Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Transposition didactique des situations problèmes d'algorithmique en première année scientifique en RD Congo.

Didactic transposition of algorithmic problem situations in the form 1 of science in DR Congo.

Bally KASAMBI BALIMWENGU[a],
Paulin BAPOLISI BAHUGA[b] ,
Deogratias MBILIZI MWISIMBWA[c],
Jean Moise MATESO MOUKE[d];
Pascal AKILIMALI BAMALEMBUKO[e],
Christian ZIGASHANE KAMBAZA[a],
Alain OMBENI LANDO[a],
Cirhuza BAKENGE BUGOYE[a],
Jean Claude MUKANGWA RAMAZANI[a],

[a] Etudiant-Chercheur en Didactique de l'Informatique à l'École Doctorale de l'Institut Supérieur Pédagogique de Bukavu (République Démocratique du Congo).

[b] Enseignant-Chercheur à l'École Doctorale de l'Institut Supérieur Pédagogique de Bukavu, (République Démocratique du Congo).

[c] Enseignant-Chercheur à l'Institut Supérieur Pédagogique de Bukavu, Unité de Recherche en Technologie de l'Information et de la Communication (République Démocratique du Congo).

[d] Enseignant-Chercheur à l'Institut Supérieur Pédagogique de Kamituga, (République Démocratique du Congo).

[e] Doctorant à l'École Doctorale de l'Institut Supérieur Pédagogique de Bukavu, Unité de Recherche en Technologie de l'Information et de la Communication (République Démocratique du Congo).

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Résumé
Cet article examine les changements apportés au programme d'enseignement scientifique en République Démocratique du Congo depuis 2019. La fusion des sections mathématiques-physiques et biologie-chimie en une seule section scientifique a marqué le début de ces réformes, suivie par l'introduction d'un nouveau programme en 2020-2021 qui intègre l'algorithmique et le codage. Cependant, des lacunes ont été observées dans la définition et l'élaboration de ces situations. Cette étude empirique vise à analyser le programme éducatif et son guide pédagogique afin de proposer des savoirs à enseigner adaptés au contexte congolais. La méthodologie s'appuie sur une analyse approfondie du programme et de ses directives méthodologiques, en tenant compte des réalités et besoins du système éducatif congolais. L'analyse révèle que certaines situations pédagogiques souffrent d'un manque de précision et d'adaptation au contexte congolais. La reformulation proposée vise à pallier ces lacunes en redéfinissant les situations pédagogiques de manière plus claire, concrète et pertinente.
Cette reformulation contribue à l'amélioration de l'enseignement des Technologies de l'Information et de la communication en RDC. Elle permettra une meilleure appropriation des concepts scientifiques par les élèves et une meilleure préparation aux exigences du monde actuel.

Mots clés : Situation pédagogique ; Transposition didactique ; Algorithme ; Algorithmique ; Scientifique.

Abstract

This article examines changes to the science curriculum in the Democratic Republic of Congo since 2019. The merger of the mathematics-physics and biology-chemistry sections into a single science section marked the start of these reforms, followed by the introduction of a new curriculum in 2020-2021 that incorporates algorithmics and coding. However, shortcomings have been observed in the definition and development of these situations.
The aim of this empirical study is to analyse the educational program and its pedagogical guide in order to propose teaching knowledge adapted to the Congolese context. The methodology is based on an in-depth analysis of the program and its methodological guidelines, taking into account the realities and needs of the Congolese education system. The analysis reveals that some teaching situations suffer from a lack of precision and adaptation to the Congolese context. The proposed reformulation aims to remedy these shortcomings by redefining pedagogical situations in a clearer, more concrete and relevant way.

This reformulation contributes to improving the teaching of Information and Communication Technologies in the DRC. It will enable students to better grasp scientific concepts and be better prepared for the demands of today's world.

Keywords: Educational situation, Didactic transposition, Algorithm, Algorithmic, Scientific.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Introduction

Dans le processus enseignement apprentissage, il existe des nombreux défis auquel l'enseignant est confronté. La transposition didactique en fait également partis autant pour des systèmes éducatifs dont la gestion s'avère évoluée que pour celle non évoluée. En République Démocratique du Congo nous sommes à l'aube d'une reforme éducative du programme éducatif national, validée et généralisée en 7e année par l'arrêté ministériel N° MINEPSP/CABMIN/1973/2018 du 26/06/2018 portant validation et généralisation des programmes éducatifs (PE) du Domaine d'Apprentissage des Sciences (DAS) pour la classe de 7e année de l'éducation de base (EB), en 8e année EB par l'arrêté ministériel N° MINEPSP/CABMIN/599/2019 du 03/07/2019 portant validation et généralisation des programmes éducatifs du DAS pour la classe de 8e année de l'EB et dans les classes de la section scientifique depuis 2019 par l'arrêté N° MINEPSP/CABMIN 600/2019 du 03/07/2019 portant mise en place de la section unique des humanités scientifiques. Ce nouveau programme introduit les notions d'algorithmiques à partir de la terminale de l'éducation de base jusqu'à la quatrième année de la section scientifique. Elles sont enseignées comme deuxième partie du cours de Technologies de l'Information et de la Communication (TIC). Ceci est une avancée majeure dans l'enseignement congolais puisque depuis le premier programme national d'informatique de 2006, qui a subi des reformes au fil des années, aucun programme éducatif officiel n'avait prescrit directement l'enseignement de l'algorithmique et/ou de la programmation (B. K. BALIMWENGU et al., 2023). Depuis 2019, cette introduction d'éléments d'algorithmiques n'a pas été appuyée des outils pédagogiques pouvant permettre aux enseignants de mieux adapter les matières par rapport au niveau des apprenants.

En effet, les programmes éducatifs existent tels que soulignés ci-dessous ; et sont désormais centré sur la mise en activité des élèves par le traitement des situations qui ont un sens pour eux et qui font appel à des savoirs essentiels pour aboutir au développement des compétences. Ainsi, l'apprenant départ son entrée en première année scientifique, selon le Programme National(EPSP2, 2016), doit posséder des bases en algorithmique [Concept de base de l'algorithmique : Algorithme, programme et langage, Instructions Variables et constantes / Instructions de base : Entrée (lecture), Sortie (écriture), branchement]; lui permettant de résoudre certains problèmes élémentaires ; et, à la sortie de cette classe, il doit traiter avec succès et de façon acceptable les situations qui relèvent de l'algorithmique et codage, il doit également utiliser les méthodes et les outils de programmation ainsi que ses techniques pour

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

résoudre les problèmes de la vie quotidienne (EPSP3, 2019). Ceci dit, l'enseignant des TIC doit puiser les matières apprises en chimie, physique et/ou mathématiques … pour montrer aux apprenants comment élaborer des algorithmes et les implémenter. C'est ainsi que le ministère a officiellement prescrit les langages de programmation Python et C comme langage d'implémentation des algorithmes. Toutefois, aucun environnement [Algobox, LARP, Scratch, …] n'a été désigné pour l'élaboration des algorithmes.

L'apprentissage de l'algorithmique en première année scientifique est un défi majeur autant pour l'apprenant que pour l'enseignant. Les programmes éducatifs étant souvent trop théoriques et abstraits l'apprentissage parait difficile et peu motivant pour les apprenants (AISSAOUI, 2011). Ainsi donc, la transposition didactique trouve sa raison d'être pour adapter les connaissances scientifiques aux besoins et aux capacités des apprenants (Balacheff, 1993; Conne, 1992; Del Notaro, 2013; Meyer & Modeste, 2022; Ravel, 2003).

Eu égard à ce qui précède, la question suivante guidera notre étude : Comment transposer les situations problèmes d'enseignement de l'algorithmique et du codage en première année scientifique?

Nous pensons que l'adaptation des situations problèmes d'enseignement de l'algorithmique et du codage au contexte socioculturel congolais et aux besoins des élèves améliorerai significativement leur compréhension des concepts scientifiques et renforcera leurs compétences en résolution de problèmes en première année scientifique en République Démocratique du Congo.

Dans le cadre de cette étude, nous explorons la possibilité de transposer didactiquement des situations problèmes d'algorithmique en première année scientifique en RDC. Nous utilisons l'outil Algobox et l'implémentation a été effectuée en langages C et Python conformément aux directives du ministère en charge de l'éducation nationale.

Notre article s'organise en quatre points : l'enseignement de l'algorithmique à l'école secondaire en RD Congo, l'analyse de la transposition didactique, les outils pédagogiques utilisés et enfin l'analyse des quelques situations d'enseignement de la première année scientifique.

Nous nous s'appuyons sur un ensemble des documents ministériels et étatiques notamment la loi cadre de l'enseignement national, les programmes éducatifs de l'informatique, les guides en appui aux programmes éducatifs et certains manuels d'enseignement.

1. Enseignement de l'algorithmique à l'école secondaire en RD Congo

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Dans l'enseignement secondaire, jusqu'à 2003, il n'existait pas de programme éducatif officiel d'informatique. Toutefois, certaines écoles avaient déjà inséré l'informatique dans leurs programmes des cours, (EPST7, 2007, p. 3). Officiellement, l'informatique a été intégrée dans le programme national de l'enseignement secondaire en 2009 par la circulaire n° 436/PR du 18 septembre 2009 du Président de la République. Dans ce programme éducatif, on ne trouve que très peu des notions d'algorithmiques et programmation dans les classes de 3ème et 4ème (toutes filières confondues), avec quelques particularités de nombre d'heures (2 heures par semaine) pour les classes de Commerciale et Informatique, Commerciale et Administrative et Secrétariat et Informatique (EPST7, 2007, p. 27).

Ce programme éducatif de 2007 qui est toujours d'application dans les autres options (Pédagogie, Sociale, Agronomie, …), on y apprend la programmation en langage Basic sauf pour la Commerciale & Gestion, Secrétariat et Informatique et Scientifique. Le langage BASIC est largement dépassé par le temps, ne rencontre plus les objectifs du siècle présent vu les nouvelles technologies auxquelles le monde scientifique fait face.

Ainsi, en 2014, des reformes ont eu lieu dans les sections Secrétariat & Informatique (SI), et ont abouti à la mise au point d'un nouveau programme éducatif en SI ; dans les options : Commerciale et Administrative (CA) et Commerciale et Informatique (CI) ; ces réformes ont abouti à la fusion de ces 2 options pour donner naissance à la section Commerciale et Gestion (CG) avec un nouveau programme éducatif, qui, du début à la fin (de la 1ère jusqu'en 4e) de son cursus, l'apprenant ne fait que manipuler l'outil à des fins bureautiques appliquées à la comptabilité , à la finance, à la correspondance, … pour ne citer que cela.

Les objectifs poursuivis par le programme éducatif (CG) ne permettent pas à l'apprenant de développer une pensée algorithmique non plus de la programmation. On peut retenir comme objectifs du programme de 1ère année [utiliser l'outil informatique pour des travaux élémentaires],(EPST7, 2007, p. 25); 2e année [Utiliser l'outil informatique pour saisir et élaborer les documents comptables et autres], (EPSPT7, 2014, p. 51) ; 3e [utiliser l'outil informatique pour saisir les documents et résoudre les problèmes comptables et financiers complexes], (EPSPT7, 2014, p. 75) et en 4e [utiliser l'outil informatique pour saisir les documents et résoudre les problèmes comptables et financiers], (EPSPT7, 2014, p. 109).Dans la même logique que le programme éducatif précédent (celui de CG), le programme éducatif de Secrétariat et Informatique est une informatique appliquée à la gestion des activités du bureau.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Depuis septembre 2016,(EPST4, 2021), d'autres réformes curriculaires ont eu lieu, cette fois en section scientifique (résultat issu de la fusion des options Mathématique-Physique et Biochimie, voir aussi (l'ARRETE N° MINEPSP/CABMIN 600/2019 du 03/07/2019), l'informatique s'appelle désormais TIC [comme pour les programmes éducatifs de 7ème et 8ème], une discipline du sous-domaine Sciences Physiques et TIC du domaine des Sciences (EPST4, 2021); et, apporte comme nouveauté l'introduction de l'algorithmique et codage (programmation) à côté des notions du tableur Excel sans spécifier les environnements d'élaboration des algorithmes, néanmoins, pour le codage (programmation) il est recommandé de les implémenter en langages Python comme en France (Meyer & Modeste, 2022, p. 114) et/ou en langage C (voir PE du DAS : Sous des sciences physiques et TIC en Section scientifique) avec l'éditeur des codes Notepad. Ce nouveau programme éducatif est appuyé d'un guide qui précise certaines notions ; en d'autres mots, il constitue un supplément au programme, (SERNAFOR1, 2018, p. 39).

Ainsi, chaque guide en appui au programme contient le Code et titre, les savoirs essentiels, les prérequis, les précisions sur les contenus ainsi que les suggestions pédagogiques ou didactiques. Il sied également de préciser qu'aucune option qualifiée de spécialité d'informatique pure ne prépare directement les apprenants à embrasser les sciences informatiques au niveau supérieur. L'option scientifique préparerait valablement les apprenants à embrasser les études informatiques au niveau supérieur mais les notions d'algorithmique et codage sont repris au curricula comme des sous points à enseigner dans le sous-domaine Technologie de l'Information et de la Communication et son volume horaire hebdomadaire est d'une heure (50 minutes).

Dans cette étude nous nous intéressons sur la transposition didactique des notions d'algorithmique et codages prévus par le programme éducatif en vigueur en République Démocratique du Congo.

2. Analyse de la transposition didactique

Dans l'enseignement de l'informatique il se pose un problème de la transposition didactique et de la transposition informatique. Face aux contraintes de la transposition didactique s'ajoutent celles de modélisation et d'implémentation informatiques: contraintes de la modélisation computable, contraintes logicielles et matérielles des supports informatiques de réalisation (Balacheff, 1993). Face à ces difficultés, l'enseignant doit inventer des stratégies pour amener les apprenants à s'approprier des savoirs complexes sans disposition informatique. Des

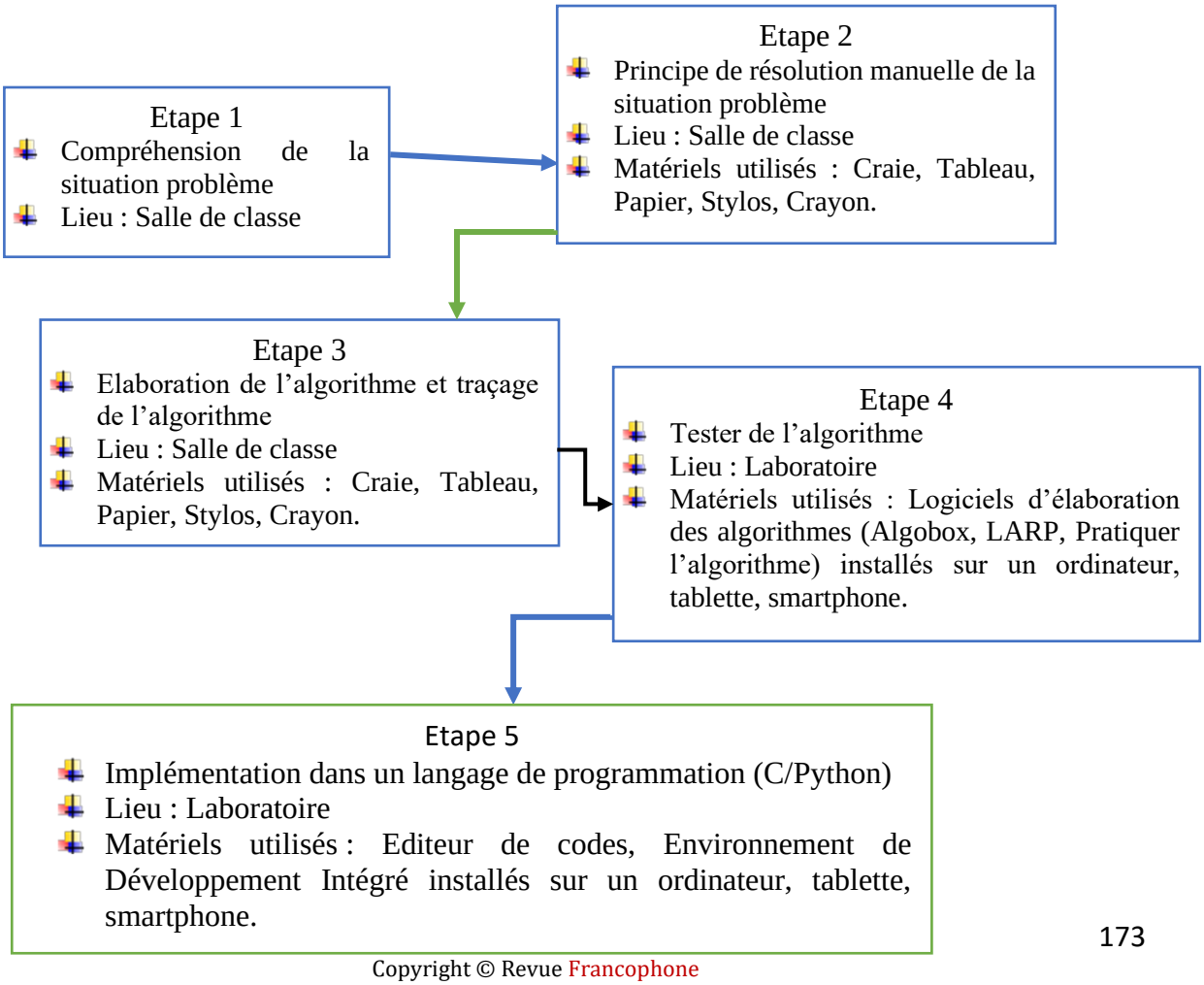
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

nombreux défis se posent dans l'enseignement de l'informatique, l'appui technique est à l'enseignement est quasi inexistant, les programmes existent mais aucun accompagnement en termes des manuels, matériels didactiques, référentiels. Même en milieu urbain, on retrouve parfois des enseignements d'Informatique où quatre ou cinq élèves sont regroupés autour d'un ordinateur (B. K. BALIMWENGU et al., 2023).

Aucune définition n'est proposée par proposée pour les concepts algorithmique, algorithme ou programme informatique dans le nouveau programme de la première année scientifique. Néanmoins, l'accent est mis sur la résolution des problèmes quotidiens auxquels l'apprenant peut être confronté, il s'agit donc de mettre en exergue les compétences en faisant recours aux nouvelles technologies de l'information et de la communication.

Pour cela, nous proposons de suivre les étapes suivantes pour l'enseignement des notions d'algorithmique en première année scientifique.

Figure 1: Modèle d'enseignement



Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Source : Notre propre confection

Cette figure [1] a été conçue par rapport aux directives méthodologiques présentées dans le programme éducatif et guide de première année scientifique.

3. Outils pédagogiques utilises

3.1. ALGOBOX

Est un environnement d'élaboration et exécution des algorithmes libre pour élèves débutants. Il a été développé en 2009 par Pascal Brachet, professeur de mathématiques au lycée Bernard Palissy (Briant & Bronner, 2015; Meyer & Modeste, 2022). Ses instructions sont écrites en français et construites pas à pas de façon hiérarchique et structurée.

Certaines recherches en ont fait usage, notamment (Meyer & Modeste, 2022; Ovono et al., 2014; Briant & Bronner, 2015).

Dans la partie [4] précédente de notre, nous utilisons ce logiciel dans la transposition informatique de quelques algorithmes conformes au programme en vigueur en RD Congo.

3.2. Les langages C et Python

Conformément aux directives du ministère nous utilisons les langages de programmations C et Python dans la transposition informatique des situations problèmes d'enseignement en première année scientifique. Le langage **C** est un langage de programmation orienté objet de plus bas niveau, rapide et puissant pour le développement des logiciels de hautes performances (Delannoy, 2016). **Python** est un langage de programmation interprété, de haut niveau et généraliste. Il est intuitif et clair pour les débutants. C'est un langage populaire pour une variété de tâches, notamment le développement d'applications Desktop, Web, mobile, la science des données et l'intelligence artificielle (Chandra & Varanasi, 2015; Eidelman, 2020). Il est recommandé en France aussi dans les lycées et collèges pour l'enseignement de l'algorithmique (Meyer & Modeste, 2022).

Par ailleurs en termes d'éditeurs de codes, nous avions utilisé Spyder pour des situations problèmes implémentées en langage Python et DevC++ en langage C.

4. Analyse des quelques situations d'enseignement de la première année scientifique

4.1. Les structures conditionnelles [MTIC3.8]

Pour l'enseignement des notions des structures conditionnelles, la situation problème est présentée :

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

« Lors de l'étude du milieu, les élèves de la 1ière année des humanités scientifiques constatent que les objets qui les entourent présentent des différences remarquables entre elles au niveau de leur état, caractéristiques et propriétés. Leur enseignant des sciences physiques profitant de l'occasion, sollicite auprès de son collègue des TIC de lui proposer une solution informatique permettant de calculer la masse moléculaire d'une substance connaissant sa formule chimique. L'enseignant des TIC demande aux élèves d'écrire un algorithme y relatif partant du tableau périodique des éléments. (EPST, 2021) »

Il est demandé de définir les concepts : test, structure, condition, booléen, structure alternative, type de test et élément de test. Et proposer des algorithmes incluant la condition « faire tant que » (EPSP3, 2019, p. 30).

En programmation les structures conditionnelles ou alternatives sont des instructions qui permettent de choisir entre deux ou plusieurs actions en fonction de la valeur d'une condition (Tchounikine, 2017).

Il existe deux types de structures conditionnelles ou alternatives :

Les instructions if

Ces instructions permettent de choisir entre deux actions, en fonction de la valeur d'une condition. Elles peuvent prendre 3 formes distinctes :

1. La première forme : **IF**

Syntaxe :

```
SI (condition) ALORS
        DEBUT_SI
                "Instructions"
        FIN_SI
```

2. La deuxième forme : **IF ELSE**

Syntaxe :

```
SI (conditions) ALORS
        DEBUT_SI
                "Instructions"
        FIN_SI
        SINON
        DEBUT_SINON
        "Instructions"
        FIN_SINON
```

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

3. La troisième forme

Syntaxe :

```
SI (conditions) ALORS
    DEBUT_SI
      "Instructions"
    FIN_SI
    SINON
     DEBUT_SINON
       SI (conditions) ALORS
        DEBUT_SI
                 " Instructions "
        FIN_SI
       SINON
        DEBUT_SINON
          " Instructions "
        FIN_SINON
      FIN_SINON
```

Les instructions switch (cas)

Ces instructions permettent de choisir entre plusieurs actions, en fonction de la valeur d'une variable.

Syntaxe :

```
switch (Conditions var):
 case (valeur de la var):
            "Instructions."
 case (valeur de la var):
            "Instructions."
 case (valeur de la var):
            "Instructions."
 default (valeur de la var):
            "default Instructions"
```

La situation choisie pour cette matrice [MTICI3.8] est mal définie, le calcul de la masse moléculaire d'une substance ne pas être calculé en utilisant les structures conditionnelles ou

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

alternatives. Cette situation est adaptée plutôt à l'enseignement de la matrice MTIC 2.10 de la 8e année éducation de base puisqu'il fait référence à un programme séquentiel (linéaire)(EPSP2, 2016). La situation similaire donnée n'est pas également adaptée, on y trouve aucune action qui peut amener l'apprenant à comprendre le fonctionnement des structures conditionnelles.

Notre contribution est élaborée en utilisant le logiciel Algobox, implémenté dans le langage de programmation C en utilisant l'IDE Dev-C++ suivant les étapes décrites à la figure 1.

Situation problème : Déterminer le plus grand nombre entre 2 nombres.

On peut le résoudre en utilisant la première forme de l'instruction IF

Etape 1 : Compréhension de la situation problème

Un nombre **x** est supérieur à la valeur d'un nombre **y** lorsque sa valeur est supérieure à l'autre ou vice versa.

Etape 2 : Principe de résolution manuelle

Pour cette étape l'enseignant explique d'abord le principe de fonctionnement de la situation problème, s'il s'agit d'une situation tirée dans le cours de mathématiques on peut présenter la formule et expliquer à l'aide d'un exemple court.

Par exemple, on peut dire que 12 est supérieur à 5, car la valeur de 12 est plus grande que la valeur de 5.

Etape 3 : Elaboration de l'algorithme et traçage de l'algorithme

Traduire la situation en pseudo-code et expliquer la trace de cet algorithme en raisonnement machine à l'aide de l'exemple ci-dessus.

```
DEBUT
    Entrer n1, n2
    Lire n1, n2
    SI n1 supérieur à n2 ALORS
    Afficher "Nombre 1 supérieur à nombre 2"
    SI NON
    Afficher "Nombre 2 supérieur à nombre 1"
FIN
```

Etape 4 : Tester l'algorithme en utilisant Algobox

Amener les élèves au Laboratoire pour tester l'algorithme dans un logiciel d'élaboration d'algorithme.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

```
FONCTIONS_UTILISEES
VARIABLES
 n1 EST_DU_TYPE NOMBRE
 n2 EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
 AFFICHER "Entrer la valeur du Nombre 1 : "
 LIRE n1
 AFFICHER "Entrer la valeur du Nombre 2 : "
 LIRE n2
 SI (n1>n2) ALORS
  DEBUT_SI
  AFFICHER "Le Nombre 1 est supérieur au Nombre 2"
  FIN_SI
  SINON
   DEBUT_SINON
   AFFICHER "Le nombre 2 est supérieur au nombre 1"
   FIN_SINON
FIN_ALGORITHME
```

Etape 5 : Implémentation de l'algorithme

En utilisant un langage de programmation dans un environnement donné (Dev-C++ ou autre) amener les apprenants à coder. Dans l'exemple présent nous utilisons Dev-C++ pour coder en C et Spyder pour coder en Python.

Figure 2: Contextualisation des notions des structures conditionnelles ou alternatives en Python.

```
3   Spyder Editor
4   Auteur : KASAMBI.
5   Projet : DEA
6   Sujet : MTIC 3.8
7   """
8   n1 =  input("Entrez la valeur du Nombre 1 : ")
9   n2 =  input("Entrez la valeur du Nombre 1 : ")
10
11  if n1 > n2:
12    print("Le nombre 1 est supérieur au nombre 2.")
13  else:
14    print("Le nombre 2 est supérieur au nombre 1.")
```

Source : Notre confection sous l'éditeur Spyder

En utilisant Spyder cela donne le résultat suivant :

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Figure 3: Résultats de la contextualisation des notions des structures conditionnelles ou

```
In [2]: runfile('C:/Users/KASAMBI/.spyder-py3/temp.py', wdir='C:/Users/
KASAMBI/.spyder-py3')
Entrez la valeur du Nombre 1 : 45
Entrez la valeur du Nombre 1 : 76
Le nombre 2 est supérieur au nombre 1.
```

alternatives en Python.

Source : Notre confection sous l'éditeur Spyder

En langage C cela donnera

Figure 4:Contextualisation des notions des structures conditionnelles ou alternatives en C.

```c
#include<stdio.h>
int main(){
    int n1, n2 = 5;
    printf("Entrer la valeur de Nombre 1: ");
    scanf("%d", &n1);
    printf("Entrer la valeur de Nombre 2:");
    scanf("%d", &n2);
    if (n1 > n2) {
      printf("Le Nombre 1 est supérieur au Nombre 2 .\n");
    } else {
      printf("Le Nombre 2 est supérieur au Nombre 1.\n");
    }
}
```

Source : Notre confection sous DevC++

Le résultat après exécution donne :

Figure 5: Résultats de la contextualisation des notions des structures conditionnelles ou alternatives en C.

```
Entrer la valeur de Nombre 1: 45
Entrer la valeur de Nombre 2:76
Le Nombre 2 est supÚrieur au Nombre 1.

--------------------------------
Process exited after 12.66 seconds with return value 0
Appuyez sur une touche pour continuer...
```

Source : Notre confection sous DevC++

Propositions

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Pour l'apprentissage élémentaire de la programmation, il est important de commencer sur les règles de base de la programmation. Python étant un langage évolué, il outrepasse certaines instructions élémentaires clés pour créer la pensée algorithmique aux apprenants. On le constate avec le petit programme ci-dessus, en python il ne pas obligatoire de déclarer une variable et définir son type pour l'utiliser, ceci saute certaines étapes de l'élaboration de l'algorithme puisqu'au départ l'apprenant doit savoir déclarer une variable conformément au type de données qu'elle pourra stocker.

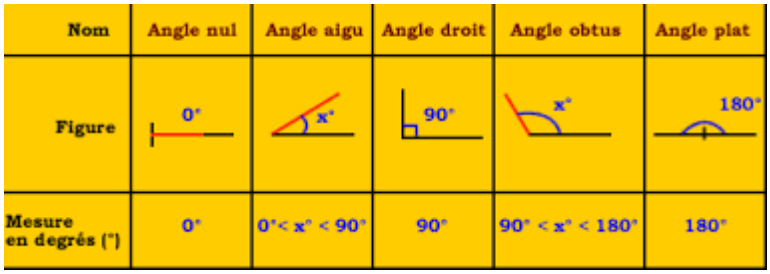
Pour enseigner la troisième forme de l'instruction SI, l'enseignant peut utiliser la situation problème en rapport avec la résolution de la détermination de l'angle à partir des degrés.

Etape 1 : Compréhension de la situation problème

L'enseignant commence par éclaircir les types d'angles et leurs degrés respectifs.

Un angle est dit **droit** lorsqu'il mesure exactement 90°. Il est formé par deux demi-droites qui se coupent perpendiculairement. Un angle est dit **aigu** lorsqu'il mesure entre 0° et 90°. Il est formé par deux demi-droites qui se coupent de telle sorte que la mesure de l'angle est inférieure à celle d'un angle droit. Un angle est dit **obtus** lorsqu'il mesure entre 90° et 180°. Il est formé par deux demi-droites qui se coupent de telle sorte que la mesure de l'angle est supérieure à celle d'un angle droit. Un angle est dit **plat** lorsqu'il mesure exactement 180°. Il est formé par deux demi-droites qui se coupent de telle sorte que les côtés opposés sont alignés.

Figure 6: Détermination des angles MTIC3.8



Etape 2 : Principe de résolution manuelle de la situation problème

A travers un exemple simple, on démontre la résolution manuelle de cet algorithme pour avoir un angle droit, obtus, plat.

Etape 3 : Elaboration de l'algorithme et traçage de l'algo

Lire le degré

Tester si (degré >0 et < 90) alors

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

```
                    Afficher "Angle Aigu"
SI non SI (degré = 90) alors
                    Afficher "Angle droit"
SI non SI (degré> 90 ET degré < 180) alors
                    Afficher "Angle Obtus"
SI non SI (degré = 180) alors
                    Afficher "Angle plat"
SI non alors
                    Afficher "Angle non spécifique"
Fin SI
```

Etape 4 : Tester l'algorithme en utilisant Algobox

```
1  FONCTIONS_UTILISEES
2  VARIABLES
3    degre EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5    AFFICHER "Entrer la valeur du degré : "
6    LIRE degre
7    SI (degre >0 ET degre <90) ALORS
8     DEBUT_SI
9     AFFICHER "C'est un angle aigu"
10    FIN_SI
11    SINON
12      DEBUT_SINON
13      SI (degre = 90) ALORS
14        DEBUT_SI
15        AFFICHER "C'est un angle droit"
16        FIN_SI
17        SINON
18          DEBUT_SINON
19          SI (degre>90 ET degre<180) ALORS
20            DEBUT_SI
21            AFFICHER "C'est un angle obtus"
```

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

```
22          FIN_SI
23          SINON
24           DEBUT_SINON
25           SI (degre = 180) ALORS
26            DEBUT_SI
27            AFFICHER "C'est un angle plat"
28            FIN_SI
29            SINON
30             DEBUT_SINON
31             AFFICHER "Angle non spécifique"
32             FIN_SINON
33           FIN_SINON
34         FIN_SINON
35       FIN_SINON
36  FIN_ALGORITHME
```

Etape 5 : Implémentation de l'algorithme en C

Figure 7: Implémentation de l'algorithme : Détermination des angles MTIC3.8 en C

MTIC3.8 Angles.c

```c
#include<stdio.h>
main(){
    int degre;
    printf("Entrer la valeur du degre : ");
    scanf("%d",&degre);
    if (degre>=0 && degre<90){
        printf("Angle aigu");
    }else
    if (degre==90){
        printf("Angle droit");
    }else
    if (degre>90 && degre<180){
        printf("Angle obtus");
    }else
    if (degre==180){
        printf("Angle plat");
    }else{
        printf("Angle non specifique");
    }
}
```

Source : Notre confection sous DevC++

Son équivalent en Python est :

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Figure 8:Implémentation de l'algorithme : Détermination des angles MTIC3.8 en Python

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Nov  4 18:04:37 2023
@author: KASAMBI
Détermine le type d'un angle en fonction dU degré
"""
def determiner_type_angle(degre):

  if degre >= 0 and degre < 90:
    return "Angle aigu"
  elif degre == 90:
    return "Angle droit"
  elif degre > 90 and degre < 180:
    return "Aigle obtus"
  elif degre == 180:
    return "Angle plat"
  else:
    return "Angle non specifique"


def main():
  """
  Programme principal.
  """

  degre = int(input("Entrez la valeur du degre : "))

  type_angle = determiner_type_angle(degre)
  print(" Cet '{}' degrés et est de type {}".format(degre, type_angle))

if __name__ == "__main__":
  main()
```

Source : Notre confection sous l'éditeur Spyder

LES STRUCTURES REPETITIVES [MTIC3.9]

Une structure répétitive est une structure de contrôle qui permet de répéter un bloc d'instructions un certain nombre de fois (Delannoy, 2016; Léry, 2022). Les structures répétitives sont utilisées lorsque l'on doit exécuter un bloc d'instructions plusieurs fois, par exemple pour traiter une liste de données ou pour répéter une action jusqu'à ce qu'une condition soit remplie (Boucherit, 2022; TAMOUM, 2018).

Il existe deux types de structures répétitives en programmation :

- La boucle **while** est une structure répétitive qui exécute un bloc d'instructions tant que la condition spécifiée est vraie.
- La boucle **for** est une structure répétitive qui exécute un bloc d'instructions un certain nombre de fois, spécifié par une variable de compteur.

Partant de l'exemple de situation donné à la matrice MTIC3.9, l'enseignant peut commencer par un exercice simple, exemple « Ecrire Bonjour » un certain nombre de fois avant de passer aux algorithmes complexes tels que le calcul du factoriel, ou de la puissance.

En nous référant toujours au modèle ci-dessus, on peut exécuter étape par étape les procédures ci-dessus.

Etape 1 : Compréhension de la situation problème

Répéter une information autant de fois que possible, ceci peut intervenir dans les situations de la vie courante faisant appel à l'impression des données sur des tissus (pagnes, argent, rideaux, …)

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Etape 2 : Principe de résolution manuelle

A travers un exemple simple, on peut faire recourt aux structures répétitives pour résoudre des problèmes de la vie courante faisant appel à ces notions.

Etape 3 : Elaboration de l'algorithme et traçage de l'algorithme

```
Déclarer le compteur
TANT QUE (condition) FAIRE
        DEBUT TANT QUE
                Afficher "Bonjour"
        FIN TANT QUE
```

Etape 4 : Tester l'algorithme en utilisant Algobox

```
1  FONCTIONS_UTILISEES
2  VARIABLES
3    compteur EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5    compteur PREND_LA_VALEUR 1
6    TANT_QUE (compteur<5) FAIRE
7     DEBUT_TANT_QUE
8     AFFICHER "Bonjour le monde"
9     compteur PREND_LA_VALEUR compteur+1
10     FIN_TANT_QUE
11 FIN_ALGORITHME
```

Etape 5 : Implémentation de l'algorithme

Figure 9: Implémentation de l'algorithme "Bonjour le monde' en C [MTIC3.9]

```c
#include<stdio.h>
main(){
    int i;
    for(i=0;i<=4; i++){
        printf("Bonjour le monde \n");
    }
}
```

Après cette introduction, l'enseignant peut alors puiser une situation dans les domaines connexes tels que les mathématiques pour expliquer davantage les notions des structures répétitives.

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Dans l'exemple ci-dessous nous résolvons une situation faisant appel au calcul du factoriel d'un nombre partant de cette situation problème :

Lors de l'étude de la statique des forces, les élèves de la première année des humanités scientifiques constatent que plusieurs forces concourantes ou parallèles peuvent être représentées par une force appelée résultante et qui fait intervenir analytiquement des carrées des forces alors qu'au cours des séquences de mathématiques, on leur a parlé des puissances des nombres et des factoriels. Émerveillés par le cours des TIC sur la leçon de codage, les élèves demandent à leur enseignant des TIC d'automatiser si possible ces trois calculs. L'enseignant profite de cette occasion pour leur demander d'écrire des algorithmes et programmes qui calculent le factoriel et la puissance n d'un nombre et la résultante de plusieurs forces concourantes connaissant leurs expressions mathématiques.

Etape 1 : Compréhension de la situation problème

La factorielle d'un nombre a de nombreuses applications en mathématiques, en statistique et en informatique. Par exemple, la factorielle est utilisée pour calculer le nombre de permutations d'un ensemble, le nombre de combinaisons d'un ensemble et la probabilité d'un événement.

Etape 2 : Principe de résolution manuelle

La factorielle d'un entier naturel **n**, notée **n!**, est le produit de tous les entiers naturels inférieurs ou égaux à n. Par exemple, 5! = 120, car 5! = 5 × 4 × 3 × 2 × 1 = 120.

La factorielle d'un nombre peut être calculée de manière récursive ou itérative.

Par la méthode récursive, la factorielle d'un nombre n est la suivante :

n! = n × (n - 1)!

Par exemple, pour calculer 5!, on peut utiliser la récursivité suivante :

5! = 5 × 4!

4! = 4 × 3!

3! = 3 × 2!

2! = 2 × 1!

1! = 1

Etape 3 : Elaboration de l'algorithme et traçage de l'algorithme

```
Déclarer var nombre et factorielle
Lire n
TANT QUE (nombre>1) FAIRE
        DEBUT TANT QUE
```

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Factorielle = factorielle X nombre

Nombre=nombre-1

FIN TANT QUE

Afficher la factorielle

Etape 4 : Tester l'algorithme en utilisant Algobox

```
1  FONCTIONS_UTILISEES
2  VARIABLES
3    n EST_DU_TYPE NOMBRE
4    fact EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6    fact PREND_LA_VALEUR 1
7    AFFICHER "Entrer un entier :"
8    LIRE n
9    TANT_QUE (n>=1) FAIRE
10     DEBUT_TANT_QUE
11     fact PREND_LA_VALEUR fact*n
12     n PREND_LA_VALEUR n-1
13     FIN_TANT_QUE
14   AFFICHER "La factorielle est : "
15   AFFICHER fact
16 FIN_ALGORITHME
```

Etape 5 : Implémentation de l'algorithme avec la boucle **WHILE**

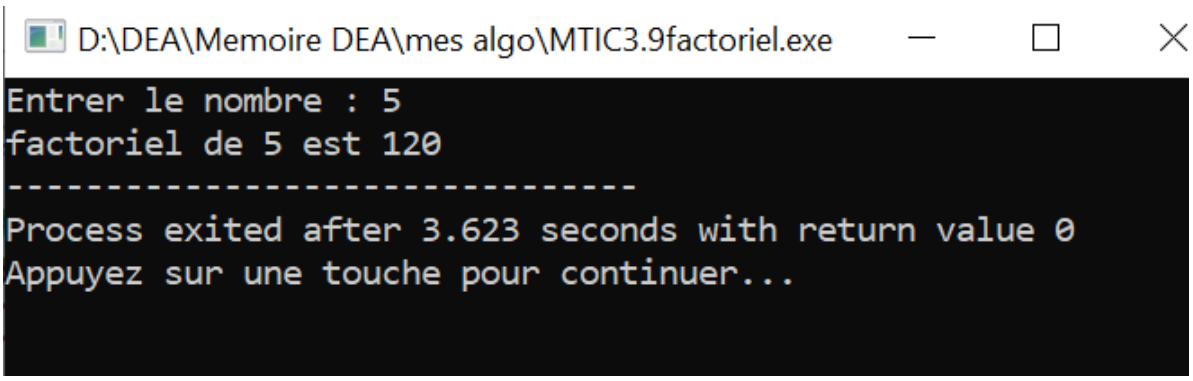
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Figure 10:Implémentation de la factorielle en C avec la boucle **WHILE** |MTIC3.9]

```c
#include<stdio.h>
main(){
    int n;
    printf("Enter le nombre : ");
    scanf("%d",&n);
    int nbr=n;
    int fct=1;
    while(n>=1){
        fct=fct*n;
        n=n-1;
        //printf("le factoriel de %d est %d \n",nbr,fct);
    }
    printf("le factoriel de %d est %d",nbr,fct);
}
```

Source : Notre confection sous DevC++

Figure 11: Résultats de la factorielle d'un nombre [MTIC3.9]



Implémentation avec la boucle FOR

Figure 12: Implémentation de la factorielle en C avec la boucle FOR |MTIC3.9]

```c
#include<stdio.h>
main(){
    int n,i;
    printf(" Enter le nombre : ");
    scanf("%d",&n);
    int nbr=n;
    int fct=1;
    for(i=1;i<=n;i++){
        fct=fct*i;
    }
    printf("le factoriel de %d est %d",nbr,fct);
}
```

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

On peut également expliquer la fonction de ce petit programme, en ceci :

Exemple nbr=3

i=1; 1<=3; oui,i=1+1=2, fct=1*1=1

i=2; 2<=3; oui, i=2+1=3, fct=1*2=2

i=3; 3<=3; oui,i=3+1=4, fct=2*3=6

i=4; 4<=3; non, stop

fct=6

LES STRUCTURES DE CAS [MTIC3.10]

Une structure de cas est une structure de contrôle qui permet de choisir entre plusieurs actions possibles en fonction d'une condition. Les structures de cas sont utilisées lorsque l'on doit effectuer une action différente en fonction de la valeur d'une variable ou d'une expression.

Il existe deux types de structures de cas en programmation :

- **La structure switch**
- **La structure if-else**

Exemple de situation : Les élèves de la première année des humanités scientifiques de l'Institut IMANI PANZI éprouvent d'énormes difficultés pour consulter le tableau périodique des éléments. L'enseignant de chimie demande à son collègue des TIC de lui proposer une solution informatique qui permet, partant d'un élément, d'une période ou d'une famille, de retrouver les caractéristiques de cet élément et les autres éléments de la famille ou de la période. En classe, L'enseignant des TIC demande à ses élèves d'écrire un programme permettant d'afficher un élément du tableau périodique connaissant son symbole, de déterminer sa famille, sa période et les autres éléments de la famille et /ou période.

Situation similaire : Ecrire un programme qui calcule la somme des 10 premiers entiers naturels.

Commentaire : Cette situation n'est pas adaptée pour une résolution avec la structure des cas, elle est plutôt adaptée pour une résolution avec la structure répétitive.

Partant de cet exemple de situation [MTIC3.10], comme les éléments du tableau périodique sont regroupés en famille et en période il s'agit de deux algorithmes et programmes différents. Le premier algorithme consistera à trouver la famille d'un élément en utilisant la structure de cas et le deuxième algorithme consistera à utiliser à trouver la période d'un élément en utilisant une structure répétitive.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Toutefois, avant d'aborder ces notions ces notions de structure de cas, il serait capital de procéder d'abord par une situation problème basique telles que :

- Détermination des jours de la semaine à partir d'un chiffre
- Détermination des mois de l'année à partir d'un nombre enter
- Déterminer la saison d'un mois d'une saisie

Etape 1 : Compréhension de la situation problème

Détermination de la saison d'un mois donné.

Etape 2 : Principe de résolution manuelle

Nous avons 2 saisons à Bukavu, la saison sèche commence souvent en juin pour terminer en septembre, et la saison de pluie, qui occupe le reste de l'année.

Etape 3 : Elaboration de l'algorithme et traçage de l'algorithme

Lire mois
Selon (mois)
    Début swicth
    Au cas où mois est 6,7,8 :
  Afficher "Saison sèche »
    Au cas où mois est 1,2,3,4,5,9,10,11,12 :
  Afficher "Saison de pluie »
Par défaut
Afficher "Ce mois n'existe pas"

Etape 4 : Tester l'algorithme en utilisant Algobox

Algobox étant limité pour cette instruction, nous implémentons directement en C

Etape 5 : Implémentation de l'algorithme en C

Figure 13: Implémentation de l'algorithme : Déterminer mois/Saison MTIC3.10 en C

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

```c
#include <stdio.h>
int main() {
  int mois;
  printf("Entrez le mois : ");
  scanf("%d", &mois);
  switch (mois) {
    case 6:case 7:case 8:
        printf("Le %d e mois est dans la saison seche a Bukavu",mois);
    break;
    case 1:case 2:case 3:case 4:case 5:case 9:case 10:case 11:case 12:
      printf("Le %d e mois est dans la Saison des pluies a Bukavu",mois);
    break;
    default:
     printf("Ce mois n'existe pas a Bukavu");
  }
  return 0;
}
```

Source : Notre confections sous DevC++

Figure 14: Implémentation de l'algorithme : Déterminer mois/Saison MTIC3.10 en Python

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  5 09:40:56 2023

@author: KASAMBI
"""

mois = int(input("Entrez le mois : "))
if mois in range (1,6):
    print ("Saison de pluie")
elif mois in range (6,9):
        print ("Saison sèche")
elif mois in range (9,13):
        print ("Saison des pluies")
else:
    print("Ce mois n'existe pas")
```

Source : Notre confection sous Spyder

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Le résultat donne :

Figure 15: Résultats de l'Implémentation de l'algorithme : Déterminer mois/Saison MTIC3.10 en Python

```
In [33]: runfile('C:/Users/KASAMBI/.spyder-py3/saisonBukavu.py', wdir='C:/Users/
KASAMBI/.spyder-py3')
Entrez le mois : 13
Ce mois n'existe pas

In [34]: runfile('C:/Users/KASAMBI/.spyder-py3/saisonBukavu.py', wdir='C:/Users/
KASAMBI/.spyder-py3')
Entrez le mois : 12
Saison des pluies

In [35]:
```

Après avoir initié les élèves à cette petite situation basique, on peut alors appliquer l'exemple de situation donné à la matrice MTIC3.10.

Etape 1 : Compréhension de la situation problème

Elaborer un algorithme permettant d'afficher la famille d'un élément sur le tableau périodique. Les éléments chimiques sont rangés sur le tableau périodique en fonction de leur numéro atomique, qui correspond au nombre de protons dans le noyau de l'atome. Les éléments sont également regroupés en fonction de leurs propriétés chimiques, qui sont déterminées par le nombre d'électrons dans la couche externe de l'atome.

Les éléments du tableau périodique sont également répartis en 18 familles, qui correspondent au nombre d'électrons de valence, qui sont les électrons situés dans la couche externe de l'atome.

Etape 2 : Principe de résolution manuelle

Les familles sont numérotées de 1 à 18, de gauche à droite.

Etape 3 : Elaboration de l'algorithme et tassage de l'algorithme

On se réfère à l'exemple précédent pour expliquer en utilisant la structure de cas.

Etape 4 : Tester l'algorithme en utilisant LARP

Figure 16: L'algorithme : Déterminer la famille d'un élément chimique MTIC3.10

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

```
\\ Module principal
DÉBUT
LIRE {position}
SÉLECTIONNER {position}
     {position entre 1-12} : {L'élément est de la famille des métaux}
     SÉLECTIONNER {expression}
          {position entre 13-17} : {L'élément est de la famille de non métaux}
          SINON
             {position 18}  ; {L'élément est de la famille de gaz}
     FINSÉLECTIONNER
     SINON
        {Cet élément n"existe pas}
FINSÉLECTIONNER
FIN
```

Source : Notre propre confection sous LARP

Etape 5 : Implémentation de l'algorithme en C

Figure 17:Implémentation de l'algorithme : Déterminer la famille d'un élément chimique MTIC3.10 en C

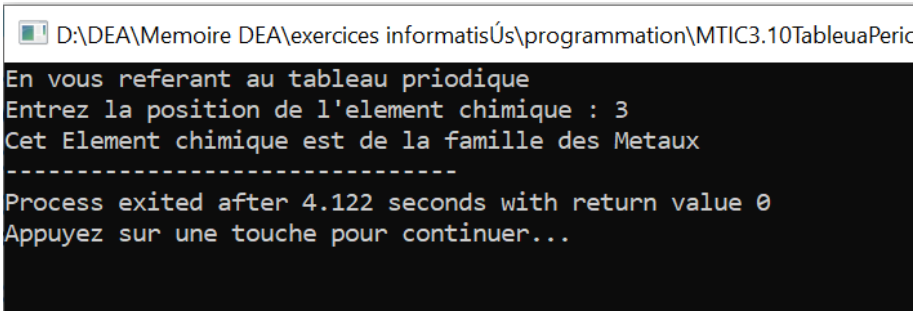
```c
#include <stdio.h>
int main() {
  int position;
  printf("En vous referant au tableau priodique\n");
  printf("Entrez la position de l'element chimique : ");
  scanf("%d", &position);
  switch (position) {
    case 1:case 2:case 3:case 4:case 5:case 6:case 7:case 8:case 9:case 10:case 11:case 12:
        printf("Cet élément chimique est de la famille des Métaux");
    break;
    case 13:case 14:case 15:case 16:case 17:
      printf("Cet element chimique est de la famille des Non Métaux");
    break;
     case 18:
      printf("Cet un gaz");
    break;
    default:
     printf("Aucun Element chimique n'occupe cette position");
  }
  return 0;
}
```

Source : Notre confection sous DevC++

Revue **Francophone**
Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Le résultat est :

Figure 18: Résultats de l'implémentation de l'algorithme : Déterminer la famille d'un élément chimique MTIC3.10 en Python



Source : Résultat de l'exécution du code sous DEVC++

Son équivalent en Python :

Figure 19:Implémentation de l'algorithme : Déterminer la famille d'un élément chimique MTIC3.10 en Python avec Case simple

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  5 12:04:30 2023

@author: KASAMBI
"""

position = int(input("Entrez la position de l'élément: "))
if position in range (1,12):
    print ("C'est un élément de la famille des métaux")
elif position in range (13,17):
        print ("C'est un élément de la famille des non-métaux")
elif position ==18:
        print ("C'est un élément de la famille de GAZ")
else:
    print("Cet élément n'existe pas sur le tableau périodique")
```

Le résultat après exécution en Spyder

```
In [26]: runfile('C:/Users/KASAMBI/.spyder-py3/tableauPeriodique.py', wdir='C:/
Users/KASAMBI/.spyder-py3')
Entrez la position de l'élément: 3
C'est un élément de la famille des métaux
```

CONCLUSION

Plusieurs études ont démontré que la pensée algorithmique tire ses origines dans les mathématiques (Briant, 2013; Briant & Bronner, 2015; Meyer & Modeste, 2022; Modeste, 2012a, 2012b), ce qui fait que certains apprenants se sentent démotiver pour s'en approprier. Le taux d'abandon du cours de programmation s'avère élevé (DEBABI, 2018; Guibert et al.,

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

2005; Kaasbøll, 1998), pour autant, l'enseignant doit inventer des mécanismes permettant aux apprenants de s'approprier les notions d'algorithmique, très capitales pour réaliser des bons programmes. Les résultats de notre étude montrent que la transposition didactique des situations problèmes d'algorithmique en première année scientifique en RDC est possible et prometteuse. En effet, la transposition visée dans cette étude permet de formater les savoirs savants en connaissances à enseigner conformément au programme éducatif en vigueur en République Démocratique du Congo.

Le nouveau programme éducatif (2019) introduit les notions d'algorithmiques à partir de la classe de terminale de l'éducation de base et continue jusque dans les classes de la section scientifique. Ceci est une avancée majeure dans l'enseignement de l'informatique en RDC, puisque depuis l'introduction officielle de l'informatique au programme national de 2007, seule le programme de la section pédagogique permettait aux apprenants d'apprendre un peu de la programmation de manière superficielle. Ce nouveau programme, bien que remplit des défis pédagogiques à relever, introduit quand bien même des notions d'algorithmique et de programmation avec comme langages de référence Python et C. Les justificatifs de Python et C n'ont pas été présentés mais toutefois nous estimons que c'est dû à la simplicité et clarté des syntaxes et facilité de trouver des ressources de ces langages.

Pour cette étude, en utilisant le logiciel Algobox, nous avions exploré les possibilités de la transposition didactique des situations problèmes présentées dans le programme éducatif. Malgré que le volume accordé soit un défi majeur pour l'enseignant qui enseigne que pour les élèves qui apprennent, nous sommes confiants que cette introduction d'éléments d'algorithmique et programmation réussira si tous les décideurs à leurs niveaux fournissent d'amples efforts pour appuyer le processus enseignement-apprentissage.

Il n'existe guère une solution magique en termes de solution informatique, nous invitons d'autres chercheurs à compléter cette étude en analysant l'impact de choix de ces langages de programmation [Python et C] sur l'apprentissage des apprenants de la section scientifique. Aussi, il est possible d'étudier l'impact que peut jouer un logiciel d'élaboration d'algorithme sur la maitrise de la programmation par les élèves du secondaire en RDC. Ainsi, des recherches futures pourraient explorer la mise en œuvre effective de ces situations pédagogiques reformulées et leur impact sur l'apprentissage des élèves.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

REFERENCES

AISSAOUI, L. (2011). *Scénarisation de l'apprentissage en algorithmique* [PhD Thesis]. https://dspace.univ-guelma.dz/jspui/handle/123456789/436

B. K. BALIMWENGU, Paulin, BAPOLISI BAHUGA, Deogratias, MBILIZI MWISIMBWA, Pascal, AKILIMALI BAMALEMBUKO, & Alain, OMBENI LANDO. (2023). Teaching algorithmic and coding in the first year of science in Bukavu : An exploratory study. *International Journal of Innovation Scientific Research and Review*, *5*(10). https://hal.science/hal-04264925v1

Balacheff, N. (1993). La transposition informatique, un nouveau problème pour la didactique. *colloque" Vingt ans de didactique des mathÃ© matiques en France", 15-17 juin 1993*, 364-370.

Boucherit, A. (2022). *Algorithmique et Structures de données I*. https://elearning.univ-eloued.dz/pluginfile.php/36767/mod_resource/content/1/cours%20Informatique%201.pdf

Briant, N. (2013). *Etude didactique de la reprise de l'algèbre par l'introduction de l'algorithmique au niveau de la classe de seconde du lycée français*. Université Montpellier II-Sciences et Techniques du Languedoc.

Briant, N., & Bronner, A. (2015). Étude d'une transposition didactique de l'algorithmique au lycée : Une pensée algorithmique comme un versant de la pensée mathématique. *Actes du Colloque EMF2015–GT3*, 231-246. https://publimath.univ-irem.fr/numerisation/ACF/ACF15115/ACF15115.pdf

Chandra, R. V., & Varanasi, B. S. (2015). *Python requests essentials*. Packt Publishing Birmingham, UK. https://www.uplooder.net/ofiles/ed3eeb642c9ce68c492ff36d183cff48/Python-Requests-Essentials.pdf

Conne, F. (1992). Savoir et connaissance dans la perspective de la transposition didactique. *Recherches en didactique des mathématiques*, *12*(2.3), 221-270.

DEBABI, W. (2018). *Conception et implémentation d'un Serious Game*. https://biblio.univ-annaba.dz/wp-content/uploads/2022/06/These-Debabi-Wassila.pdf

Del Notaro, C. (2013). Transposition didactique de quelques critères de divisibilité dans le manuel de 6P (8Harmos). *Math-Ecole*, *219*, 4-9.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Delannoy, C. (2016). *Programmer en langage C : Cours et exercices corrigés*. Editions Eyrolles. https://www.fnac.com/a9824782/Claude-Delannoy-Programmer-en-langage-C-5e-edition

Eidelman, A. (2020). Python Data Science Handbook. *Statistique et Société*, *8*(2), 45-47.

EPSP2. (2016). Programme éducatif du Domaine d'Apprentissage des Sciences Classe de 8ème année de l'Education de Base. *Direction des Programmes Scolaires et Matériel Didactique*. https://www.eduquepsp.education/v1/programme-scolaire/#1561183235144-40a47fd3-7f89

EPST. (2021). *Programme éducatif du Domaine d'Apprentissage des Sciences Classe de 1ère année des Humanités Scientifiques*. Direction des Programmes Scolaires et Matériel Didactique. https://www.eduquepsp.education/v1/programme-scolaire/#1561183235144-40a47fd3-7f89

EPSP5. (2019). Guide en appui au Programme Éducatif du Domaine d'Apprentissage des Sciences Classe de première année Scientifique. *Direction des Programmes Scolaires et Matériel Didactique*. https://www.eduquepsp.education/v1/programme-scolaire/#1561183235144-40a47fd3-7f89

Guibert, N., Guittet, L., & Girard, P. (2005). Initiation à la Programmation «par l'exemple» : Concepts, environnement, et étude d'utilité. *Acte de colloque EIAH'05*, 461-466. https://www.researchgate.net/profile/Patrick-Girard-4/publication/255641814_Initiation_a_la_Programmation_par_l'exemple_concepts_environnement_et_etude_d'utilite/links/55705c8108ae193af41ff4ae/Initiation-a-la-Programmation-par-lexemple-concepts-environnement-et-etude-dutilite.pdf

Kaasbøll, J. J. (1998). Exploring didactic models for programming. *NIK 98–Norwegian Computer Science Conference*, 195-203.

Léry, J.-M. (2022). *Algorithmique en C, C++, Java, Python et PHP*. Editions Ellipses. https://books.google.com/books?hl=fr&lr=&id=neCDEAAAQBAJ&oi=fnd&pg=PR1&dq=Le+langage+de+programmation+PHP&ots=hlw4q1Z0L4&sig=5O3uW1DXoJoDlV_Cl7Ka8v4rI9k

Meyer, A., & Modeste, S. (2022). Rôle d'un logiciel dans la transposition didactique du concept d'algorithme : Le cas du logiciel AlgoBox en France et des programmes du lycée entre 2009 et 2019. *Cahiers d'histoire du Cnam*, *15*(1), 97-131.

Volume : 2 Numéro : 2 Page: 167 à 196
Doi: 10.5281/zenodo.11132956 Reçu : 28/03/2024 ; Révisé : 10/04/2024 ; Accepté: 27/04/2024

Modeste, S. (2012a). *Enseigner l'algorithme pour quoi? Quelles nouvelles questions pour les mathématiques? Quels apports pour l'apprentissage de la preuve?* [PhD Thesis, Université de Grenoble]. https://theses.hal.science/tel-00783294/

Modeste, S. (2012b). La pensée algorithmique : Apports d'un point de vue extérieur aux mathématiques. *Actes du colloque espace mathématique francophone*.

Ravel, L. (2003). *Des programmes à la classe : Etude de la transposition didactique interne. Exemple de l'arithmétique en Terminale S spécialité mathématique.* [PhD Thesis, Université Joseph-Fourier-Grenoble I]. https://theses.hal.science/tel-00162790/

TAMOUM, M. (2018). *Informatique*. https://lmd.sahla-dz.com/oackoafa/2023/08/SAHLA-DZ.COM-Les-cours-de-module-Informatique-1-Dr.-Mohammed-TAMOUM.pdf

Tchounikine, P. (2017). Initier les élèves à la pensée informatique et à la programmation avec Scratch. *Laboratoire d'informatique de Grenoble. En ligne:* http://ligmembres.imag.fr/tchounikine/PenseeInformatiqueEcole.html.