Προεπεξεργαστής της C

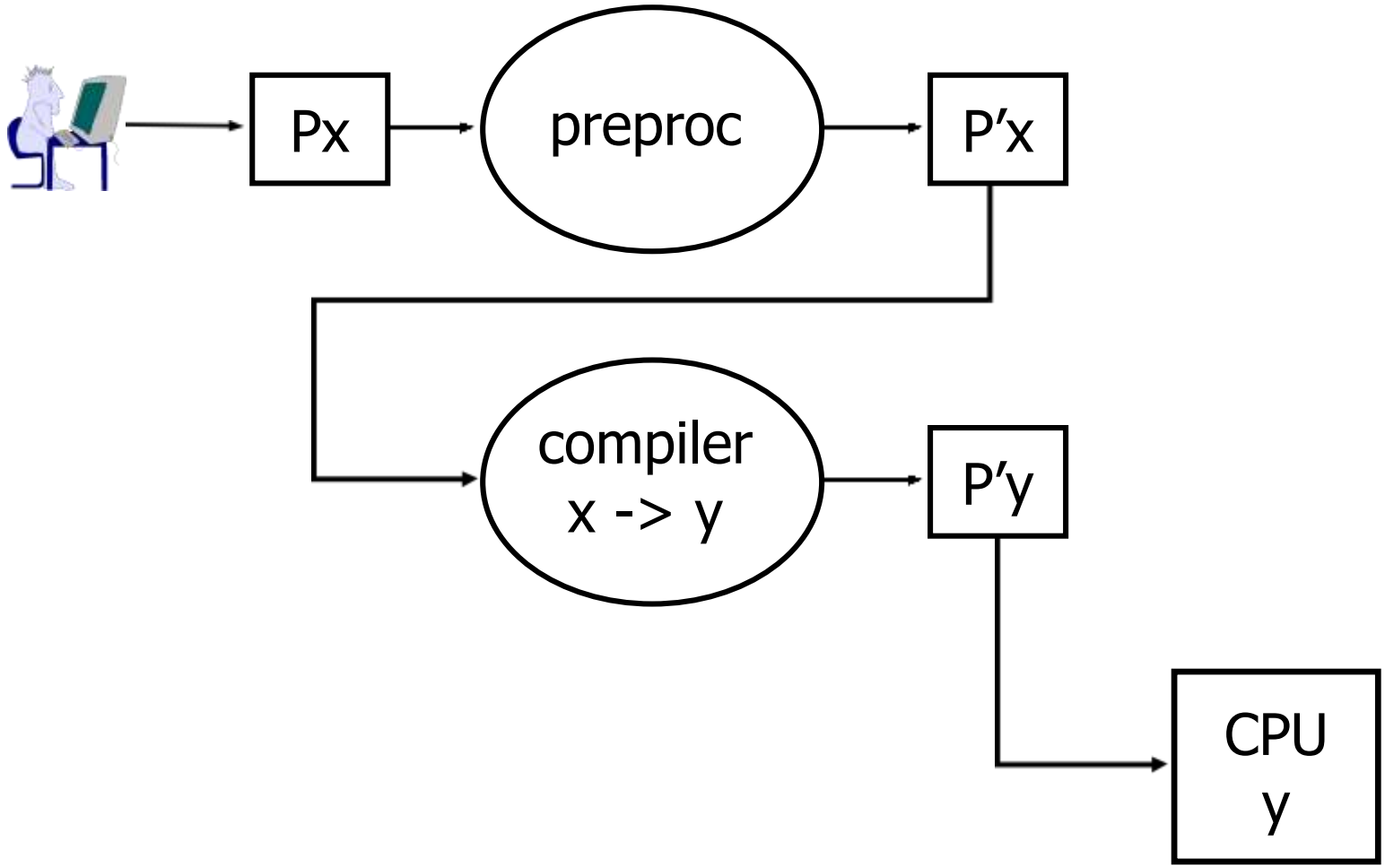
C Preprocessor

Τι κάνει ο προεπεξεργαστής;

- Ο προεπεξεργαστής (pre-proccesor) της C είναι ένα πρόγραμμα που μετασχηματίζει τον πηγαίο κώδικα
  - προτού περαστεί στον μεταγλωττιστή (gcc -E)
- Πραγματοποιεί **αλλαγές σε επίπεδο κειμένου**, με την μορφή «απλών» αντικαταστάσεων χαρακτήρων
- **Δεν** γίνεται έλεγχος της συντακτικής ορθότητας

Οι πιο χαρακτηριστικές χρήσεις:

- μακροεντολές
- εισαγωγή κειμένου υπό συνθήκη
- αντιγραφή κειμένου από άλλα αρχεία


Px
preproc
P'x
compiler
x -> y
P'y
CPU
y

Ορισμός μακροεντολών

- `#define <macro> <expr>`
- Ορίζει μακροεντολή `<macro>` που θα **αντικατασταθεί** (κάθε φορά που εντοπίζεται στον κώδικα του προγράμματος) με την έκφραση `<expr>`
- Οι μακροεντολές **δεν** είναι συναρτήσεις
- **Σημείωση 1:** χρειάζεται προσοχή (παρενθέσεις) έτσι ώστε να αποφεύγονται **λάθη με τελεστές** που ακολουθούν στο κείμενο του προγράμματος
- **Σημείωση 2:** χρειάζεται προσοχή με «παραμέτρους» των macros καθώς αυτές **αποτιμώνται κάθε φορά** που εμφανίζονται στην έκφραση

```
/* symbolic constant */

#define N 100
…
int t[N];
…
for (i=0; i < N; i++ ) {
  …
}
…
if (i == N) {
  …
}
…
```

```
/* symbolic constant, one more time */

#define N 10*10
…
int t[N];
…
for (i=0; i < N; i++ ) {
  …
}
…
if (i == N) {
  …
}
…
```

```
/* sequence of symbolic constants */

#define A 10

#define B 20

#define C (A+B)

#define D (C*B)
```

```
/* macro */

#define LOWERCASE(a) (a- 'A' + 'a')

int main(int argc, char *argv[]) {
  char c;

  do {
    c=getchar();
    if ((c >= 'A') && (c <= 'Z')) { c=LOWERCASE(c); }
    putchar(c);
  } while (c != '\n');

  return(0);
}
```

```
/* macro, differently  */

#define LOWERCASE(a) (((a>='A') && (a<='Z')) ? a-'A'+'a' : a)

int main(int argc, char *argv[]) {
  char c;

  do {
    c=getchar();
    putchar(LOWERCASE(c));
  } while (c != '\n');

  return(0);
}
```

```
/* macro, yet another option */

#define PUTLOWERCASE(a) ( \
  { if ((a<'A') || (a>'Z')) { putchar(a); } \
    else { puchar(a-'A'+'a'); } \
  } \
)

int main(int argc, char *argv[]) {
  char c;

  do {
    c=getchar();
    PUTLOWERCASE(c);
  } while (c != '\n');

  return(0);
}
```

```
/* the wrong way: too lazy to use brackets */

#define PLUSONE(x) x+1

int main(int argc, char *argv[]) {
  int i,j;

  scanf("%d",&i);
  j=PLUSONE(i)*2;
  printf("%d\n",j);

  return(0);
}
```

```
/* the right way */

#define PLUSONE(x) (x+1)

int main(int argc, char *argv[]) {
  int i,j;

  scanf("%d",&i);
  j=PLUSONE(i)*2;
  printf("%d\n",j);

  return(0);
}
```

```
/* repeated evaluation of pseudo-parameters */

#define SQUARE(a) ((a)*(a))

int main(int argc, char *argv[]) {
  int i,j;

  scanf("%d",&i);
  j=SQUARE(++i);
  printf("%d\n",j);

  return(0);
}
```

Προ-ορισμένες μακροεντολές

`__DATE__`	date of compilation as a string literal in "MMM DD YYYY" format
`__TIME__`	time of compilation as a string literal in "HH:MM:SS" format
`__FILE__`	source code filename as a string literal
`__func__`	function name as a string literal
`__LINE__`	line number in the source file as a decimal constant
`__STDC__`	equal to the decimal value 1 if the compiler complies with the ANSI standard

```c
#include <stdio.h>

int main (int argc, char *argv[]) {
  printf("File: %s\n", __FILE__ );
  printf("Date: %s\n", __DATE__ );
  printf("Time: %s\n", __TIME__ );
  printf("Function:%s\n", __func__);
  printf("Line: %d\n", __LINE__ );
  printf("ANSI: %d\n", __STDC__ );

  return(0);
}
```

```
File: predefined.c
Date: Dec 21 2014
Time: 23:57:58
Function: main
Line: 8
ANSI: 1
```

Stringize (#)

- Μετατρέπει παράμετρο μακροεντολής σε string

```
#include <stdio.h>

#define  message_for(a, b, c)  \
         printf(#a " and " #b " and " #c ": hello\n")

int main(int argc, char *argv[]) {
   message_for(Carole, Debra, 42);
   return 0;
}
```

```
Carole and Debra and 42: hello
```

Συνένωση tokens (##)

- Συνενώνει παραπάνω από μία παραμέτρους της μακροεντολής σε μία

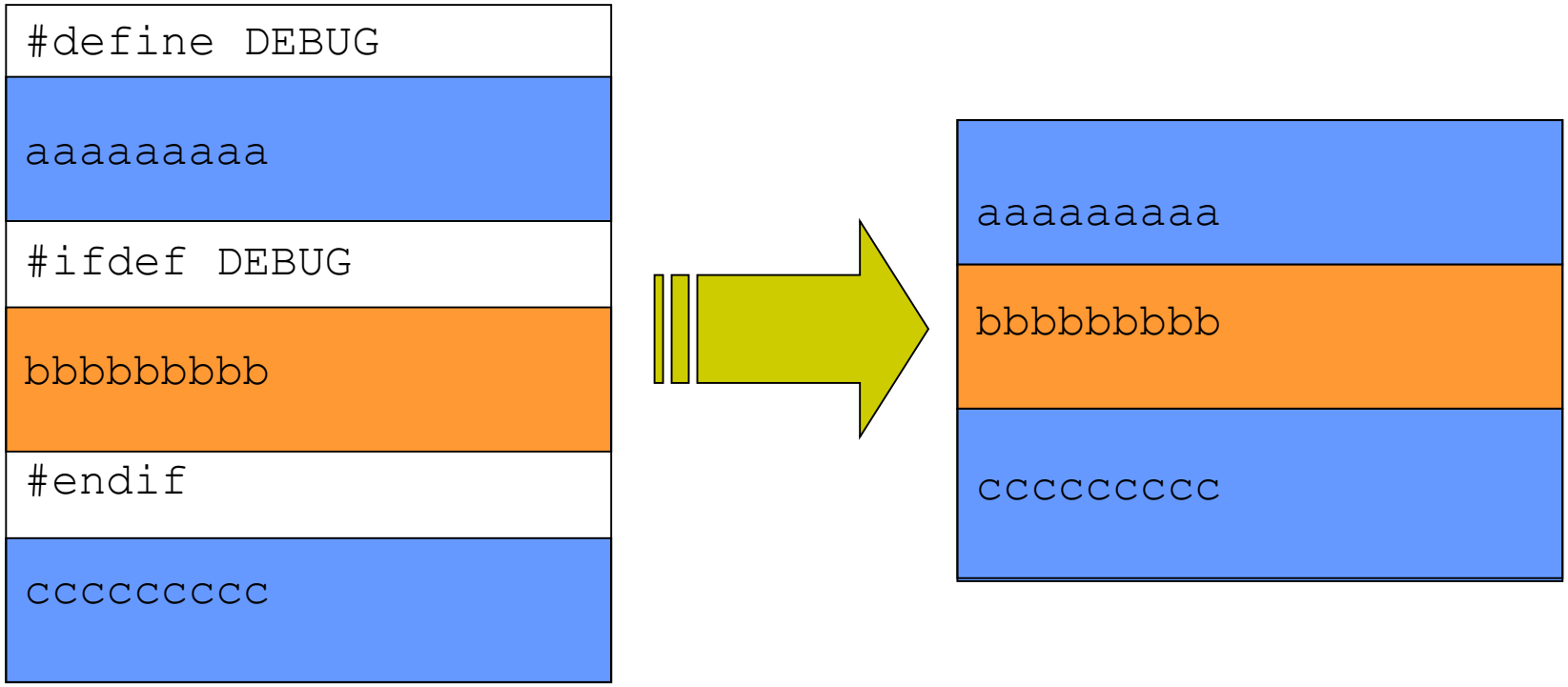
```
#include <stdio.h>

#define tokenpaster(n) \
        printf ("token%s = %d\n", #n, token##n)

int main(int argc, char *argv[]) {
   int token34 = 40;

   tokenpaster(34);
   return 0;
}
```

```
token34 = 40
```

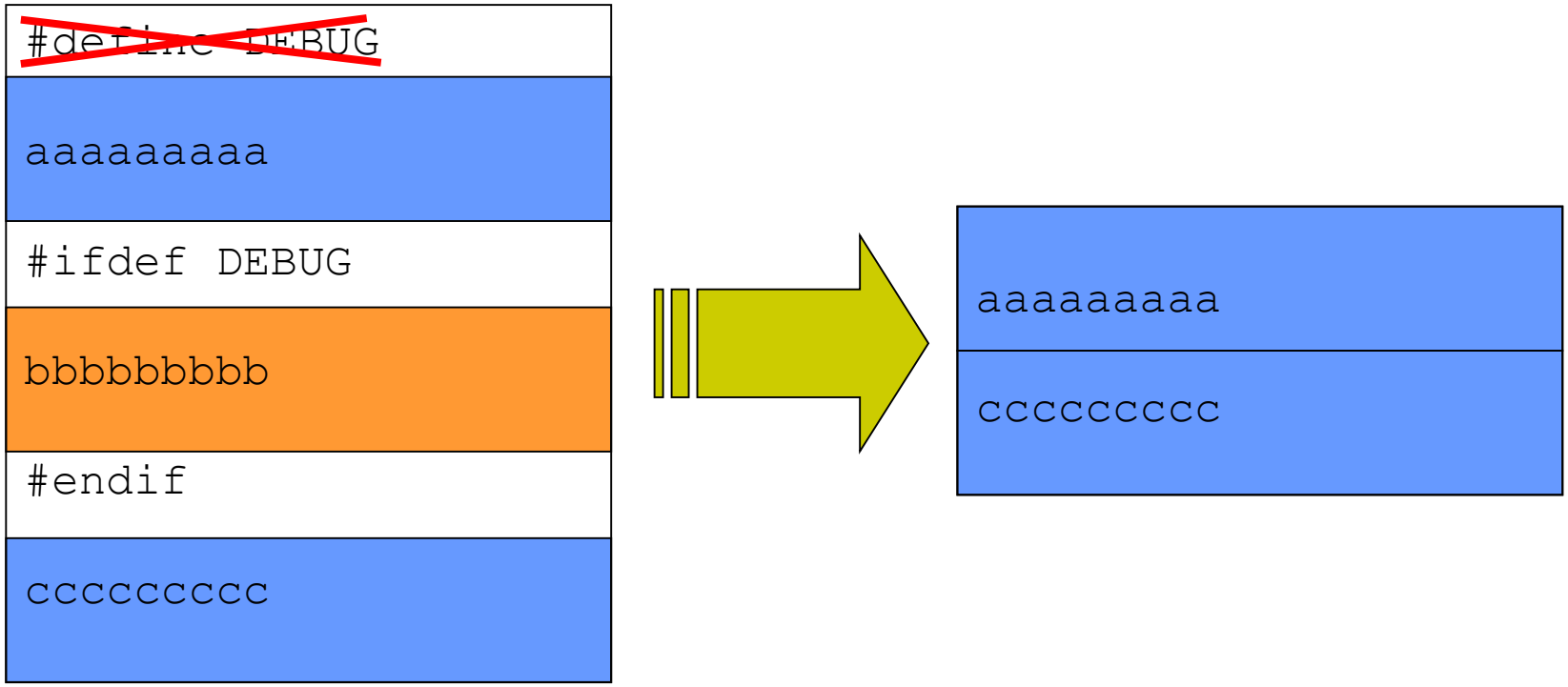
Εισαγωγή υπό συνθήκη

- `#if, #else, #elif, #endif`
- `#ifdef, #ifndef`
- Συνθήκες πάνω σε συμβολικά ονόματα
- Αν μια συνθήκη ισχύει τότε το αντίστοιχο κείμενο συμπεριλαμβάνεται για μετάφραση, διαφορετικά όχι
  - το κείμενο που συμπεριλαμβάνεται / αφαιρείται μπορεί να είναι οτιδήποτε (δεν χρειάζεται να αντιστοιχεί σε κάποια συντακτική ενότητα)
- Κλασικές χρήσεις:
  - επιλογή ανάμεσα σε διαφορετικές εκδόσεις του κώδικα (κώδικας αποσφαλμάτωσης, υποστήριξη για διαφορετικές πλατφόρμες κλπ)
  - αποφυγή διπλής εισαγωγής κειμένου (βλέπε `#include`)

```
#define DEBUG

…

#ifdef DEBUG   /* ή #if defined(DEBUG) */

… /* my debugging code */

#endif

…
```


#define DEBUG
aaaaaaaaa
#ifdef DEBUG
bbbbbbbbb
#endif
ccccccccc
aaaaaaaaa
bbbbbbbbb
ccccccccc

```
#define DEBUG

…

#ifdef DEBUG   /* ή #if defined(DEBUG) */

… /* my debugging code */

#endif

…
```


#define DEBUG
aaaaaaaaa
#ifdef DEBUG
bbbbbbbbb
#endif
ccccccccc
aaaaaaaaa
ccccccccc

```c
// test.c
#include <stdio.h>

int main(int argc, char *argv[]) {
#ifdef DEBUG
   printf("Debug code\n");
#else
   printf("Release code\n");
#endif
}
```

```
$ gcc -D DEBUG test.c -o test
$ ./test
Debug code
$
$ gcc test.c -o test
$ ./test
Release code
$
```

```c
// test.c
#include <stdio.h>

int main(int argc, char *argv[]) {
#if OPTION == 1
   printf("Code option 1\n");
#elif OPTION == 2
   printf("Code option 2\n");
#else
   printf("Default code\n");
#endif
}
```

```
$ gcc -D OPTION=2 test.c -o test
$ ./test
Code option 2
$
$ gcc test.c -o test
$ ./test
Default code
$
```

```
#ifndef INCL_X    /* ή #if !defined(INCL_X) */

#define INCL_X   /* avoid recursive inclusion */

… /* my code */

#endif
```

Εισαγωγή αρχείων

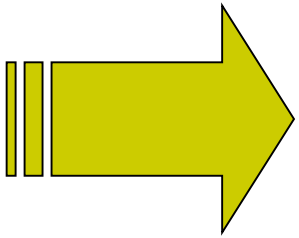
- `#include "filename"`

- Τα περιεχόμενα του αρχείου `filename` εισάγονται **ακριβώς στο σημείο** που εμφανίζεται η εντολή
  - αν το αρχείο δεν βρίσκεται στον κατάλογο εργασίας, ο κατάλογος πρέπει να δίνεται ως μέρος του ονομάτος
  - αν το όνομα δίνεται σε `< >` αντί `" "`, ο προεπεξεργαστής θεωρεί ότι το αρχείο βρίσκεται στον κατάλογο όπου υπάρχουν τα header files της C, π.χ. `/usr/include`

- Αν το αρχείο που εισάγεται περιέχει και αυτό με την σειρά του εντολές `#include` τότε γίνεται εισαγωγή **και αυτών των αρχείων** μέσα στο αρχείο

```
/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
aaaaaaaaaa
```

```
/* αρχείο file2 */

#include "file1"

bbbbbbbbbb
bbbbbbbbbb
bbbbbbbbbb
```

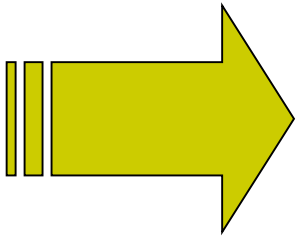


```
/* αρχείο file2 */

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
aaaaaaaaaa

bbbbbbbbbb
bbbbbbbbbb
bbbbbbbbbb
```

```
/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
aaaaaaaaaa
```

```
/* αρχείο file2 */

#include "file1"

bbbbbbbbbb
bbbbbbbbbb
bbbbbbbbbb
```

```
/* αρχείο file3 */

#include "file2"

cccccccccc
cccccccccc
cccccccccc
```



```
/* αρχείο file3 */

/* αρχείο file2 */

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
aaaaaaaaaa


bbbbbbbbbb
bbbbbbbbbb
bbbbbbbbbb

cccccccccc
cccccccccc
cccccccccc
```

```
/* myprog_min.h */
#define min(a,b) ((a)<(b) ? (a) : (b))
```

```
/* myprog_max.h */
#define max(a,b) ((a)>(b) ? (a) : (b))
```

```
/* myprog_main.c */

#include "myprog_min.h"
#include "myprog_max.h"

int main(int argc, char *argv[]) {
  int a,b;
  scanf("%d %d",&a,&b);
  printf("%d %d\n",min(a,b),max(a,b));

  return(0);
}
```

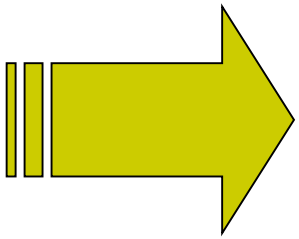
Αποφυγή επανειλημμένης εισαγωγής

- Ένα αρχείο μπορεί να εισάγει με την σειρά του (έμμεσα) επιπλέον αρχεία, τα οποία μπορεί να εισάγονται ξανά από το κυρίως πρόγραμμα
- Η διπλή εισαγωγή μπορεί να είναι ανεπιθύμητη
  - π.χ. διπλή εισαγωγή μεταβλητών και συναρτήσεων που οδηγεί (στη μετάφραση) σε συντακτικά λάθη
- Λύση 1: ο προγραμματιστής πρέπει να γνωρίζει ποιά αρχεία εισάγουν (έμμεσα) τα αρχεία που εισάγει στο πρόγραμμα του (άμεσα)
  - μη πρακτικό, άκομψο
- Λύση 2: το ίδιο το αρχείο φροντίζει να μην μπορεί να εισαχθεί δύο φορές στο ίδιο κείμενο
  - κλασική περίπτωση χρήσης του `#ifndef`

```
/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
```

```
/* αρχείο file2 */

#include "file1"

bbbbbbbbbb
bbbbbbbbbb
```

```
/* αρχείο file3 */

#include "file2"
#include "file1"

cccccccccc
cccccccccc
```

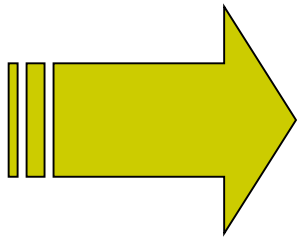


```
/* αρχείο file3 */

/* αρχείο file2 */

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa

bbbbbbbbbb
bbbbbbbbbb

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa

cccccccccc
cccccccccc
```

```
#ifndef INC_FILE1
#define INC_FILE1
/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa
#endif
```

```
/* αρχείο file2 */

#include "file1"

bbbbbbbbbb
bbbbbbbbbb
```

```
/* αρχείο file3 */

#include "file2"
#include "file1"

cccccccccc
cccccccccc
```



```
/* αρχείο file3 */
/* αρχείο file2 */
/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa

bbbbbbbbbb
bbbbbbbbbb

cccccccccc
cccccccccc
```

```c
/* myprog_min.h */
#ifndef INCL_MYPROGMIN
#define INCL_MYPROGMIN
#define min(a,b) ((a)<(b) ? (a) : (b))
#endif
```

```c
/* myprog_max.h */
#include "myprog_min.h"
#define max(a, b) ({\
  int res;\
  if ((a) == min((a),(b))) res = (b); else res = (a); \
  res; \
})
```

```c
/* myprog_main.c */
#include "myprog_min.h"
#include "myprog_max.h"

int main(int argc, char *argv[]) {
  int a,b;
  scanf("%d %d",&a,&b);
  printf("%d %d\n",min(a,b),max(a,b));
  return(0);
```