

Chapitre 3

Structure de contrôle : if...then...else / switch

Les structures de contrôles (ou instructions conditionnelles) permettent de réaliser des tests, et suivant le résultat de ces tests, d'exécuter des parties de code différentes.

1 if ... else

le if, dit opérateur de test se présente sous la forme basique suivante :

```
if (condition) {  
    instruction1 ;  
}
```

L'instruction *instruction1* n'est exécutée que si la condition entre parenthèse est vérifiée.

L'opérateur de test peut aussi s'écrire :

```
if(condition) {  
    instruction1 ;  
}  
else {  
    instruction2 ;  
}
```

on exécute l'instruction *instruction1* si la condition entre parenthèse est vérifiée, Sinon on exécute l'instruction *instruction2*.

On peut aussi écrire des versions plus complexes avec de multiples conditions :

```
if (condition1) {  
    instruction1 ;  
}  
else if(condition2) {  
    instruction2 ;  
}  
else if(condition3) {  
    instruction3;  
}  
else {  
    instruction_par_defaut;  
}
```

Si c'est la *condition1* qui est vérifiée, on exécute *instruction1*, Si c'est la *condition2* qui est vérifiée, on exécute *instruction2*, Si c'est la *condition3* qui est vérifiée, on exécute *instruction3*, Sinon, si aucune des conditions précédentes n'est vérifiée on exécute l'instruction *instruction_par_defaut*.

Avant de présenter quelques exemples, il nous faut aussi définir les opérateurs permettant de réaliser des tests, le plus

souvent il s'agit des opérateurs logiques :

Opérateur	Signification
==	c'est le test d'équivalence
!=	différent (non équivalence)
>	supérieur
>=	supérieur ou égal
<	inférieur
<=	inférieur ou égal
&&	ET logique
	OU logique
!	NOT logique
^	XOR logique

Exemple 1 :

```
if (a == b) {
    printf("les variables a et b sont égales\n");
}
```

Si a est égal à b , on affiche la phrase "les variables a et b sont égales".

ATTENTION : Ne pas confondre $a == b$ avec $a = b$:

- $a == b$ est une comparaison qui vérifie si a et b sont égaux.
- $a = b$ est une affectation, on affecte à la variable a la valeur de la variable b .

Exemple 2 :

```
if (!(a == b)) {
    printf("les variables a et b ne sont pas égales \n");
}
```

A cause du NOT (!) qui inverse le résultat de l'expression $(a == b)$, on affiche la phrase "les variables a et b ne sont pas égales" si l'expression $a == b$ n'est pas vérifiée, en d'autres termes si a et b ne sont pas égaux.

Exemple 3 :

```
if ((a == b) || (a < b)) {
    printf("a n'est pas supérieur à b \n");
}
```

Si a est égal à b OU a est inférieur à b , on affiche la phrase "a n'est pas supérieur à b".

Exemple 4 :

```
if (a == b) {
    printf("les variables a et b sont égales \n");
}
else {
    printf("les variables a et b ne sont pas égales \n");
}
```

Si a est égal à b , on affiche la phrase "les variables a et b sont égales" sinon on affiche "les variables a et b ne sont pas égales".

Exemple 5 :

```

if (a > b) {
    printf(" a est supérieur à b \n");
}
else if(a == b) {
    printf(" a est égal à b \n");
}
else {
    printf("a est inférieur à b \n");
}

```

Si a est supérieur à b , on affiche la phrase "a est supérieur à b", si a est égal à b on affiche "a est égal à b", sinon on affiche "a est inférieur à b".

2 switch ... case

Pour éviter les imbrications d'instructions *if*, le C possède une instruction qui permet d'explorer plusieurs cas en même temps : c'est l'instruction *switch*. La syntaxe du *switch* est comme suit :

```

switch(variable){
    case valeur1 :
        instruction10 ;
        instruction11 ;
        break;

    case valeur2 :
        instruction12 ;
        instruction13 ;
        break;

    ...

    default :
        instruction_par_defaut ;
}

```

Si *variable* prend la valeur *valeur1* alors on exécute les instructions *instruction10* et *instruction11*, si elle prend la valeur *valeur2* on exécute les instructions *instruction12* et *instruction13*, etc.

Par défaut (c'est à dire, si aucune des valeurs ci-dessus ne correspond à la variable), alors on exécute l'instruction *instruction_par_defaut*.

ATTENTION : Une fois l'exécution des instructions commencée à partir d'un "case" les autres instructions sont exécutées séquentiellement y compris celles des "case" suivants jusqu'à rencontrer l'instruction **break** qu'il ne faut donc surtout pas oublier.

3 Exercices**3.1 if then else****Question 3-1 Minimum** → exercice de cours

Ecrivez un programme qui :

1. initialise 2 variables a et b (ou demander à l'utilisateur de saisir les valeurs).
2. déclare une variable *nbmin* également entière.
3. à l'aide d'un *if*, fait en sorte que la variable *nbmin* contienne la valeur minimale de a et b .

4. affiche cette valeur minimale ainsi déterminée.

Question 3-2 Minimum (2) → exercice d'assimilation

Même exercice mais pour 3 nombres entiers.

Question 3-3 Maximum → pour aller plus loin

Ecrire un programme C qui calcule le maximum entre 3 entiers a , b et c .

Question 3-4 Signe du produit → exercice d'assimilation

Ecrivez un programme qui affiche le signe du produit de a et b sans faire la multiplication. Pour cela, vous testerez uniquement si le résultat est inférieur à 0.

Question 3-5 Age → exercice d'assimilation

Initialiser une variable entière qui correspondra à l'âge d'une personne, si la personne a au moins 18 ans, alors on affiche "peut voter", sinon, on affiche "ne peut pas voter". Refaire la même chose en le formulant négativement : si la personne n'a pas 18 ans, elle ne peut pas voter, sinon elle peut.

Question 3-6 Mention → exercice d'entraînement

Nous désirons afficher la mention obtenue par un élève en fonction de la moyenne de ses notes. S'il a une moyenne strictement inférieure à 10, il est recalé. S'il a une moyenne entre 10 (inclus) et 12, il obtient la mention passable. S'il a une moyenne entre 12 (inclus) et 14, il obtient la mention assez bien. S'il a une moyenne entre 14 (inclus) et 16, il obtient la mention bien. S'il a une moyenne supérieure à 16 (inclus) il obtient la mention très bien. Écrire les instructions nécessaires.

Question 3-7 Parité → exercice d'assimilation

Écrire un programme C qui teste si un nombre *entier* est pair. Dans le cas où le chiffre est pair, vous afficherez un message du type "Le chiffre X est pair", où X sera le chiffre que vous aurez préalablement déclaré. Modifiez ensuite votre programme pour afficher également un message si le chiffre est impair.

Vous utiliserez pour cela la fonction modulo (%). Rappel : l'opérateur "%" (ou modulo) renvoie le reste de la division (entière) entre 2 nombres (ex : $11\%3 = 2$).

Question 3-8 Année bissextile → pour aller plus loin

Si l'année A n'est pas divisible par 4, alors elle n'est pas bissextile Si A est divisible par 4, l'année est bissextile **sauf** si A est divisible par 100 **et pas** par 400.

Exemples :

- 1901 n'est pas bissextile car non divisible par 4
- 2004 est bissextile car divisible par 4 et pas par 100
- 2100 n'est pas bissextile car divisible par 4, divisible par 100 mais pas par 400
- 2000 est bissextile car divisible par 4, par 100 et par 400

Écrire un programme qui détermine si une année est bissextile ou non.

Question 3-9 Parité 2 → pour aller plus loin

Trouvez ensuite un moyen de résoudre l'exercice 3-7 sans la fonction modulo mais avec l'opération division entière (/ sur des int)

3.2 switch

Question 3-10 En toute lettre → exercice de cours

Ecrire un programme qui demande à l'utilisateur de taper un chiffre et qui l'écrit ensuite en toute lettre à l'écran. Par exemple, si l'utilisateur tape le chiffre 9, le programme affichera **neuf**.

Note : on ne s'occupera que des chiffres et pas de nombres en dehors de l'intervalle $[0 - 9]$.

Question 3-11 Echelle de Richter → exercice d'entraînement

L'échelle de Richter permet de décrire la magnitude des tremblements de terre :

1	Micro tremblement de terre, non ressenti
2	Très mineur. non ressenti mais détecté
3	Mineur. causant rarement des dommages
4	Léger. Secousses notables d'objets à l'intérieur des maisons
5	Modéré. Légers dommages aux édifices bien construits
6	Fort. Destructeur dans des zones allant jusqu'à 180 kilomètres à la ronde si elles sont peuplées
7	Majeur. Dommages modérés à sévères dans des zones plus vastes.
8	Important. Dommages sérieux dans des zones à des centaines de kilomètres à la ronde
9	Dévastateur. Dévaste des zones sur des milliers de kilomètres à la ronde

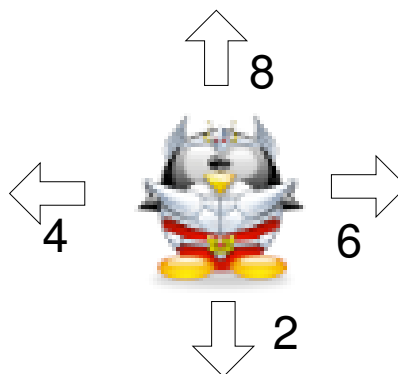
Si le nombre n'est pas compris entre 1 et 9 c'est qu'il y a erreur de saisie (si inférieur à 1) ou que c'est l'apocalypse (si supérieur à 9).

Vous écrirez un programme permettant à l'utilisateur de saisir une valeur d'échelle et qui en réponse affichera à l'écran la description associée à ce nombre. Vous n'oublierez pas de gérer le cas où le nombre tapé par l'utilisateur est "hors-échelle".

Question 3-12 Saisie des déplacements du personnage → en vue du projet

Dans le cadre de votre projet, vous aurez à récupérer les touches tapées par l'utilisateur pour savoir dans quelle direction déplacer votre personnage. Ecrivez un petit programme qui demande à l'utilisateur de saisir un **nombre** puis qui en fonction du nombre saisi :

- 6 : affiche "le personnage va à droite".
- 4 : affiche "le personnage va à gauche".
- 8 : affiche "le personnage va en haut".
- 2 : affiche "le personnage va en bas".
- dans le cas d'un autre caractère, affiche : "erreur de saisie, le personnage ne bouge pas".



4 Validation des compétences acquises à l'issue de cette séance

Je maîtrise les compétences demandées à l'issue de cette séance si **je suis capable** de :

- ☐ exprimer une condition **booléenne** à l'aide des opérateurs de comparaison `<`, `>`, `<=`, `>=`, `==`, `!=` et des opérateurs booléens `&&`, `||`, `!`
- ☐ réaliser une série d'instructions si le résultat de l'évaluation de la condition est vérifié, ceci à l'aide du mot-clef `if`
- ☐ réaliser une série d'instructions si le résultat de l'évaluation de la condition est vérifié ou une autre série d'instructions dans le cas contraire, à l'aide des mot-clefs `if` et `else`
- ☐ enchaîner des conditions à l'aide de `if`, `else if...`, `else`.
- ☐ utiliser la structure `switch...case...default` afin de réaliser différentes séries d'instructions suivant la valeur prise par une variable donnée
- ☐ écrire un programme C bien indenté