Les structures de contrôle

Table des matières

I. Contexte

Durée : 60 minutes

Environnement de travail : PHP avec Replit

Prérequis : aucun

II. Découvrir les structures de contrôles

A. Découvrir les structures de contrôles

Principe des tests conditionnels

En informatique comme en mathématiques, nous appelons « *expression* » une ligne de calcul. La plupart du temps, ces expressions sont littérales, c'est-à-dire qu'elles contiennent des variables. Par exemple, « *3 + a = 4* » est une expression littérale, dans laquelle « *a* » est une variable. Si cette variable vaut 1 alors l'égalité est correcte, tandis que si cette variable est différente de la valeur 1 l'égalité est incorrecte. Nous ne pouvons pas savoir à l'avance si une expression sera correcte ou incorrecte. Pour le savoir, l'expression doit être évaluée, c'est-à-dire qu'elle doit être calculée en tenant compte des valeurs des variables. Quand une expression est évaluée comme vraie, on dit qu'elle vaut True, ce qui signifie vrai en anglais. Quand une expression est évaluée comme fausse, on dit qu'elle vaut False, ce qui signifie faux en anglais. Une expression littérale qui est évaluée à True ou False est une expression booléenne.

Ces notions ont été travaillées par George Boole au XIX^{e} siècle en Angleterre. Il est considéré comme le père de la logique moderne et son nom est désormais utilisé comme adjectif en informatique en son honneur. Dans ce cours, nous utiliserons des expressions booléennes, c'est-à-dire des expressions qui sont évaluées à True ou False.

Une expression booléenne contient souvent un opérateur de comparaison, comme "==" pour tester l'égalité, ou "<" qui signifie "plus petit que", ou encore ">" qui signifie "plus grand que". Vous découvrirez d'autres opérateurs ultérieurement. À l'aide de ces opérateurs de comparaison, nous pouvons réaliser des tests, c'est-à-dire comparer deux éléments entre eux afin d'obtenir en résultat une valeur booléenne True ou False. Ce résultat booléen permet alors de définir une condition. En effet, le principe est de réaliser des actions uniquement SI le résultat du test vaut True. Dans ce contexte, nous parlons de "test conditionnel".

Évaluer des expressions booléennes

Nous illustrons ce cours avec des captures d'écran du site replit.com en utilisant le langage de programmation PHP. Dans la capture ci-dessous, nous avons écrit sur la gauche un fichier nommé "test.php". Dans ce fichier, nous avons écrit 6 commandes "var_dump" avec une expression booléenne. La commande "var_dump" en PHP affiche le type d'information – dans ce cas "bool" pour signifier qu'il s'agit d'une valeur booléenne - avec le résultat de l'évaluation – dans ce cas True ou False. À la fin de chaque ligne, à partir du caractère dièse, un commentaire explique la commande indiquée. Enfin, à droite de l'écran, on a exécuté ce fichier dans un panneau "Console". Vous y voyez donc le résultat de chaque ligne.

Principe du “Si”

En langue française, le mot “si” introduit une condition, et le mot “alors” amène aux conséquences. Voyons quelques exemples d’usage :

- Si le temps est très chaud, alors il faut s’hydrater davantage.
- Si tu as suffisamment d’argent, alors tu peux réaliser cet achat.
- Si tu as bien appris ta poésie, alors tu pourras la réciter.

Les mots entre “si” et “alors” forment la condition qui est évaluée à Vrai (True) ou Faux (False). Dans le cas où la condition est évaluée à True, les mots après “alors” forment les actions à réaliser.

En informatique, nous utilisons le mot anglais “if” qui correspond à “si”. Le mot “alors” était traduit par “then” dans les anciens langages, mais les langages actuels ont abandonné ce mot-clé par une syntaxe plus épurée. Nous verrons des exemples plus concrets dans la seconde partie de ce cours.

Principe du “Sinon”

L’évaluation d’une expression booléenne lors d’un test conditionnel amène une alternative binaire : soit le résultat est True - et dans ce cas on exécute les actions spécifiques indiquées - soit le résultat est False. Dans ce dernier cas, il est également possible d’exécuter des actions spécifiques. Pour cela, on utilise le mot “sinon”. Voici quelques exemples, en langue française, pour illustrer ce principe :

- Si la neige tombe, alors ne pas sortir dehors, sinon partir en voiture.
- Si l’arbre est plein de fruits mûrs, alors les cueillir, sinon attendre qu’ils mûrissent.
- Si tu mesures plus de 1,20 m, alors tu peux accéder à ce manège, sinon son accès est interdit.

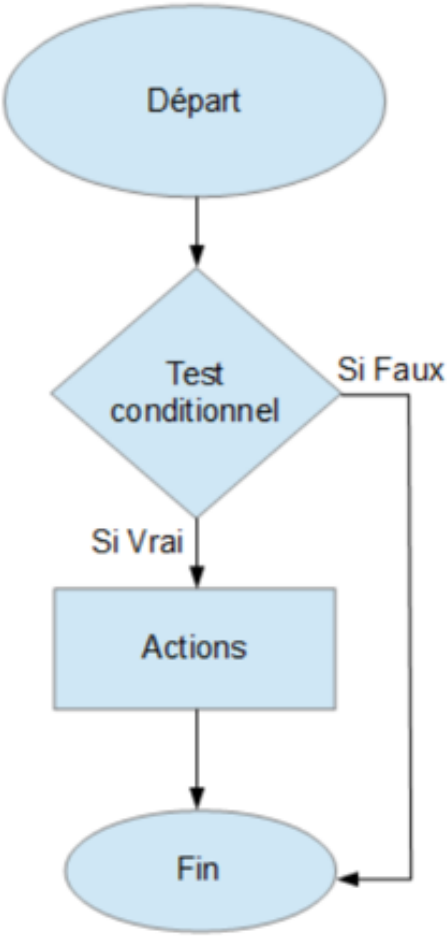
En informatique, nous utilisons le mot anglais *else* qui correspond à “sinon”. Quand on définit une condition avec “si” (*if*), le cas “sinon” (*else*) est facultatif. Nous verrons des exemples plus concrets dans la seconde partie de ce cours.

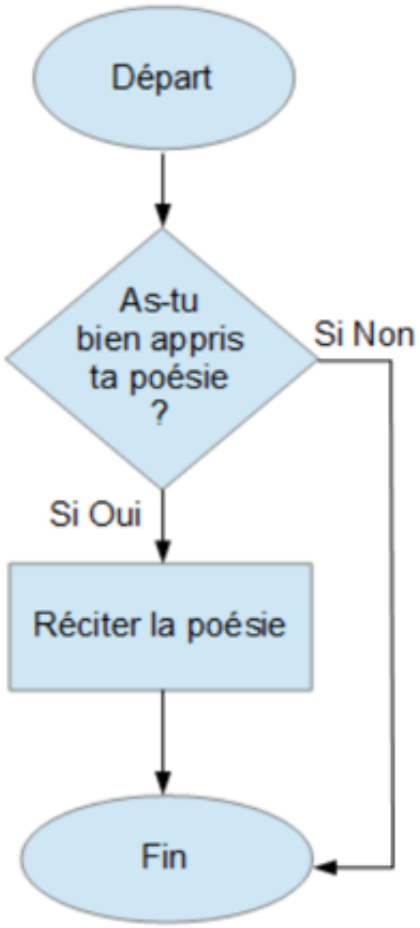
Présentation des logigrammes

Un logigramme est un diagramme logique, c’est-à-dire un schéma dans lequel des flèches relient des boîtes. Les flèches représentent le sens d’exécution des actions successives. Les boîtes sont de forme rectangulaire pour représenter des actions, en forme de losange pour représenter des choix (comme nos tests conditionnels), et en forme d’ovale pour désigner les états de départ et d’arrivée de la séquence.

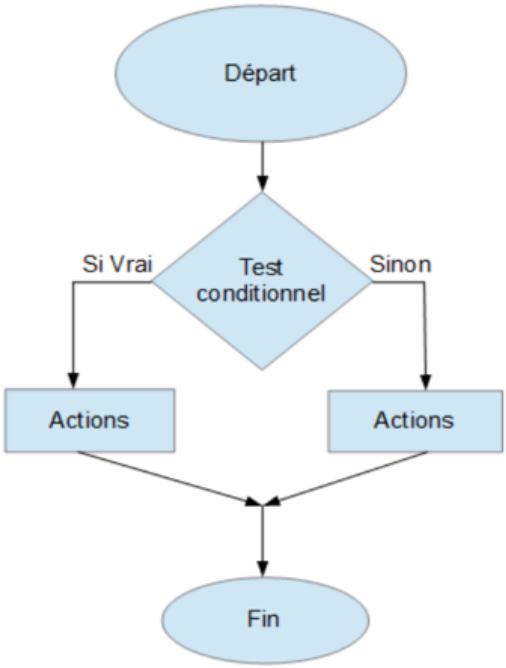
Voici deux premiers exemples de logigrammes :

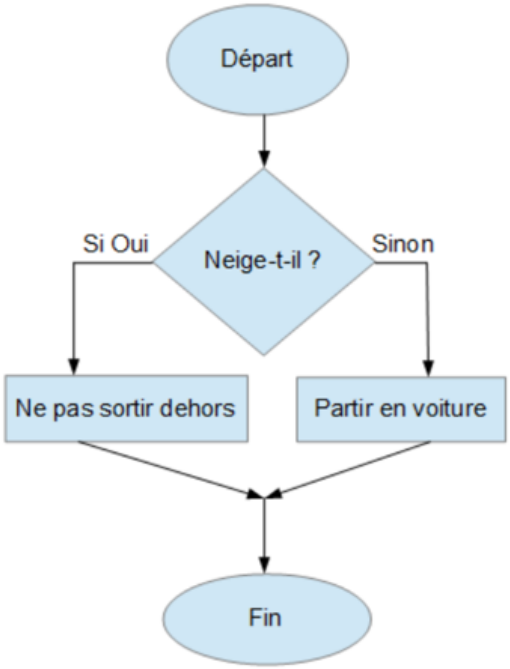
- Le premier, sur la gauche, présente de façon générique le cas du “Si”,
- Le second, sur la droite, présente notre exemple précédent de l’apprentissage d’une poésie.





Voici maintenant deux exemples de logigrammes présentant le cas du “Sinon” : le premier de façon générique, et le second avec notre exemple précédent de la chute de neige.





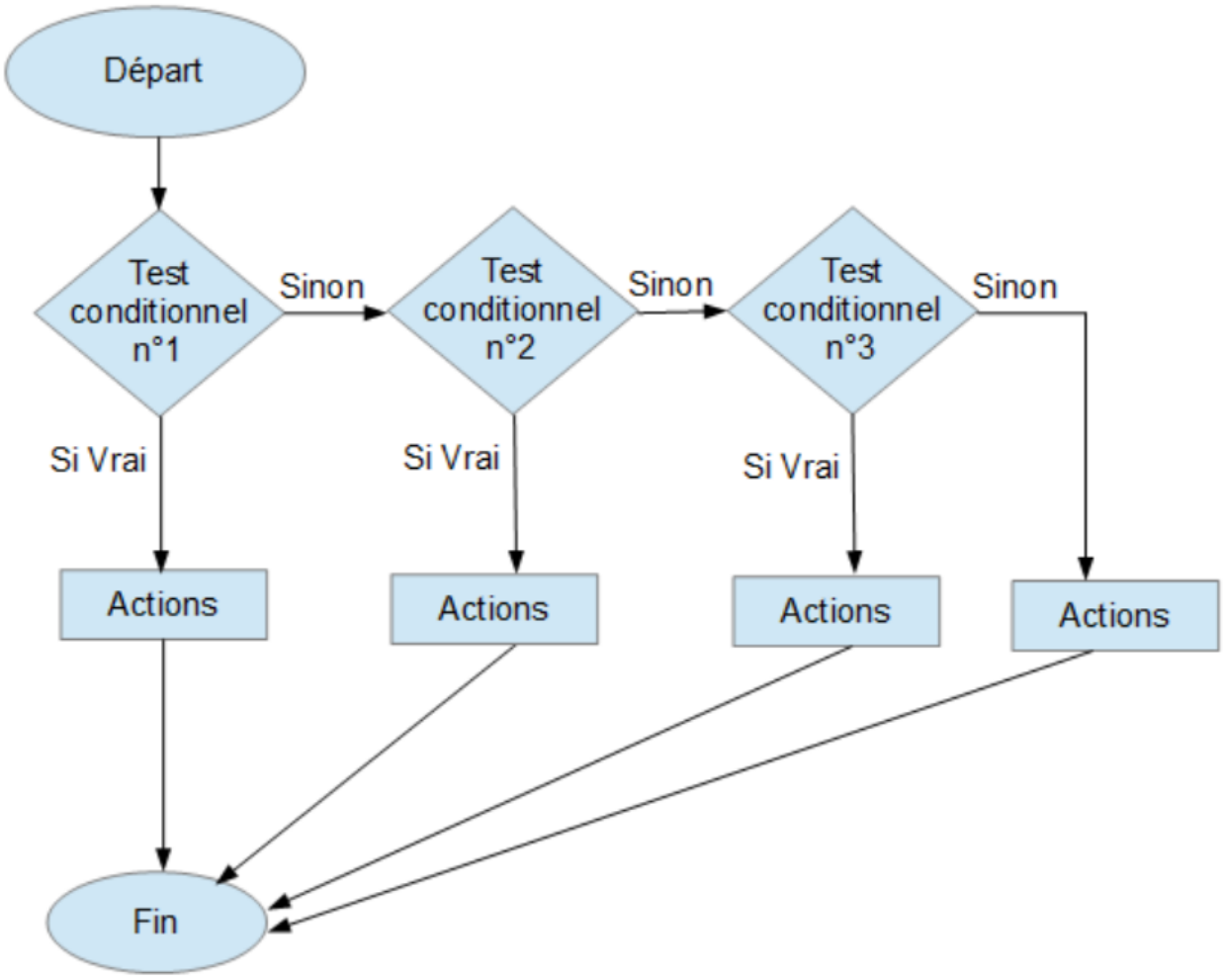
Principe du “Sinon Si”

Certaines situations ne correspondent pas avec une approche binaire Vrai/Faux, mais requièrent plus de nuances, afin d’avoir des actions adéquates. Dans ce cas, nous utilisons les mots “sinon si”. Voici deux exemples pour illustrer ce principe :

- Si la température est supérieure à 40 °C, alors le bain est trop chaud, sinon si la température est inférieure à 25 °C, alors le bain est trop froid, sinon le bain est à la bonne température.
- Si ta note est inférieure à 10, alors tu n’as pas ton diplôme, sinon si ta note est inférieure à 12, alors tu as ton diplôme sans mention, sinon si ta note est inférieure à 14, alors tu as ton diplôme avec la mention AB, sinon si tu as une note inférieure à 16, alors tu as ton diplôme avec la mention B, sinon tu as une mention TB.

En informatique, nous utilisons les mots anglais "else if” pour exprimer ce concept. Certains langages utilisent une contraction de ces mots avec “elif”. Il est possible de définir autant de “sinon si” que l’on veut, mais le code doit avant tout rester lisible et compréhensible. Nous verrons des exemples plus concrets dans la seconde partie de ce cours.

Le principe du “Sinon Si” peut se représenter selon le logigramme suivant :



Principe du “Selon”

Dans certains cas, une succession de *else if* peut devenir fastidieuse, surtout s’il s’agit de comparer un unique élément avec différentes valeurs. Dans ce cas, nous utilisons en français le mot “selon”. Voici deux exemples en langue française pour présenter le principe du “selon” :

- Selon que votre température vaut :
  - 37 °C, alors tout va bien,
  - 38 °C, alors vous êtes légèrement fiévreux,
  - 39 °C, alors vous êtes vraiment fiévreux,
  - 40 °C, alors vous êtes très fiévreux.
- Indiquer un fruit, selon que son initiale vaut :
  - “a”, alors écrire “ananas”
  - “b”, alors écrire “banane”
  - “c”, alors écrire “cerise”
  - “d”, alors écrire “datte”
  - Etc.

Vous constatez alors que cette situation permet de prendre en compte toute une liste de valeurs différentes, pour un unique élément (la température pour le premier exemple, et l’initiale pour le second exemple). Le principe du “selon” fonctionne comme un aiguillage, afin de faire prendre un chemin parmi plusieurs chemins possibles. Cet aiguillage se traduit en anglais par le mot *switch*. Les différents cas possibles sont appelés *case* en anglais.

En langage PHP et JavaScript, l’implémentation du “selon” utilise les mots “switch” et “case”. Le “switch” introduit l’élément à comparer (dans nos exemples, la température et l’initiale du fruit), tandis que le “case” annonce les différents cas possibles à considérer.

Une particularité du “Selon”, le “Stop”

Lors de l’exécution d’un *switch* dans un programme informatique, le comportement est de tester les différentes valeurs possibles (les *cases*) dans l’ordre, du premier au dernier. Si la valeur ne correspond pas, les actions associées à ce “case” ne sont pas exécutées, et le fonctionnement passe au test de la valeur suivante (le *case* suivant). Si la valeur correspond, alors les actions associées à cette case sont exécutées. Cependant, toutes les actions de tous les *cases* suivantes sont également exécutées. C’est le comportement normal de ces langages. Or, la plupart du temps, on ne souhaite exécuter que les actions du “case” qui correspond et non toutes les actions de tous les *cases* suivantes. Pour cela, nous utilisons en français un “stop” dans les actions, qui a pour effet de terminer l’exécution du “switch”. En anglais, et dans les langages de programmation illustrés dans ce cours, le mot-clé pour cela est “break”. En général, nous utilisons la commande *break* à la fin de chaque *case*. Cela permet de s’assurer que seules les actions du cas correspondant sont exécutées.

Exemple **Concret avec Switch, Case, et Break**

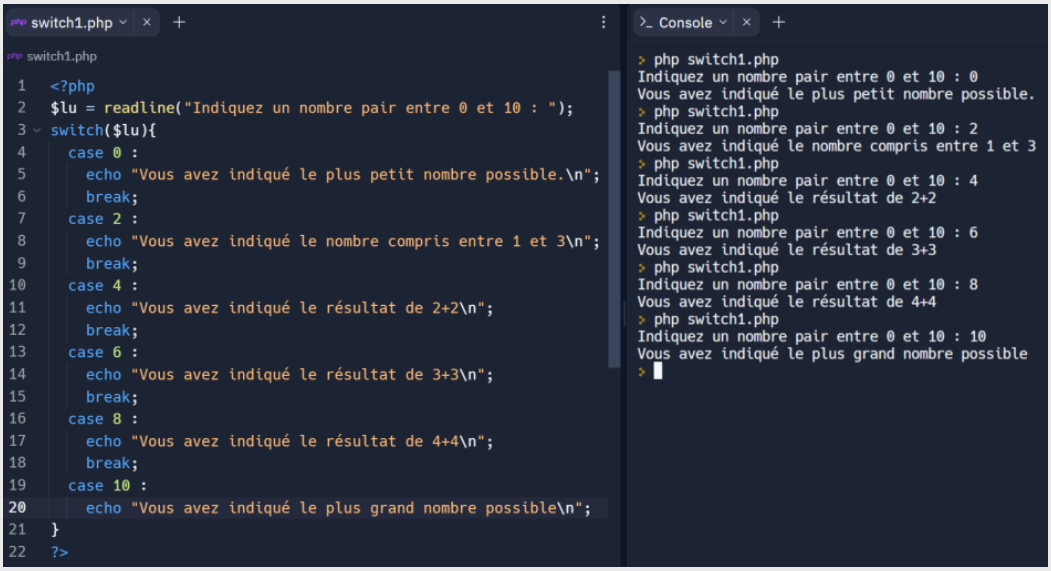
Pour illustrer ce que nous venons de dire à propos du “Selon” et du “Stop”, vous avez ci-dessous une capture d’écran d’un petit programme PHP écrit dans Replit. Dans le panneau de gauche, un fichier nommé “switch1.php” a été rédigé. Dans le panneau de droite, la console, ce fichier “switch1.php” a été exécuté 6 fois.

Voyons en détail ce que fait ce programme :

- La commande “readline” affiche à l’écran le texte indiqué et attend que l’utilisateur saisisse une information. L’information saisie par l’utilisateur est alors enregistrée dans la variable nommée “$lu”.
- Nous avons programmé un “switch” sur la variable “$lu”, et défini 6 “case” correspondant aux 6 valeurs attendues (0, 2, 4, 6, 8, 10).

- Pour chaque “case”, nous utilisons la commande “echo” qui affiche à l’écran le texte indiqué, puis nous indiquons la commande “break”.

Dans le panneau de droite, le programme a été exécuté 6 fois afin de vous faire voir le résultat de chaque “case”.



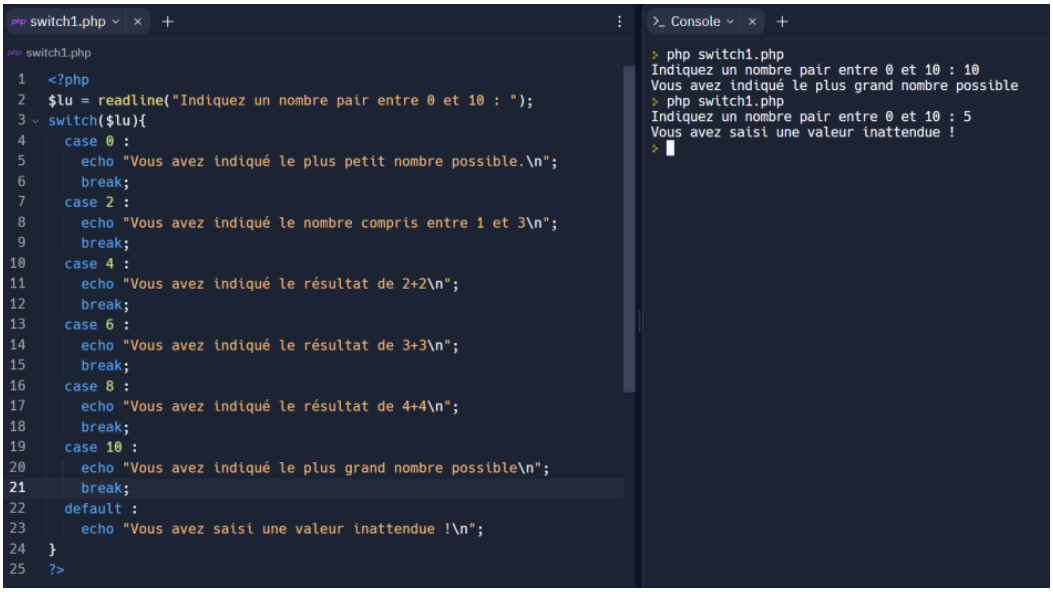
La commande “break” est ici indispensable. Si elle était absente, le fait de saisir la valeur “0” afficherait les 6 phrases correspondant aux 6 “cases”.

Principe du “Selon” et “Sinon”

Nous avons vu, lors de la présentation du “Si” (if), qu’il est possible de définir une clause “Sinon” (else). Cette dernière est exécutée si aucune des conditions précédentes n’a pu être exécutée. Elle peut être considérée comme le cas échéant.

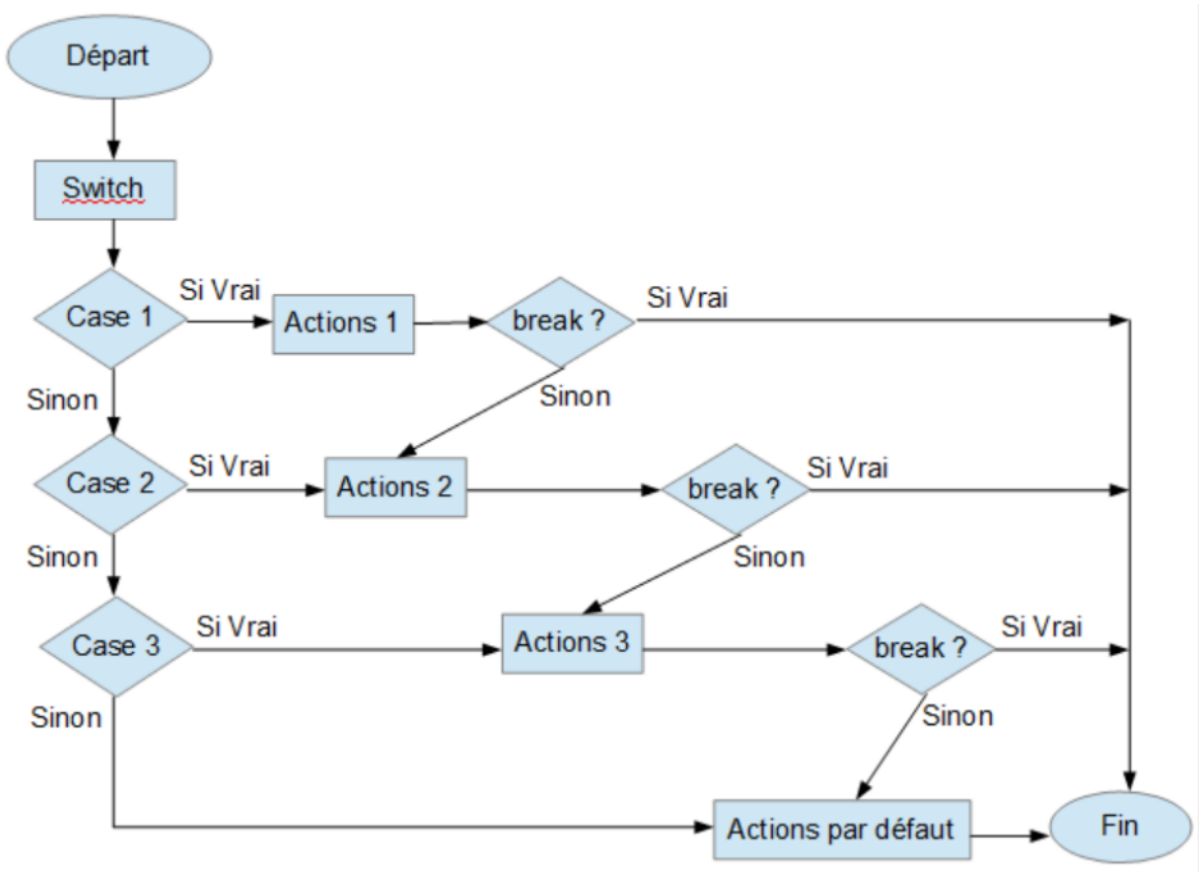
Dans le cas du “Selon” (*switch*), il est possible également de définir un cas exécuté si aucun autre “case” n’a été exécuté. Il s’agit du cas par défaut, qui est introduit à l’aide du mot-clé “default”.

Dans la capture d’écran ci-dessous, nous avons complété l’exemple précédent avec un cas par défaut. Nous avons également exécuté le programme dans la console pour vous rendre compte du fonctionnement.



Logigramme du “Switch”

Dans l’illustration ci-dessous, 3 “cases” ont été définis. Ces derniers apparaissent alignés à gauche à la verticale. Lorsqu’un “case” satisfait un test du “switch”, un groupe d’actions est exécuté. Ensuite, si la commande “break” est indiquée, alors le traitement du “switch” est terminé. Mais si la commande “break” est absente, alors les actions des autres “case” sont exécutées. Enfin, si aucun “case” n’a été satisfait, les actions par défaut sont exécutées (si elles ont été définies).



Différences entre “Selon” et “Sinon Si”

De manière générale, une démarche de programmation avec la structure “Sinon Si” est suffisante, c’est-à-dire que la structure du “Selon” n’est pas nécessaire. D’ailleurs, le langage Python qui est plus récent que PHP et JavaScript ne possède pas de structure “Selon”. Il est donc tout à fait possible d’écrire des programmes sans ces structures “Selon”. Cependant, elles font partie des principes de base de la programmation et sont utilisées depuis des décennies. Il est donc utile de vous les faire découvrir.

Il y a 3 différences majeures entre le “switch” et le “if… else if… else” :

- **Le “break” :** dans une démarche “Sinon Si”, seules les actions associées à un “if” sont exécutées, tandis que dans une démarche “Selon”, toutes les actions après le cas correspondant sont effectuées, à moins de placer la commande “break” pour arrêter leur exécution.
- **L’élément comparé :** dans un “switch”, on compare un unique élément - rappelez-vous nos premiers exemples sur cette structure - la température et l’initiale d’un fruit - alors qu’après un “if”, on peut indiquer toute sorte de tests conditionnels, indépendamment des tests précédents.
- **La comparaison :** dans un “switch”, la comparaison effectuée est un test d’égalité, tandis qu’après un “if”, le test conditionnel peut utiliser divers opérateurs de comparaison, comme le “plus petit que” (<) ou le “plus grand que” (>) par exemple.

III. Manipuler les structures de contrôles

A. Manipuler les structures de contrôles

Principe de cette section

Dans cette seconde partie du cours, nous allons nous entraîner à manipuler les structures de contrôle décrites avant. Nous allons soumettre des situations, en langue française, et vous êtes invité à réfléchir à la manière de conceptualiser ces situations à l'aide des structures de contrôles. Une ou plusieurs solutions sont proposées pour vous permettre de comparer votre démarche.

Méthode **Manipuler le "if"**

Pour ce premier exercice, nous nous plaçons dans le contexte d'un étudiant qui vient d'obtenir sa note pour la délivrance de son diplôme. Nous ne connaissons pas la valeur de sa note, mais nous savons qu'elle est au maximum de 20. Nous représenterons cette note par la lettre N. Lors de l'affichage du résultat, sur écran ou sur listing papier, le libellé "reçu" doit apparaître dès que l'étudiant a obtenu au moins la moyenne, c'est-à-dire une note au moins égale à 10. Prenez maintenant le temps de réfléchir à la structure à utiliser dans ce cas, pour afficher le libellé "reçu", en utilisant les mots anglais indiqués précédemment.

Voici une réponse qui peut convenir : IF (N supérieur ou égal à 10), Afficher "reçu". Nous verrons plus loin en vidéo l'implémentation en PHP de ce petit scénario.

Pour le deuxième exercice, nous nous rapprochons du jeu du pendu, qui consiste à trouver un mot, lettre par lettre. Nous nous intéressons plus particulièrement au moment où le joueur a saisi une lettre et que le programme doit tester la présence de cette lettre dans le mot mystère. Si cette lettre est présente, il convient de l'afficher. Nous décidons de désigner le mot mystère avec la lettre M, et la lettre saisie par la lettre L. À vous de réfléchir maintenant à la structure à utiliser, avant de vous proposer une solution.

Voici une réponse qui peut convenir : IF (L appartient à M), Afficher L dans M. Nous verrons ce jeu plus en détail par la suite.

Méthode **Manipuler le "if… else"**

L'exercice suivant consiste à vérifier la disponibilité d'un solde suffisant sur votre compte bancaire pour réaliser un achat à l'aide de votre carte de paiement. Nous considérons qu'il s'agit d'une carte à contrôle de solde, qui ne permet pas d'être à découvert. Nous ne connaissons bien évidemment pas, à l'avance, ni le montant de la transaction ni le solde du compte bancaire au moment du contrôle. Nous représentons le montant par la lettre M et le solde par la lettre S. Si le solde est suffisant, alors le débit est autorisé et l'achat est réalisé. Si le solde est insuffisant, alors l'autorisation de débit est refusée et l'achat n'est pas réalisé. En utilisant les mots décrits précédemment, réfléchissez à la structure à utiliser pour mettre en œuvre la situation décrite.

Voici une réponse qui peut convenir :

IF (S supérieur à M), Autoriser la transaction, ELSE Refuser la transaction.

Dans la capture d'écran ci-dessous, nous avons implémenté cette situation en PHP dans Replit. Pour cette démonstration pédagogique, nous ne réalisons pas l'autorisation bancaire, mais affichons juste un texte qui indique l'action réalisée. Afin d'avoir un code opérationnel pour tout montant d'achat et tout solde bancaire, ces deux informations sont demandées à l'utilisateur au début du programme. Dans le panneau de droite, le programme a été exécuté deux fois : la première fois avec un solde suffisant (600 €) pour le montant saisi (50 €), la deuxième fois avec un solde insuffisant (400) pour le montant spécifié (800 €).

Complément

Voici un nouvel exercice qui se déroule dans un manège. Vous pouvez contextualiser le sujet comme il vous plaît : un simple carrousel pour enfants sur la place du marché de votre quartier, ou une grande attraction pour adulte dans votre parc favori. Le but est d'afficher en permanence le bon message à l'usager, qui doit rester tout le temps en sécurité. Ainsi, quand le manège est en cours d'exécution, il faut absolument rester assis. Tandis que lorsque le manège est à l'arrêt, ce qui signifie que le tour est terminé, il convient de se lever pour céder la place, et quitter l'attraction. Nous pouvons représenter le manège par la lettre M. Vous devez maintenant réfléchir pour formaliser cette situation à l'aide de la structure de contrôle, et afficher un message approprié en fonction de la situation.

Voici une manière qui pourrait convenir :

IF (M a terminé son tour), Afficher "Levez-vous et sortez maintenant", ELSE Afficher "Restez assis".

Il est possible, bien évidemment, de l'aborder d'une autre manière, par exemple :

IF (M en cours d'exécution), Afficher "Restez assis", ELSE Afficher "Quittez l'attraction".

Il n'y aura jamais une seule façon de solutionner une situation. L'important est d'être méthodique et de conceptualiser votre démarche. Vous pouvez vous aider avec un logigramme pour les situations les plus complexes. Vous finirez toujours par trouver une structure opérationnelle en effectuant des tests fonctionnels.

Méthode **Manipuler le "if… else if… else"**

L'exemple suivant a été évoqué dans la première partie de ce cours, mais nous allons maintenant l'approfondir. Il consiste précisément à déterminer la mention obtenue lors d'un examen scolaire. Il convient de considérer tous les cas suivants :

- Pour une note strictement inférieure à 10 : l'examen est "non obtenu".
- Pour une note comprise entre 10 et strictement inférieure à 12 : l'examen est obtenu avec "mention passable".
- Pour une note comprise entre 12 et strictement inférieure à 14 : l'examen est obtenu avec "mention assez bien".
- Pour une note comprise entre 14 et strictement inférieure à 16 : l'examen est obtenu avec "mention bien".
- Pour une note au moins égale à 16 : l'examen est obtenu avec "mention très bien".

Vous devez réfléchir à la structure de contrôle à utiliser, avec les différents tests conditionnels. L'action à implémenter, dans ce cas précis, est simplement un affichage de la mention, comme indiqué ci-dessus entre guillemets. Nous pouvons représenter la note d'examen par la lettre N.

Il y a bien sûr plusieurs manières de faire. Nous vous en présentons deux pour mieux comprendre les différentes approches possibles :

- IF (N<10), Afficher “non obtenu”

  ELSE IF (N<12), Afficher “mention passable”

  ELSE IF (N<14), Afficher “mention assez bien”

  ELSE IF (N<16), Afficher “mention bien”

  ELSE Afficher “mention très bien”

- IF (N>=16), Afficher “mention très bien”

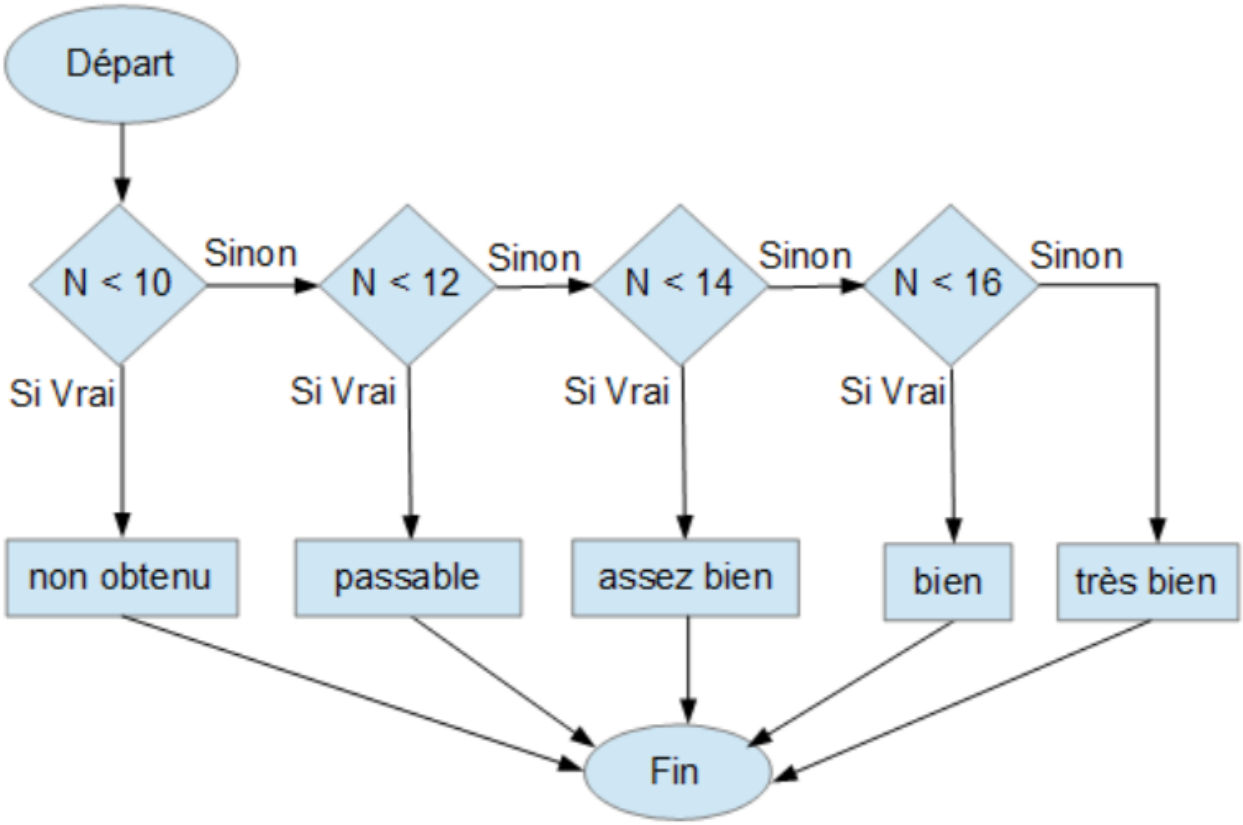
  ELSE IF (N>=14), Afficher “mention bien”

  ELSE IF (N>=12), Afficher “mention assez bien”

  ELSE IF (N>=10), Afficher “mention passable”

  ELSE Afficher “non obtenu”

Vous trouverez ci-dessous un logigramme pour cet exemple précis, qui met en œuvre la première approche décrite. Prenez le temps de bien le comprendre, et imaginez-vous en train de l’écrire vous-même.



Nous allons également voir cet exercice en vidéo juste après, avec une implémentation en PHP dans Replit.

Manipuler le “Switch” de base

Nous allons maintenant nous entraîner avec la structure de contrôle “switch”, en nous appuyant sur la capture d’écran ci-dessous. Dans cet exemple, nous commençons par demander à l’utilisateur de saisir un nombre parmi 3 choix possibles : 2, 4, ou 12. Nous utilisons ensuite le nombre saisi par l’utilisateur dans un switch, dans lequel chaque choix possible correspond à un “case". Enfin, chaque “case” déclenche l’affichage d’une affirmation à propos du nombre saisi :

- Dans le cas où le nombre 2 est saisi, le message “Vous avez indiqué un multiple de 2.” s’affiche.
- Dans le cas où le nombre 4 est saisi, le message “Vous avez indiqué un multiple de 4.” s’affiche.
- Dans le cas où le nombre 12 est saisi, le message “Vous avez indiqué un multiple de 3.” s’affiche.

Si je vous demandais de réfléchir à l’implémentation de cette situation, avec le nombre saisi représenté par la lettre N, vous imagineriez probablement l’approche suivante, en respectant l’ordre de l’énoncé :

SWITCH (N)

CASE 2, Afficher “Vous avez indiqué un multiple de 2.”, BREAK.

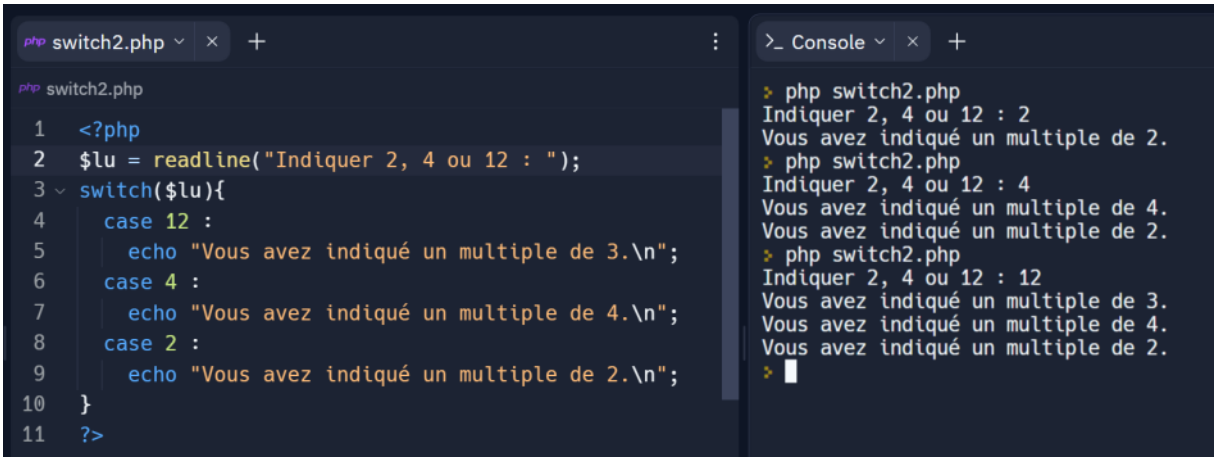
CASE 4, Afficher “Vous avez indiqué un multiple de 4.”, BREAK.

CASE 12, Afficher “Vous avez indiqué un multiple de 3.”, BREAK.

Cependant, le but principal de cet exercice est de ne pas utiliser la commande “break”, mais uniquement “switch” et “case”. Avec cette contrainte supplémentaire, il faut faire preuve d’ingéniosité afin de trouver l’ordre des “case” qui convient. En effet, sans la commande break, les actions des “case” vont s’exécuter les unes à la suite des autres à partir du moment où un “case” est satisfait. Pour bien le comprendre, vous pouvez regarder à nouveau le logigramme du “switch” présenté dans la première partie de ce cours.

Dans la capture ci-dessous, le programme est écrit dans le panneau de gauche. Dans le panneau de droite, le programme a été exécuté 3 fois correspondant à chacun des 3 cas possibles :

- Dans le cas où le nombre 2 a été saisi, seul le message “Vous avez indiqué un multiple de 2.” S’affiche.
- Dans le cas où le nombre 4 a été saisi, deux messages s’affichent : “Vous avez indiqué un multiple de 4.” Et “Vous avez indiqué un multiple de 2.”
- Dans le cas où le nombre 12 a été saisi, trois messages s’affichent : “Vous avez indiqué un multiple de 3.”, puis “Vous avez indiqué un multiple de 4.”, et “Vous avez indiqué un multiple de 2.”



Cet exemple vous révèle une particularité du “switch”, apportant la capacité simultanée de distinguer des cas spécifiques tout en permettant d’exécuter du code commun. Il est intéressant d’avoir rencontré ce cas de figure pour votre raisonnement, même si, dans la pratique, le “switch” s’utilise le plus souvent avec le “break”.

Méthode **Manipuler “break” avec “switch”**

Dans ce nouvel exercice, vous allez réfléchir à la syntaxe à utiliser avec l’exemple que nous avons donné plus haut, selon que votre température vaut :

- 37 °C, alors tout va bien,
- 38 °C, alors vous êtes légèrement fiévreux,
- 39 °C, alors vous êtes vraiment fiévreux,
- 40 °C, alors vous êtes très fiévreux.

Prenez maintenant quelques secondes pour vous imaginer à rédiger un code qui implémente cette situation, en utilisant la lettre T pour représenter la température.

Vous avez pu, par exemple, imaginer ceci :

SWITCH (T)

CASE 37, Afficher “va bien”, BREAK

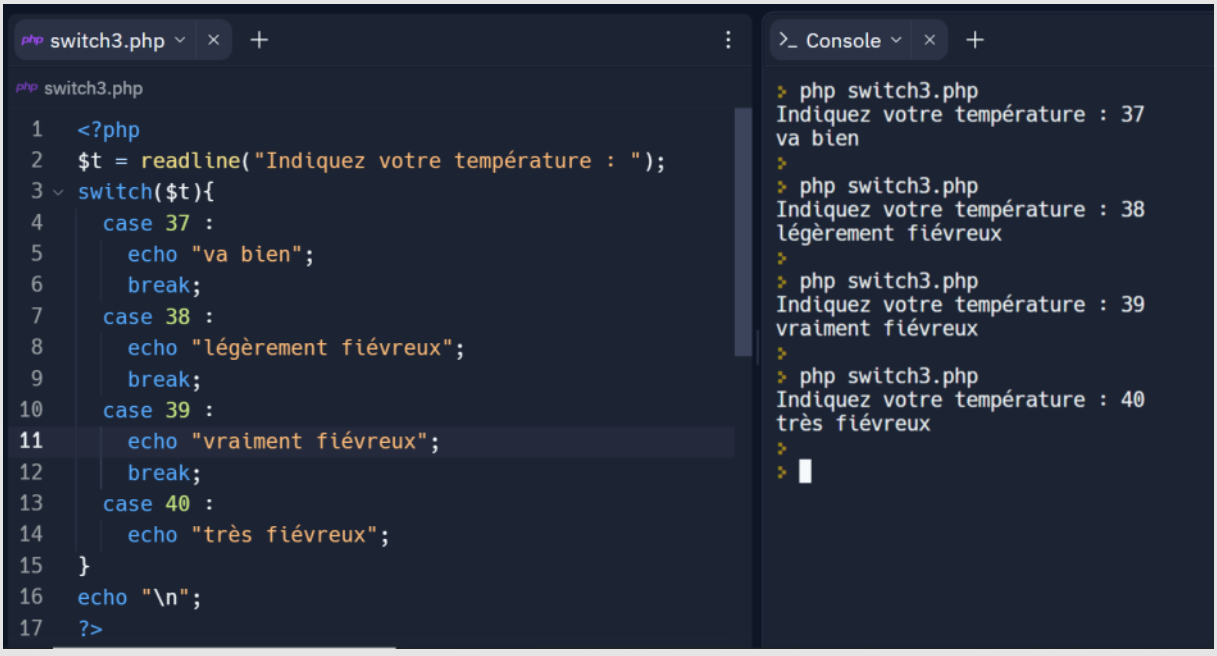
CASE 38, Afficher “légèrement fiévreux”, BREAK

CASE 39, Afficher “vraiment fiévreux”, BREAK

CASE 40, Afficher “très fiévreux”, BREAK

L’enjeu de cet exercice était bien évidemment de penser à la commande “break” à la fin de chaque “case”. Si vous les avez oubliés, ce n’est pas bien grave à ce stade : lorsque vous programmerez vous-même en vrai, vous vous rendrez vite compte de vos oublis, et les corrigerez rapidement.

Afin de parfaire cette section, vous trouverez ci-dessous une capture d’écran de l’implémentation de ce contexte en PHP sous Replit. À gauche, le code du programme. À droite, l’exécution du programme, faisant apparaître les 4 cas d’utilisation.



Méthode **Manipuler “default” avec “switch”**

Nous allons maintenant nous intéresser au cas par défaut. Cela est loin d’être anecdotique. En effet, en informatique nous devons toujours être vigilants à tous les scénarios envisageables, même ceux qui sont censés ne jamais se produire. En ce sens, le cas par défaut du “switch” permet aisément de parer à tout comportement, qu’il soit prévu, ou imprévu.

Dans cet exercice, nous souhaitons coder un programme qui indique si l'initiale d'un mot donné commence par une lettre minuscule ou une lettre majuscule. Pour rester dans une approche simpliste, nous allons regarder uniquement les lettres "e" et "E". Néanmoins, il convient de prendre en compte le cas où le mot saisi ne commence ni par "e" ni par "E".

En utilisant "switch", "case" et "default", et en notant par la lettre I l'initial du mot donné, prenez maintenant quelques secondes pour imaginer la syntaxe du pseudo-code décrivant cette situation.

Vous avez pu, par exemple, imaginer cette approche :

SWITCH (I)

CASE e, Afficher "Votre mot commence par une minuscule.", BREAK

CASE E, Afficher "Votre mot commence par une majuscule.", BREAK

DEFAULT, Afficher "Je n'ai pas pu déterminer votre lettre."

Dans la capture d'écran ci-dessous, un programme correspondant à cet exercice a été écrit dans le panneau de gauche. On commence par demander un mot à l'utilisateur, puis on indique au "switch" la première lettre de ce mot. Dans le panneau de droite, nous avons exécuté 4 fois ce programme.

Pour les deux premières exécutions, nous avons indiqué des mots qui satisfont respectivement chacun des deux cas : en minuscule puis en majuscule. Pour les deux cas suivants, nous avons indiqué des mots avec accent. Il s'agit donc de lettres modifiées, qui ne sont pas reconnues par notre programme, et qui déclenchent par conséquent la clause par défaut. Notez que, dans ce cas simpliste, tout mot commençant par autre chose que "e" et "E" déclenche la clause par défaut. Ces exemples avec les mots ''école'' et ''ÊTRE'' présentent un intérêt particulier, car de premier abord on pourrait être tenté de s'attendre que le programme reconnaisse la minuscule de ''école'' et la majuscule de ''ÊTRE'', mais il n'en est rien. Le cas par défaut a donc son utilité démontrée.

IV. Auto-évaluation

A. Exercice

Le code suivant permet de jouer à retrouver un mot mystère selon la célèbre méthode du pendu. Cependant, 4 morceaux de ce code ont été remplacés par « *??????* ». Vous devrez écrire les morceaux manquants pour exécuter le programme. De nombreux commentaires ont été écrits pour nous permettre de comprendre plus aisément le fonctionnement du programme.

```php
<?php
echo "Jeu du pendu : retrouvez le mot mystère !\n" ;
$mot_a_trouver = "PROGRAMMATION" ;
$mot_a_completer = "____________" ;
$essais_restants = 12 ; # le nombre d'erreurs autorisées
# la commande while permet d'exécuter en boucle tout le code compris entre accolades,
# tant que les expressions booléennes entre parenthèses sont évaluées à True
```

```php
while($mot_a_trouver != $mot_a_completer && $essais_restants > 0){
    echo "****************************************\n";
    echo "Mot à compléter : {$mot_a_completer}\t\t" ;
    echo "Nombre d'essais restants : {$essais_restants}\n" ;
    # la commande readline permet à l'utilisateur de saisir une information au clavier
    # l'information saisie est mémorisée dans la variable $lettre
    $lettre = readline("Indiquez votre lettre en majuscule, et validez avec la touche Entrée :
") ;
    $lettre_absente = true ;
    # cette variable booléenne nous servira pour savoir si la lettre saisie est absente du mot
mystère
    # elle est initialisée à true pour signifier "la lettre saisie est absente du mot mystère"
    #  la commande for permet de répéter l'exécution du code entre accolades
    # un certain nombre de fois, dans ce cas précis 13 fois, car le mot mystère contient 13
lettres
    for($i=0;$i<13;$i+=1){
        ?????? ($mot_a_trouver[$i]==$lettre){
            # dans le cas où la i-ème lettre du mot mystère correspond à la lettre saisie
            # on complète le mot, à la i-ème position, par la lettre saisie
            $mot_a_completer[$i]=$lettre ;
            # on indique que la lettre saisie n'est pas absente du mot mystère
            $lettre_absente = false ;
        }
    } # fin de la boucle for
    # chaque lettre du mot mystère a été comparée à la lettre saisie
    # à ce moment, si la lettre saisie est toujours absente du mot mystère
    # c'est qu'il s'agit d'une lettre en erreur et il convient de décompter le nombre d'essais
restants
   ?????? $essais_restants -= 1 ;
} # fin de la boucle while
# l'expression booléenne du while est évaluée à False
# si l'utilisateur a trouvé le mot, c'est qu'il avait encore un droit à l'erreur
# si l'utilisateur a utilisé toutes les erreurs autorisées, il n'a pas trouvé le mot
?????? echo "Bravo ! Vous avez trouvé le mot mystère : {$mot_a_trouver}\n" ;
?????? echo "Pendu ! Le mot mystère était {$mot_a_trouver}\n" ;
?>
```

Question 1

[solution n°1 p.19]

Identifiez les quatre morceaux ?????? à réécrire. Déterminez les mots-clés à employer. Expliquez votre raisonnement qui permettra d'écrire les tests conditionnels manquants.

Question 2

[solution n°2 p.19]

Écrivez les deux morceaux manquants. Vous pourrez ensuite exécuter le jeu avec un proche pour partager vos nouvelles compétences.

B. Test

Exercice 1 : Quiz

[solution n°3 p.20]

Question 1

La commande IF effectue un test conditionnel, dont la valeur est :

- ○ Numérique
- ○ Booléenne
- ○ Quelconque

Question 2

On veut comparer les âges d'une fratrie. Pour cela, quelle structure de contrôle est idéale ?

- Le IF
- Le SWITCH
- Aucune, les deux peuvent être utilisées

Question 3

Trois étudiants ont conceptualisé la notion de ponctualité. Un rendez-vous est considéré ponctuel en acceptant un écart de 5 minutes. L'écart est signé par rapport à l'horaire prévu, de sorte que la valeur 10 correspond à un retard de 10 minutes et la valeur - 10 correspond à un délai d'attente (en avance) de 10 minutes. Choisissez l'unique bonne réponse parmi les 3 propositions.

- IF (écart < 5), Afficher "ponctuel", ELSE Afficher "en retard"
- IF (écart < -5), Afficher "trop en avance", ELSE IF (écart < 5), Afficher "à l'heure", ELSE Afficher "trop en retard"
- IF (écart > 5), Afficher "trop tard", ELSE IF (écart > 0), Afficher "ponctuel", ELSE IF (écart = 0), Afficher "pile à l'heure"

Question 4

Le programme PHP suivant est utilisé par la caisse d'un musée pour calculer le tarif de la clientèle. L'hôtesse de caisse saisit 20 personnes. Quel montant doit être payé ?

```php
<?php
$individuel = 10 ;
$groupe = 5 ;
$n = readline("Saisir le nombre de personnes : ") ;
if ($n<20) {
  echo "Tarif individuel appliqué. Total : ".$individuel*$n ;
}
else {
  echo "Tarif groupe appliqué. Total : ".$groupe*$n;
}
echo "\n";
?>
```

- 20
- 100
- 200

Question 5

Nous considérons le programme suivant. Quel affichage est produit par la saisie du nombre 4 ?

```php
<?php
$n = readline("Indiquez un nombre entre 1 et 4 : ");
switch($n){
  case 3 :
    echo "Vous avez indiqué un nombre impair.\n";
  case 1 :
    echo "Vous avez indiqué le plus petit entier positif non nul.\n";
    break;
  case 4 :
    echo "Vous avez indiqué un nombre plus grand que 3.\n";
  case 2 :
    echo "Vous avez indiqué un nombre pair.\n";
```

```
13     break;
14   default :
15     echo "Vous avez saisi une donnée non reconnue.\n";
16 }
17 ?>
```

- [ ] Vous avez indiqué un nombre plus grand que 3
- [ ] Vous avez indiqué un nombre pair
- [ ] Vous avez saisi une donnée non reconnue

Solutions des exercices

p. 16 Solution n°1

```php
<?php
echo "Jeu du pendu : retrouvez le mot mystère !\n" ;
$mot_a_trouver = "PROGRAMMATION" ;
$mot_a_completer = "_____________" ;
$essais_restants = 12 ; # le nombre d'erreurs autorisées
# la commande while permet d'exécuter en boucle tout le code compris entre accolades,
# tant que les expressions booléennes entre parenthèses sont évaluées à True
while($mot_a_trouver != $mot_a_completer && $essais_restants > 0){
    echo "****************************************\n";
    echo "Mot à compléter : {$mot_a_completer}\t\t" ;
    echo "Nombre d'essais restants : {$essais_restants}\n" ;
    # la commande readline permet à l'utilisateur de saisir une information au clavier
    # l'information saisie est mémorisée dans la variable $lettre
    $lettre = readline("Indiquez votre lettre en majuscule, et validez avec la touche Entrée : ") ;
    $lettre_absente = true ;
    # cette variable booléenne nous servira pour savoir si la lettre saisie est absente du mot mystère
    # elle est initialisée à true pour signifier "la lettre saisie est absente du mot mystère"
    #  la commande for permet de répéter l'exécution du code entre accolades
    # un certain nombre de fois, dans ce cas précis 13 fois, car le mot mystère contient 13 lettres
    for($i=0;$i<13;$i+=1){
       if ($mot_a_trouver[$i]==$lettre){
            # dans le cas où la i-ème lettre du mot mystère correspond à la lettre saisie
            # on complète le mot, à la i-ème position, par la lettre saisie
            $mot_a_completer[$i]=$lettre ;
            # on indique que la lettre saisie n'est pas absente du mot mystère
            $lettre_absente = false ;
        }
    } # fin de la boucle for
    # chaque lettre du mot mystère a été comparée à la lettre saisie
    # à ce moment, si la lettre saisie est toujours absente du mot mystère
    # c'est qu'il s'agit d'une lettre en erreur et il convient de décompter le nombre d'essais restants
   if $essais_restants -= 1 ;
} # fin de la boucle while
# l'expression booléenne du while est évaluée à False
# si l'utilisateur a trouvé le mot, c'est qu'il avait encore un droit à l'erreur
# si l'utilisateur a utilisé toutes les erreurs autorisées, il n'a pas trouvé le mot
if echo "Bravo ! Vous avez trouvé le mot mystère : {$mot_a_trouver}\n" ;
else echo "Pendu ! Le mot mystère était {$mot_a_trouver}\n" ;
?>
```

p. 16 Solution n°2

```php
<?php
echo "Jeu du pendu : retrouvez le mot mystère !\n" ;
$mot_a_trouver = "PROGRAMMATION" ;
$mot_a_completer = "_____________" ;
$essais_restants = 12 ; # le nombre d'erreurs autorisées
# la commande while permet d'exécuter en boucle tout le code compris entre accolades,
# tant que les expressions booléennes entre parenthèses sont évaluées à True
while($mot_a_trouver != $mot_a_completer && $essais_restants > 0){
```

```php
    echo "****************************************\n";
    echo "Mot à compléter : {$mot_a_completer}\t\t" ;
    echo "Nombre d'essais restants : {$essais_restants}\n" ;
    # la commande readline permet à l'utilisateur de saisir une information au clavier
    # l'information saisie est mémorisée dans la variable $lettre

    $lettre = readline("Indiquez votre lettre en majuscule, et validez avec la touche Entrée : ") ;
    $lettre_absente = true ;

    # cette variable booléenne nous servira pour savoir si la lettre saisie est absente du mot mystère
    # elle est initialisée à true pour signifier "la lettre saisie est absente du mot mystère"
    #  la commande for permet de répéter l'exécution du code entre accolades
    # un certain nombre de fois, dans ce cas précis 13 fois, car le mot mystère contient 13 lettres

    for($i=0;$i<13;$i+=1){
       if ($mot_a_trouver[$i]==$lettre){
            # dans le cas où la i-ème lettre du mot mystère correspond à la lettre saisie
            # on complète le mot, à la i-ème position, par la lettre saisie

            $mot_a_completer[$i]=$lettre ;
            # on indique que la lettre saisie n'est pas absente du mot mystère

            $lettre_absente = false ;
        }
    } # fin de la boucle for
    # chaque lettre du mot mystère a été comparée à la lettre saisie
    # à ce moment, si la lettre saisie est toujours absente du mot mystère
    # c'est qu'il s'agit d'une lettre en erreur et il convient de décompter le nombre d'essais restants
   if ($lettre_absente) $essais_restants -= 1 ;
} # fin de la boucle while
# l'expression booléenne du while est évaluée à False
# si l'utilisateur a trouvé le mot, c'est qu'il avait encore un droit à l'erreur
# si l'utilisateur a utilisé toutes les erreurs autorisées, il n'a pas trouvé le mot
if ($essais_restants > 0)  echo "Bravo ! Vous avez trouvé le mot mystère : {$mot_a_trouver}\n" ;
else echo "Pendu ! Le mot mystère était {$mot_a_trouver}\n" ;
?>
```

Exercice p. 16 **Solution n°3**

Question 1

La commande IF effectue un test conditionnel, dont la valeur est :

- ○ Numérique
- ◉ Booléenne
- ○ Quelconque

Un test conditionnel est toujours évalué à True ou False. Il s'agit donc d'une valeur booléenne.

Question 2

On veut comparer les âges d'une fratrie. Pour cela, quelle structure de contrôle est idéale ?

- (●) Le IF
- ( ) Le SWITCH
- ( ) Aucune, les deux peuvent être utilisées

La structure "switch" ne permet de comparer qu'une seule information avec plusieurs valeurs. Pour comparer plusieurs âges entre eux, la structure "if" est nécessaire.

Question 3

Trois étudiants ont conceptualisé la notion de ponctualité. Un rendez-vous est considéré ponctuel en acceptant un écart de 5 minutes. L'écart est signé par rapport à l'horaire prévu, de sorte que la valeur 10 correspond à un retard de 10 minutes et la valeur - 10 correspond à un délai d'attente (en avance) de 10 minutes. Choisissez l'unique bonne réponse parmi les 3 propositions.

- ( ) IF (écart < 5), Afficher "ponctuel", ELSE Afficher "en retard"
- (●) IF (écart < -5), Afficher "trop en avance", ELSE IF (écart < 5), Afficher "à l'heure", ELSE Afficher "trop en retard"
- ( ) IF (écart > 5), Afficher "trop tard", ELSE IF (écart > 0), Afficher "ponctuel", ELSE IF (écart = 0), Afficher "pile à l'heure"

La bonne réponse est celle qui tient compte du signe du retard : dans l'énoncé un retard négatif indique un temps d'attente, qui est excessif au-delà de 5 minutes.

Question 4

Le programme PHP suivant est utilisé par la caisse d'un musée pour calculer le tarif de la clientèle. L'hôtesse de caisse saisit 20 personnes. Quel montant doit être payé ?

```php
<?php
$individuel = 10 ;
$groupe = 5 ;
$n = readline("Saisir le nombre de personnes : ") ;
if ($n<20) {
  echo "Tarif individuel appliqué. Total : ".$individuel*$n ;
}
else {
  echo "Tarif groupe appliqué. Total : ".$groupe*$n;
}
echo "\n";
?>
```

- ( ) 20
- (●) 100
- ( ) 200

Le test du IF est $n<20. Or l'hôtesse saisit 20. Donc la condition du IF n'est pas satisfaite, et les actions du ELSE sont exécutées. Le calcul utilise donc le tarif groupe à 5 € par personne multiplié par 20 personnes. L'affichage indique donc 100.

Question 5

Nous considérons le programme suivant. Quel affichage est produit par la saisie du nombre 4 ?

```php
<?php
$n = readline("Indiquez un nombre entre 1 et 4 : ");
switch($n){
  case 3 :
    echo "Vous avez indiqué un nombre impair.\n";
  case 1 :
    echo "Vous avez indiqué le plus petit entier positif non nul.\n";
    break;
  case 4 :
    echo "Vous avez indiqué un nombre plus grand que 3.\n";
  case 2 :
    echo "Vous avez indiqué un nombre pair.\n";
    break;
  default :
    echo "Vous avez saisi une donnée non reconnue.\n";
}
?>
```

- [x] Vous avez indiqué un nombre plus grand que 3
- [x] Vous avez indiqué un nombre pair
- [ ] Vous avez saisi une donnée non reconnue

À la fin du traitement des actions du “case 4”, le programme continue avec le traitement des actions de la case suivant, c’est-à-dire du “case 2”. À la fin de ce dernier, le programme rencontre un “break” et finit.