COURS INFO1 — 26.X.2020

Résumé

Méthodologie

- cycle de développement d'un logiciel
  CdC - Analyse - Ecriture - Test
- Génération du code
  écriture - Compilation [preprocessing / compilation / Link]
  (exécutable)

Langage.

- Témoin
- type de données (entier / réel)
- variables, constantes
- affichage, saisie

Structure de contrôle

contrôle le comportement du logiciel en fonction de certains événements.

if

```
if ( condition )
```

↳ VRAIE $\neq \emptyset$
FAUSSE $\emptyset$

```
if ( 1 )...      condition VRAIE
if ( ∅ )...      condition FAUSSE
if (42)          condition VRAIE
```

opérations de test :

==	égalité	if (x == 1) ... if (x == y) ...
!=	différent	if (x != 1) ... if (x != y) ...
< <= > >=		if (x <= y) ...

not equal

Algo 17

associer les conditions

c1 && c2 ET logique if (c1 && c2) ...

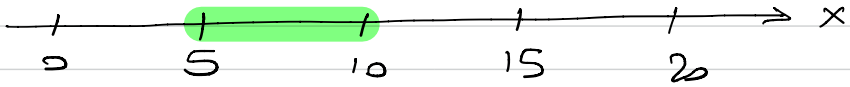
c1 || c2 OU logique if (c1 || c2) ...

```
if ( x==0 || y==0) ...
```

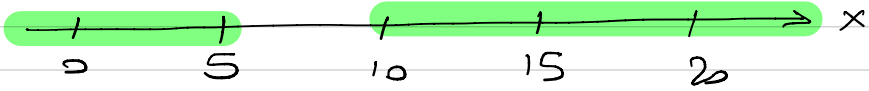
gauche à droite

```
int x;
```

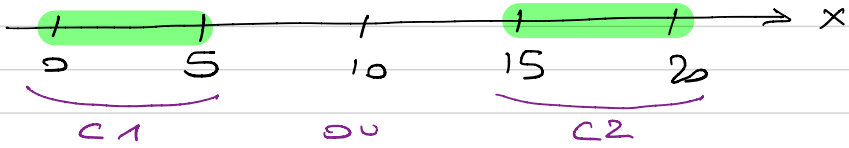
```
if ( x==10 ) ...
```




```
if ( 5<=x && x<=10) ...
```




```
if ( x<=5 || 10<=x ) ...
```




```
if ( (0<=x && x<=5) || (15<=x && x<=20))...
```

c1 c2

Structure if

- if (cond) { bloc }

  Le bloc est exécuté si cond VRAIE

```
if ( x==1 )
{
    puts ("Ok");
}
```

bloc

- if (cond) { blocV }
  else { blocF }

  le blocV est exécuté si cond VRAIE
  sinon le blocF est exécuté.

```
if ( x==1)
{
    puts ("ok");
}
else
{
    puts ("ERROR");
}
```

```
if (x==1)
  puts ("ok");
else
  puts ("ERROR");
```

{ } sont facultatives si 1 ligne de code dans le bloc.

Recommandation : toujours le mettre par sécurité.

```
if (cond 1)        { bloc 1 }
else if (cond 2)   { bloc 2 }
else               { bloc 3 }
```

le bloc 1 est exécuté si cond1 VRAIE

le bloc 2 ——— cond1 FAUSSE et cond2 VRAIE

le bloc 3 ——— cond1 FAUSSE et cond2 FAUSSE

⚠ Si cond1 vrai, cond2 non évaluée.

TD 20201026

↳ https://github.com/Info1-TIN-A-2020-2021/TD

↳ TD20201026.zip

[makefile
  t.json

Amélioration de la partie testing.

1. Saisir une tension en [V]
2. afficher un message fonction de la valeur de la tension

Valeur	0 ... 50	50 ... 1500	1500 ... 15kV	> 15kV
Message	BT	MT	HT	THT

15'

14h07

Mode INTERACTIF

> ./app ↵

Enter the voltage [V] : 500 ↵
MT

interactif → le logiciel a besoin de l'utilisateur lors de son exécution.

mode NON INTERACTIF

> ./app 500 ↵

MT

non interactif → le logiciel utilise les argum^ts lors de son exécution.

Suite du TD : programmer le mode interactif ou non

	mode interactif	non interactif
argc	1	2

14h21

2 Erreurs fréquentes.

- if (x = ∅) { bloc }

  Pb pour la condition
  1) on met ∅ dans x
  2) condition x=∅ est évalué comme toujours fausse !

  bloc jamais exécuté !

  if (x = 1) { bloc }

  ① on met 1 dans x
  ② la condition x = 1 est évalué comme toujours vraie

  bloc toujours exécuté !

- Pas de point. virgule après le if( )

  if (x == ∅) ;
  {
  ≡ bloc
  }

  c'est le ; qui est concerné par la condition

  ⇒ le bloc sera toujours exécuté !

test égalité

2x =

if (x == ∅)

Après le if jamais de ; pour terminer la ligne.

Conseil pour test d'égalité : if (∅ == x) { ... }

ne compile pas ⇒ ← if (∅ = x)

Structure de contrôle switch / case

NE S'APPLIQUE QU'À DES ENTIERS

IDÉE : Simplifier l'écriture et aussi d'optimiser le temps de traitement.

int x ;

valeur x	0	1	2	3	autres valeurs
message	x	y	z	T	_

```
if (x==0)
{
  puts ("x");
}
else if (x==1)
{
  puts("y");
}
else if (x==2)
{
  puts("z");
}
else if (x==3)
{
  puts("T");
}
else {
puts ("_");
}
```

```
switch (x)
{
  case 0:
    puts ("x");
    break;
  case 1:
    puts ("y");
    break;
  case 2:
    puts ("z");
    break;
  case 3:
    puts ("T");
    break;

  default:
    puts ("_");
    break;
}
```

un entier ?

valeurs numériques

Si x vaut 3
→ 4 conditions qui seront évaluées

Table

0	adresse mémoire du code à exécuter
1	
2	
3	

Table d'indirection → + rapide