REVUE HYBRIDE DE L'ÉDUCATION

L'apprentissage de la programmation : Par quoi commencer et dans quel ordre?

Patrick Giroux,
Professeur au département des sciences de l'éducation et responsable du Laboratoire de formation et de recherche sur la littératie numérique de l'Université du Québec à Chicoutimi
Patrick_Giroux@uqac.ca

Maxime Boivin,
Conseiller pédagogique en mathématique (Commission scolaire des Rives-du-Saguenay)
Candidat à la maitrise en éducation (Université du Québec à Chicoutimi)
maxime.boivin@csrsaguenay.qc.ca

Vincent Porta Scarta,
Diplômé de la maîtrise en informatique
Université du Québec à Chicoutimi
vincent.porta-scarta1@uqac.ca

Résumé

Les technologies occupent une part de plus en plus importante dans nos vies et la programmation est une compétence qui gagne en importance. De nombreux enseignants du préscolaire, du primaire et du secondaire intègrent des activités d'apprentissage de la programmation dans leurs activités en lien, notamment, avec les mathématiques. Ce texte a comme objectif d'aider les enseignants à planifier des activités de formation à la programmation en définissant les principaux concepts à maîtriser pour apprendre à programmer, en suggérant un ordre fondé sur l'expérience et en donnant des exemples de stratégies ou d'activités pour initier les jeunes à la programmation.

Mots-clés : Apprentissage, programmation, robotique, pensée computationnelle, résolution de problème.

Mise en contexte

La place occupée par les technologies et le numérique ne cesse d'augmenter dans nos vies. Le ministère de l'Économie, de la Science et de l'Innovation du Québec (MESI, 2016) place d'ailleurs, dans son plan d'action en économie numérique, un axe d'intervention visant à acquérir les compétences numériques requises. Pour sa part, le ministère de l'Éducation et de l'Enseignement supérieur (MÉES, 2018) propose un ambitieux plan d'action numérique devant notamment permettre de « soutenir le développement des compétences numériques des jeunes et des adultes ». Lorsqu'il est question de numérique et d'acquisition de nouvelles compétences, la programmation est un sujet qui se présente rapidement et c'est sans surprise que le plan d'action numérique du MÉES y fait directement référence. Apprendre la programmation est une compétence qui, au fil des ans, gagne en importance dans notre société (Karsenti et Bugmann, 2017). Bien que l'école québécoise n'ait pas encore l'obligation de l'enseigner, de plus en plus d'initiatives à différents niveaux, autant dans les milieux scolaires qu'à l'extérieur de ceux-ci, voient le jour. C'est le cas de Kidscode[1], un organisme canadien qui vise à aider les jeunes et leur famille à développer leurs compétences numériques, du Réseau Technoscience[2] qui a mis en place des camps d'été d'apprentissage de la programmation et de Studio XP[3] qui offre des cours de création de jeux vidéo.

En général, lorsqu'il est temps d'associer la programmation à une discipline scolaire précise, les mathématiques semblent être la discipline qui est la plus naturellement reliée. Tout comme pour cette dernière, l'apprentissage de la programmation nécessite de se familiariser avec de nouveaux concepts. Les apprenants sont confrontés à une nouvelle logique et à une terminologie qui peut sembler imposante. Cette nouvelle logique constitue ce qu'on appelle la pensée computationnelle. Ce mode de pensée est présenté par Wing (2006) comme impliquant la résolution de problèmes ainsi que la compréhension du comportement humain en se basant sur des concepts fondamentaux des sciences informatiques. Cette école de pensée inclut des outils mentaux qui reflètent l'ampleur des sciences informatiques (comme l'abstraction, la décomposition en sous-problème ou en sous-tâches ou la recherche efficace d'informations pertinentes pour résoudre un problème par exemple) et permettent, entre autres, de résoudre des problèmes qui sembleraient autrement inaccessibles. Pour Wing (2006), il s'agit d'habiletés fondamentales que nous devrions tous développer et qui devraient être enseignées à tous les jeunes pour compléter leurs habiletés analytiques (avec la lecture, l'écriture et l'arithmétique).

[1] http://kidscodejeunesse.org/fr/a_propos.html
[2] https://technoscience.ca
[3] https://www.studioxp.ca/

L'apprentissage et l'acquisition de ce mode de pensée ne se font pas instantanément. Dans le cas de la programmation, une exposition trop rapide à plusieurs nouveaux concepts pourrait dérouter les élèves et les surcharger. Il est donc pertinent de se poser les questions suivantes : de quelle façon devons-nous initier les jeunes à la programmation et par quels concepts et apprentissages débuter ? C'est à ces questions que ce texte tentera de répondre en proposant une progression des apprentissages permettant de se lancer dans l'univers de la programmation. Il vise donc à servir de point de départ aux enseignants ou aux professionnels désirant débuter des activités de formation rattachées à la programmation.

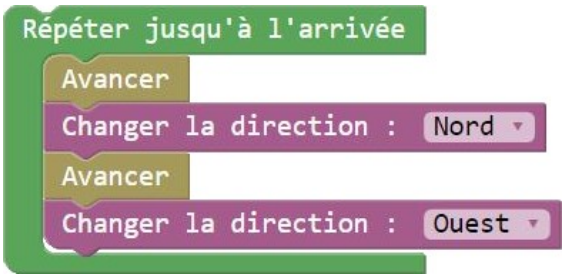
La progression présentée ici est inspirée des conclusions tirées du développement et de l'expérimentation de trois trousses d'activités visant à permettre aux jeunes du primaire et du début du secondaire d'être initiés à la programmation et de l'expérience gagnée en planifiant la mise en place de clubs de codage pour le secondaire[4]. Au fil des phases de développement et d'expérimentations, des constats concernant ces apprentissages sont apparus et c'est de ces derniers que la séquence suivante est inspirée. Cette progression semble idéale en raison de l'interdépendance de certains concepts ; il est facilitant de connaître certains concepts précis pour apprendre et utiliser ceux qui vont suivre. En effet, en programmation, les mêmes notions de base reviennent fréquemment et sont souvent employées simultanément. Une fois ces notions maîtrisées, le niveau de difficulté peut être augmenté graduellement et presque indéfiniment.

En plus de présenter un ordre logique, ce texte présente une définition de chacun des concepts qui sont présents dans la séquence. Les explications contenues dans ce texte sont indépendantes d'un langage informatique précis : elles se contentent de présenter une définition générale des différents concepts. De plus, la progression proposée distingue les apprentissages possibles dès le primaire de ceux qui devraient être réservés pour le secondaire.

[4] La création et la mise à l'essai de ces trousses d'apprentissage de la programmation et la planification des clubs de codage pour le secondaire ont bénéficié de l'aide financière du Programme NOVA Science du ministère de l'Économie, de la Sciences et de l'Innovation (MESI) du Québec et du Programme PromoScience du Conseil de Recherches en Sciences Naturelles et en Génie (CRSNG) du Canada.

Qu'est-ce que la programmation ?

Avant d'enseigner comment programmer, il convient d'abord d'initier les jeunes à la programmation elle-même. Programmer, c'est donner des instructions précises à un ordinateur, une machine ou un robot. Ces instructions sont transmises en respectant une syntaxe particulière, propre au langage informatique que l'on utilise. Il existe une panoplie de langages pouvant être utilisés et certains sont plus populaires[5] que d'autres. Parmi les langages les plus populaires actuellement, on retrouve : Java, C++, Python et Swift. Tous les langages ne sont pas égaux, notamment en termes de complexité et de difficulté. Il existe ainsi des langages utilisant la programmation visuelle (blocs) tel que Scratch ou Blockly, qui conviennent très bien pour les jeunes du primaire et même pour ceux du secondaire. Avec ce type de programmation, il suffit de joindre des blocs représentant chacun une commande afin de concevoir un programme complet. L'image suivante présente un exemple de programmation par blocs avec Blockly.



Une alternative pour les plus jeunes et les débutants : La programmation sans écran

Pour pouvoir programmer avec un ordinateur, il est souvent nécessaire d'utiliser une interface (logiciel) qui est en mesure de comprendre les instructions qui lui sont envoyées. Cette interface requiert elle-même une phase d'apprentissage et d'appropriation. Il faut aussi souvent savoir lire et écrire… Lors de l'initiation des plus jeunes, par exemple au préscolaire ou au premier cycle du primaire, la programmation sans écran est ainsi une avenue à considérer. Les grands concepts de la programmation et les fondements de la pensée computationnelle peuvent tous être acquis lors de différentes activités qui ne nécessitent pas d'ordinateur ou d'écran. Un exemple concret pourrait être une leçon ayant pour but d'apprendre aux jeunes comment fonctionne un robot. Le robot exécute les commandes qui lui sont envoyées. Dans une activité sans écran, les jeunes, jumelés en équipe de deux, pourraient à tour de rôle jouer le robot en effectuant les commandes ou le programmeur en formulant les commandes. En réalisant cette activité où les jeunes doivent donner des instructions à leur collègue robot pour qu'il se rende d'un point A à un point B et en les exposant à des défis de plus en plus importants (le trajet peut gagner en complexité, en longueur, devoir contourner des obstacles, etc.), il serait ainsi possible de les initier à plusieurs concepts importants en programmation. Des robots

[5] https://www.tiobe.com/tiobe-index/

très simples et très efficaces comme le Beebot[6] et le Ozobot[7] peuvent aussi être utiles pour initier jeunes et adultes à la programmation sans recourir à des écrans.

Une progression conceptuelle pour l'initiation à la programmation
Il apparaît d'abord pertinent de se familiariser avec certains mots de vocabulaire qui peuvent être entendus fréquemment. Certains de ces termes sont souvent utilisés dans des contextes similaires et peuvent ainsi être perçus, dans certains cas, comme des synonymes. Les premières activités de programmation présentées aux jeunes devraient couvrir cette terminologie. La complexité inhérente à la terminologie permet, par ailleurs, de structurer les apprentissages et suggère une progression dans les apprentissages à faire, apprentissages que l'on peut débuter dès le préscolaire.

Commande, bogue et une stratégie de résolution de problème
Une commande est une instruction que l'on donne à un ordinateur comme : « Fais quelque chose ». Dans le cas d'un robot, une commande pourrait être : avance de 2 mètres ou tourne à droite de 90 degrés. En réalité, une simple addition de deux valeurs constitue une commande (ex. : additionne 2+2). Les commandes doivent donc être perçues comme étant des instructions claires et précises que l'on donne à une machine numérique.

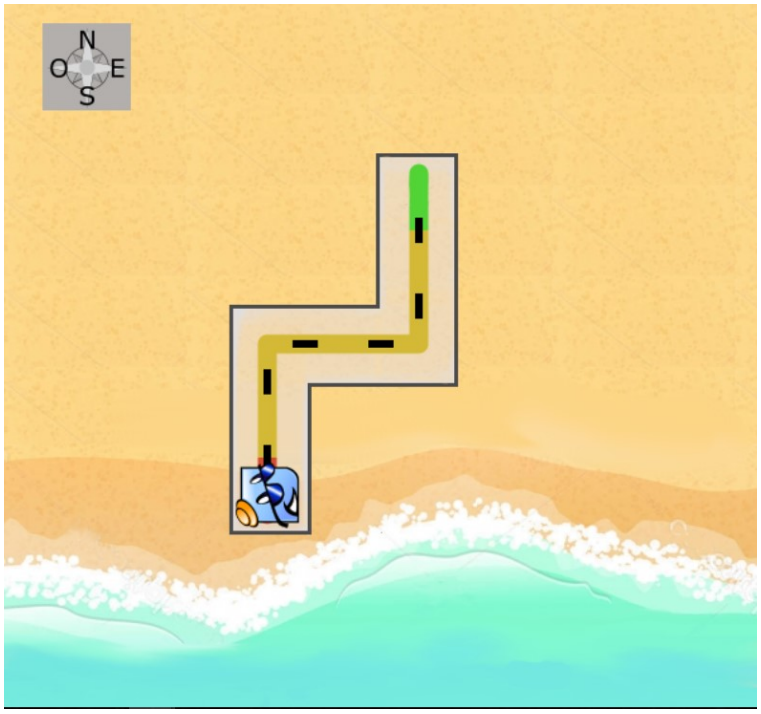
Lors d'une initiation à la programmation, demandez aux apprentis de tester ou de valider leurs commandes une à la fois (ex. : Je demande au robot de tourner à droite et j'attends de voir ce qui va en résulter lors de l'exécution de la commande.). Ils pourront ainsi apprendre à anticiper les conséquences associées aux commandes et comprendre qu'elles se succèdent ou s'additionnent dans l'ordre exact de leur émission. Dans tous les cas, avec ou sans écran, qu'ils travaillent avec des robots ou des ordinateurs, les apprentis programmeurs auront la chance de recevoir une rétroaction immédiate qu'ils doivent absolument apprendre à observer, analyser et utiliser pour s'autoévaluer et s'ajuster. Cette rétroaction leur permettra de rapidement réaliser s'il y a ou non bogue dans le programme, c'est-à-dire un défaut de conception ou une commande erronée. L'enseignant devrait accompagner le jeune et l'encourager à s'engager dans le processus impliquant l'observation, l'analyse, l'évaluation et la correction nécessaire au développement de la pensée computationnelle et très proche de la pensée mathématique et scientifique.

Programme
Il s'agit d'un mot qui est utilisé dans plusieurs circonstances. De façon générale on peut dire qu'il représente un ensemble complet de commandes

[6] https://www.bee-bot.us/

[7] https://ozobot.com/

permettant d'exécuter une ou plusieurs tâches. Rapidement, les apprentis programmeurs comprendront le concept de commandes et pourront créer des programmes en additionnant des commandes. Par exemple, dans la situation imagée ici, les commandes qui permettent de guider le personnage jusqu'au bout du chemin sont « avance, avance, tourne vers l'est, avance, avance, tourne vers le nord, avance et avance ». Cet ensemble de commandes est un programme. Avec de la pratique, les apprentis arriveront à écrire des programmes de plus en plus longs et complexes.

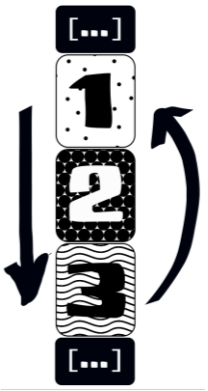


Code

Le mot « code » est majoritairement utilisé pour représenter le fichier ou le texte contenant les commandes d'un programme. Il est fréquent d'entendre parler de « lignes de code » qui désignent les lignes de commandes qui composent ledit programme. Dans le langage courant, le verbe coder peut aussi être utilisé pour définir l'action de « programmer ».

Boucles

Les boucles sont des commandes qui sont exécutées plusieurs fois (ex. : avance de 4, tourne à droite de 90 degrés et recommence 3 fois). Au lieu d'écrire les mêmes lignes de commandes de façon répétée, il suffit de les indiquer une seule fois et de préciser à l'ordinateur le nombre d'exécutions désiré. Les boucles sont d'une grande utilité parce qu'elles permettent de simplifier les programmes. Il s'agit d'un concept plutôt simple que l'on peut apprendre facilement dès le premier cycle du primaire, mais qui peut aussi rapidement gagner en complexité en y intégrant d'autres concepts comme, par exemple, les conditions (présentées plus bas).



Algorithme

Un algorithme est un ensemble de commandes permettant de réaliser une tâche particulière. À chaque fois que le même algorithme est utilisé, le même résultat devrait être obtenu. L'utilisation du mot algorithme peut être semblable à celle du mot programme dans plusieurs situations. Cependant, l'algorithme réfère plus souvent à un petit nombre de commandes permettant de réaliser une tâche précise contrairement aux programmes qui peuvent être beaucoup plus imposants. Il arrive aussi que les algorithmes soient des boucles fréquentes auxquelles on a donné un nom que l'on utilise pour que la boucle s'exécute. Par exemple, la boucle décrite plus haut pourrait aussi être l'algorithme « fait un carré de taille 4 ». Chaque fois que l'on demanderait à l'ordinateur ou au robot de « faire un carré de taille 4 » il ferait la boucle décrite précédemment.

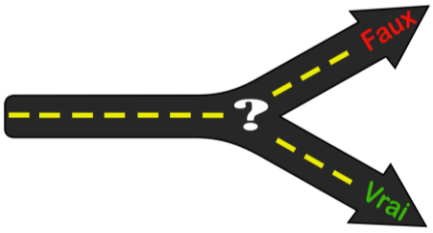
Dans la vie quotidienne, nous sommes souvent confrontés à des algorithmes. Par exemple, « faire pocher des œufs » est un algorithme qui résume plusieurs étapes (ex. : porter 4 tasses d'eau à ébullition, ajouter une cuillère à table de vinaigre, etc.) qui mènent toujours au même résultat, peu importe que l'on prépare des œufs à la bénédictine ou simplement des œufs pochés pour manger avec des toasts.

Pour initier les jeunes aux algorithmes, une bonne stratégie est de les inciter à créer des algorithmes qui résument plusieurs commandes et permettent de simplifier un programme ou de le rendre plus court. Par exemple, dans la situation imagée lors de la présentation du concept de programme, les commandes qui permettent de guider le personnage jusqu'au bout du chemin étaient « avance, avance, tourne vers l'est, avance, avance, tourne vers le nord, avance et avance ». On pourrait par contre utiliser un algorithme pour simplifier. En utilisant l'algorithme « avance » qui veut toujours dire « avancer le robot de 2 cases » dans mon programme, j'obtiens le programme suivant : « avance, tourne vers l'est, avance, tourne vers le nord et avance » qui permet lui aussi d'atteindre la fin. L'algorithme « avance » remplace un ensemble de 2 commandes. Le

processus de création d'algorithmes nécessite d'analyser son programme, de repérer les successions de commandes qui se répètent et de trouver un moyen de les combiner.

Conditions

Une condition est une commande ou un algorithme qui ne s'exécute que lorsqu'un énoncé est vrai. Par exemple, je pourrais demander à un robot d'avancer et de ne s'arrêter que lorsqu'il rencontrera une ligne blanche. Il n'effectuera la commande « s'arrêter » que lorsqu'il aura rencontré une ligne blanche et pas avant. Comme les boucles, les conditions sont aussi accessibles pour les jeunes dès le premier cycle du primaire, mais ce concept peut rapidement gagner en complexité. Au moment de concevoir la condition, il est par exemple possible de combiner plusieurs éléments en utilisant des opérateurs logiques comme le ET et le OU. On se situe alors probablement plus à la fin du deuxième, voire même au troisième cycle du primaire. Le ET aura pour effet d'exiger que tous les éléments placés dans la condition soient vrais, et ce même s'il y en a une cinquantaine. Dès qu'un élément sera faux, la condition ne sera pas respectée et les commandes contenues à l'intérieur ne seront pas exécutées. L'opérateur OU aura pour effet d'exiger qu'au moins un des éléments placés dans la condition soit vrai. Dès qu'un d'entre eux sera vrai, même s'il y en a une cinquantaine et que les autres sont tous faux, les commandes à l'intérieur seront exécutées.



On peut rapidement initier les apprentis programmeurs aux conditions sans avoir recours à des écrans ou du matériel complexe. Un exemple pourrait être un jeu où des mots sont associés à des mouvements. Si l'enseignant dit « saute » les jeunes doivent faire un saut sur place. Si l'enseignant dit « main » les enseignants doivent lever la main. Si l'enseignant dit « pomme » les élèves se placent en boule par terre. Pour faire plus simple, l'enseignant peut concevoir des énoncés plus explicites comme « Si mon nom est Coralie, je lève la main » ou « Si j'ai 8 ans, je viens à l'avant de la classe » que les élèves devraient exécuter. Dans cette activité les jeunes jouent le rôle de l'ordinateur vérifiant si les conditions sont vraies ou fausses et exécutent ou non la commande.

Les boucles sont souvent utilisées avec une condition qui indique à l'ordinateur quand arrêter de répéter la même série de commandes. Lorsque le nombre de boucles à exécuter peut varier d'une exécution à

l'autre, il est souvent plus approprié de placer une condition d'arrêt (ex. : Exécute cet algorithme **tant que** l'utilisateur n'a pas entré le mot « terminé ».). Cette possibilité est pratique puisqu'ainsi il n'est pas nécessaire de redémarrer régulièrement le programme. Il suffirait seulement d'indiquer le moment où on veut l'arrêter. Dans notre exemple précédent, nous aurions pu utiliser une boucle avec une condition plutôt que l'algorithme « avance » qui voulait toujours dire « avancer le robot de 2 cases » dans le programme créé. En effet, nous aurions pu dire au robot « avance jusqu'à la prochaine intersection » ou « avance tant que c'est possible ». La boucle combinée à une condition permet de gérer une certaine part d'incertitude. Par exemple, si je ne sais pas de combien de cases le robot doit avancer avant de tourner, les boucles proposées précédemment permettraient tout de même de composer un programme sans bogue.

Les variables

Les variables constituent un incontournable pour toute personne voulant devenir un programmeur sérieux. C'est aussi un concept plus complexe à appréhender que nous recommandons d'aborder seulement après une bonne initiation aux concepts précédemment introduits. Il est aussi souvent plus facile de l'introduire lors de l'apprentissage de la programmation textuelle (à l'ordinateur) et avec un langage précis (parce que chaque langage a ses particularités).

En mathématiques, les variables sont souvent vues comme étant des lettres pouvant prendre plusieurs valeurs. En informatique, l'idée diffère un peu. Les variables peuvent être perçues comme des tiroirs contenant une information sur lesquels il est possible d'apposer une étiquette. L'étiquette représente le nom de la variable. Un exemple pourrait être un pointage dans un jeu, la variable « pointage » conserverait cette valeur en mémoire et pourrait l'actualiser selon le déroulement du jeu. Un autre exemple serait un programme désirant mettre en mémoire le prénom, le nom et l'âge d'une personne. Dans ce contexte trois variables seraient nécessaires, une pour chaque information.

Types de variables
Le contenu d'une variable peut être de différents types selon le langage utilisé. En général, on retrouve : les nombres entiers, les nombres décimaux et les chaînes de caractères (mots). Ainsi, une variable peut mettre en mémoire l'information uniquement associée à son type. Il est cependant possible d'effectuer des conversions d'un type vers un autre selon les besoins (changer un nombre décimal en nombre entier, ou encore changer un nombre entier en chaîne de caractère). La particularité des chaînes de caractères est d'être considérée uniquement comme du texte par l'ordinateur. Ainsi, si on place un nombre dans une chaîne de caractère, l'ordinateur ne sera pas en mesure de réaliser un calcul mathématique avec ce dernier puisqu'il le perçoit comme un symbole et non pas comme un nombre.

Les opérations sur les variables
Une fois familiarisé avec les variables, il est pertinent de savoir quelles opérations peuvent être exécutées sur ces dernières. Il est possible de réaliser les opérations mathématiques de base sur les nombres (+, -, ×, ÷). Il est aussi possible d'effectuer des vérifications avec les opérateurs suivants : (<, ≤, >, ≥). Bien qu'il ne soit pas possible d'effectuer des opérations mathématiques à proprement parler sur les chaînes de caractères, l'addition de deux variables de ce type aura pour effet de les fusionner dans certains langages de programmation.

S'initier au concept de variable
Le côté abstrait du concept de variable le rend difficile à aborder avant la fin du primaire, voir le début du secondaire et seulement après avoir été initié et après avoir bien compris les concepts précédents.

Pour une initiation sans écran au concept de variable, il est possible de réaliser une activité dans laquelle les jeunes devraient placer une information dans des boîtes représentant chacune une variable. Pour chacune d'elles, les jeunes devraient déterminer le type de cette variable selon l'information qui ira à l'intérieur. Une boîte portant le nom « Âge » devrait être associée au type : nombres entiers (ou seulement nombre pour les plus jeunes). Une autre boîte ayant pour nom « Ma couleur préférée » devrait être associée au type : chaîne de caractère (ou « mot » pour les plus jeunes). Après avoir identifié le type de chacune des boîtes, les jeunes seraient invités à placer dans chacune d'elles l'information leur correspondant, en l'écrivant sur un bout de papier ou en plaçant un objet à l'intérieur (dans le cas de jeunes ne sachant pas encore écrire). Au final, les jeunes seraient ainsi déjà initiés au concept de variable qui représente une boîte dans laquelle on peut placer une information d'un certain type. Ensuite, il serait possible d'explorer les opérations et les combinaisons possibles sur les variables en imaginant des commandes, par exemple :

- Additionnons les âges des élèves de la classe.

- Vérifions si l'âge de Mathieu est supérieur à celle de Thomas.
- Peut-on multiplier l'âge de Corinne par sa couleur préférée ?

Combiner « variables » et « conditions »

Les concepts de variables et de conditions peuvent ensuite être combinés pour amener la programmation à un autre niveau. Un exemple de programme simple avec des variables et des conditions pourrait être une calculatrice simple dans laquelle l'utilisateur doit indiquer deux nombres et dire à l'ordinateur ce qu'il doit faire parmi les choix suivants : addition, soustraction, multiplication ou division. L'ordinateur exécutera le cas correspondant à ce qui lui est demandé. Voici concrètement à quoi pourrait ressembler un tel programme :

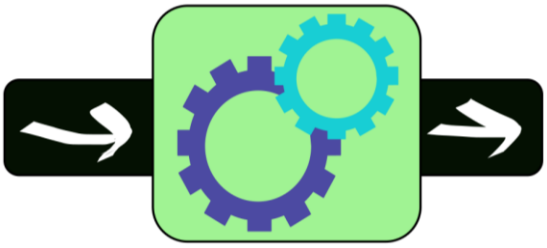
Demander d'entrer un premier nombre et le placer dans une variable : nombre1
Demander d'entrer un deuxième nombre et le placer dans une variable : nombre2
Demander d'entrer le nom de l'opération et le placer dans une variable : opération
Si opération est égal à « addition »
    Fait : nombre1 + nombre2
Sinon, si opération est égal à « soustraction »
    Fait : nombre1 - nombre2
Sinon, si opération est égal à « multiplication »
    Fait : nombre1 * nombre2
Sinon, si opération est égal à « division »
    Fait : nombre1/nombre2
Sinon
    Affiche : Ce n'est pas une opération valide

Dans cet exemple précis, dès que l'ordinateur reconnaîtra la valeur « vrai » pour un des cas, il l'exécutera et ne vérifiera pas les cas suivants. Le dernier cas est placé pour les situations où l'utilisateur entrerait un mot différent de ceux désirés. Puisque les quatre choix possibles sont tous présents dans les premières conditions, le dernier cas s'exécutera uniquement s'il ne s'agit pas d'une opération valide.

Fonctions

En mathématiques, les fonctions sont associées à une relation entre deux variables dans lesquelles, lorsqu'on place une valeur de la variable indépendante, on obtient une seule valeur de la variable dépendante. Mathématiquement parlant, lorsqu'on place toujours la même valeur dans une fonction, on obtient toujours le même résultat. En informatique, la définition diffère un peu, bien qu'elle soit similaire sur l'idée de fond. Il s'agit

d'un sous-programme ou d'un algorithme, qui est souvent utilisé et auquel on donne un nom afin de raccourcir le programme. Lorsqu'elle est utilisée, la fonction effectue toujours la même tâche ou donnera toujours le même résultat si les variables sont identiques.



Encore une fois, il est possible d'initier les apprentis programmeurs au concept de fonction sans avoir recours à un écran en associant les fonctions à des routines. Ainsi les jeunes se retrouvent à utiliser déjà des fonctions sans le savoir. Un exemple concret pourrait être un parent qui demande à son enfant d'aller se coucher. Aller se coucher est une fonction qui comprend plusieurs étapes : aller à la salle de bain, brosser ses dents et aller se placer dans son lit. Les parents n'ont pas besoin de répéter chacune des étapes, ils ont simplement besoin de dire qu'il faut « aller se coucher ». Il est donc possible d'envisager plusieurs activités mettant à profit les fonctions informatiques. Dans le cas de jeunes jouant au robot, un nom particulier (exemple : « trajet spécial » pourrait signifier une série de mouvements comme : avance d'un pas, tourne à droite, avance de deux pas et tourne à gauche.).

En ce qui concerne la programmation sur les ordinateurs, la fonction est le plus souvent écrite en aparté du programme ou « à côté » du programme dans une bibliothèque ou une « classe » (le concept de classe est présenté plus bas) et le programme ne fait que référence à la fonction. Les fonctions peuvent avoir besoin de paramètres en entrée (des variables) et peuvent retourner des valeurs. Dans certains cas, elles n'ont cependant besoin de rien et ne retourneront rien. Dans le cas d'un robot, une fonction pourrait être conçue afin de le faire avancer d'une distance fixe (exemple : 10 cm). Ainsi, pour faire avancer le robot de 10 cm, il serait seulement nécessaire d'insérer la fonction « avancer » dans le programme, fonction qui serait disponible dans un répertoire ou une bibliothèque intégrée au robot ou accessible par ce dernier. Un exemple de fonction nécessitant des paramètres pourrait être une fonction « moyenne de trois nombres ». Pour être exécutée, elle aurait besoin d'avoir en entrée trois nombres. À la fin de son exécution, elle retournerait un seul nombre (la moyenne des trois premiers). Voici l'exemple :

Fonction : « moyenne de trois nombres » utilisant les variables nombre1, nombre2 et nombre3
moyenne = (nombre1 + nombre 2+ nombre3) ÷ 3
retourner la valeur de la moyenne

L'utilité des fonctions est la même que pour les boucles ou les algorithmes, il s'agit de simplifier le programme final. En programmation, lorsqu'une série de commandes a été créée pour faire une tâche précise, les programmeurs cherchent à éviter d'avoir à entrer les mêmes commandes et utilisent donc des fonctions, des boucles ou des algorithmes pour sauver du temps.

Les classes

Les classes sont les derniers des concepts rencontrés dans cette progression des apprentissages. Elles viennent de pair avec les fonctions et permettent de créer des programmes plus complexes. Il est d'ailleurs recommandé de limiter l'utilisation des classes aux élèves plus âgés (fin primaire et début secondaire). De plus, l'exploitation des classes est plus conviviale avec la programmation textuelle sur ordinateur.

Les classes sont utilisées lorsqu'il est question d'objets ayant des caractéristiques précises. En fait, elles sont la façon que les programmeurs ont de permettre à un ordinateur de se représenter ces objets. Elles contiennent des caractéristiques (variables) et des actions possibles (fonctions). Prenons l'exemple d'une bouteille d'eau. Au niveau des caractéristiques, la bouteille a une hauteur, une largeur, une capacité et plusieurs autres caractéristiques. Au niveau des actions, la bouteille d'eau peut être remplie ou vidée. Bien qu'il y ait d'autres caractéristiques et potentiellement d'autres actions qu'il soit possible d'envisager pour une bouteille d'eau, les programmeurs conserveront seulement celles qui leur seront utiles pour réaliser la tâche qu'ils veulent accomplir. Dans un jeu où il faudrait remplir une bouteille jusqu'à ce qu'elle soit pleine, les caractéristiques requises seraient la capacité de la bouteille et son action serait le fait de remplir cette dernière.

Dans cet exemple, l'ordinateur sait donc à quoi ressemble et à quoi sert une bouteille d'eau. Par la suite, pour introduire un ou plusieurs de ces objets dans son programme, le programmeur indiquera à l'ordinateur d'utiliser un élément de la classe « bouteille d'eau » avec des valeurs précises pour chacune des caractéristiques. Exemple :

- Petite bouteille est un objet de la classe bouteille d'eau qui a une capacité de 300 ml.
- Grande bouteille est un objet de la classe bouteille d'eau qui a une capacité de 1000 ml.

Apports et perspectives

Voici donc l'éventail des concepts à acquérir lorsque l'on débute en programmation. L'ordre suggéré pour leur apprentissage apparaît logique étant donné les connaissances nécessaires pour chacun des nouveaux concepts.

Il est possible de commencer très tôt l'apprentissage de ces concepts en utilisant des activités « sans écran ». Ainsi, même si les jeunes n'étaient pas en mesure concrètement de programmer sur un ordinateur, ils seraient familiers avec la terminologie et la logique sous-jacentes. L'apprentissage sans écran rend beaucoup plus facile pour eux l'apprentissage de la programmation sur un ordinateur par la suite. Surtout, avec un encadrement adéquat, ils développeront leur pensée computationnelle et seront mieux outillés pour comprendre le fonctionnement des machines numériques, ainsi que pour résoudre des problèmes de plus en plus complexes.

En ce qui a trait à la programmation sur ordinateur, avec cette séquence, il ne reste plus qu'à faire un choix de langage de programmation et à se lancer dans l'apprentissage, étape par étape, des syntaxes précises pour chacun des concepts présentés. Il est important de bien expérimenter et de pratiquer un concept avant de se lancer dans un autre. Les jeunes vont rapidement se sentir déboussolés et vont facilement mélanger plusieurs notions s'ils sont exposés à un trop grand nombre de nouveaux concepts dans un court laps de temps. L'expérience acquise dans la création, la mise à l'essai et l'amélioration de plusieurs trousses d'apprentissage de la programmation nous a maintes fois démontré l'importance de ne pas progresser trop vite et de procéder par étapes en augmentant lentement le niveau de difficulté, en incluant un à un les concepts et en combinant chaque nouveau concept introduit avec ceux déjà maîtrisés pour améliorer des programmes déjà conçus ou résoudre de nouveaux problèmes en composant des programmes plus complexes.

Quant à l'inclusion de la programmation dans les établissements scolaires, les portes d'entrée sont nombreuses. Au sens le plus large, la quasi-totalité des compétences transversales peut être mobilisée dans des activités concernant l'apprentissage de la programmation. Au niveau préscolaire, l'apprentissage de la programmation en lien avec la robotique est particulièrement intéressant pour travailler la latéralité et le positionnement dans l'espace. En ce qui a trait au domaine de la mathématique, la première compétence visant la résolution de situations problèmes s'y rattache très bien. Il est aussi possible d'envisager des activités de programmation liées à tous les champs à l'intérieur du domaine de la mathématique, du début du primaire jusqu'à la fin du secondaire. En bref, les possibilités sont nombreuses et il y a place à l'innovation. De nombreuses communautés de partage se sont par ailleurs créées sur le Web autour de logiciels comme Scratch ou Blockly, de robots comme le

BeeBot ou le Ozobot. N'hésitez pas à explorer ces communautés, à poser des questions et à échanger ! Au plaisir de vous y croiser !

Références

Karsenti, T., & Bugmann, J. (2017). Une brève histoire des technologies en éducation. Dans T. Karsenti, & J. Bugmann (Éds.), *Enseigner et apprendre avec le numérique* (p. 15-40). Montréal : Les Presses de l'Université de Montréal.

Ministère de l'Économie, de la Science et de l'Innovation (2016). *Plan d'action en économie numérique*. Québec : Gouvernement du Québec.

Ministère de l'Éducation, du Loisir et du Sport (2007). *Programme de formation de l'école québécoise : enseignement secondaire, deuxième cycle*. Québec : Gouvernement du Québec.

Ministère de l'Éducation du Québec (2006). *Programme de formation de l'école québécoise : Enseignement secondaire, premier cycle*. Québec : Gouvernement du Québec.

Ministère de l'Éducation et de l'Enseignement Supérieur (2018). *Plan d'action numérique en éducation et en enseignement supérieur.* Québec : Gouvernement du Québec.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*, 33–35