

Les structures de contrôle et d'exécution

Table des matières

I. Différents types de structures de contrôle	3
II. Exercice : Quiz	7
III. Exécution des structures de contrôle	7
A. Structure conditionnelle SI	7
B. Structure conditionnelle SELON.....	9
IV. Exercice : Quiz	11
V. Essentiel	12
VI. Auto-évaluation	12
A. Exercice	12
B. Test.....	13
Solutions des exercices	13

I. Différents types de structures de contrôle

Contexte

Afin de pouvoir résoudre une panoplie de problèmes à la fois, les mathématiciens arabes ont mis en place, au début du neuvième siècle, un système dénommé « *algorithmique* ». Avec ce système, il est possible d'élaborer une série d'opérations chronologiques pour parvenir à la résolution d'un problème (algorithme). Aujourd'hui, avec l'évolution de la technologie et de l'intelligence artificielle, les algorithmes peuvent être utilisés dans l'informatique, les mathématiques, la planification, les traitements de fichiers, la santé et plusieurs autres domaines. Cependant, avant toute exécution d'un quelconque algorithme, il est important de veiller à la saisie effective et correcte des différentes commandes. De ce fait, avant de procéder à une programmation, il serait bien de connaître les différentes structures de contrôle ainsi que les moyens d'exécutions afin de bénéficier de son efficacité.

Définition

Dans le domaine algorithmique, la structure de contrôle désigne la commande qui contrôle l'effectivité des algorithmes et leur chronologie d'exécution. En effet, il en existe de plusieurs types.

Les structures de contrôle séquentielles

Une structure de contrôle séquentielle est une commande qui contrôle l'ordre dans lequel les différentes instructions d'un algorithme ou d'un programme informatique sont exécutées.

Séquence : les commandes de programmation algorithmiques sont généralement exécutées d'une manière séquentielle, c'est-à-dire que la structure séquentielle traite le programme et ensuite incrémente le compteur ordinal avant de passer enfin au chargement la commande suivante. Il s'agit ici donc d'un facteur pertinent dans l'exécution d'un programme puisque c'est elle qui ordonne et même corrige les commandes ou les syntaxes mal écrites (les ponctuations et espaces surtout). Lorsqu'une erreur de syntaxe est repérée, le script ne s'exécute pas et une indication est donnée sur la ligne à laquelle l'erreur a été détectée.

Pour illustrer cette idée d'incrémentation, on pourrait imaginer un algorithme qui décrit une action à effectuer pour préparer un appareil à cake. On partira du principe suivant : Tant que la pâte n'est pas homogène, ajouter de la farine (20 g. par 20 g.) et du chocolat fondu (10 g. par 10 g.). La pâte sera homogène au bout de 10 répétitions du script. On demandera alors à l'algorithme d'afficher le dosage final des deux ingrédients.

Voici comment ceci se traduirait en Python :

```
1 homogene = 0
2 farine = 0
3 chocolat = 0
4 while homogene <10:
5     farine = farine + 20
6     chocolat = chocolat + 10
7     homogene = homogene +1
8 print(farine)
9 print(chocolat)
```

On voit ainsi que l'incrémentation de la valeur homogène est de 1 ($\text{homogene} = \text{homogene} + 1$) jusqu'à atteindre 10 ($\text{while homogene} < 10$). Une fois ce cycle terminé, l'exécution du script prend fin.

Arrêt du programme : lorsque les principes de base des algorithmes sont respectés, le programme établi doit s'arrêter après avoir exécuté la commande finale. Néanmoins, il existe plusieurs programmes aujourd'hui pouvant vous proposer des commandes raccourcies afin de stopper manuellement l'exécution du programme et ce, dans un langage approprié. Par ailleurs, cet arrêt manuel, d'après les principes du système d'exploitation, peut conduire à un arrêt ou une coupure définitive de l'exécution du programme afin de permettre au microprocesseur de stocker les données et de finaliser leur traitement.

Par exemple, un raccourci fréquent pour stopper l'exécution d'un script est d'entrer la combinaison des touches Ctrl + C. Il peut arriver que certains programmes nécessitent juste de taper la lettre « Q » pour « Quit », parfois encore, ce sera sur la touche « Échap » qu'il faudra appuyer. Il ne faut pas hésiter à consulter les documentations en ligne ou les forums pour connaître la bonne technique.

Les structures de contrôle itératives

Une structure de contrôle itérative est une séquence d'instructions qui est destinée à être répétée plusieurs fois.

- **Commande de blocs :**

Au fil du temps, les blocs d'instruction ont été classés en deux grandes catégories :

- Les blocs alternatifs : ils exécutent tout un bloc d'instructions après que ce dernier a rempli toutes les conditions requises.
- Les boucles : ils exécutent le bloc d'instructions d'une façon répétée. Ce type d'exécution résulte de la programmation planifiée.

- **Bloc d'instructions :**

Un bloc d'instruction regroupe plusieurs commandes de programmation algorithmique. En effet, il en existe de deux familles : la première est composée des commandes composées de deux expressions clés : l'expression débutante et l'expression finale. Cette dernière expression dépend du langage de programmation.

Algol 68, bash : en miroir (if.....fi, case...esac, etc)

Ada : end suivi de l'espace et de l'expression initiale (if...end if, when.....end when, etc)

Fortran 77 : end suivi de l'expression initiale (if...endif, so....endso, etc)

Turbo basic : chaque commande à son expression finale spécifique (for...next, while...wend, etc)

Exemple

Un algorithme de vérification du signe d'un nombre pourrait se rédiger ainsi :

```
Demander un nombre
SI nombre > 0, alors
Afficher "Ce nombre est positif"
SI nombre < 0, alors
Afficher "Ce nombre est négatif"
SINON
Afficher "Ce nombre est neutre"
FIN DE SI
```

Une fois traduit en Bash, par exemple, cet algorithme donnerait le résultat suivant :

```
1 read -p 'Entrez un nombre : ' n
2 if [ $n -gt 0 ]; then
3 echo -e "Ce nombre est positif."
4 elif [ $n -lt 0 ]; then
5 echo -e "Ce nombre est négatif."
6 else
7 echo -e "Ce nombre est neutre."
8 fi
```

La deuxième famille regroupe toutes les commandes qui opèrent préalablement sur une instruction atomique. C'est à dire qu'elle sera exécutée entièrement, sans que le processeur ne l'interrompe pour une autre action. Le côté linguistique intervient dans la définition d'un nouveau processus afin d'identifier les blocs susceptibles d'être utilisés.

Mots clés : begin....end, do...done

Ponctuation : \$ (...\$)

Blocs alternatifs : s'agit des commandes dont les structures de programmation réalisent un test logistique sur une condition et permettent d'effectuer un choix entre les blocs existants tout en se basant sur le résultat du test. Lorsque le test est positif, exécuter de façon continue et croissante des instructions. Le **test si** est la forme de vérification la plus simple mais il y a aussi la forme **test selon** qui consiste à spécialiser la commande « **si non si** », ce qui permet de faire le tri du bloc en cours d'exécution en fonction des valeurs des variables concernées.

Exemple

Test si

```
SI Test
  Instruction 1
```

```
FIN SI
Instruction 2
```

Or

```
SI Test
  Instruction 1
```

```
SINON
  Instruction 2
```

```
FIN SI
Instruction 3
```

Or

```
if ( test1 )
  Instruction1
```

```
else if ( test2 )
  Instruction2
```

```
Instruction3
```

Test selon

```
SI Test 1
  Instruction 1
```

```
SINONSI Test 2
  Instruction 2
```

```
FIN SI
Instruction 3
```

Or

```
BASIC: IF condition THEN instruction
```

```
C : if (condition) instruction
```

```
Python : if condition : instruction
```

Boucles : il s'agit d'une structure de contrôle qui favorise l'exécution d'un aspect du code à plusieurs reprises. Selon les règles normales, son exécution varie d'un cas à un autre (soit le nombre de fois, soit jusqu'à l'obtention d'une condition d'ouverture de la boucle). Ici, un minimum d'attention est demandé puisqu'il suffit d'une erreur de programmation et cela fausse toutes les commandes bouclées. Tout comme les blocs alternatifs, les langages proposent également plusieurs sortes de boucles :

- Boucle précondition : la boucle ne passe pas sans une vérification de la condition,
- Boucle post condition : la vérification passe après la boucle,
- Boucle condition d'arrêt : la vérification est faite au milieu de la boucle.

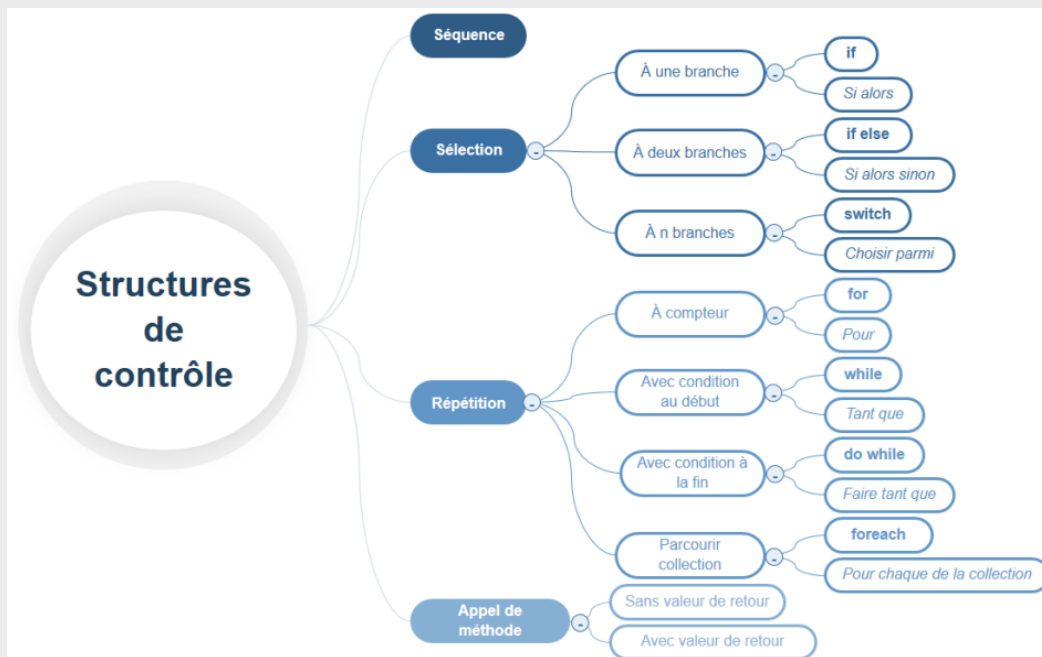
Exemple

Boucle précondition

```
SELON Variable 1
CAS Valeur 1 :
  Instruction 1
CAS Valeur 2 :
  Instruction 2
FIN SELON
Instruction 3
```

Boucle post condition

```
TANTQUE Test
  Instruction 1
FIN TANTQUE
Instruction 2
```



Exercice : Quiz

[solution n°1 p.15]

Question 1

L'attribution des étiquettes est assurée par trois familles d'instruction.

- Vrai
- Faux

Question 2

Les commandes à étiquettes sont celles qui assurent le traitement ordinal des instructions.

- Vrai
- Faux

Question 3

Le saut inconditionnel renvoie l'instruction vers l'étiquette.

- Vrai
- Faux

Question 4

Au niveau des blocs alternatifs, les instructions sont exécutées sous condition.

- Vrai
- Faux

Question 5

Boucle précondition : la boucle passe sans une vérification de la condition.

- Vrai
- Faux

III. Exécution des structures de contrôle**A. Structure conditionnelle SI****Structure conditionnelle**

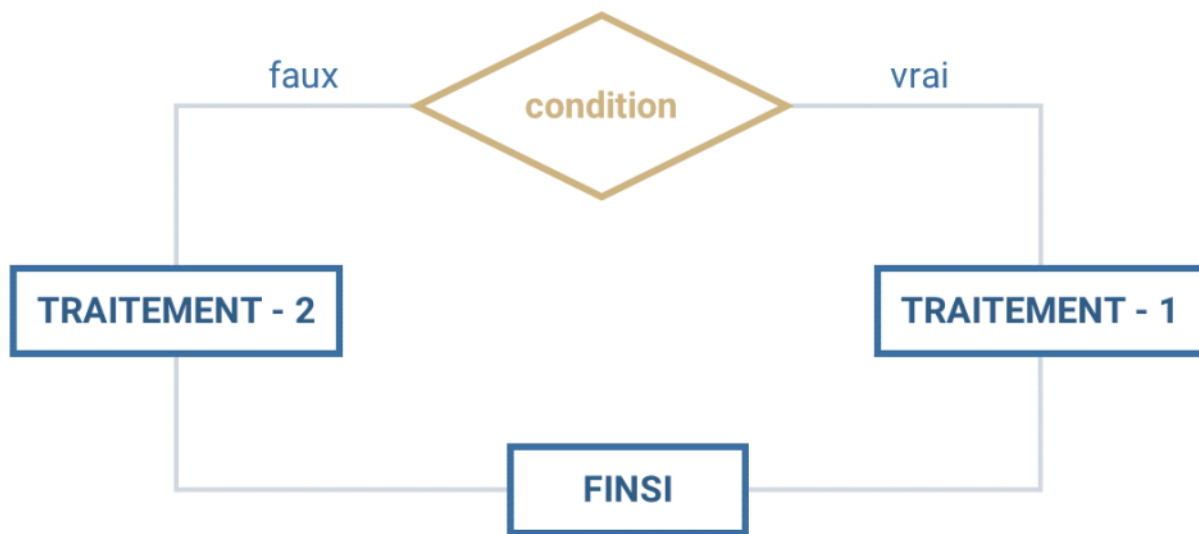
Commande :

```
SI <Condition> ALORS
```

```
<Suite d'action(s)-1>
```

```
[SINON <suite d'actions(s)-2>] FINSI
```

Schéma illustratif :



La <condition> est une variable dichotomique, c'est-à-dire qu'elle peut être vrai ou faux, selon les valeurs des paramètres qui la constituent.

Lorsque la condition est vérifiée, autrement dit, lorsque la valeur est vraie c'est la <suite d'actions-1> qui sera exécutée. Ensuite, le système passe à l'exécution juste après le FINSI.

Par ailleurs, lorsque la condition n'est pas vérifiée, autrement dit, lorsque la valeur de la condition est fausse, c'est la <suite d'actions-2> qui s'exécute, en cas où celle-ci existe (facultative). Dans le cas contraire, le système passe directement à l'instruction qui suit le FINSI.

La condition doit toujours être notée dans des parenthèses. Elle peut être une condition simple ou même très complexe c'est-à-dire composée d'autres sous conditions. Les guillemets sont eux aussi très importants et capitaux, surtout dans les cas où vous désirez exécuter plusieurs commandes à la fois. Il est également important de savoir que lorsqu'une seule commande est vérifiée, alors tous les autres peuvent s'en suivre.

Les structures conditionnelles if else peuvent subir une imbrication les unes dans les autres comme suit :

```
if (condition 1){
    traitement 1; }
else{
    if(condition 2)
        traitement 2;
    else
        traitement 3;
}
```

Il est possible d'imbriquer autant de « if else » que l'on veut. De plus, les accolades du premier else sont obligatoires dans ce cas, car le bloc contient plusieurs instructions.

Opérateur ternaire

Il s'agit d'une écriture abrégée de la structure if else. Il est beaucoup plus utilisé lorsqu'on souhaite retourner un résultat en fonction de l'état d'une condition. La structure est relevée comme ceci :

```
(condition)?(traitement 1):(traitement 2);
```

Explication : si la condition est vraie, alors le premier bloc délimité par les parenthèses (traitement 1) est exécuté. Dans le cas contraire, c'est le deuxième bloc qui est exécuté (traitement 2).

L'opérateur ternaire appliqué dans la situation précédente produit le code ci-après :

```
a=10;
b=(a%2==0)?("Nombre pair"):(("Nombre impair"));
```

Ici la variable b peut recevoir soit « *Nombre pair* » ou « *Nombre impair* » selon la condition (est-ce que la variable a est divisible par 2).

Arrêt prématuré

Il peut arriver de souhaiter arrêter une boucle avant de parvenir à la valeur finale du compteur. On parle donc d'un arrêt prématuré (ou improvisé) qui est exprimé avec le mot clé break. Cet arrêt s'applique généralement aux boucles for et while.

Exemple

```
str="";
i=1;
while(i<5){
    if(i==3){
        break;
    }
    str+=i;
    i++;
}
```

À la fin de la boucle, le str vaudra 12. En effet, si la boucle se terminait normalement alors la variable str vaudrait 1 234. Mais comme on a arrêté prématurément la boucle avec l'instruction break une fois, le compteur i arrive à 3. Alors la boucle s'arrête aussitôt sans exécuter le reste du code après rupture.

B. Structure conditionnelle SELON

Commande :

```
SELON (sélecteur) FAIRE
Cas <liste de valeurs-1> : <suite d'action (s)-1>
[Cas <liste de valeur-2> : <suite d'action (s)-2>
..... ] [SINON : <suite d'action (s)-n> ]
FINSELON
```

Le sélecteur de commande peut être une variable unique ou une expression arithmétique ou logique, ceci dépend du contexte. La structure SELON examine le « *sélecteur de commande* » ensuite passe à la comparaison de ce dernier respectivement avec les valeurs dans les listes. Dans les cas où on remarque une égalité avec une valeur, les extensions qui correspondent sont devant cette valeur et sont directement exécutées.

Devant « *Cas* », il peut y avoir une seule variable, une suite de variables séparées par des virgules et/ou un intervalle de variables. Après avoir traité la suite d'actions correspondante, l'exécution se déclenche aussitôt après le FINSELON.

Remarque

Le sélecteur doit avoir le même type que les variables.
Ces valeurs ne doivent être ni réelles ni sous forme de chaîne de caractères.

Structures répétitives

En algorithmique, les difficultés habituelles ne portent pas uniquement sur des séquences d'opérations, sous ou sans préalable. Il arrive des situations où l'on est obligé d'exécuter un traitement à plusieurs reprises, c'est-à-dire d'une façon répétée. En effet, pour relever les notes d'un étudiant et calculer sa moyenne, on est amené à saisir N variables avec N désignant le nombre de notes relevées. Ensuite, il faut faire la somme et la diviser par N. Ce problème est résolu à l'aide des structures répétitives.

La boucle POUR

Elle exprime la répétition d'un traitement un nombre de fois.

Commande :

POUR Vc DE Vi A Vf [PAS Vp] FAIRE. <Traitement>
FINFAIRE

- Avec Vc, une variable entière, qui compte le nombre de répétitions du <Traitement>,
- Vi la valeur initiale à laquelle Vc est initialisée,
- Vf la valeur finale à laquelle se termine Vc,
- Vp la valeur du pas, c'est la valeur qu'on rajoute à Vc à chaque fin de traitement.

Remarque

La boucle POUR est utilisée lorsque l'on connaît le nombre de répétitions du <Traitement> d'avance.
La valeur du pas peut être positive ou négative et par conséquent, il faut, au départ de la boucle, que $V_i \leq V_f$ ou $V_i \geq V_f$ selon la positivité ou la négativité de cette valeur.
Valeur du pas est égale à 1 par défaut.
Exécution :
1. Initialiser Vc par la valeur de Vi (comme si on avait $V_c = V_i$).
2. Vérifier si Vi dépasse (\pm) Vf (du côté supérieur ou inférieur, selon la positivité ou la négativité du pas). Si oui, alors la commande s'arrête et l'exécution continue après le FINFAIRE.
Sinon, Exécuter : <Traitement>
Incrémenter ou décrémenter le Vc par la valeur du pas.

La boucle Répéter ... Jusqu'à

Commande :

Répéter <Traitement>
Jusqu'à (condition d'arrêt)

Exécution :

1. Exécuter du <Traitement>,
2. Tester la valeur de la <condition d'arrêt>,
3. Si elle est vérifiée Alors la boucle s'arrête.

Sinon Retour à l'étape 1.

La boucle TANT QUE

Commande :

TANT QUE (condition d'exécution)

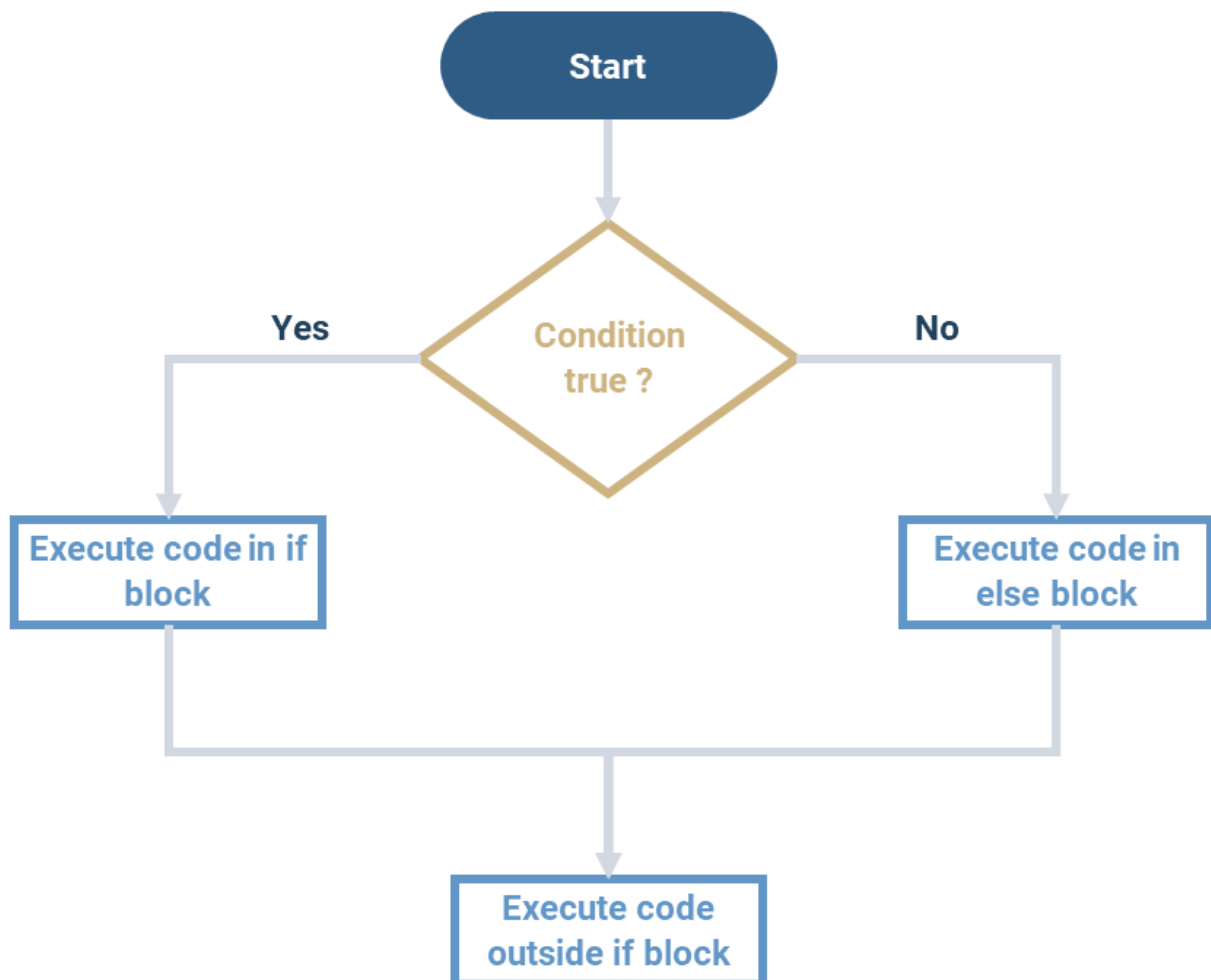
FAIRE <Traitement>

FIN FAIRE

Exécution :

1. Tester la valeur de la <condition d'exécution>,
2. Si elle est vérifiée Alors,
3. Exécution du <Traitement>.

Sinon, Retour à l'étape 1, Arrêt de la boucle.

**Méthode** Écrire un algorithme simple**Exercice : Quiz**

[solution n°2 p.15]

Question 1

La condition est une variable unique.

- Vrai
- Faux

Question 2

On dit que la valeur de la condition est fausse lorsque la condition n'est pas vérifiée.

- Vrai
- Faux

Question 3

Les accolades du premier else dans la structure if else sont nécessaires.

- Vrai
- Faux

Question 4

En algorithmie, les difficultés quotidiennes portent uniquement sur des séquences d'actions avec ou sans condition.

- Vrai
- Faux

Question 5

Quelle est la structure conditionnelle qui examine le sélecteur de commande ?

- SI
- SELON

V. Essentiel

Les trois principales structures de programmation algorithmique sont la structure séquentielle, la structure de sélection et la structure d'itération.

La structure séquentielle se charge d'exécuter toutes les commandes tout en respectant l'ordre dans lequel elles ont été écrites.

La structure de sélection est chargée d'exécuter une commande après avoir vérifié si un certain nombre de conditions d'éligibilité sont remplies.

La structure itérative est chargée de la réalisation d'un automate fini qui n'est rien d'autre qu'un modèle arithmétique utilisé dans la programmation et qui part de la conception des programmes et des différentes étapes d'analyse séquentielle pour les applications et les canaux de communication tout en passant par le langage et le contrôle de processus.

VI. Auto-évaluation

A. Exercice

Un magasin de photocopie facture 2 € les dix premières photocopies, 1,50 € les vingt suivantes et 1 € au-delà.

Question 1

[solution n°3 p.16]

Écrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées puis affiche le montant correspondant. Rédigez un script en Python pour l'exécuter et vérifier qu'il est fonctionnel.

Un camping souhaite savoir la saison en fonction du numéro du mois, cela dans le but de facturer plus facilement car le prix dépend de la saison.

Question 2

[solution n°4 p.17]

Écrire un algorithme permettant d'afficher la saison en introduisant le numéro du mois. Rédigez un script en Python pour l'exécuter et vérifier qu'il est fonctionnel.

B. Test**Exercice 1 : Quiz**

[solution n°5 p.18]

Question 1

Les trois principales structures de programmation sont Java, Python, Visual Basic.

- Vrai
- Faux

Question 2

Quelle est la structure de contrôle qui se charge de l'exécution d'une commande après vérification de la condition par l'ordinateur ?

- Boucle
- Séquence
- Sélection
- Fonctionnel

Question 3

Quelle est la structure de contrôle qui se charge de s'assurer que chaque commande soit exécutée dans l'ordre ?

- Séquence
- Boucle
- Sélection

Question 4

La structure de contrôle séquentielle est celle qui exploite la commande do/while.

- Vrai
- Faux

Question 5


La boucle est chargée de répéter un code à plusieurs reprises.

- Vrai
- Faux

Solutions des exercices


Exercice p. 7 Solution n°1**Question 1**

L'attribution des étiquettes est assurée par trois familles d'instruction.

- Vrai
- Faux
-  On utilise deux familles de commandes pour l'adressage des étiquettes. Il s'agit de sauts sans condition et conditionnels.


Question 2

Les commandes à étiquettes sont celles qui assurent le traitement ordinal des instructions.

- Vrai
- Faux
-  Une commande labellisée est celle qui possède un identifiant (habituellement un numéro) associé à la commande source. On l'utilise souvent comme cible pour la structure de contrôle qui manquait auparavant au programme.


Question 3

Le saut inconditionnel renvoie l'instruction vers l'étiquette.

- Vrai
- Faux
-  Communément appelé *goto*, un saut sans condition aide à renvoyer l'exécution à la balise.


Question 4

Au niveau des blocs alternatifs, les instructions sont exécutées sous condition.

- Vrai
- Faux
-  Il s'agit d'ordres pour lesquels les structures de programmation effectuent un essai logique à une condition et vous permettent de faire un choix parmi les blocs existants sur la base du résultat du test.

Question 5

Boucle précondition : la boucle passe sans une vérification de la condition.

- Vrai
- Faux
-  La boucle précondition signifie que la boucle ne passe pas sans un contrôle d'état.


Exercice p. 11 Solution n°2

Question 1

La condition est une variable unique.

Vrai

Faux


 Il s'agit d'une variable dichotomique qui signifie qu'elle peut être vraie ou fausse, en fonction des valeurs des paramètres qui la composent.

Question 2

On dit que la valeur de la condition est fausse lorsque la condition n'est pas vérifiée.

Vrai

Faux


 Lorsque la condition n'est pas cochée, cela sous-entend directement que la valeur de la condition est fausse.

Question 3

Les accolades du premier else dans la structure if else sont nécessaires.

Vrai

Faux


 Les crochets bouclés du premier else dans la structure if else sont obligatoires, étant donné que le bloc contient plusieurs instructions.

Question 4

En algorithmie, les difficultés quotidiennes portent uniquement sur des séquences d'actions avec ou sans condition.

Vrai

Faux


 Dans les algorithmes, les difficultés journalières ne sont pas seulement des enchaînements d'actions, sous ou sans condition. Il y a des situations où vous devez faire le traitement encore et encore, c'est-à-dire à plusieurs reprises.

Question 5

Quelle est la structure conditionnelle qui examine le sélecteur de commande ?

SI

SELON

 La structure SELON examine le « sélecteur de contrôle » et se déplace ensuite pour comparer ce dernier avec les valeurs dans les listes respectivement.

Algorithme Facture :

Demander d'entrer un nombre de photocopies pour définir la variable « *nombre* ».

Établir une suite de conditions

Condition 1 :

Si la variable est inférieure ou égale à 10 :

Alors multiplier le nombre par 2

Condition 2 :

Si la variable est inférieure ou égale à 30 :

Alors créer la variable « *a* » égale à « *nombre* » -10

Multiplier « *a* » par 1,5 et ajouter 20

Si la variable est supérieure à 30 :

Alors créer la variable « *b* » égale à « *nombre* » -30

Ajouter 50 à « *b* »

Script :

```

1 nombre = int(input("Entrez le nombre de photocopies : "))
2 if nombre <= 10:
3     print("Le montant est de : ", nombre*2, " €")
4 elif nombre <=30:
5     a = (nombre-10)
6     print("Le montant est de : ", a*1.5+20, " €")
7 elif nombre >30:
8     b = (nombre-30)
9     print("Le montant est de ", b+50, " €")

```

p. 13 Solution n°4

Créer une boucle

Tant que la boucle est valide :

Demander d'entrer un numéro de mois de 1 à 12

Si le numéro est 3, 4 ou 5

Alors afficher : la saison est printemps

Si le numéro est 6, 7 ou 8

Alors afficher : la saison est été

Si le numéro est 9, 10 ou 11

Alors afficher : la saison est automne

Si le numéro est 12, 1 ou 2

Alors afficher : la saison est hiver

Sinon

Afficher : ce numéro ne correspond à aucun mois

La boucle est fausse

Script :

```


1 boucle = True
2 while boucle == True:
3     m = int(input("Entrez un numéro de mois de 1 à 12 : "))
4     if m == 3 or m == 4 or m == 5:
5         print("La saison est : PRINTEMPS")
6     elif m == 6 or m == 7 or m == 8:
7         print("La saison est : ETE")
8     elif m == 9 or m == 10 or m == 11:
9         print("La saison est : AUTOMNE")
10    elif m == 12 or m == 1 or m == 2:
11        print("La saison est : HIVER")
12    else:
13        print("Ce numéro ne correspond à aucun mois")
14 boucle = False

```

Exercice p. 13 Solution n°5


Question 1

Les trois principales structures de programmation sont Java, Python, Visual Basic.

- Vrai
- Faux
-  Les trois principales structures de programmation sont la séquence, la sélection et l’itération.


Question 2

Quelle est la structure de contrôle qui se charge de l’exécution d’une commande après vérification de la condition par l’ordinateur ?

- Boucle
- Séquence
- Sélection
- Fonctionnel
-  Il s'agit de la structure de sélection, car elle est chargée de mener un test logique selon une condition. Il vous permet de choisir entre les blocs existants selon le résultat du test.

Question 3

Quelle est la structure de contrôle qui se charge de s’assurer que chaque commande soit exécutée dans l’ordre ?

- Séquence
- Boucle
- Sélection
-  La structure séquentielle traite le programme puis incrémente le compteur ordinal avant de finalement se déplacer pour charger la prochaine commande.

Question 4

La structure de contrôle séquentielle est celle qui exploite la commande do/while.

- Vrai
- Faux
- Seule la structure de boucle utilise le contrôle de faire/tout.

Question 5

La boucle est chargée de répéter un code à plusieurs reprises.

- Vrai
- Faux
- C'est une structure de contrôle qui favorise l'exécution d'un aspect du code de manière répétitive.