TD1

swap(x, y)

x : variable → valeur / variable / adresse / lieu de stockage

passage par val. ⇌ C

C → passage par valeur

passage par réf.

C++ ...

syntaxe

&

⚠ — utilisé dans les déclarations / types de fct

val ~ x non modifiable

```
void incr(int *px)
int &x
```

C ~> adresse / pointeurs / ...

C++ ~ demande le passage par réf

⚠

```
int i = 3;
```

&i → adresse de i

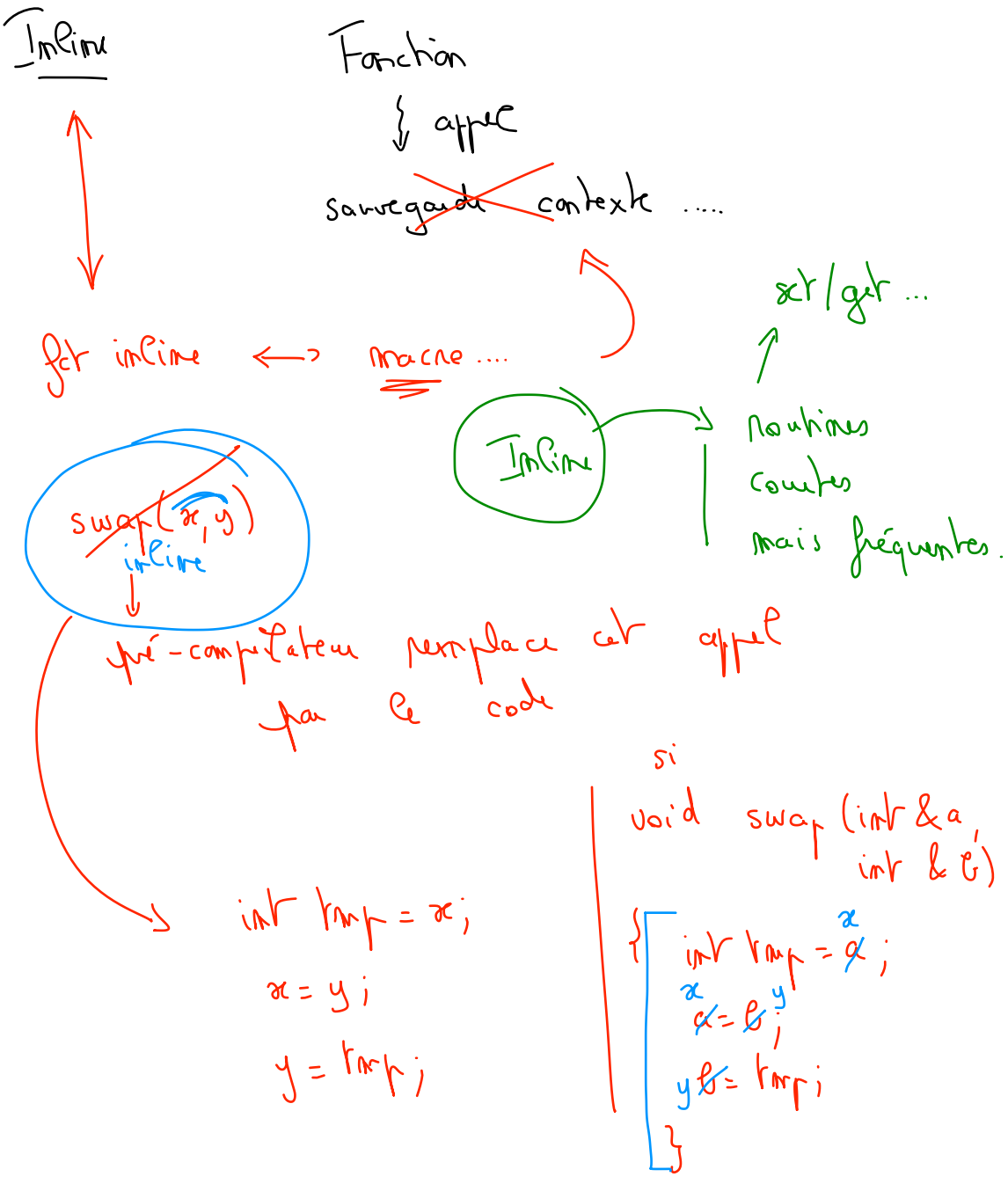
swap.h
swap.cpp | main.cpp / .hpp → templates ...

① (-c) swap.o

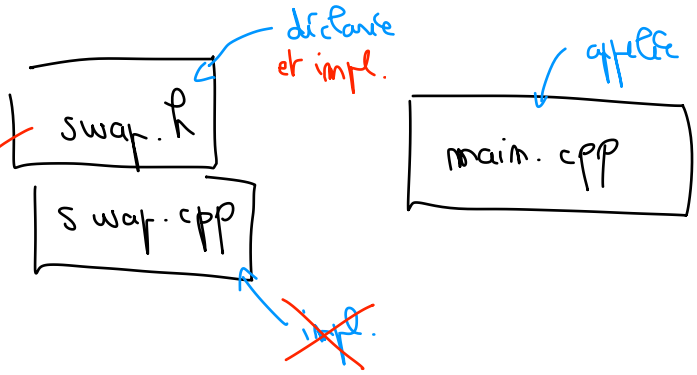
② (-c) main.o

③ main

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

all: main

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

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

main: swap.o main.o
	$(CC) $(CCOPTS) -o main main.o
swap.o

clean:
	rm *.o main
```



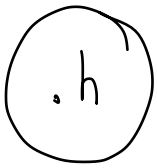
Inline



swap
|
inline

```
inline void swap ( .... )
{ .... }
  ↑ code.
```

Partie II



classes C++

⊕ (héritage / polymorphisme)

code générique C++ (template)

ex : listes de 

↓

var. de type.

Classe → données / types → private / (protected) / public

→ données (membres)

→ fonctions (méthodes). → oblig. → constructeurs / destruct.

nom ↕ classe

```
class Tore {
  private:
    int _x;
  public:
①→  Tore(int i)    { _x = i; }
②→  Tore(float y)   { _x = int(y); }
    :
};
```

→ appelés à la déclaration.

main // ①

```
Tore t(3);
```

↳ appel du constr. prenant 1 entier ⇝ ①

// ②

```
Tore tt(3.5);
```

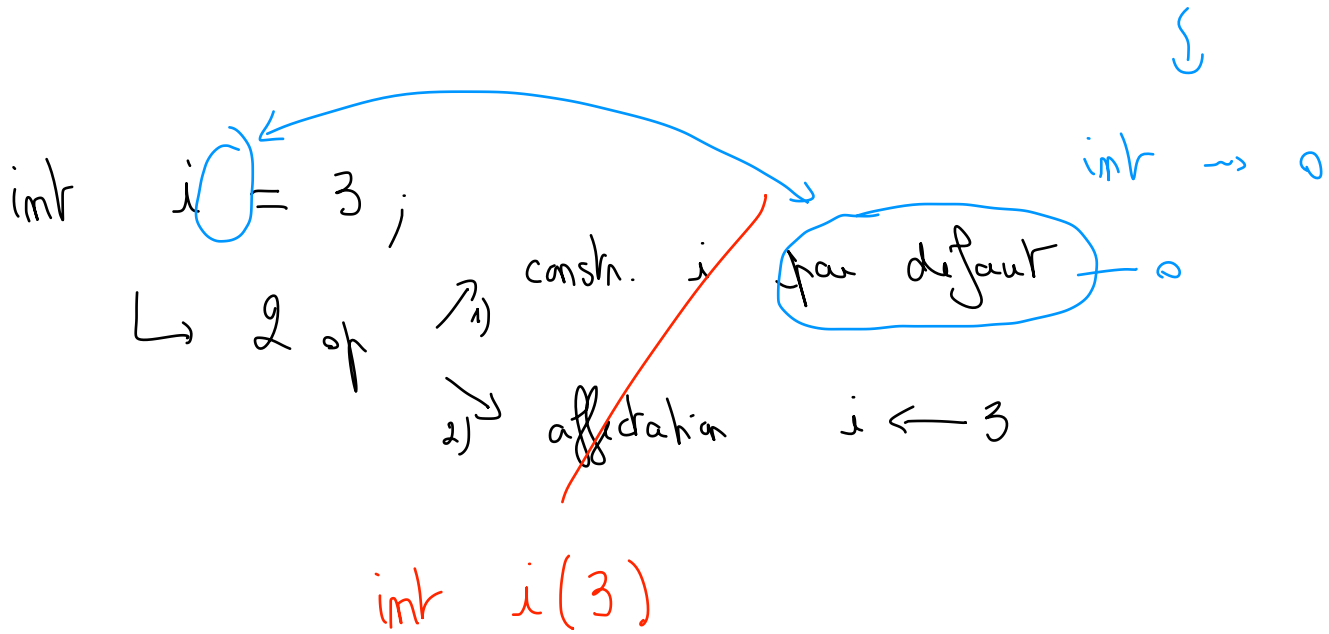
---

int i; int → classe ....

constructeur de int appelé ⇝ pas d'args

⇓

constr. par défaut




```
class V3 {
private:
	int _x, _y, _z, _norme ;

	void maj_norme() ;

public:
	V3(int x=0, int y=0, int z=0) ; // fait le constructeur std + par
défaut
}
```

→ .h

```
V3::V3(int x, int y, int z)
{
	_x = x ;
	_y = y ;
	_z = z ;
	maj_norme() ;
}

void V3::maj_norme()
{
	_norme = sqrt(_x*_x + _y*_y + _z*_z) ;
}
```

→ .cpp

```
/* main ...

 V3 v(1,2,3);
 V3 v2; // appel du constructeur par defaut

 */
```

ds le main