Linguaggi e Traduttori
a.a. 2005/2006
Tesina n° 2

Titolo: Format C
Descrizione

Si vuole costruire un traduttore che compie la formattazione di un file C.
Il file prodotto deve avere le seguenti caratteristiche:

- Un'indentatura è composta da 2 spazi. I caratteri di "*{*" fanno aumentare l'indentatura di altri 2 spazi, mentre il carattere "*}*" fa diminuire l'indentatura di 2 spazi.
  Es.
  ```
  main() {
    fz() {
      int a;
    }
  }
  ```
- Tutti i blocchi devono avere il carattere "*{*" nella stessa riga della keyword e separata da quest'ultima da uno spazio. Il carattere di "*}*" deve essere allineato alla keyword.
  Es.
  ```
  main() {
  }
  ```
- Il carattere "," deve essere seguito da uno spazio, tranne nel caso di commenti e di stringhe: caratteri racchiusi da "".
  Es.
  ```
  int a, b;
  ```
- I caratteri =, ==, <=, >= e != sono preceduti e seguiti da uno spazio.
- Le dichiarazioni di variabili dovranno essere riscritte in ordine di tipo. L'ordine in cui dovranno essere riscritti i tipi è l'ordine alfabetico.
  Es.
  Il file:
  ```
  int a, b, c;
  struct pippo a, b;
  struct myStruct d;
  float a, c;
  myVar c;
  ```

  Dovrà essere così riscritto:
  ```
  float a, c;
  int a, b, c;
  myVar c;
  struct myStruct d;
  struct pippo a, b;
  ```
  NB: Nel caso che il primo tipo sia uguale, come nel caso di struct, viene ordinato alfabeticamente in base al secondo tipo
- I commenti devono essere riscritti così come sono nel file di ingresso. L'unico vincolo sui commenti è la gestione corretta dei rientri. Si consiglia in ogni caso di gestire i commenti tramite lo scanner, il quale accedendo ad una classe globale riesce a capire quanto vale il rientro attuale (rientro attuale che viene impostato dal parser).

- Come ultima parte dell'esercizio si richiede di scrivere un modulo che esegue il commento del codice. Ogni funzione deve essere preceduta da un commento che riporta il nome della funzione, il prototipo della funzione, le variabili passate alla funzione, quella restituita e un campo descrizione. I commenti dovranno essere iniziati dal carattere /*formatC per fare in modo che, al secondo utilizzo del traduttore, non vengano riscritti 2 volte i commenti alle funzioni, ma vengano riportati solo quelli aggiornati. Le dichiarazioni delle variabili dovranno essere precedute dal commento /*FormatC dich:<nome_funz>*/ mentre il codice dal commento /*FormatC code:<nome_funz>*/

  Es.

  Il file:

```
float* fz(int a, float* b, int c[]) {
        int a;
        printf("%d\n",a);
       return (float)a;
}
```

  Dovrà essere riscritto come:

```
/*formatC*****************************************************
 * NOME: fz
 * PROTOTIPO: float* fz(int a, float* b, int c[])
 * IN:
 * int a:
 * float *b:
 * int c[]:
 * OUT:
 * float* fz:
 * DESCRIZIONE:
 *
 ***********************************************************/
float* fz(int a, float* b, int c[]) {
  /*formatC dich:fz */
  int a;
  /*formatC code:fz */
  printf("%d\n",a);
  return (float)a;
}
```

Assieme al programma dovrà essere allegata una breve relazione contenente una veloce descrizione delle strutture dati utilizzate, di come il programma funziona e delle specifiche/limiti di funzionamento. Dovranno poi essere allegati dei **file di esempio** per verificare il corretto funzionamento del programma.

L'esercizio può essere risolto con una sola scansione del file di ingresso, in ogni caso soluzioni in cui si fanno due scansioni sono considerate corrette.