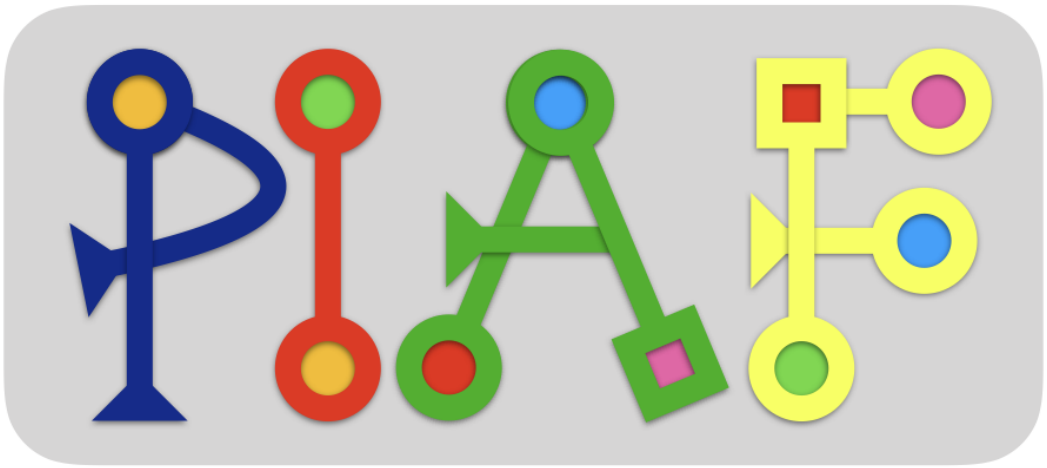

PIAF : développer la Pensée Informatique et Algorithmique dans l'enseignement Fondamental

Référentiel de compétences

Annexe 1: Description et exemples

Erasmus+

LIÈGE université

UNIVERSITÉ DE LORRAINE

UNIVERSITÉ DU LUXEMBOURG

UNIVERSITÄT DES SAARLANDES

Version 2 – Juillet 2021

Table des matières

Compétence 1 : Définir des abstractions / généraliser

C 1.1 Nommer des objets et (séquences d') actions

Définition
Être capable de donner des noms à des objets, des actions et des séquences d'actions.

Contexte
Il est important de prendre l'habitude de donner des noms aux objets / actions existants afin de pouvoir y référer et faciliter l'expression des solutions.

À noter
Certaines valeurs (objets) peuvent être connues et ne pas changer (par exemple un nom de famille, un numéro de rue), dans ce cas celles-ci sont appelées constantes. Certaines valeurs ne sont pas connues et / ou peuvent changer en fonction du contexte (par exemple la température), et sont donc appelées variables. Ce concept de variable est abordé dans la compétence C 1.6 ci-dessous.

Exemples illustrant la compétence

1. Étant donné une histoire avec différents personnages présentant des similitudes (par exemple représentant tous des sportifs), mais distinguables, être capable donner des noms afin de les identifier de manière univoque, et de les intégrer dans une histoire.

2. Un apprenant réalise une action, un autre la voit et doit simplement la raconter à un troisième, qui ne voit pas le premier apprenant. Ce troisième apprenant doit alors être capable de reproduire cette action.

3. Deux apprenants s'accordent sur quatre mots correspondant à quatre (séquences d') actions simples. Un troisième apprenant arrive. Le premier dit un des quatre mots et le deuxième exécute les commandes correspondantes devant le troisième apprenant, qui doit être capable d'apprendre le langage défini par les quatre mots (et par la suite d'exécuter les actions demandées).

4. Après avoir défini les étapes nécessaires pour s'habiller, être capable d'appeler cette séquence d'actions "s'habiller".

C 1.2 Différencier (i) objet et action, et (ii) actions atomiques et non-atomiques

Définition
Être capable de dire si quelque chose (image, expression, phrase) fait référence à un objet, à une action, ou bien à une séquence d'actions.

Contexte
Les objets et les actions peuvent être considérés comme des métaphores des concepts informatiques d'*expression* et d'*instruction*. Ces derniers se distinguent par le fait que les expressions ont une valeur alors que les instructions n'en ont pas (elles changent l'état de l'environnement [mémoire, écran, etc.]).

Cette compétence fait référence à la capacité de distinguer les parties d'un algorithme (ex. recette, mouvement, etc.) qui ont des valeurs (généralement représentées par des noms) de celles qui n'en ont pas (généralement représentées par des verbes).

Par ailleurs, certaines actions peuvent être décomposées en sous-actions, auquel cas elles sont appelées actions *non-atomiques*. Les actions qui ne peuvent pas être décomposées davantage sont, elles, appelées actions *atomiques*.

Exemples

1. À partir de mots (ou groupes de mots) donnés, pouvoir dire s'il s'agit d'un objet (en général un nom, par exemple une montagne, un arbre, une image, le chiffre 5) ou d'une action (généralement un groupe verbal, par exemple sauter, enfiler quelque chose, chanter, réfléchir).

2. À partir de cartes sur lesquelles figurent des mots, être capable de classer ces cartes en deux groupes (objets *vs*. actions), puis essayer de voir quelles actions peuvent s'appliquer à quels objets.

3. Imaginer (i) quelles actions on peut effectuer sur un objet ou (ii) sur quels objets une action peut être effectuée.

4. Classer les actions suivantes en fonction de leur atomicité : se préparer pour l'école, mettre une chaussette, danser, ordonner les animaux par taille, tourner à gauche de 90 degrés, multiplier un nombre par deux, donner la première lettre de l'alphabet.

C 1.3 Identifier les paramètres d'entrée d'une séquence d'actions

Définition
Être capable de dire ce qui est requis (objet ou séquence d'actions) pour effectuer une séquence d'actions.

Contexte
Les séquences d'actions peuvent être considérées comme des fonctions (au sens mathématique). Ses paramètres d'entrée sont souvent appelés simplement ses entrées.
Lors de l'apprentissage de l'algorithmique, il est recommandé de commencer par établir la liste des entrées et le résultat de l'algorithme avant d'écrire l'algorithme lui-même.

Exemples
1. Si je veux dire qui est la plus grande personne parmi trois individus, j'ai besoin de connaître leurs tailles.

2. Pour m'habiller, je dois connaître les vêtements à ma disposition.

3. Pour réaliser une recette, je dois connaître et disposer des ingrédients nécessaires.

4. Pour couper des cheveux, j'ai besoin d'une paire de ciseaux et d'une personne chevelue.

C 1.4 Décrire le résultat d'une séquence d'actions

Définition
Après avoir exécuté une séquence d'actions (qu'il s'agisse d'une séquence informelle, d'un algorithme, ou d'un programme), être capable de décrire cette séquence. L'apprenant doit décrire la situation finale (quels objets ont éventuellement été modifiés / créés / supprimés, ou quelle solution à un problème a été trouvée).

Contexte
Cette compétence est fortement liée à la compétence C1.3 ci-dessus. Ces deux compétences sont nécessaires pour définir des séquences d'actions qui fonctionnent.

À noter
La construction d'une séquence d'actions ne fait pas partie de cette compétence, elle est abordée en compétence C 2.3.

Exemples
1. À partir d'un programme (par exemple en Scratch) donné, l'apprenant peut dire que le chat s'est déplacé trois fois de haut en bas, que la tortue est dans la voiture ou que la boîte contient 3 stylos.

2. À partir d'une recette (sans le titre), l'apprenant est capable de dire le nom du plat qui a été préparé.

3. Voici une carte (grille 3x3), voici le point de départ (position (1,1)), et voici le point d'arrivée (position (3,3)) :

		Arrivée
Départ		

Voici un algorithme :

Monter
Monter
Tourner à droite
Tourner à droite

Imaginons qu'un robot soit mis en position (1,2),
L'apprenant doit être capable d'exécuter cet algorithme et de dire si celui-ci permet au robot de rejoindre le point d'arrivée ou non.

C 1.5 Prédire le résultat d'une séquence d'actions

Définition

Être capable de dire, à partir d'une séquence d'actions, ce qui se passera si elle est exécutée. Contrairement à la compétence 1.4, cette compétence consiste à fournir une prédiction sans exécuter réellement la séquence d'actions.

Contexte

Cette compétence est importante en particulier pour éviter que les apprenants n'abusent d'une approche par « essais et erreurs » (c'est-à-dire, changer le programme tant qu'il échoue sans essayer de comprendre pourquoi).

Exemples

1. En reprenant l'exemple de la grille 3x3 donné en compétence C 1.4 ci-dessus, être capable de prédire si l'algorithme fourni permet au robot d'atteindre le point d'arrivée, mais avec la contrainte que le programme correspondant à l'algorithme ne doit pas être réellement exécuté.

2. Donner aux apprenants une position de départ sur une grille d'une certaine taille (par exemple 10x10) et un programme construit sur les actions *Monter*, *Tourner à droite*, *Descendre*, et *Tourner à gauche,* et leur demander quelle figure la tortue dessinera si ce programme est lancé depuis cette position initiale (en utilisant des figures facilement interprétables telles que des chiffres ou formes géométriques).

3. Donner aux apprenants un critère de comparaison entre personnes (par exemple « plus grand que ») et demander ce que donnera une suite de comparaisons deux à deux d'un ensemble de personnes, telle qu'à chaque comparaison, on garde la plus grande des deux personnes comparées.

C 1.6 Utiliser des objets dont la valeur peut changer

Définition
Être capable de manipuler des valeurs qui peuvent varier en fonction du contexte, en utilisant des abstractions nommées (appelées également des *identificateurs* ou encore *identifiants*).

Contexte
Cette compétence introduit la notion de variable informatique, qui correspond à l'association d'un nom avec un emplacement en mémoire. L'introduction du nom est appelé « *déclaration de variable* » et son association à une valeur est appelée « *affectation* ».

Exemples

1. Donner aux apprenants des cartes avec des chiffres. L'enseignant énonce oralement une addition, nombre après nombre. Les enfants doivent calculer le résultat de l'addition à la volée (c'est-à-dire, sans connaître ses éléments à l'avance et sans noter l'addition) en choisissant / mettant à jour la carte représentant le résultat.

2. Créer un programme Scratch dans lequel, tous les cinq clics, le chat miaule, ou bien, dans lequel le score du joueur correspond au nombre de fois que le chat a été « cliqué ».

C 1.7 Reconnaître, parmi des objets et séquences d'actions connus, lesquels peuvent être utilisés pour atteindre un nouvel objectif

Définition
Étant donné un nouveau problème (c'est-à-dire dont la solution est a priori inconnue), être capable de réaliser qu'il est similaire à un problème connu dont il est possible de réutiliser tout ou une partie de la solution pour résoudre ce nouveau problème.

Contexte
La pensée informatique comprend la possibilité de s'appuyer sur des solutions existantes plutôt que de tout réinventer / réécrire.

À noter
La modification effective de l'algorithme relève de la compétence C 2.5.

Exemples

1. Les apprenants reçoivent une liste de recettes : soupe aux tomates, gâteau aux fraises, curry de légumes. Il leur est demandé de faire une soupe aux oignons. Être capable de dire de quelle(s) recette(s) il est possible de se servir.

2. On donne aux apprenants un algorithme pour calculer la plus grande de trois valeurs. L'enseignant demande ensuite quelle partie de celui-ci peut être réutilisée pour calculer la plus petite de ces trois valeurs.

Compétence 2: Composer/décomposer une séquence d'actions

C 2.1 Ordonner une séquence d'actions pour atteindre un objectif

Définition

Étant donné une liste non ordonnée d'actions et un but, être capable de combiner ces actions dans un ordre valide pour construire une séquence qui permet d'atteindre ce but.

Contexte

Avant de pouvoir définir une séquence d'actions entière (cf compétence C 2.3), une activité en lien avec cette compétence mais plus facile à réaliser consiste à *ordonner* les parties déjà existantes d'une séquence d'actions. Ainsi, l'apprenant n'a pas besoin d'identifier toutes les parties nécessaires pour atteindre l'objectif, mais seulement de les mettre dans le bon ordre.

Exemples

1. Donner aux apprenants l'objectif de cuire un gâteau, être capable d'ordonner les différentes étapes (actions) parmi les suivantes : mesurer la quantité de farine, prendre un bol, casser trois œufs, etc.

2. Donner aux apprenants les différentes étapes pour s'habiller et leur demander de les ordonner (plusieurs séquences sont acceptables).

C 2.2 Compléter une séquence d'actions pour atteindre un objectif simple

Définition

Étant donné une séquence d'actions limitée et un objectif simple (c'est-à-dire ne s'appuyant pas sur des concepts avancés comme les conditions et les boucles), être capable d'*ajouter les actions manquantes* à la séquence pour atteindre l'objectif.

Contexte

En algorithmique, l'apprentissage commence souvent par s'inspirer de solutions (ou problèmes) existantes. Dans ce contexte, il est important de pouvoir évaluer dans quelle mesure une solution est éloignée d'une solution existante, et de pouvoir la compléter si nécessaire.

Exemples

1. Une forme géométrique en cours de traçage sur une grille est presque terminée et l'apprenant doit ajouter la (ou les) action(s) manquante(s).

2. Considérant que le but est d'amener un personnage à un endroit de l'écran et que le programme fourni réalise une partie du chemin, l'apprenant doit compléter les actions manquantes (éventuellement en copiant et collant les actions déjà existantes) pour amener le personnage à l'endroit voulu.

C 2.3 Créer une séquence d'actions pour atteindre un objectif simple

Définition
Étant donné un objectif simple (c'est-à-dire ne s'appuyant pas sur des concepts avancés tels que les conditions et les boucles), être capable d'identifier toutes les actions nécessaires et les mettre dans le bon ordre.

Contexte
Cette capacité peut être considérée comme une extension de la capacité d'ordonner (voir C 2.1) ou de compléter (voir C 2.2) une séquence d'actions existante.

Exemples

1. Placer l'apprenant dans une pièce vide et lui donner un ensemble d'actions réalisables. Lui demander la liste ordonnée des actions à réaliser pour ouvrir la porte de la pièce, par exemple : se tourner vers la porte, avancer jusqu'à la porte, prendre la poignée, l'actionner vers le bas, tirer la porte vers soi.

2. Dessiner un carré : Avancer, tourner à droite, avancer, tourner à droite, avancer, tourner à droite, avancer, tourner à droite.

C 2.4 Créer une séquence d'actions pour atteindre un objectif complexe

Définition
Étant donné un objectif complexe (c'est-à-dire un objectif exigeant des conditions simples ou complexes ou des boucles), être capable de définir une séquence d'actions permettant de réaliser cet objectif (i.e., identifier et combiner correctement les actions nécessaires).

Contexte
Cette compétence est une extension de la compétence C 2.3 décrite ci-dessus.

Exemples

1. Décrire les actions permettant à un personnage placé dans un labyrinthe (c'est-à-dire une grille dont certaines cases sont des murs infranchissables) de trouver son chemin vers la sortie.

2. Décrire les actions permettant de faire l'addition de deux nombres entiers naturels.

3. Étant donné un plan (carte de quartier par exemple), un point de départ, un objectif (différents points à visiter,...) et des règles (obstacles,…), donner les actions à faire pour satisfaire l'objectif tout en respectant les règles.

C 2.5 Combiner des séquences d'actions pour atteindre un objectif

Définition
Étant donné des séquences d'actions permettant de réaliser certains objectifs, être capable de dire si celles-ci sont pertinentes pour un nouvel objectif, et si oui de les combiner (ordonner) pour réaliser celui-ci.

Contexte
Ici, l'accent est mis sur la *combinaison* de séquences. Il y a une autre couche d'abstraction. Il ne s'agit pas seulement de combiner des actions ou des objets, mais des séquences (généralement désignées par un nom). Cette compétence est un prolongement de la compétence C1.7 qui consistait juste à identifier les parties réutilisables.

Exemples

1. Prendre les séquences d'actions pour préparer des plats individuels, et les combiner (dans le bon ordre) pour obtenir un menu complet de l'apéritif au dessert.

2. À partir de séquences d'actions permettant de dessiner un carré et un triangle, les combiner pour dessiner une maison.

3. Créer un jeu complexe (avec plusieurs niveaux) composé de jeux plus simples qui existent déjà.

4. Sachant que max(a,b) donne le maximum de a et b, comprendre que max(max(a,b),c) donne le maximum de a, b et c.

C 2.6 Décomposer des objectifs en sous-objectifs plus simples

Définition
Être capable de diviser un objectif complexe en plusieurs sous-objectifs qui peuvent être atteints plus facilement. Ceux-ci peuvent alors être réalisés par différentes personnes.

Contexte
Lorsqu'ils conçoivent des programmes complexes, les apprenants sont souvent confrontés au syndrome de la "feuille blanche", et se demandent par où commencer. Il est important de pouvoir s'attaquer à des problèmes complexes en les divisant d'abord en problèmes plus petits (et plus faciles à résoudre).
Il s'agit ici non plus de construire (adapter) une solution à un problème, mais d'adapter (déconstruire) le problème pour le ramener à une séquence de solutions.

Exemples

1. Quelles sont les principales étapes pour préparer le sac d'école pour le lendemain ? (par exemple, chercher son sac d'école, vérifier les cours prévus à l'emploi du temps du lendemain, mettre le matériel pour chaque cours dans le sac).

2. Commander une liste de plats pour un groupe de personnes (par exemple obtenir la commande de chaque personne, les regrouper, envoyer la liste ainsi compilée en cuisine).

3. Étant donné une forme géométrique complexe, calculer sa surface en décomposant celle-ci en une combinaison de formes simples.

Compétence 3: Contrôler une séquence d'actions

C 3.1 Répéter une séquence d'actions un nombre donné de fois

Définition
Être capable d'utiliser de manière appropriée les répétitions d'actions (ou de séquences d'actions) *un certain nombre de fois*, afin d'atteindre un objectif.

Contexte
Une première famille connue de répétitions en informatique sont celles qui sont exécutées un nombre donné (connu à l'avance) de fois. Ce type de répétitions correspond à une « boucle **pour »** en programmation.

Exemples

1. Demander à des apprenants d'écrire un algorithme qui permet d'ajouter 3 jaunes d'œufs dans un bol. Si les apprenants ne voient pas l'avantage d'une boucle, leur demander d'ajouter 25 jaunes d'œufs.

2. Étant donné un robot ne pouvant tourner que de 45° à gauche, définir une séquence d'actions lui permettant de suivre un déplacement donné (chaque fois que le robot devra tourner, il faudra répéter le virage de 45° à gauche un certain nombre de fois en fonction de l'angle voulu).

C 3.2 Répéter une séquence d'actions jusqu'à ce qu'un objectif soit atteint

Définition
Être capable de contrôler les répétitions d'actions (ou de séquences d'actions) *jusqu'à ce qu'une condition soit remplie*, afin d'atteindre un objectif.

Contexte
Les répétitions qui sont exécutées jusqu'à ce qu'une condition soit remplie (le nombre de répétitions n'est donc pas connu à l'avance) constituent également un type important de répétitions en informatique. Cela correspond à la « boucle **tant que »** en programmation.

Exemples

1. Mélanger les ingrédients *jusqu'à* l'obtention d'une pâte lisse.

2. Avancez *jusqu'à* ce que vous atteignez un mur (dans ce cas, il y a un capteur pour détecter la proximité d'un obstacle).

C 3.3 Intégrer une condition simple dans une séquence d'actions

Définition
Être capable d'utiliser des conditions simples (reposant sur **un** critère) pour permettre (ou interdire) certaines séquences d'actions.

Contexte
Le contrôle de l'exécution de certaines séquences d'actions au moyen de conditions est central dans la pensée informatique, voir aussi compétence C 4.1 ci-dessous.

Exemples
1. **Si** la pâte est épaisse, **alors** ajouter 20 cl d'eau et mélanger.

2. **Si** le robot fait face au mur, **alors** il recule vers l'arrière, **sinon** il s'arrête.

3. Demander à une personne de donner son âge et lui dire (**selon la réponse**) "vous êtes un adulte maintenant" **ou** "attendez un peu avant de conduire une voiture".

C 3.4 Intégrer une condition complexe dans une séquence d'actions

Définition
Être capable d'utiliser des conditions complexes (c'est-à-dire reposant sur plusieurs critères, qui sont combinés au moyen des opérateurs logiques « *et* », « *ou* », « *ne pas* ») pour permettre (ou interdire) certaines séquences d'actions.

Contexte
Lors de la conception d'algorithmes, il est souvent nécessaire de combiner plusieurs conditions pour décrire une solution à un problème.

Dans l'acquisition de la pensée informatique, il est par ailleurs important de prendre conscience que toute condition est un objet comme un autre (en informatique, on les appelle des booléens). Les objets booléens peuvent être constants (comme l'expression "2>1" qui est toujours vraie) ou variables ("il fait plus de 10° aujourd'hui" n'est pas toujours vrai).

Exemples
1. Si *x est strictement inférieur à 0* **et** *strictement supérieur à 24*, alors ce n'est pas une heure valide.

2. Si *la pâte est épaisse* **et** que *la température est inférieure à 10°C* **et** que *nous sommes un samedi* **ou** *un dimanche*, alors aller au magasin ouvert 7j/7 **et** acheter du lait.

3. Si *le robot arrive face à un mur* **et** qu'*il y a aussi un mur à sa gauche* **ou** que *la température est supérieure à 10°C,* alors le robot s'arrête sinon il tourne deux fois de 45°.

Compétence 4: Évaluer des objets ou des séquences d'actions

C 4.1 Comparer deux objets selon un critère donné

Définition
Étant donné deux objets et un critère de comparaison, être capable de comparer ces objets par rapport au critère.

Contexte
En informatique, il est courant de raisonner sur des objets disposant d'une relation d'ordre, comme c'est le cas par exemple des nombres (ordre sur les entiers ou les réels) ou des mots (ordre lexicographique). Être capable d'appliquer cette notion de relation d'ordre sur des objets divers (par exemple, ordre de taille, ordre d'âge entre personnes) fait partie de la pensée informatique.

Exemples

1. L'enseignant choisit le critère (taille, poids, largeur, etc.) et les apprenants doivent ordonner les objets par rapport à celui-ci.

2. Créer des cartes représentant des « objets » (personnes, animaux, choses que l'on trouve dans une cuisine) et demander aux apprenants de les regrouper en plusieurs catégories en fonction d'un critère qu'ils ont choisi (taille, nombre de pattes, choses qu'ils mangent, couleur de la peau, matériel), puis leur demander de modifier le critère et le groupement. On notera qu'ici, contrairement à l'exemple 1, il n'y a pas d'ordre (comme c'est le cas par exemple pour les couleurs).

3. Comparer des poids au moyen d'une balance de Roberval. Cette balance est un bon outil car elle permet de comparer deux valeurs à la fois (comme un processeur qui applique des opérations binaires). Elle permet d'éviter l'écueil de vouloir comparer toutes les valeurs en même temps comme ont souvent tendance à le faire des néo-apprenants.

C 4.2 Comparer deux séquences d'actions selon un critère donné

Définition
Être capable de comparer des séquences d'actions selon divers critères : en termes de lisibilité, de nombre de lignes, de temps d'exécution ou autre.

Contexte
Lorsqu'une solution (séquence d'actions) est trouvée, il est tentant de ne pas remettre en question la qualité de cette solution (c'est-à-dire de ne pas la comparer à d'autres solutions potentielles). La pensée informatique consiste aussi à savoir qu'un problème peut avoir 0, une ou plusieurs solutions, ainsi qu'à être critique vis-à-vis des solutions (ce qui nécessite des moyens de les comparer).

Exemples
1. Si vous souhaitez voyager de Nancy à Liège, plusieurs itinéraires sont possibles. Trouvez celui qui est le plus court, le plus rapide, le moins cher,…

2. Vous avez deux robots, un qui avance vite mais tourne lentement et l'autre qui tourne vite mais qui avance lentement. Parfois, selon le robot, un chemin plus long est plus efficace qu'un chemin plus court.
   Donnez plusieurs labyrinthes aux élèves présentant des caractéristiques différentes (par exemple, le nombre de mouvements de rotation requis) et laissez-les trouver le robot qui convient à tel ou tel labyrinthe.

3. Je sais faire du vélo, conduire et marcher. On me donne un voyage à faire, être capable de choisir le meilleur moyen de transport en fonction du stationnement, de la distance, de la circulation,….

C 4.3 Améliorer une séquence d'actions par rapport à un critère donné

Définition
Être capable de réfléchir à la possibilité d'améliorer une séquence d'actions par rapport à un critère et modifier cette séquence d'actions en conséquence.

Contexte
Á la suite de la compétence C 4.2, il est important, une fois qu'une solution (séquence d'actions) est connue comme n'étant pas optimale par rapport à un critère donné, de pouvoir rechercher des moyens de l'améliorer.

Exemples
1. Donner à des élèves un algorithme permettant à un robot de passer d'un point A à un point B, et leur demander de trouver un chemin avec moins de virages/sauts/étapes.
2. Fournir une recette avec des actions répétitives (par exemple ajouter un œuf plusieurs fois) et demander de rendre le code plus lisible (par exemple, en utilisant une boucle pour les œufs).

Compétence 5: Manipuler des représentations formelles

C 5.1 Représenter des objets ou séquences d'actions au moyen d'une représentation formelle

Définition
Être capable de représenter un objet ou une action en utilisant un mode de représentation défini avec précision (non ambigu).
Contexte
La pensée informatique inclut également la capacité à utiliser un langage défini avec précision (appelé généralement *langage* formel), et qui a la propriété d'être traité sans ambiguïté par un autre humain ou par une machine. Cette compétence 5.1 correspond donc à la capacité de représenter un mot, une image ou un nombre (ou une combinaison de ceux-ci) au moyen d'un code / langage (par exemple nouvel alphabet, symbole, etc.) défini avec précision. Ce choix de représentation est généralement appelé *encodage* en informatique.
Exemples

1. Utiliser différents pictogrammes ayant une signification précise (liée à une action) pour décrire une séquence d'actions (mouvements d'un robot sur une grille par exemple) énoncée oralement et informellement.

2. Associer un nombre (en utilisant par exemple la table ASCII) à un caractère du clavier.

3. Envoyer un message secret à quelqu'un en le chiffrant à l'aide de codes (c'est-à-dire de tables de correspondances).

C 5.2 Traduire des objets ou séquences d'actions entre représentations formelles

Définition
Être capable d'utiliser un autre langage / code formel pour représenter des objets ou séquences d'actions formellement définis.

Contexte
Cette compétence est une extension de la compétence C 5.2 ci-dessus, avec la contrainte supplémentaire que les systèmes de représentation des informations sont basés sur des règles définies avec précision (représentations formelles).

Exemples

1. Traduire le chiffre 2 de la notation arabe vers une autre représentation formelle (par exemple une représentation en chiffres romains, en langue des signes, etc.)

2. Utiliser un organigramme pour représenter les actions qui ont été décrites en pseudo-code.

3. Avoir deux langages différents pour un robot : d'une part "aller au nord, au sud, à l'est, à l'ouest" et, d'autre part, "avancer, tourner à gauche, tourner à droite". Demander aux enfants de traduire un programme de l'un de ces deux langages vers l'autre.

Compétence 6: Construire une séquence d'actions de manière itérative

C 6.1 Vérifier si une séquence d'actions atteint un objectif donné

Définition
Être capable de vérifier si une séquence d'actions donnée produit ou non le résultat escompté.

Contexte
Certaines séquences d'actions répondent à un problème particulier, d'autres à un problème général (dépendant de paramètres). Dans ce dernier cas, le fait de pouvoir vérifier pour quels paramètres une séquence d'action fonctionne (donne le résultat escompté) ou non, est une compétence recherchée ici. Cette compétence vise entre autres à développer chez l'apprenant l'habitude de vérifier et d'amender itérativement une séquence d'actions, en fonction de tests (paramètres et résultat associé).

Exemples

1. Prendre le mot "Toto", prendre la séquence d'actions qui permet de calculer le nombre de lettres du mot, et vérifier que celle-ci retourne 4..

2. Prendre un mot quelconque, une séquence d'actions qui permet de calculer le nombre de lettres du mot, et vérifier que le résultat retourné est celui attendu.

3. Soit une carte (grille 3x3), avec un point de départ (position (1,1)) et un point d'arrivée (position (3,3)) :

		Point d'arrivée
Point de départ		

Voici un algorithme :

Monter
Monter
Aller à droite
Aller à droite

Vérifiez que lorsque le robot est au point de départ (1.1), et que l'algorithme est exécuté (le robot s'est déplacé), que celui-ci le conduit au point de sortie.

C 6.2 Repérer des erreurs dans une séquence d'actions

Définition
En cas de séquence d'actions erronée, être capable de localiser la ou les erreurs.

Contexte
Les tests ne sont pas seulement utiles pour vérifier si une séquence d'actions atteint le résultat escompté. Lorsque ceux-ci sont bien choisis, ils permettent également de mieux comprendre pourquoi une séquence donnée est erronée et d'émettre des hypothèses sur les causes de l'erreur.

Exemples

1. Pour transformer 120 grammes en kilogrammes.
   La séquence suggérée : Prenez le nombre de grammes, reculez de deux chiffres et mettez une virgule → le résultat est 1,2kg. L'élève doit déterminer que ceci nécessite une correction.

2. Soit une carte (grille 3x3), avec un point de départ (position (1,1)) et un point d'arrivée (position (3,3)) :

		Point d'arrivée
Point de départ		

   Voici un algorithme :

   Monter
   Aller à gauche
   Aller à droite
   Aller à droite

Être capable d'identifier la ou les erreurs.

C 6.3 Corriger une séquence d'actions pour atteindre un objectif donné

Définition
À partir d'une séquence d'actions donnée, qui ne permet pas d'atteindre un but donné, être capable de modifier cette séquence d'actions pour atteindre le but.

Contexte
Une fois que les tests ont permis de localiser l'erreur, il est important de la corriger, éventuellement en réitérant le processus. À noter qu'il est important de garder à l'esprit que le fait de fournir le résultat escompté sur un exemple ne garantit pas que la séquence d'actions est correcte/généralisable, ni qu'il s'agisse de la ou des meilleures solutions (ex. en termes d'étapes intermédiaires).

Exemples

1. Supposons qu'il y ait un labyrinthe 3x3 avec (1,1) étant le point de départ et (3,3) celui d'arrivée :

		Point d'arrivée
Point de départ		

L'algorithme suivant ne permet pas de passer du point départ au point d'arrivée. Corrigez-le.

Aller à droite
Aller à gauche
Monter
Aller à droite

C 6.4 Étendre ou modifier une séquence d'actions pour atteindre un nouvel objectif

Définition
À partir d'une séquence d'actions donnée dont le résultat est connu (la séquence est complète par rapport à un objectif précédent), pouvoir la réutiliser et l'adapter pour atteindre un autre (nouvel) objectif.

Contexte

En informatique, pour réaliser des programmes complexes, il est judicieux de commencer par un objectif plus simple. On réalise le programme correspondant, simple mais fonctionnel, puis on l'étend peu à peu en ajoutant des fonctionnalités, tout en s'assurant que les modifications apportées ne vont pas compromettre le reste du programme. C'est ce qu'on appelle une démarche itérative.

A noter

Cette compétence diffère de C 2.2 car ici on ne part pas d'une séquence d'actions incomplète (qui n'atteint donc pas l'objectif souhaité) mais on enrichit progressivement une séquence d'actions fonctionnelle en se rapprochant peu à peu d'un objectif final.

Exemples

1. Après avoir programmé un dessin (par exemple un carré dans Scratch), l'utiliser dans un autre programme pour réaliser un dessin plus avancé (par exemple une rosace composée de carrés en légère rotation les uns par rapport aux autres).

2. Les apprenants partent d'un programme (qu'on leur donne ou qu'ils ont eux-mêmes programmé) réalisant un jeu de labyrinthe où le joueur doit déplacer un personnage de l'entrée à la sortie sans toucher les murs. Étendre ce programme pour ajouter un fantôme qui se déplace sur l'écran en passant à travers les murs, tel que si le fantôme touche le personnage on a perdu.