Les conditions permettent de **tester** les variables que nous avons déjà appris à créer. Sans elles, les programmes informatiques feraient toujours la même chose !

Condition `>| if... else`

La condition `>| if... else` permet par exemple de dire à la machine :

> Si la variable bidule est égale à 50, fais X !

C'est la base de la condition `>| if/else`. Cependant, il également pouvoir tester si la variable est inférieure ou égale, inférieure, supérieure, etc...

Voici le tableau récapitulatif des symboles utilisés pour `>| if/else`, à apprendre par coeur :

Symbole	Signification
==	est égal à
>	est supérieur à
<	est inférieur à
>=	est supérieur ou égal à
<=	est inférieur ou égal à
!=	est différent de

⚠ `>| =` et `>| ==` n'ont pas la même signification en C. `>| =` donne une nouvelle à la variable qu'on essaye de comparer, ce qui est contre productif.

Ouvrir une variable `>| if`

Pour ouvrir une variable `>| if`, on formate le code comme suit :

- On écrit `>| if` ;
- Entre parenthèses, on écrit notre condition ;
- Entre accolades, on rédige les instructions à exécuter si la condition est vérifiée.

Soit :

```
`if (/*la condition*/)
    {
        // instructions à excécuter si la condition est vraie
    }`
```

Si on veut ajouter une instruction alternative (`>| else`), on la déclare comme `>| if`, mais sans spécifier de condition entre parenthèses.

Exemple :

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

int main(int argc, char *argv[])

{

int age=16;

if (age>= 18) // si la variable est vrai, alors...
    {
        printf ("You can legally drink !\n"); // ... fais ceci
    }

else // sinon...

    {
        printf ("Please ask an adult to buy your alcohol \n"); //... fais cela !
    }

}
```

On peut faire de même avec un `scanf` :

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

int main(int argc, char *argv[])

{

int age=0;

printf ("Please enter your age\n");
scanf ("%d", &age);

if (age>= 18) // si la variable est vrai, alors...
{
    printf ("You can legally drink !\n"); // ... fais ceci
}

else // sinon...

{
    printf ("Please ask an adult to buy your alcohol \n"); //... fais cela !
}

 return 0;

}
```

L'instruction alternative : `else if` & tester des conditions multiples.

Mais comment faire alors si on veut plus de deux conditions sur une seule variable et valeur ? Inutile de multiplier les `if/else`, on peut simplement utiliser `else/if`.

En gros, ça consiste à dire à la machine :

SI la variable vaut A, alors fais B,
SINON SI la variable vaut C, ALORS fais D,
SINON SI la variable vaut E, ALORS fais F,
SINON fais G.

⚠ On peut mettre autant de `else if` qu'on veut.

De même, si on veut tester plusieurs conditions sur une même variable, on peut utiliser les symboles suivants pour former des contraintes :

Symbole	Signification	Usage
&&	ET	Teste deux conditions à la fois
||	OU	Teste si une condition est vrai pour X ou Y valeur.
!	NON	Teste l'inverse d'une condition

Exemple

```c
int main(int argc, char *argv[])

{

    int age = 0;

    printf("Please enter your age\n");
    scanf("%d", &age);

    if (age >= 18) // si la variable est vrai, alors...
    {
        printf("You can legally drink !\n"); // ... fais ceci
    }

    else // sinon...

    {
        printf("Please ask an adult to buy your alcohol \n"); //... fais cela !
    }

    return 0;
}
```

⚠ Quelques erreurs de débutant à éviter...

→ Ne pas confondre `=` et `==.` Avec un seul `=`, on attribue la valeur déclarée à la variable, ce qui n'est pas ce qu'on veut faire.
→ Il ne faut pas conclure une condition par `;`. Le point virgule se met derrière une instruction, pas une condition.

Faire des condition vrai/faux avec des booléens

En C, il n'y a pas de variable type "booléen". On gère ces derniers avec un type entier, comme `int`. Un booléen est une variable à laquelle on fait prendre les valeurs `0` et `1`, où 0 = faux et 1 = vrai.

On utilise les booléens car ils sont bien plus lisibles pour les humains.

Honnêtement ????? revoir avec Max

Optimiser son code avec `switch`

`If/else` a beau être une des conditions les plus utilisées en C, il peut s'avérer répétitif et peu pratique. Pour éviter de se répéter à l'infini quand on teste la valeur d'une variable unique, on peut utiliser `switch`.

Au lieu de répéter des `if/else` comme dans l'exemple plus haut, on procède de la façon suivante :

→ On note `switch(variable)`,
→ On ouvre `{}`,
→ Au sein des accolades, on décrit tous les cas qu'on veut prévoir : `case 1:`, `case 456:`... le chiffre est égal à la valeur de la variable.

⚠ Après chaque `case`, on note un `break;`, sinon l'ordinateur lira les instructions pour tous les cas. `break;` indique à l'ordinateur qu'il doit reprendre le test du début, il doit "sortir" des accolades.

⚠ Une fois qu'on a listé tous les cas, on clôture la condition par `default;`, qui affiche un message lambda pour tous les cas qui ne sont pas listés.

Exemple : voir exercice "Switch" > switch.c

Exemple de code :

```
switch (age)
{
    case 2:
      printf("Salut bebe !");
      break;
    case 6:
      printf("Salut gamin !");
      break;
    case 12:
      printf("Salut jeune !");
      break;
    case 16:
      printf("Salut ado !");
      break;
    case 18:
      printf("Salut adulte !");
      break;
}
```

Les conditions condensées : les ternaires

On peut faire des conditions d'une troisième façon, plus rarement utilisée : les **expressions ternaires.** Ces dernières sont entièrement facultatives, et certains ne les utilisent pas, car elles rendent malaisée la lecture du texte par les humains. Concrètement, il s'agit d'une expression `if/else`, mais qui tient en une ligne.

Admettons une variable booléenne `majeur` dont on veut modifier la valeur pour qu'elle soit `= 18` si `majeur` est vraie, et `= 17` si elle est fausse. Si on veut le faire avec if/else, on procède de la façon suivante :

```
{

int age = 0, majeur = 18;

if (majeur)
    age = 18;

else
    age = 17;

}
```

⚠ On a ôté les accolades, facultatives s'il n'y a qu'une instruction

Pour faire de même en ternaire, il suffira d'indiquer :

```
int age = 0, majeur = 18;

age = (majeur) ? 18 : 17; // expression ternaire
```

En bref

→ Les **conditions** sont la base de tout programme. Elles permettent à l'ordinateur de prendre une décision en se basant sur la valeur d'une variable.
→ Les mots-clés `>| if`, `>| else if`, `>| else` signifient "si", "sinon si", et et "sinon". On peut noter autant de `>| else if` que désiré.
→ Un **booléen** est une variable qui a deux états, `>| vrai (1)` ou `>| faux (0)`. On utilise `>| int` pour stocker les booléens : ils ne sont que des nombres.
→ `>| Switch` est une alternative au `>| if/else` pour analyser la valeur d'une variable. Il permet d'améliorer la lisibilité du code quand on a besoin de vérifier de nombreux cas.
→ Les **ternaires** sont des conditions courtes qui permettent d'affecter rapidement une valeur à variable en fonction du résultat d'un test. On les utilise peu car elles nuisent à la lisibilité du texte.