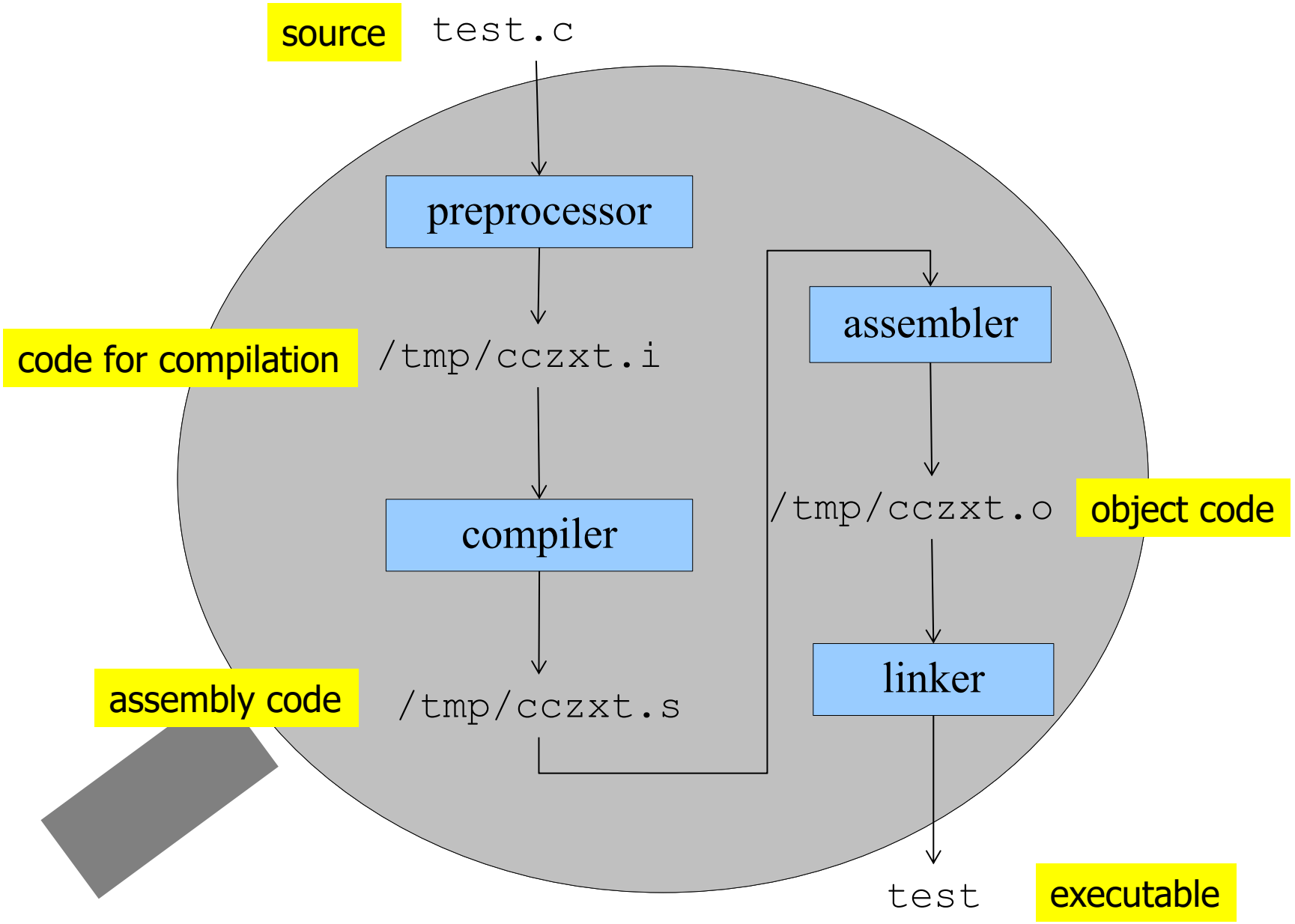
Προγραμματισμός II (ECE116)

#3 προεπεξεργαστής C (C preprocessor)

Σπύρος Λάλης
lalis@uth.gr


source
test.c
preprocessor
code for compilation
/tmp/cczxt.i
compiler
assembly code
/tmp/cczxt.s
assembler
/tmp/cczxt.o
object code
linker
test
executable

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**
  - «απλές» αντικαταστάσεις κειμένου
- **Δεν** γίνεται έλεγχος συντακτικής ορθότητας των εντολών του προγράμματος
- **Δεν** γίνεται μετάφραση σε γλώσσα μηχανής

Βασικές χρήσεις:

1. μακροεντολές
2. συμπερίληψη κειμένου υπό συνθήκη
3. εισαγωγή κειμένου από άλλα αρχεία

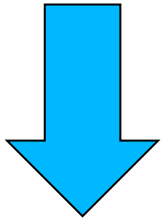
Macros

Ορισμός μακροεντολών

- `#define <macro> <expr>`
- Το `<macro>` θα **αντικατασταθεί**, οπουδήποτε αυτό εντοπίζεται στον κώδικα του προγράμματος, με την έκφραση `<expr>`

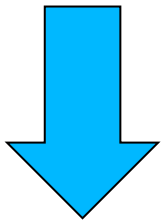
- Οι μακροεντολές **δεν** είναι συναρτήσεις
- **Σημείωση 1:** χρειάζεται προσοχή (παρενθέσεις) έτσι ώστε να αποφεύγονται **λάθη με τελεστές** που ακολουθούν στο κείμενο του προγράμματος
- **Σημείωση 2:** χρειάζεται προσοχή με «παραμέτρους» των macros καθώς αυτές **αποτιμώνται κάθε φορά** που εμφανίζονται στην έκφραση

```
/* symbolic constant */

#define N 25

for (i=0; i<2*N; i++ ) {
  printf("%d\n",i);
}
```



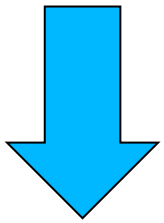
```
for (i=0; i<2*25; i++ ) {
  …
}
```

```
/* symbolic constant */

#define N 20+5

for (i=0; i<2*N; i++ ) {
  printf("%d\n",i);
}
```



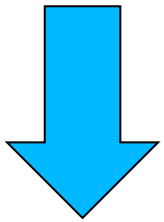
```
for (i=0; i<2*20+5; i++ ) {
  printf("%d\n",i);
}
```

```
/* symbolic constant */

#define N (20+5)

for (i=0; i<2*N; i++ ) {
  printf("%d\n",i);
}
```



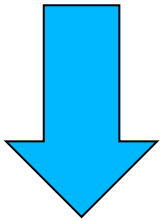
```
for (i=0; i<2*(20+5); i++ ) {
  printf("%d\n",i);
}
```

```
/* too lazy to use brackets */

#define PLUSONE(x) x+1

j = PLUSONE(i)*2;
```



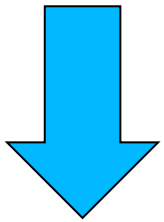
```
j = i+1*2;
```

```
/* the right way */

#define PLUSONE(x) (x+1)

j = PLUSONE(i)*2;
```



```
j = (i+1)*2;
```

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

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

j = SQUARE(i++);
```



```
j = ((i++)*(i++));
```

```
/* macro for conversion to lowercase, without check */

#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 for printing in lowercase, with check */

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

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

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

  return(0);
}
```

γιατί δεν χρειάζεται αυτό;

Προκαθορισμένα macros

`__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
```

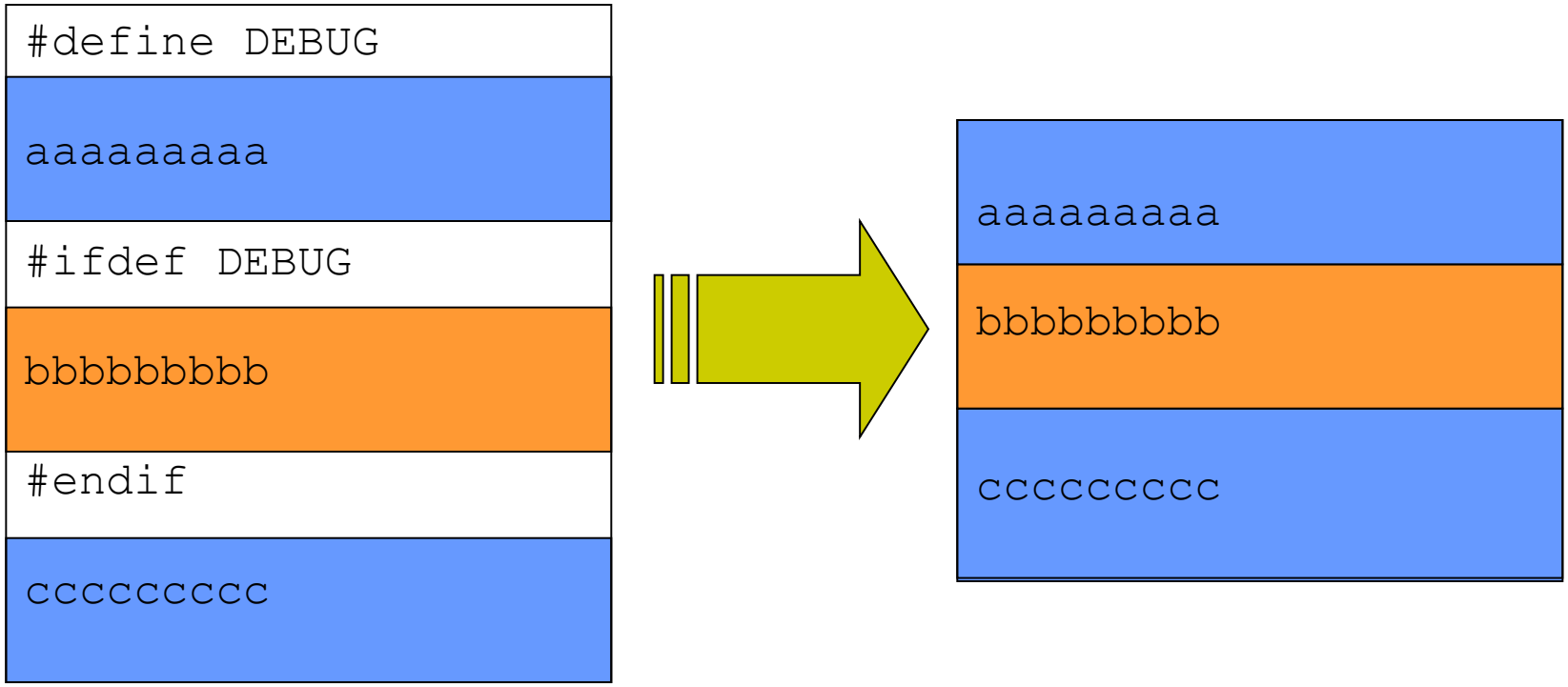
Conditional inclusion

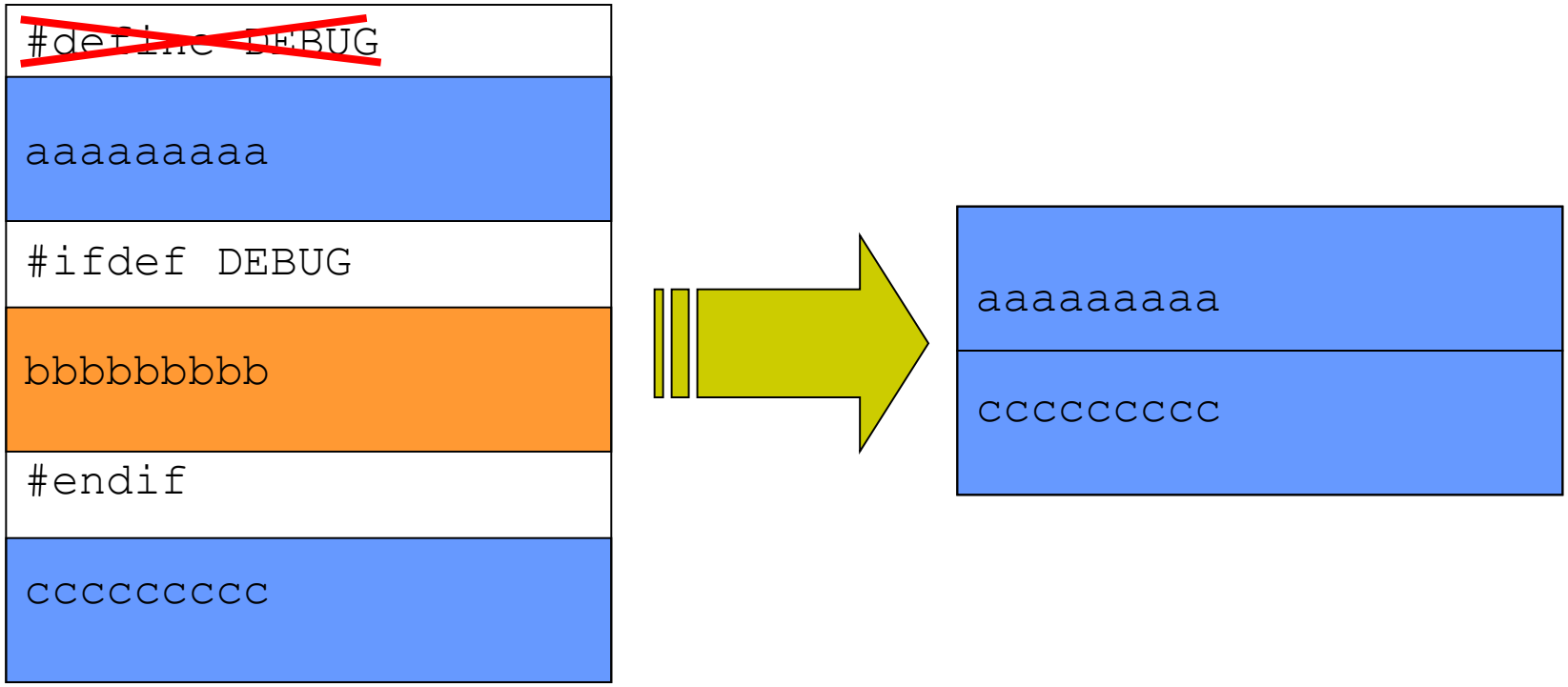
Συμπερίληψη κειμένου υπό συνθήκη

- `#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 (ή παραλείπεται) μπορεί να είναι οτιδήποτε – δεν χρειάζεται να αντιστοιχεί σε συντακτική ενότητα

Κλασικές χρήσεις:

- Επιλογή διαφορετικών εκδόσεων του κώδικα
  - κώδικας αποσφαλμάτωσης, διαφορετικές πλατφόρμες, …
- Αποφυγή διπλής εισαγωγής κειμένου
  - ιδιαίτερα χρήσιμο για το `#include` (βλέπε αργότερα)


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


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

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

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

```
$ gcc -D DEBUG test.c -o test
$ ./test
Debug message
Normal message
$
```

```
$ gcc test.c -o test
$ ./test
Normal message
$
```

```
// 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
$
```

File inclusion

Εισαγωγή αρχείων

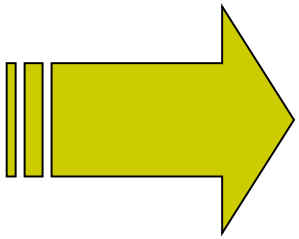
- `#include "filename"`

- Τα περιεχόμενα του αρχείου `filename` εισάγονται **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 `< >`, ο προεπεξεργαστής αναζητά το αρχείο στον συμβατικό κατάλογο για τα header files, 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
```

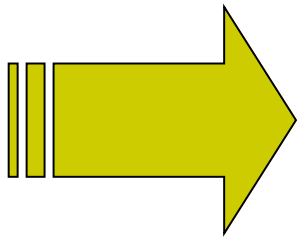


```
/* αρχείο file2 */

/* 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
```

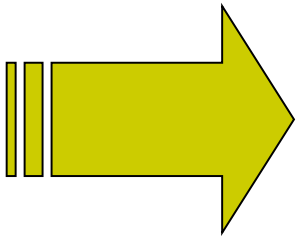
Αποφυγή επανειλημμένης εισαγωγής

- Ένα αρχείο μπορεί να εισάγει με την σειρά του (έμμεσα) επιπλέον αρχεία, τα οποία μπορεί να εισάγονται **ξανά** από το κυρίως πρόγραμμα
- Η πολλαπλή εισαγωγή μπορεί να είναι ανεπιθύμητη
  - π.χ. διπλή εισαγωγή δηλώσεων τύπων, μεταβλητών και συναρτήσεων που οδηγεί (στη μετάφραση) σε συντακτικά λάθη
- **Λύση 1:** ο προγραμματιστής γνωρίζει / ελέγχει ποια αρχεία εισάγουν (έμμεσα) τα αρχεία που εισάγει ο ίδιος στο πρόγραμμα του (άμεσα)
  - μη πρακτικό, άκομψο, ιδιαίτερα επιρρεπές σε λάθη
- **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
```

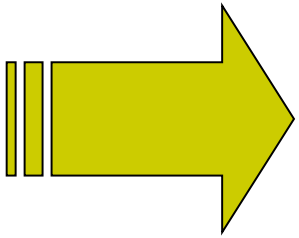


```
/* αρχείο file3 */

/* αρχείο file2 */

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa

bbbbbbbbbb
bbbbbbbbbb

/* αρχείο file1 */

aaaaaaaaaa
aaaaaaaaaa

cccccccccc
cccccccccc
```

```
#ifndef _FILE1
#define _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
```

```
/* my_min.c */
#ifndef __MYMIN
#define __MYMIN
int min(int a, int b) {
  if (a < b) return(a); else return (b);
}
#endif
```

τι θα γίνει αν παραληφθούν αυτές τις εντολές;

```
/* my_max.c */
#include "my_min.c"
int max(int a, int b) {
  if (min(a,b) == a) return(b); else return(a);
}
```

```
#include <stdio.h>
#include "my_min.c"
#include "my_max.c"

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);
}
```