Comment faire pour que le programme exécute certaines instructions ou certaines autres en fonction de la situation qu'il rencontre ?

Langage C

LES BRANCHEMENTS CONDITIONNELS

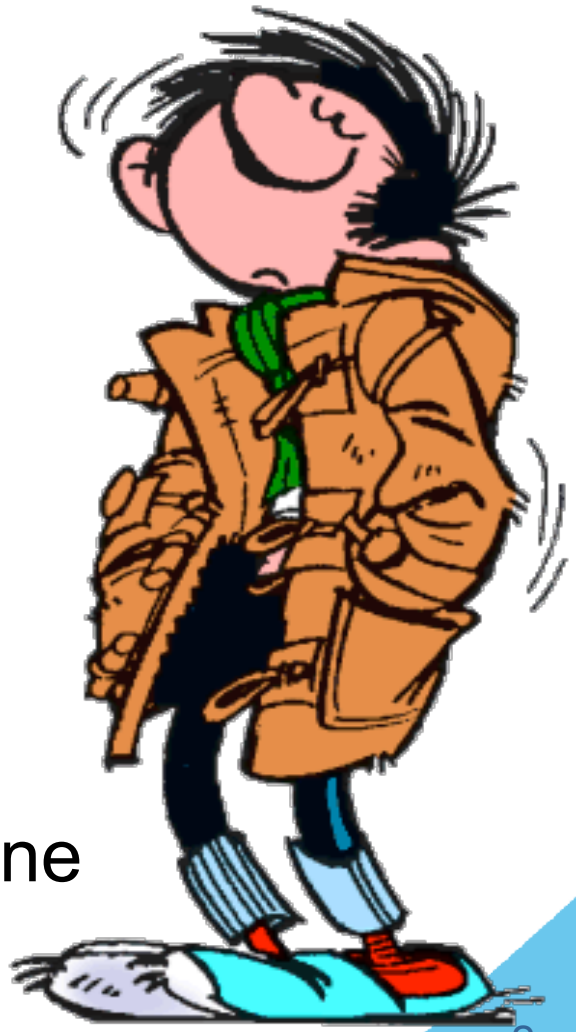
Introduction

Voici comment Gaston s'habille avant de sortir:

Il **met son pull**.

Il **enfile un manteau** s'il fait moins de 6°C à l'extérieur.

Il **sort**



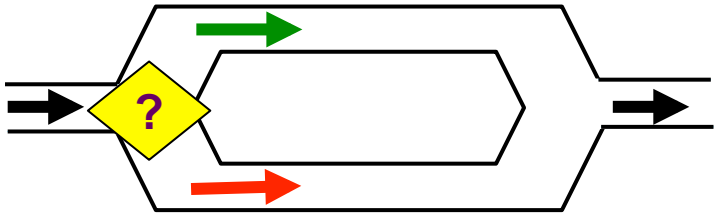
Les instructions à exécuter dépendent d'une certaine **condition**.

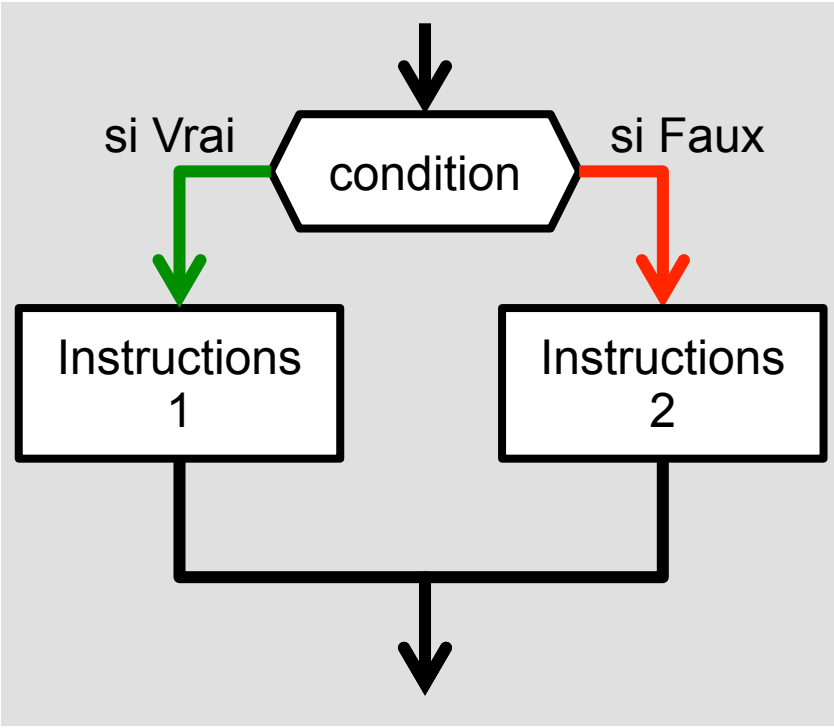
Qu'est-ce qu'une <condition> ?

Une <condition> est une expression
qui vaut VRAI ou FAUX
au moment de son évaluation

Ex : a > 2 t_celsius < 6 ...

L'instruction **if - else**






```
if ( <condition> ) {
        <instructions 1>
} else {
        <instructions 2>
}
```

// Accolades optionnelles pour un bloc ne contenant qu'une seule instruction

Exemple

```
int age;

printf("\nEntrez l'âge : ");
scanf("%d", &age);

printf("La personne est ");
if (age >= 18) {
    printf("majeure.");
} else {
    print("mineure.");
}
```


A vous de jouer

Ecrire le code de **ppetit.c**

- L'utilisateur saisit 2 entiers (positifs ou négatifs).
- Le programme affiche le plus petit des 2.

Corrigé ppetit.c

```
int a, b;
int ppetit;

printf("\nPremier nombre : ");
scanf("%d", &a);
printf("Second nombre : ");
scanf("%d", &b);

if (a < b)
    ppetit = a;
else
    ppetit = b;

printf("Le nombre le plus petit est %d.", ppetit);
```


A vous

Ecrire le code qui affiche ce que fait Gaston en fonction de la température (`int t_celsius`).

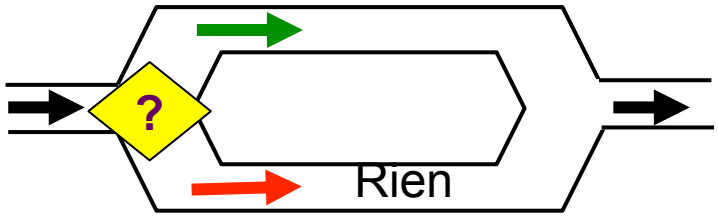
```
Entrez la température : 8
Gaston met son pull.
Gaston sort.
```

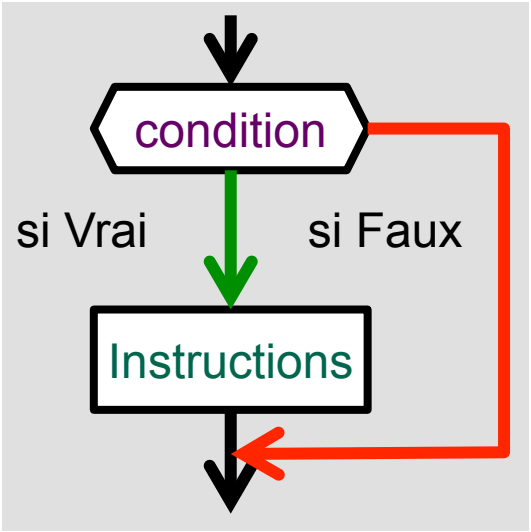
```
Entrez la température : 3
Gaston met son pull.
Gaston met son manteau.
Gaston sort.
```

Corrigé

```
int t_celsius;
printf("\nEntrez la température : ");
scanf("%d", &t_celsius);

printf("Gaston met son pull");

if ( t_celsius < 6) {
    printf("Gaston met son manteau");
} else {
}

printf("Gaston sort.");
```

else pas obligatoire si rien






```
printf("Gaston met son pull");

if ( t_celsius < 6) {
    printf("Gaston met son manteau");
}

printf("Gaston sort.");
```

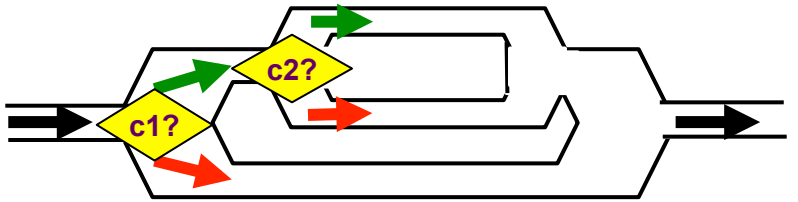
Et si on avait ?

Voici comment Gaston s'habille avant de sortir:

Il met son pull. S'il fait moins de 6°C à l'extérieur, il enfile son manteau puis sort. Dans le cas ou il gèle, il ajoute une écharpe.

Objectif : trouver l'algorithme qui décrit comment Gaston s'habille

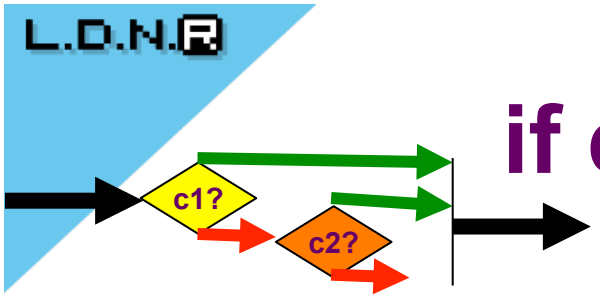
If "imbriqué"



Tout bloc peut comporter un if, on peut donc les imbriquer tant que nécessaire (avec ou sans **else**).

```
printf("Gaston met son pull");
if ( t_celsius < 6) {
   printf("Gaston met son manteau");
   if (t_celsius < 0) {
        printf("Gaston met son écharpe");
   }
}
printf("Gaston sort.");
```

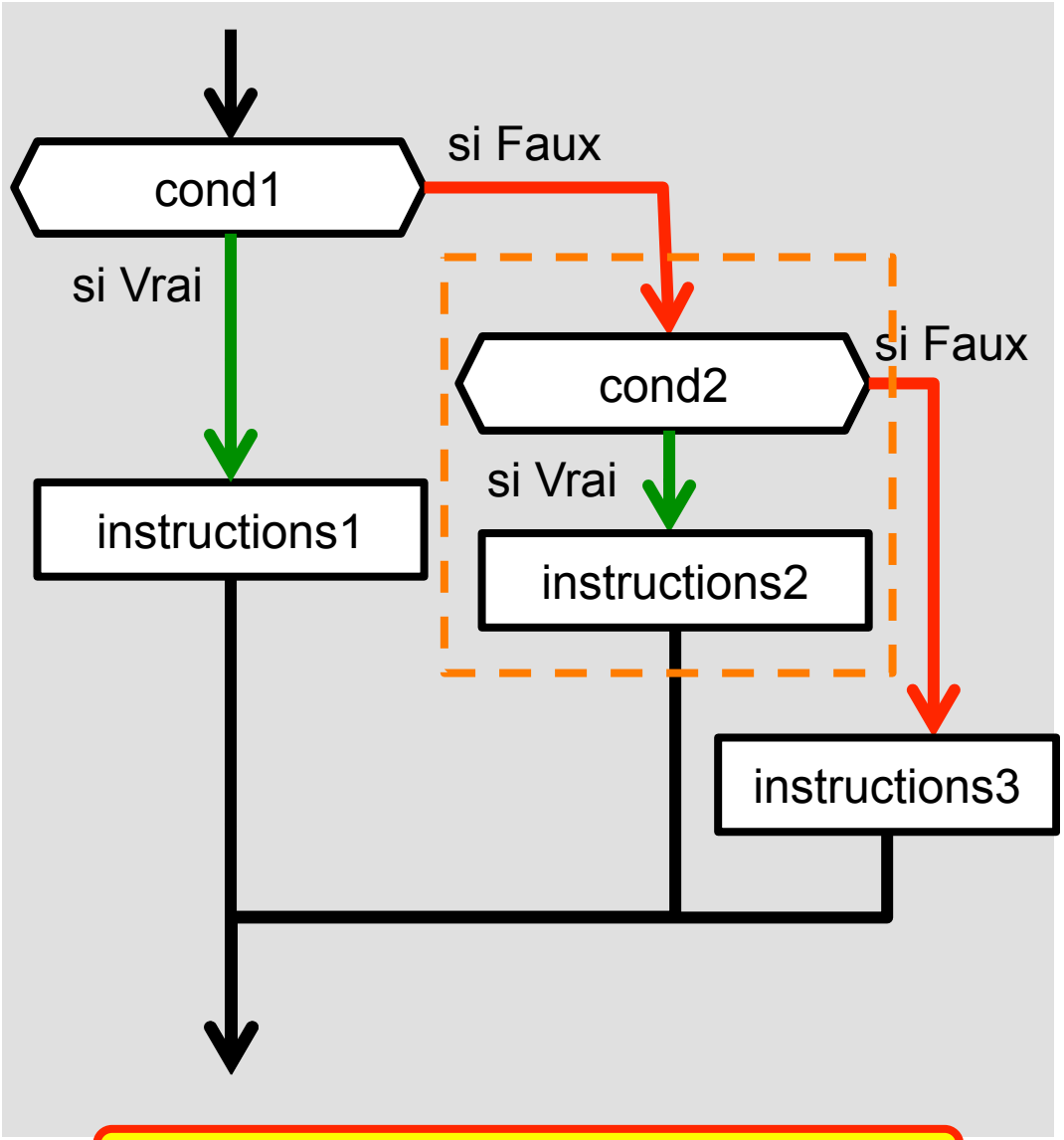
Il existe **if else-if** plus lisible mais qui ne remplace pas tous les cas.



if else-if

```
if ( <condition1> ) {
        <instructions 1>
} else if (<condition2> ) {
        <instructions 2>
} else {
        <instructions 3>
}
```

Facilite l'écriture de certains if imbriqués.



Le **else** final est optionnel

Ajoutez autant de **else if** que nécessaire

Exemple

```
printf("le nombre est ");
if (n == 0) {
    printf("nul");
} else if (n % 2 == 0) {
    printf("pair");
} else {
    printf("impair");
}
```

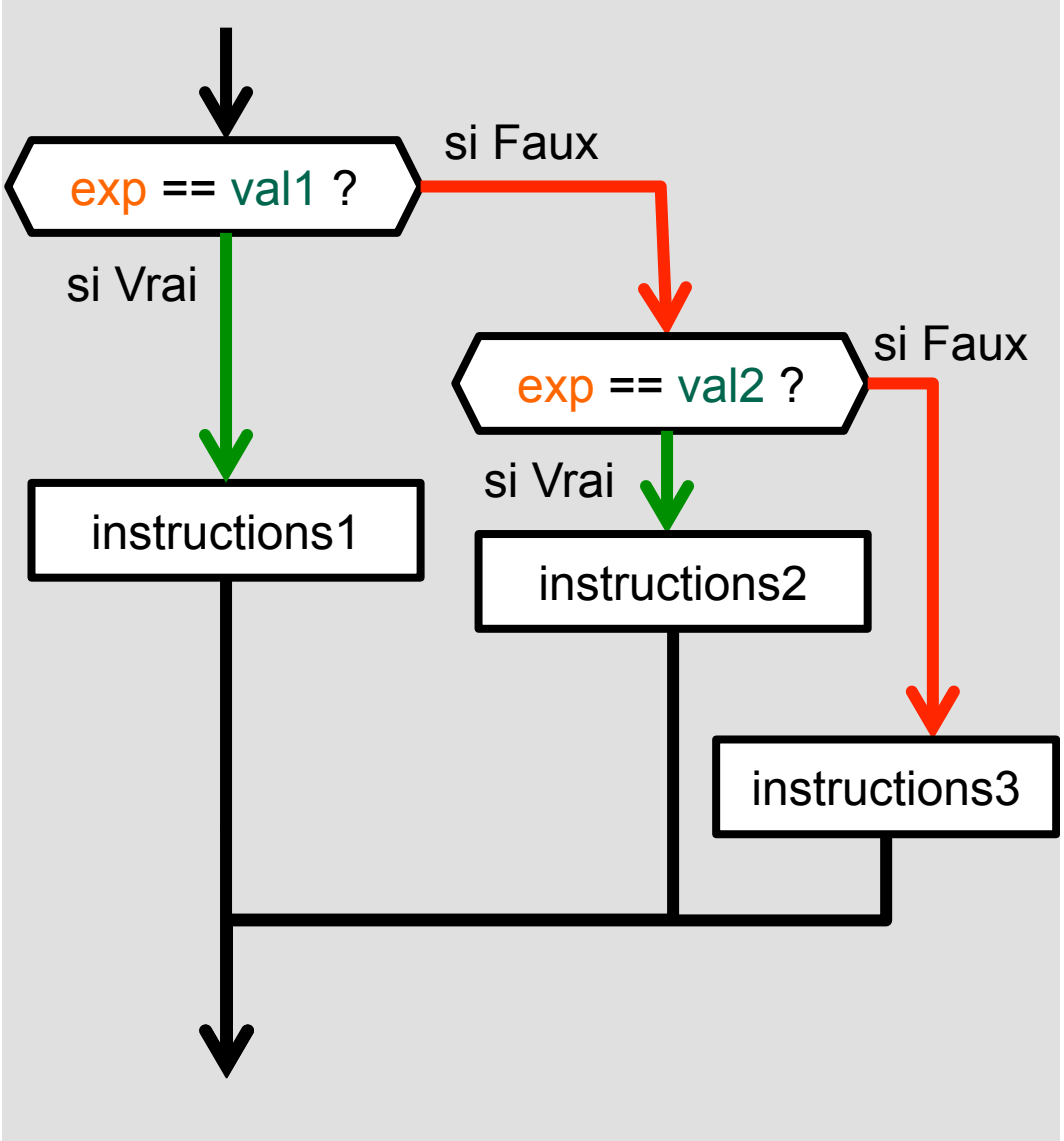

Notre exemple avec le **if else-if**

```
printf("Gaston met son pull");
if ( t_celsius < 0) {
    printf("Gaston met son manteau");
    printf("Gaston met son écharpe");
} else if ( t_celsius < 6) {
    printf("Gaston met son manteau");
}
printf("Gaston sort.");
```

La solution **if** imbriqués était meilleure (moins de redondance de code)

L'instruction switch

```
switch ( <expression> ) {
    case <valeur1> :
        <instructions 1>
        break;
    case <valeur2> :
        <instructions 2>
        break;
    default :
        <instructions3>
}
```



Facilite l'écriture de certains if else-if.
<valeurN> sont des valeurs constantes possibles de <expression>
Ajouter autant d'instructions **case** que nécessaire.

Exemple

```
int codeSaisi;
int codeGagnant = 3, codeConsolation = 7;

printf("Saisissez un code (0-9)");
scanf("%d", &codeSaisi);

switch (codeSaisi) {

    case codeGagnant:
        printf("Le gros lot");
        break;

    case codeConsolation:
        printf("Le lot de consolation");
        break;

    default:
        printf("Aucun gain");
}
```

Switch

Que se passe-t-il si on oublie une instruction **break**?

Les conditions sont au coeur des décisions

Langage C

LES CONDITIONS

Condition simple

En comparant des expressions numériques

expression1 <COMP> expression2

==	égal à
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieurs ou égal à
!=	différent de

Identifie les écritures valides de conditi

Est-ce une écriture valide de condition?

Si oui, dans quels cas vaudra-t-elle VRAI?

```
3 < 5
5 < 3
ab != ba
annee >= 2018
2018 <= annee
annee > 2019
annee - 2018 > 0
3 < nombreDecibels < 8
(anneeFin - anneeDebut) < nbAnneesMax
```


Ecrire le condition à partir des données fournies

Il gèle

```
int t_celsius;
```

La personne est majeure

```
int agePersonne;
```

Le discrimant (b^2-4ac) est strictement négatif

```
double a, b, c;
```

Le code saisi est correct

```
int codeSaisi, codeAttendu;
```

On a le droit de diviser par denomin

```
int denomin;
```


Ecrire des conditions. A vous !

le caractère a pour code ascii 65

```
char carac;
```

Les variables contiennent le même caractère

```
char carac1, carac2;
```

Paul a 2 ans de moins que Virginie

```
int âgePaul, âgeVirginie;
```

La durée entre le début et la fin est dans la limite

```
int anneeDebut, anneeFin, nbAnneesMax;
```


A vous de jouer !

```
int t_celsius;
printf("\nEntrez la température : ");
scanf("%d", &t_celsius);

printf("Gaston met son pull");
if ( t_celsius < 6) {
   printf("Gaston met son manteau");
   if (t_celsius < 0) {
        printf("\nGaston met son écharpe");
   }
}
printf("Gaston sort.");
```

Toutes les températures sont-elles acceptables?
Quelle(s) condition(s) nécessaire?

Condition composée

Construite à partir d'autres conditions avec les connecteurs logiques :

- `!` NON logique
- `&&` ET logique
- `||` OU logique

c1	c2	c1 && c2	c1 || c2
V	F	F	V
V	V	V	V
F	F	F	F
F	V	F	V

c	! c
V	F
F	V

Exemple

```
int t_celsius;
printf("\nEntrez la température : ");
scanf("%d", &t_celsius);
if (t_celsius < -60 || t_celsius > 60) {
    printf("Température incorrecte.");;
    return EXIT_FAILURE;
}
printf("Gaston met son pull");
if ( t_celsius < 6) {
  printf("Gaston met son manteau");
  if (t_celsius < 0) {
      printf("\nGaston met son écharpe");
  }
}
printf("Gaston sort.");
```

A vous de jouer

Est-ce une écriture valide de condition?

Si oui, dans quels cas vaudra-t-elle VRAI?

```
a < b && b < a;
a == 2 || a == 3;
! (a != 2 && a != 3);
ab > 3 && ba < 2;
```


Ecrire des conditions. A vous !

La vitesse est entre 50 et 80

```
int vitesse;
```

La personne est majeure et le code est correct

```
int âgePersonne;
int codeSaisi, codeAttendu;
```

carac contient une lettre (non accentuée) majuscule

```
char carac;
```

La vitesse est de 50 plus ou moins une tolérance

```
char vitesse, tolerance;
```

C'était une année de guerre mondiale

```
int annee;
```


Le cas particulier du C

Dans le C d'origine, il n'existe pas de valeur VRAI ou FAUX.

En C, une condition est toute expression numérique avec les conventions suivantes :

- La valeur 0 correspond à FAUX
- Toute valeur ≠ 0 correspond à VRAI

Les opérateurs de comparaison du C sont conçus pour que tout ce qui a été dit avant reste valide.

Exemples

L'écriture est-elle valide?

Lorsqu'elle sera exécutée, dans quels cas la condition vaudra-t-elle VRAI?

```
45
5 - ( 3 + 2)
2 * b - 2
a = 0
maFonction(0)
maFonction(1)
estBissextile(2019)
```


Exemples

Diagramme bulle des fonctions ci-dessous pour qu'elles retourne la valeur attendue ?

L'année 2019 est bissextile

```
estBissextile(2019)
```

Le feu est allumé

```
estAllume(leFeu)
```

La chaîne de caractère vaut bonjour

```
! strcmp(chaine, "bonjour")
```


Levée d'ambiguïté du else non apparié

```
if ()
    inst
if ()
    inst2
else {
}
```

Bonne pratique!

Mettre des accolades pour éviter toute ambiguïté.

Le else se rapporte au if le plus extérieur (≠ Java).

```
if ( ) {
  inst1
  if ( )
    inst2
} else <instructions>
```

A vous de jouer

Ecrire le code de la fonction `estBissextile(int)`

Note : Année bissextile si elle est divisible par 4 et non divisible par 100, ou si l'année est divisible par 400.

Test: bissexile (2000, 2012), non bissextile (1900, 2013)

```
int annee;

printf("\nAnnee : ");
scanf("%d", &annee);

if (estBissextile(annee))
    printf("%d est bissextile.", annee);
else
    printf("%d n'est pas bissextile.", annee);
```


Corrigé estBissextile(int)

```
int estBissextile(int annee) {

   return (annee % 400 == 0) ||
     ((annee % 4 == 0) && (annee % 100 != 0));
}
```