Partie 1

swap → C → PB : passage par valeurs

↳ sol C : passer des ptrs.

C++ →

passage par réf. ⚠

↓

int i ⇝ passage par val.

int &i ⇝ ——— réf.

≠

dans main | int i

&i = adresse de i ....

référence (≈ peu ≈ adresse variable. ≈ très)

int i → valeur / rvalue : j = (i)

→ variable / lvalue : (i) = 3

↓

référence.

---

swap → take des réfs.

→ const

C++ include → .h — pas .hpp / ~~.hxx~~ / .hh

implémentation → .cpp / .cxx / .cc

↓ template / généricité.

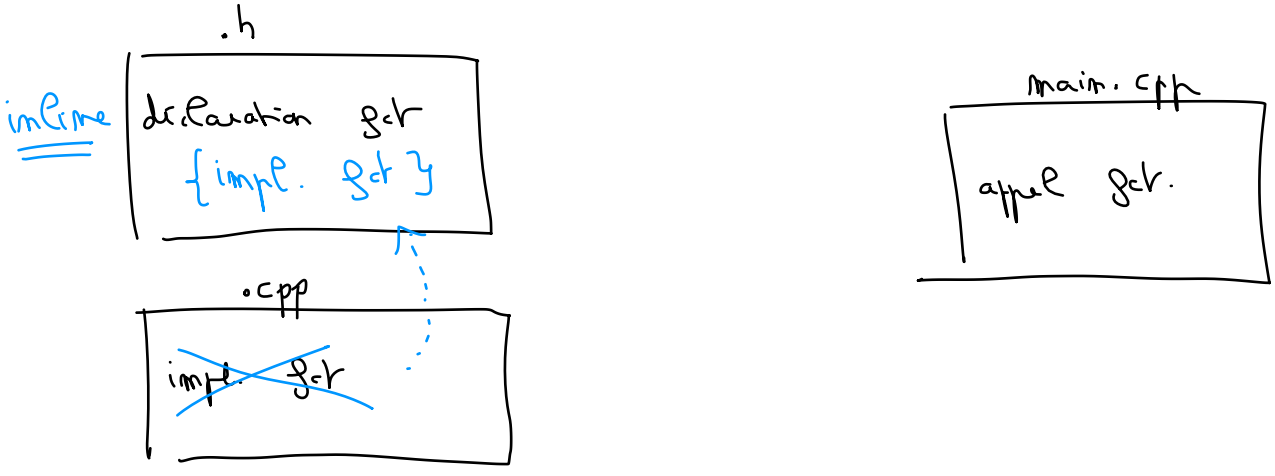
Exemple de Makefile

```
CC = g++
CCOPTS = -Wall

all: main1

partie1.o: partie1.h partie1.cpp
	$(CC) $(CCOPTS) -c partie1.cpp

main1.o: main1.cpp
	$(CC) $(CCOPTS) -c main1.cpp

main1: partie1.o main1.o
	$(CC) $(CCOPTS) -o main1 main1.o
partie1.o

clean:
	rm *.o main1
```

---

inline s/ fct

Sans inline



Avec inline → implémentée ds .h

fct — macro → ~~appel fct~~

↳ code à la précompilat.

inline

routine courte et fréquentes.

main ~~swap(x, y)~~ ;

précompil

remplacé par le code de swap ....

pas d'appel de fct sauvegarde de contexte .....

Partie II → classes

Classe — données → variables) ~ struct C
↳ données membres

fcts → méthodes
↳ fct membres
↳ méthode.

Classe — visibilité / utilisateur → private, public, (protected)

méthodes "allégées" de classe

constructeur ↔ pas de retour

même nom que la classe

↓

quand on crée une instance | variable objet | du type de la classe

↓

ex : ds main

Complexe z ; ∅ pas d'arg

↓

constructeur par défaut des Complexes appelés.

Plusieurs constructeurs ....

Complexes

pas d'args → rien → défaut → construit → 0 + i.0

2 double → x + iy

z → constructeur par copie → z' = z.

Entiers

C

main

(int i ; déclaration ↓ alloc.

↝ crée i

↝ pas initialisée.

C++

(int i0 ; déclaration ↓ alloc + appel constr.

classe int

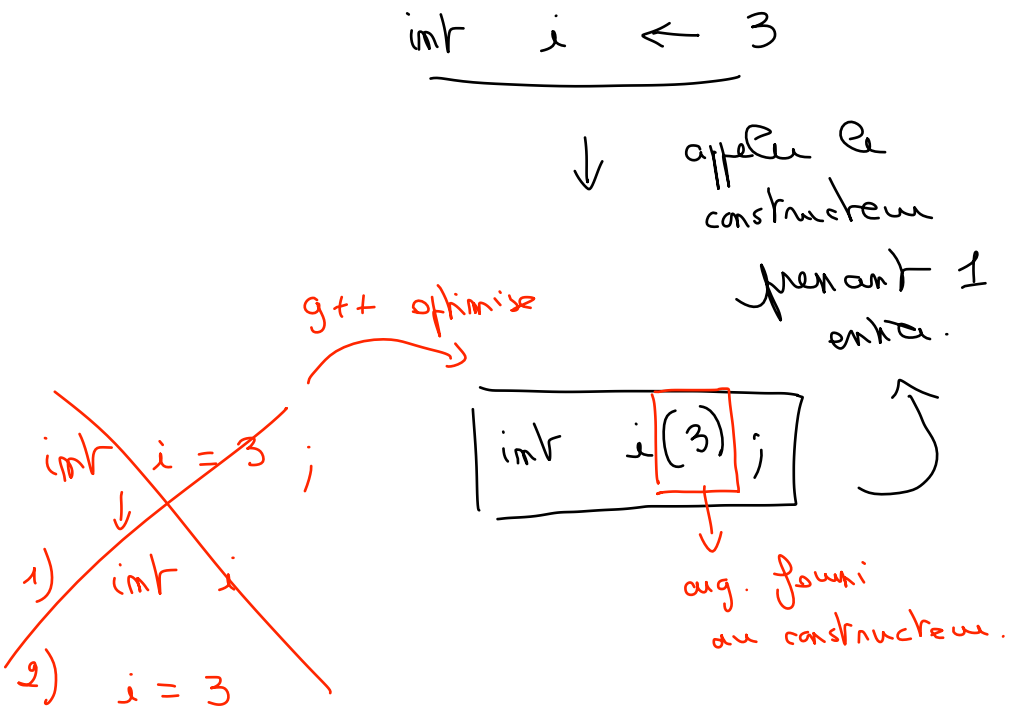
↝ crée un obj. de classe int

↓

appel d'un constructeur.

par défaut

i ← 0.


int i ← 3
appelle le constructeur prenant 1 entier.
g++ optimise
int i = 3 ;
1) int i ;
2) i = 3
int i(3) ;
arg. fourni au constructeur.