Макроассемблер. Локальные метки 2

Макрогенерация представляет собой развитый механизм замены текста. При использовании макросредств с помощью макрокоманд в текст программы можно вставлять последовательности строк, которые логически могут быть данными или командами.

Предположим, что в макросе содержатся локальные метки. В таком случае, когда макрокоманда вызывается несколько раз, то в процессе макрогенерации возникает ситуация, когда в программе один идентификатор (в нашем случае метка) будет определён несколько раз, что, естественно, будет распознано транслятором как ошибка. Для выхода из подобной ситуации в ассемблере предусмотрена директива local, имеющая следующий синтаксис:

local идентиф1, идентиф2....идентифN

Данная директива задаётся непосредственно за заголовком макроопределения. Результат работы данной директивы будет генерация в каждом экземпляре макрорасширения уникальных имён для всех идентификаторов, в след виде ??xxxx, где xxxx означает шестнадцатеричное число.

Замена меток ведётся следующим образом:
К примеру, в макроопределении содержится некоторое описание меток:

```
local метка1, метка2, ... меткаN
```

В таком случае при первом вызове макрокоманды метка1 заменяется на ??0000, метка2 на ??0001, метка11 на ??000a и.т.д. При следующих вызовах замены осуществляются аналогично, с толь лишь только разницей что к идентификатору прибавляется общее количество меток предыдущих вызовов.

Написать макроподстановщик, реализующий директиву local.

Входные данные: На вход подаётся листинг в следующем виде:

```
ммя_макроса1 macro
local метка1, метка2, ... меткаN
метка1:
       команда1 пар1, пар2 <или> команда_усл/безусл_перехода метка1..N
       команда2 пар1, пар2 <или> команда_усл/безусл_перехода метка1..N
       ..................
       командаN пар1, пар2 <или> команда_усл/безусл_перехода метка1..N
метка2:
       -//-
. . . . . . . . . . . .
меткаN:

endm
exit macro

ммя_макроса2 macro
-//-
```

```
. . . . . .
ммя_макросаN macro
-//-

main proc
     команда1 пар1, пар2  <или> имя_макроса[1..N]
     . . . . . .
     командаN пар1, пар2  <или> имя_макроса[1..N]
main endp
```

Длины строк не более 80-и символов. Входной файл может содержать пустые строки, а идентификаторы и ключевые слова разделены любым количеством пробелов.

Выходные данные: На выходе должен быть листинг следующего вида:

```
. . . . . .
??xxxx:
 команда1 пар1, пар2 <или> команда_усл/безусл_перехода ??xxxx
 команда2 пар1, пар2 <или> команда_усл/безусл_перехода ??xxxx
 . . . . . .
 командаN пар1, пар2 <или> команда_усл/безусл_перехода ??xxxx
. . . . . .
```

Причём метки (??xxxx) выводятся с начала строки, а команды с отступом в один пробел.

Пример:

Sample input

```
one macro
 local m1, m2, m3
m3:
     mov ax, bx
m1:
     mov dx, bx
     mov cx, ax
     cmp ax, bx
     jne m1
m2:
     mov dx, bx
     cmp ax, bx
     jne m2
     mov cx, ax
     cmp ax, bx
     je m3
endm
exit macro

two macro
 local m1, m2, m3
m3:
     cmp ax, bx
m1:
     cmp dx, bx
     cmp cx, ax
     cmp ax, bx
     jne m1
m2:
     cmp dx, bx
     cmp ax, bx
     jne m2
```

```
     cmp cx, ax
     cmp ax, bx
     je m3
endm
exit macro

main proc
     mov ax, bx
     two
     mov ax, bx
     one
main endp
```

Sample output

```
 mov ax, bx
??0002:
 cmp ax, bx
??0000:
 cmp dx, bx
 cmp cx, ax
 cmp ax, bx
 jne ??0000
??0001:
 cmp dx, bx
 cmp ax, bx
 jne ??0001
 cmp cx, ax
 cmp ax, bx
 je ??0002
 mov ax, bx
??0005:
 mov ax, bx
??0003:
 mov dx, bx
 mov cx, ax
 cmp ax, bx
 jne ??0003
??0004:
 mov dx, bx
 cmp ax, bx
 jne ??0004
 mov cx, ax
 cmp ax, bx
 je ??0005
```