etat) {
      case ETAT_DBT_LIGNE:
        switch (c) {
          case ' ':
          case '\t':
            break;
          default:
            putchar(c);
            etat=ETAT_NORMAL;
            break;
        }
        break;
      case ETAT_NORMAL:
        switch (c) {
          case '\n':
            putchar('\n');
            etat=ETAT_DBT_LIGNE;
            break;
          default :
            putchar(c);
            break;
        }
        break;
    }
  }
}
```