

1 Module [ .c  
.h ]

- .h → #pragma once protection multi-include
- .c → implementation des fonctions

[ Trigo . c ]

void test (void) {

=  
}

[ errorManager . c ]

void test (void) {

=  
}

link → double définition de la fonction test FAILED

Solution: Restreindre la visibilité des 2 fonctions

**static**

void **test**(void) {

=  
}

**static**

void test (void) {

=  
}

Le "nom" tor est unique  
visible dans le module  
où il est déclaré **static**

Reprendre TD 20200505

Ajouter 2 fonctions b1 & b2 non static → **COMPILE FAILED**

Ajouter le mot **static**  
Verifier le compilateur ISNS

C trigo.c ×

```
TD > TD20200505 > C trigo.c > test(void)
33
34 void test(void) {
35
36     printf("[trigo.c | test] \n");
37 }
38
```

C errorManager.c ×

```
TD > TD20200505 > C errorManager.c > test(void)
35
36 void test(void) {
37
38     printf("[errorManager.c | test] \n");
39 }
```

```
gcc -std=c99 -Wall -g -Iinclude -c errorManager.c -o obj/errorManager.o -MMD -MF obj/errorManager.d
gcc -std=c99 -Wall -g -Iinclude -c trigo.c -o obj/trigo.o -MMD -MF obj/trigo.d
gcc -o app obj/errorManager.o obj/ TD20200505.o obj/trigo.o -lm
obj/trigo.o: In function `test':
/home/coder/project/TD/TD20200505/trigo.c:34: multiple definition of `test'
obj/errorManager.o:/home/coder/project/TD/TD20200505/errorManager.c:36: first defined here
collect2: error: ld returned 1 exit status
Makefile:23: recipe for target 'app' failed
make: *** [app] Error 1
```

Ajouter "static" devant !

```
static void test(void) {
```



## Classes de stockage

- ✓ • volatile
- ✓ • static
- ✓ • const
- ✓ • auto
- ✓ • register.

auto

auto int k=∅; → EQUIV.  
int k=∅;

constants

const double g=∅;

Valeur non modifiable

register

register uint8\_t dir=∅;  
force le compilateur à placer  
"dir" dans un registre CPU

Volatile

volatile uint8\_t sens=1;

static

① dans les fonctions

static int cpt=42;

- 1) cpt est initialisé à 42 bss au 1<sup>er</sup> appel de la fonction
- 2) la valeur de cpt est conservée d'un appel sur l'autre

PC



static (en-tu) pour les variables globales.

variables locales → déclarées DANS les fonctions  
globale → EN dehors

int32\_t z = 38;

void f(void) {

    int32\_t k = 0;

=

}

accessible dans toutes les fonctions

TD 2020 Q5Q5

① créer une variable global g (uint32\_t)  
initialisée à 42 dans le module fonction

② dans display() → meth g ← φ  
③ dans test → meth g ← 1

ISN 40

## Trigo.c

```
#include "trigo.h"
int32_t gT = 42; Variable globale

sPoint rotate(const sPoint p1, const double angle){
    sPoint p2 = {0., 0.};
    double alpha = angle * PI / 180.;

    gT = 0; gT = 0

    // rotate p1 around (0,0) with angle=alpha (radians)
    p2.x = p1.x * cos(alpha) - p1.y * sin(alpha);
    p2.y = p1.x * sin(alpha) + p1.y * cos(alpha);

    test(); gT = 1
    return p2;
}

void pointDisplay(const char *name, const sPoint p) {
    printf("%s: (%+.3lf, %+.3lf)\n", name, p.x, p.y);
    return;
}

static void test(void) {
    printf("[trigo.c | test]\n");
    gT = 1; gT = 1
    return;
}
```

$gT = 42$

$gT = \phi$

$gT = 1$

$gT = 1$

$gT = 1$

↑  
↓

Ver globale  
modifiable  
partout dans  
le module

```
#include "TD20200505.h"

int main(int argc, char const *argv[])
{
    sPoint p1 = {0., 0.};
    sPoint p2 = {0., 0.};

    // call the displayError function of the errorManager module
    displayError(E_NO_ERROR);

    p1.x = 1.;
    p1.y = 1.;

    p2 = rotate(p1, -45); gT = 3

    pointDisplay("P1", p1);
    pointDisplay("P2", p2);

    return 0;
}
```

$T\phi : gT = 42$

① exporter  $gT$   
 ② mettre  $gT \leftarrow 3$  à la fin du main.

IS: 53

$gT = 1$

Dans le fichier trigo.h

extern int32\_t gT;

VARIABLES GLOBALES → déclaration UNIQUEMENT dans le .c

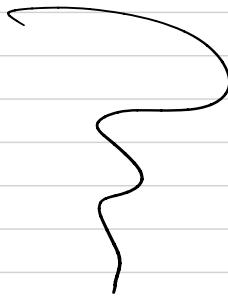
## Tdloc

static int32\_t gT =  $\phi$ ;

si variable global "static"  
==> pas d'extern ou

## Error Manager

static int32\_t gT =  $\phi$ ;



# Makefile et génération de l'app.

**Makefile** → (txt) appelé par l'outil make

```

EXEC=app ← nom de l'exécutable
CC=gcc → nom du compilateur/linker
CFLAGS+= -std=c99 -Wall -g ] options de compilation
CFLAGS+= -Iinclude ] options de compilation
LDLIBS+= -lm ) librairies complémentaires link
ODIR:=obj ) répertoire pour les fichiers .o
SRC := $(wildcard *.c) ← liste des .c à compiler
OBJS = $(patsubst %,$(ODIR)/%, $(SRC:.c=.o)) ← définition des fichiers .o à partir des fichiers .c

all: $(EXEC) target (cible)

-include $(ODIR)/*.d dépendance .h/.c
$(EXEC): $(OBJS) ← règle de link
      $(CC) -o $@ $^ $(LDFLAGS) $(LDLIBS)

$(ODIR)/%.o: %.c | $(ODIR) ← règle de compilation.
      $(CC) $(CFLAGS) -c $< -o $@ -MMD -MF $(@:.o=.d)

$(ODIR):
  mkdir $@

clean: target
  $(RM) $(EXEC)
  $(RM) -rf $(ODIR)

test: all make test
  ~project/unittest/unittest.py -i t.json

.PHONY: clean all test { syntaxe
.DEFAULT: all make
  
```

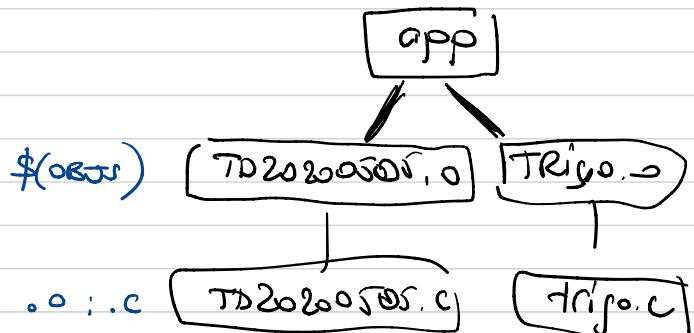
fm: lib math,

ps \* .c

— définition des fichiers .o à partir des fichiers .c

test.c → obj / test.o

make [all] cible  
make app



→ gcc -std=c99 -Wall -g -Iinclude -c TD2020S05.c -o objs/TD2020S05.o

→ gcc -o app objs/TD2020S05.o objs/Trigo.o -lm

```
#include <stdio.h>
#include <stdlib.h>

int test(int a, int * b, int * c, int * d)
{
    a = *b;
    *b = *b + 5;
    *c = a + 2;
    d = c;
    return *d;
}

int main()
{
    int a = 0, b = 100, c = 200, d = 300, e = 400;
    e = test(a, &b, &c, &d);
    printf("a:%d, b:%d, c:%d, d:%d, e:%d\n", a, b, c, d, e);
    getch();
}
```

- A Terminieren pow ke jenach 14 ~