MODIFICA 1 per esercitazione 5

Scaricate la vostra esercitazione da gitlab. Create subito un **branch su GitLab** con il vostro nome e cognome (ad es. michele-rossi) e fatene il checkout sulla vostra macchina. Verificate di star lavorando sul branch appena creato (questo è il branch su cui andrete a fare il commit alla fine).
Il commit va fatto una sola volta alla fine **ed in presenza del docente**.

Assicuratevi che le vostre modifiche eseguano correttamente sugli esempi dati in questa traccia ed altri da voi creati.

Si modifichi il compilatore NewLang per aggiungere la seguente istruzione **init-do-for-step**:

`init` **(** *init-expression*$_{opt}$ **)** **`do`** {*statements*} **`for`** **(***cond-expression*$_{opt}$ **)** **`step`** (*loop-expression*$_{opt}$) ;

La *init-expression*$_{opt}$ può essere nulla o avere una lista di identificatori (separati da virgola) iniziati con un solo nome di tipo.
La *cond-expression*$_{opt}$ è una qualsiasi espressione booleana o nulla.
La *loop-expression*$_{opt}$ è una lista di espressioni separate da virgola o nulla.

Si noti che questo aggiunge un altro livello di scoping.

Esempio valido:

```
init (integer i << 1, j << 1)
do {
        ("ciao, ") -->;
        (i)-->;
        (", ") -->;
        (j)-->!;
   }
for(i < 20 and j >-20)
step(i<<i+1, j<<j-1);
```

che nella traduzione in C diventa

```
{
     int i = 1, j=1;
     do{
       printf("ciao, %d, %d\n", i, j); // o una serie di printf
       i=i+1; j=j-1;
     }while (i < 20 && j >-20);
}
```

Altri esempi:

```
_____________________
init ()
do {}
for()
step();
Che diventa in C -> {do{}while(1);}
_____________________

init(var i<< 1.1)
do{
        ("ciao, ") -->;
        (i)-->!;    }
```

```
for(i < 10.1)
step (i<<i+1);
```

Che diventa in C:

```
{float i = 1.1;
do{
     printf("ciao, %f\n", i);
     i=i+1;
} while(i<10.1);}
```