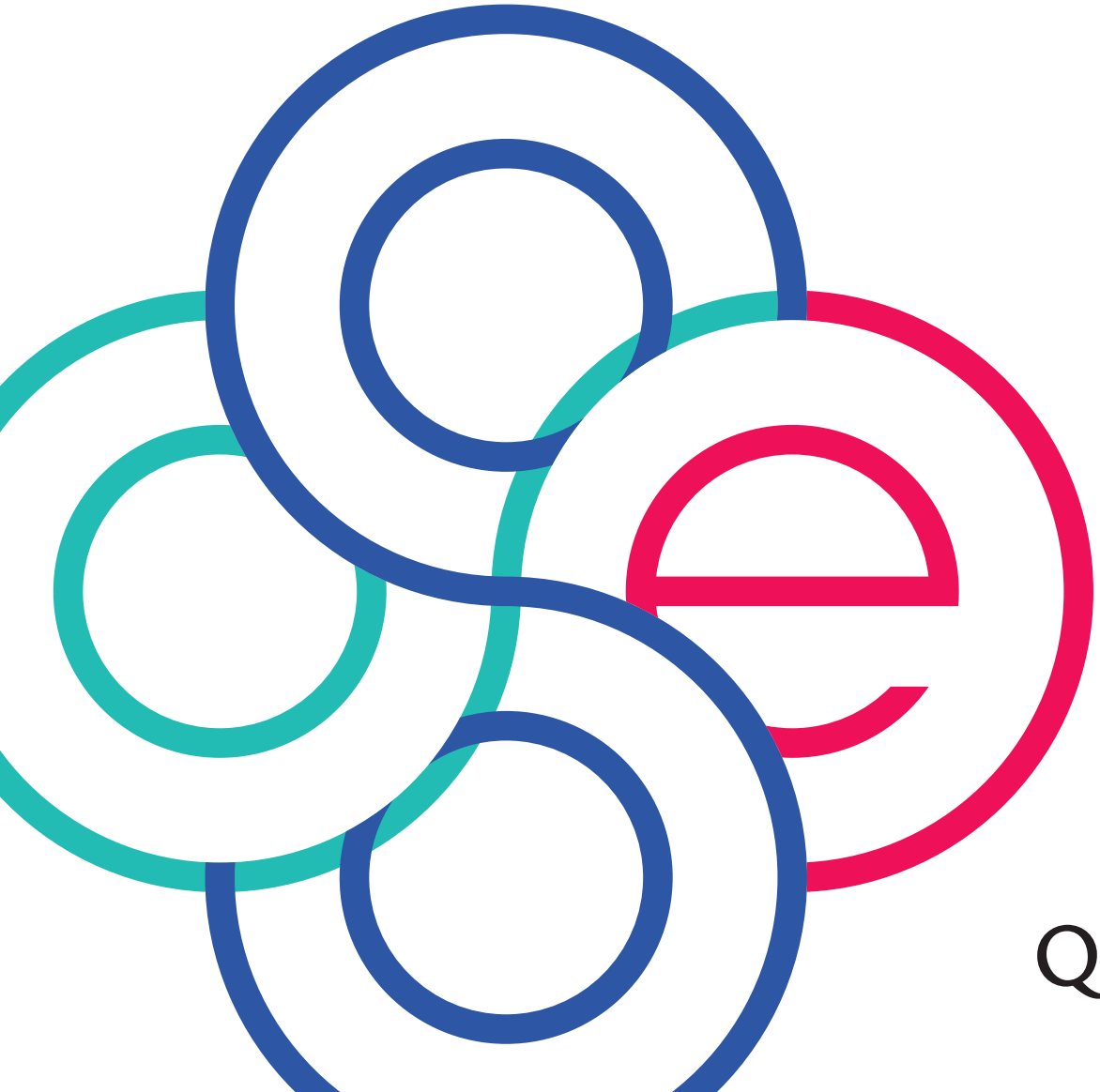
Novembre 2020

CONSEIL SUPÉRIEUR DE L'ÉDUCATION

État des connaissances sur l'apprentissage et la pratique de **la programmation informatique en contexte scolaire**

Études et recherches

Document préparatoire pour le ***Rapport sur l'état et les besoins de l'éducation 2018-2020*** du Conseil supérieur de l'éducation.



Québec

Le Conseil supérieur de l'éducation a confié la coordination de la préparation et de la diffusion du présent document de recherche à sa présidence. Ce document et les positions qu'il peut contenir n'engagent pas le Conseil ni ses instances consultatives.

Rédaction et recherche
Hugo Couture, agent de recherche

Révision linguistique
Mélissa Guay

Comment citer cet ouvrage :
Couture, Hugo (2020). *État des connaissances sur l'apprentissage et la pratique de la programmation informatique en contexte scolaire*, Études et recherches, Québec, Conseil supérieur de l'éducation, 17 p.

Dépôt légal
Bibliothèque et Archives nationales du Québec, 2020
ISBN : 978-2-550-87920-6 (version PDF)

Ce document a été produit dans l'esprit d'une rédaction épicène, c'est-à-dire d'une représentation équitable des femmes et des hommes.

Avis aux lectrices et aux lecteurs

Pour accomplir sa mission, le Conseil supérieur de l'éducation peut mener ou faire effectuer des études et des recherches qu'il juge nécessaires à la préparation de ses avis et de ses rapports. Le cas échéant, il peut décider de rendre publiques de telles productions s'il estime que la richesse et l'utilité potentielle des renseignements colligés le justifient. C'est dans cette perspective que le Conseil rend public le présent document.

Cette publication reprend les éléments d'un mandat accompli dans le cadre de la production du Rapport sur l'état et les besoins de l'éducation 2018-2020. Ce rapport présentera une réflexion sur le numérique en éducation et sur les responsabilités nouvelles que ce contexte implique pour le système éducatif.

Déposé le 10 mai 2019 à la 4e réunion du Comité du rapport sur l'état et les besoins de l'éducation, le document original visait à ce que le comité et le Conseil prennent connaissance d'un état des lieux sur l'enseignement de la programmation informatique.

Table des matières

Introduction

L'idée d'enseigner la programmation informatique aux enfants en contexte scolaire remonte aux années 1960 (Papert, 1980). Le langage LOGO, développé par Seymour Papert et ses collègues du Massachusetts Institute of Technology (MIT), a été créé pour enseigner la programmation en tant que compétence et outil de résolution de problèmes. Papert concevait LOGO à la fois comme environnement de codage et environnement d'apprentissage des mathématiques et de développement de la pensée procédurale (Grover et Pea, 2013; Pea, 1983). « Programmer un ordinateur ne signifie rien de plus ni moins que de communiquer avec lui dans un langage que l'utilisateur humain et lui peuvent "comprendre". Et l'apprentissage des langues est l'une des choses que les enfants font le mieux. » (Papert, 1980, p. 5-6.) L'enthousiasme entourant la programmation s'est propagé rapidement, car on lui prêtait de nombreuses vertus par rapport à l'acquisition de diverses compétences cognitives, dont la planification et la résolution de problème.

À la suite de visites dans des écoles primaires où ce langage était enseigné dans les années 1980, le Conseil supérieur de l'éducation (CSE) estimait qu'il fallait « généraliser le plus possible l'accès au langage LOGO et à l'utilisation du traitement de texte. Tandis que le traitement de texte permettra aux élèves de produire des textes dans toutes les disciplines du régime pédagogique, le langage LOGO les aidera à se familiariser avec la logique de l'ordinateur. Le langage LOGO est un outil dont les élèves peuvent se servir pour mettre en forme, utiliser et évaluer des procédures informatiques. Ces exercices sont de nature à favoriser la rigueur de la pensée de l'élève et à permettre les premiers apprentissages de la méthode scientifique » (CSE, 1984, p. 8).

Dans les années 1990, l'intérêt pour la programmation s'essouffle au profit de l'enseignement de nouvelles compétences informatiques, comme le traitement de texte et la recherche sur Internet (Moreno-León, Robles et Román-González, 2016). Il faudra attendre le nouveau millénaire pour que surgisse un regain d'intérêt pour le codage dans le cadre du développement des compétences dites du 21e siècle. L'intérêt pour ces compétences « *is due, in large part, to numerous reports and whitepapers published in the last decade, and well-organized, well-funded education initiatives launched by non-governmental*

and not-for-profit organizations » (Howard, 2018, p. 1)[1]. L'influence de ces travaux et l'arrivée d'outils de programmation adaptés aux plus jeunes apprenantes et apprenants conduiront plusieurs pays à réintroduire cet apprentissage dans leur programme d'enseignement primaire, notamment l'Estonie, la Grèce, l'Angleterre, l'Australie, ainsi que certains États américains et provinces canadiennes.

Ce retour de la programmation serait attribuable à trois raisons (Popat et Starkey, 2019). La première répondrait à un impératif économique devenu leitmotiv : préparer la future main-d'œuvre pour l'industrie des nouvelles technologies. La deuxième évoque un idéal entrepreneurial dans lequel les étudiantes et les étudiants sont appelés à devenir des « producteurs d'innovations ». La troisième raison veut que la maîtrise des bases de la programmation et l'apprentissage de la pensée computationnelle (ou pensée informatique) soient des incontournables pour le développement des compétences du 21e siècle.

1 « dû en grande partie aux nombreux rapports et livres blancs publiés au début des années 2000, et à des initiatives de réformes éducatives organisées et financées par des organisations non gouvernementales et des organisations à but non lucratif » (notre traduction).

1 L'enseignement de la programmation ici et ailleurs

Plusieurs chercheuses et chercheurs se sont intéressés aux expériences internationales en matière de robotique éducative et d'enseignement de la programmation en contexte scolaire. Moreno-León, Robles et Román-González (2016) regroupent les différentes façons d'implanter l'apprentissage de la programmation selon trois catégories.

Certains pays fondent leur approche éducative sur des études ou des **représentations sociales** qui anticipent une éventuelle rareté des ressources humaines dans le domaine des technologies. La programmation apparaît pour ces pays comme une fin en soi.

D'autres juridictions incorporent la programmation à l'enseignement des technologies de l'information et de la communication (TIC) ou à des disciplines associées aux sciences informatiques. Par exemple, le nouveau curriculum français intègre la programmation dans les cours de sciences et technologies du primaire, de technologie au secondaire et de mathématiques aux deux ordres d'enseignement.

Un dernier groupe accorde davantage d'attention aux aspects pédagogiques de l'apprentissage de la programmation. La programmation y est utilisée comme moyen pour apprendre dans d'autres disciplines. C'est le cas en Nouvelle-Zélande, pays qui a adopté une approche interdisciplinaire. C'est de ce dernier modèle que semble vouloir s'inspirer le Québec.

Il n'existe que très peu de données permettant de distinguer les approches les plus efficaces pour l'enseignement et l'apprentissage des compétences dites du 21e siècle (Lamb et Maire, 2017), dont fait souvent partie la programmation dans les nouveaux curriculums. Il n'y a pas non plus de consensus sur l'âge idéal pour introduire les rudiments de la programmation ni sur la nature de cet apprentissage. Certains pays initient leurs élèves dès le primaire (Estonie et Israël), d'autres attendent au secondaire (Irlande et Autriche). Certaines juridictions ne la mettent au programme que lorsqu'elle est liée aux choix vocationnels des apprenantes et des apprenants (République tchèque). Enfin, certains pays n'offrent la programmation qu'en option (Lituanie et Pologne). Ces différences dans la manière d'implanter l'apprentissage de la programmation s'expliqueraient en partie par «*a lack of empirical research on the topic that could provide evidence to policy makers*» (Moreno-León, Robles et Román-González, 2016,

p. 284)[2]. Malgré cette absence de consensus, plusieurs chercheuses et chercheurs de l'éducation et de l'informatique estiment que le contexte technologique actuel justifie l'acquisition de compétences numériques de base en enseignement obligatoire.

Au Canada, les bases du codage sont inscrites au programme de la maternelle à la 3e année en Nouvelle-Écosse (2015-2016). Depuis 2016, la Colombie-Britannique modifie progressivement son curriculum afin que les élèves du préscolaire, du primaire et du secondaire soient initiés officiellement au codage informatique. Au Québec, l'informatique et l'initiation aux technologies sont intégrées au Programme de formation de l'école québécoise (PFEQ) (MEES, 2018b)[3] par l'entremise de la compétence transversale *Exploiter les TIC*. Toutefois, la programmation n'apparaît pas dans le PFEQ[4].

Par ailleurs, la robotique pédagogique est de plus en plus utilisée au Québec. Certains établissements d'enseignement privés avaient déjà formellement intégré le codage informatique dans leurs pratiques avant la parution de la Stratégie numérique du Québec, qui poursuit un objectif de « développement des compétences numériques de tous et toutes ». La deuxième mesure du Plan d'action numérique en éducation et en enseignement supérieur (PAN) consiste d'ailleurs à « accroître l'usage pédagogique de la programmation informatique » comme moyen de développer la compétence numérique. Le ministère souhaite que la programmation soit « dans la majorité des écoles primaires et secondaires du Québec, tant publiques que privées, d'ici l'année scolaire 2020-2021 » (MEES, 2018a, p. 27), comme c'est le cas dans certaines provinces canadiennes et certains pays européens cités en exemple. « Étant donné le potentiel interdisciplinaire de la programmation informatique, les élèves y seront initiés de façon transversale » (*ibid*.) suivant les conclusions d'un rapport commandé par le ministère. Ce dernier indiquait que : « L'enseignement de la programmation favorise le développement de compétences transversales, qu'elles soient d'ordre intellectuel, méthodologique, personnel et social ou communicationnel. Toutes, sans exception, sont mises à profit : exploiter l'information, résoudre des problèmes, exercer son jugement critique, mettre en œuvre sa pensée créatrice, se donner des méthodes de travail efficaces, exploiter les TIC et actualiser son potentiel » (Barma, 2018, p. 117).

L'intégration d'une nouvelle compétence transversale au PFEQ pour développer une pensée informatique est aussi une avenue qui fait écho au consensus général sur la pertinence de l'enseignement de la programmation. Ainsi, et contrairement à d'autres pays qui intègrent la programmation dans le cadre des sciences informatiques ou qui implantent sa pratique au sein de cours obligatoires, le Québec vise plutôt l'utilisation de la programmation informatique à des fins pédagogiques et didactiques. La publication récente d'un document du ministère, qui vise à démystifier la programmation informatique, précise son potentiel pédagogique : « Bien qu'elles puissent aussi faire l'objet d'un apprentissage spécifique, la programmation et la robotique peuvent être utilisées comme modalités pédagogiques au même titre que la création plastique, la conception de vidéos, le jeu ou la recherche d'information. Ainsi utilisées, elles

2 « une absence de recherche empirique sur le sujet, susceptible de fournir des preuves aux décideurs » (notre traduction).

3 Au secondaire, des cours à option en programmation informatique peuvent être offerts au 2e cycle selon les programmes locaux des établissements. Ces cours n'apparaissent pas dans la liste des matières à option pour lesquelles le ministre a établi un programme d'études (MEES, 2017).

4 La programmation était enseignée au secondaire sous l'ancien régime pédagogique dans le cadre du cours Introduction à la science de l'informatique et du cours Initiation à un langage de programmation (1982).

permettent de développer plusieurs compétences du Programme de formation de l'école québécoise (PFEQ), tant sur le plan des compétences transversales que sur celui des compétences disciplinaires » (MEES, 2020).

L'approche du Québec conçoit en somme la programmation comme une « modalité pédagogique supplémentaire » qui peut susciter l'engagement, la créativité, la résolution de problèmes, la familiarisation au numérique et la structuration de la pensée, par exemple.

2 La pertinence de l'apprentissage de la programmation

Le rôle de la programmation dans le développement de la pensée informatique expliquerait l'engouement actuel pour la programmation et sa réintroduction dans plusieurs curriculums nationaux. L'apprentissage et la pratique de la programmation permettraient de développer des compétences transférables dans d'autres domaines (telles que la pensée créative, l'esprit critique et la résolution de problèmes), connexes ou non à l'informatique.

Même si les recherches ne sont pas unanimes, plusieurs tendent à confirmer cette hypothèse, ce qui justifierait l'enseignement de la programmation pour éduquer aussi bien au numérique que par le numérique. Les auteurs d'une analyse de la place des sciences informatiques au sein des curriculums du Royaume-Uni, de la Nouvelle-Zélande, de l'Australie, d'Israël et de la Pologne (Webb, Davis, Bell et autres, 2016), estiment que l'omniprésence du numérique exige des connaissances et des compétences essentielles pour assurer une participation pleine et entière à la société actuelle.

2.1 Pensée informatique et programmation

La pensée informatique désigne « […] *a universally applicable attitude and skill set* […]. *Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science* » (Wing, 2006, p. 33)[5]. Bref, la pensée informatique impliquerait le traitement systématique, logique et efficace de l'information afin de réaliser certaines tâches.

5 « ensemble d'attitudes et d'acquis universellement applicables […] [qui] conduit à résoudre des problèmes, à concevoir des systèmes et à comprendre le comportement humain, en s'appuyant sur les concepts fondamentaux de la discipline et en y incluant une large collection d'outils intellectuels qui reflètent l'étendue de la science qu'est l'informatique » (traduction de Pierre Lescanne, autorisée par l'auteure).

Les compétences en programmation comprennent celles qui sont nécessaires pour créer, modifier et évaluer du code, ainsi que les connaissances relatives aux procédures de programmation (objets et algorithmes, par exemple) (Scherer, Siddiq et Sánchez Viveros, 2018). Ces deux dimensions sont appelées concepts computationnels ou informatiques (ceux que les programmeuses et programmeurs utilisent comme des variables et de la syntaxe) et pratiques informatiques (pratiques de résolution de problèmes qui interviennent lors du processus de programmation). Le traitement de l'information mobilise à la fois des compétences et des connaissances préexistantes en plus d'en créer de nouvelles lors du processus.

Malgré la relation de proximité entre les compétences en programmation et la pensée informatique, les deux concepts ne sont pas identiques, car la pensée informatique couvre plus large : « En principe, la pensée informatique ne nécessite pas l'apprentissage de la programmation bien qu'en réalité, elle représente une façon de mettre en pratique la méthode de résolution de problème par l'apprenant et qu'elle permet son évaluation par l'enseignant » (Webb, Davis, Bell et autres, 2016).

Aux compétences et aux connaissances procédurales citées précédemment s'ajoute la perspective informatique. Cette dernière dimension désigne la capacité réflexive de l'apprenante et de l'apprenant à l'égard des technologies qui l'entourent, de ses relations avec autrui et de sa compréhension de lui-même. En d'autres termes, il s'agit de pensée critique (Lye et Koh, 2016) ou encore de culture numérique.

Développer la pensée informatique implique de décomposer en étapes simples un problème quelconque, de le traiter le plus efficacement possible afin de pouvoir généraliser ce traitement systématique de l'information à d'autres situations. Toutefois, la transférabilité de ces apprentissages n'aurait pas encore été complètement validée et documentée par des recherches empiriques rigoureuses (Grover et Pea, 2013). Les études menées pendant les années 1980 n'ont pas permis de démontrer les effets bénéfiques de la programmation pour le développement de compétences générales (Popat et Starkey, 2019).

L'introduction de la pensée computationnelle au sein des curriculums obligatoires accompagnée « de discours idéologiques qui véhiculent des attentes souvent disproportionnées chez les acteurs politiques, professionnels et scientifiques » (Collin, Guichon et Gabin Ntébutsé, 2015) par rapport à l'enseignement de la programmation a fait l'objet de critiques au sein des milieux éducatifs pendant la dernière décennie. Ainsi, le concept de pensée informatique est jugé imprécis et flou. De plus, son insertion dans des curriculums déjà surchargés apparaissait problématique, et ce, d'autant plus que l'on n'avait pas réfléchi à la meilleure manière de l'inclure (comme sujet général, comme discipline ou de façon pluridisciplinaire). On s'est aussi demandé si la pensée informatique était à ce point différente des autres formes de pensée à développer, prévues dans certains curriculums, ou encore si son développement était utile pour les jeunes ne se destinant pas aux domaines des sciences, à la technologie, à l'ingénierie et aux mathématiques (STIM). À ce sujet, Grover et Pea (2013) expliquent que les promoteurs de la pensée informatique reconnaissent que celle-ci partage des éléments avec les mathématiques, l'ingénierie et le design, mais ils considèrent qu'elle diffère en ce qu'elle focalise sur le traitement systématique de l'information. Ces auteurs estimaient par ailleurs que plusieurs éléments restaient à documenter pour justifier l'introduction de la pensée informatique dans les curriculums. Ils proposaient alors de reprendre et de mettre à jour, dans le contexte des compétences du 21e siècle, le bagage de connaissances issu des premiers travaux des années 1980. Ces travaux portaient sur les aspects cognitifs de l'apprentissage des concepts informatiques. Ils rappelaient aussi d'étudier la question de l'usage de l'informatique en tant que moyen pour enseigner d'autres matières (transfert de compétences en résolution de problèmes dans d'autres domaines).

> *Clearly, much remains to be done to help develop a more lucid theoretical and practical understanding of computational competencies in children. What, for example, can we expect children to know or do better once they've been participating in a curriculum designed to develop CT and how can this be evaluated? These are perhaps among the most important questions that need answering before any serious attempt can be made to introduce curricula for CT development in schools at scale. It is time to redress the gaps and broaden the 21st-century academic discourse on computational thinking* (Grover et Pea, 2013, p. 42)[6].

D'autres auteures et auteurs partagent cette opinion. Ils invitent ainsi les chercheuses et les chercheurs en éducation à évaluer les apports cognitifs de l'apprentissage de la programmation (Scherer, 2016).

2.2 Apports cognitifs de la programmation, compétences transversales et effets de transfert

Une recension des écrits sur les effets de l'apprentissage de la programmation (*cognitive educational outcomes*) (Popat et Starkey, 2019) utilise la taxonomie révisée de Bloom-Krathwohl qui comprend six niveaux répartis en deux paliers : niveaux cognitifs de base (se rappeler, comprendre et appliquer) et hauts niveaux cognitifs (analyser, évaluer et créer). Sur cette base d'analyse se dégagent cinq éléments décrivant les apports de l'enseignement de la programmation :

1. Les études recensées suggèrent que la pratique de la programmation peut favoriser l'apprentissage de la résolution de problèmes (principalement mathématiques). Cependant, les résultats ne démontrent pas que la programmation est plus efficace qu'un enseignement traditionnel.

2. L'idée voulant que les activités de programmation en classe favorisent la collaboration (compétence transversale de coopération) entre les apprenantes et les apprenants et qu'elles stimulent leurs habiletés sociales reste à démontrer. Les études recensées n'ont pas isolé la dynamique de classe, ce qui rend les résultats discutables.

3. Certaines études suggèrent que l'apprentissage de la programmation peut inciter les étudiantes et les étudiants à devenir des apprenantes et des apprenants actifs grâce aux activités et outils en ligne mis à leur disposition. Ces études reconnaissent par ailleurs que certaines notions permettant aux élèves de comprendre comment mener à bien des activités d'apprentissage ne peuvent être inculquées que par des enseignantes et des enseignants.

6 De toute évidence, il reste beaucoup à faire pour développer une compréhension théorique et pratique des compétences computationnelles chez les enfants. Par exemple, que souhaitons-nous que les enfants sachent davantage ou accomplissent mieux une fois qu'ils auront participé à un programme d'études conçu pour développer les compétences informatiques? Comment peut-on les évaluer? Savoir ce que l'on souhaite développer et comment on l'évaluera est sans doute l'une des questions les plus importantes à poser avant que soient introduites massivement les compétences informatiques dans les curriculums. Il est temps de combler ces lacunes et d'élargir le discours autour des compétences du 21e siècle et du développement de la pensée informatique chez les enfants (Grover et Pea, 2013, p. 42, notre traduction).

4. Les études recensées suggèrent que certains aspects de la pensée critique peuvent être développés chez les apprenantes et les apprenants par des activités de programmation. Les auteurs mentionnent que les étudiantes et les étudiants devraient être encouragés à tester, évaluer et modifier du code afin de développer leur pensée critique, ce qui impliquerait des tâches de haut niveau cognitif de la taxonomie de Bloom-Krathwohl appliquées au contexte numérique.

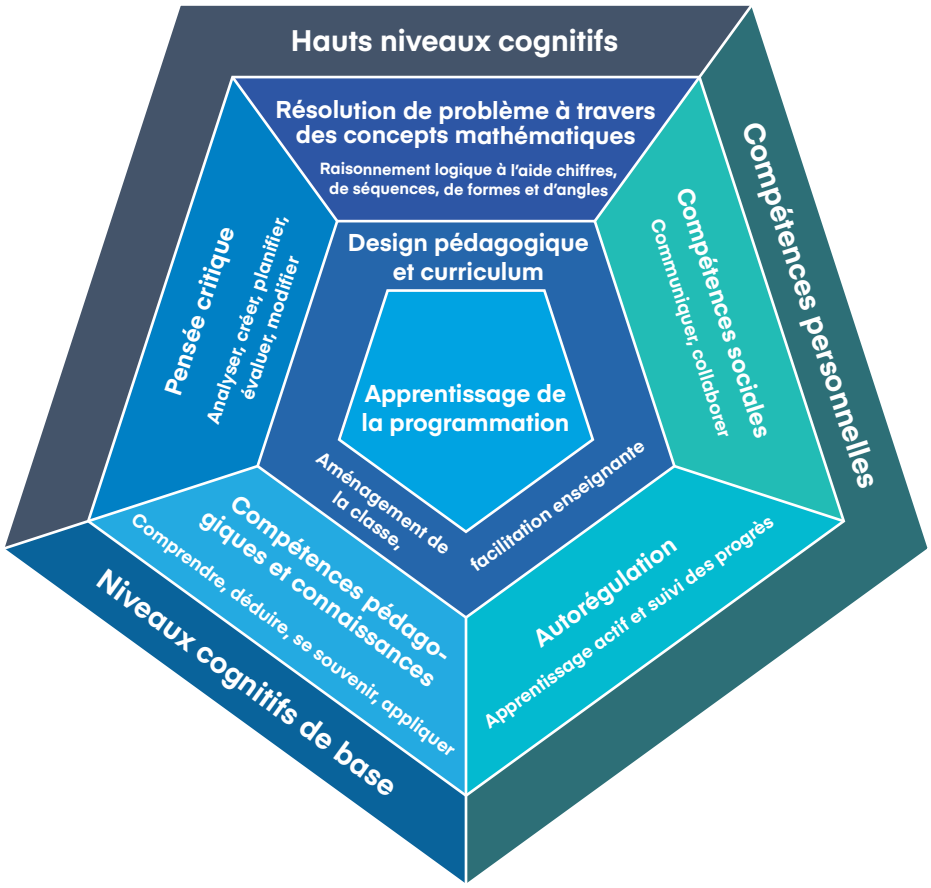
5. Certaines évaluations retenues dans la revue systématique de Popat et Starkey (2019) observent que la programmation a des effets bénéfiques sur le développement de compétences et l'acquisition de connaissances dans d'autres domaines que ceux liés à l'informatique ou aux mathématiques (notamment en lecture, en épellation, en arts ou en histoire). Toutefois, l'acquisition de ces compétences par la programmation serait **largement tributaire des stratégies pédagogiques de l'enseignante ou de l'enseignant**. Ainsi, la manière d'enseigner la programmation et les contextes pédagogiques influencent les effets positifs (résultats) de son apprentissage chez les apprenantes et les apprenants. Par exemple, l'organisation spatiale de la salle de classe peut favoriser la collaboration, la communication et le travail en équipe entre les élèves, peu importe la matière enseignée.

La recension des écrits suggère « *that through learning to code, students use a range of skills beyond coding. However, these skills are developed further through the deliberate teaching and inclusion in the curriculum and pedagogical design* » (Popat et Starkey, 2019, p. 370)[7]. Ainsi, l'apprentissage de la programmation et sa pratique en contexte scolaire impliquent des compétences cognitives « de base ainsi que des compétences de haut niveau cognitif ». Le schéma suivant illustre l'influence de l'apprentissage de la programmation en fonction de la taxonomie de Bloom-Krathwohl.

7 « suggère qu'en apprenant à programmer, les élèves utilisent une gamme de compétences allant au-delà des activités de codage. Le développement de ces compétences transversales est d'autant plus favorisé lorsqu'il est intégré au curriculum et lorsqu'il fait l'objet d'un enseignement intentionnel accompagné de stratégies pédagogiques appropriées » (Popat et Starkey, 2019, p. 370, notre traduction).

Figure 1

Modélisation de l'influence de la programmation



Source : traduit et adapté de Popat et Starkey, 2019, p. 370.

Certaines compétences de haut niveau cognitif peuvent être aussi bien développées chez les apprenantes et les apprenants sans passer par des activités de programmation. C'est le cas, notamment, des compétences en résolution de problèmes d'ordre mathématique, pour lesquelles un enseignement direct traditionnel serait plus efficace toujours selon la même recension. Comme la plupart des activités de codage destinées aux jeunes apprenantes et apprenants utilisent la programmation comme un ensemble de problèmes auxquels les enfants doivent trouver les solutions « correctes », les études recensées par Popat et Starkey ne démontrent pas d'effets bénéfiques de la programmation sur la créativité. Toutefois, comme nous le verrons plus loin, d'autres auteures et auteurs rapportent des résultats différents.

Les habiletés personnelles que la programmation contribuerait à développer comprennent les compétences sociales, l'autogestion et l'apprentissage actif. Le postulat voulant que la programmation permette le travail collaboratif et favorise le développement de compétences sociales est répandu, il est même intégré dans certains curriculums (Australie et Pologne) (Webb, Davis, Bell et autres, 2016). Or, il n'y a pas de consensus sur l'importance et le rôle que revêtent les compétences sociales dans le contexte

de l'apprentissage des sciences informatiques. La recension de Popat et Starkey (2019) indique que ces habiletés relèvent davantage du contexte d'apprentissage, du design pédagogique et des intentions pédagogiques et didactiques de l'enseignante et de l'enseignant que d'un effet direct de l'apprentissage de la programmation. Cette observation est également valable pour les compétences d'autogestion et d'apprentissage actif.

La conclusion principale de cette recension systématique est que le design pédagogique et l'intégration de l'apprentissage de la programmation dans les curriculums sont les deux éléments à prendre en considération; ils déterminent les résultats (*educational outcomes*) escomptés de l'apprentissage de la programmation.

Ce dernier aspect souligne l'importance du rôle de l'enseignante et de l'enseignant, de ses intentions didactiques et de ses compétences pédagogiques ainsi que des situations d'apprentissages qu'il crée à l'aide des technologies numériques. Les activités et les tâches organisées et planifiées par le personnel enseignant favorisent un meilleur apprentissage des sciences informatiques (plutôt que par apprentissage autonome), soutiennent l'apprentissage actif et améliorent le travail collaboratif et la communication entre les apprenantes et les apprenants.

Concernant les habiletés associées aux niveaux taxonomiques de base (se rappeler, comprendre, appliquer des connaissances), peu d'études se sont penchées sur l'applicabilité des compétences acquises en programmation dans d'autres domaines que les mathématiques ou les sciences de l'informatique. En fait, la présomption du transfert des compétences acquises par la pratique de la programmation remonte aux années 1980-1990 (Scherer, 2016). Sans preuve à cette époque, les chercheuses et chercheurs présumaient que les compétences développées par la programmation étaient transférables et généralisables (Pea, 1983), car elles partageaient des caractéristiques avec la compétence transversale de résolution de problèmes ainsi que la pensée créative.

Considérant les résultats mitigés des travaux récents, quelques chercheuses et chercheurs ont remis en doute cette idée dans une méta-analyse fort éclairante portant sur le transfert d'apprentissages lié à la programmation (Falloon [2016] et Denning [2017], cités dans Scherer, Siddiq et Sánchez Viveros, 2018). Selon cette méta-analyse, une part des disparités observées dans les conclusions des études relèvent des méthodologies employées : « *Moderator analyses revealed significantly larger transfer effects for studies with untreated control groups than those with treated (active) control groups. Moreover, published studies exhibited larger effects than gray literature* » (Scherer, Siddiq et Sánchez Viveros, 2018, p. 1)[8]. Au-delà de ces variations, la méta-analyse démontre l'existence d'un effet global de transfert statistiquement significatif et qualifié de « modéré » des compétences acquises par la programmation. L'interrelation des concepts et des compétences associés à la programmation et à la pensée informatique ainsi qu'à diverses autres compétences transversales explique ce résultat (*ibid*).

Ainsi, l'enseignement de la programmation favoriserait l'acquisition de la pensée informatique tout en étant évidemment efficace pour l'apprentissage des habiletés en programmation. L'effet de transfert n'est toutefois pas distribué également sur toutes les compétences transversales et aptitudes cognitives. Plus la situation d'apprentissage est éloignée du domaine informatique, moins le transfert de compétences serait

8 « Les analyses modératrices ont révélé des effets de transfert significativement plus importants pour les études avec des groupes témoins non traités que ceux avec des groupes témoins traités (actifs). De plus, les études publiées ont montré des effets plus importants que la littérature grise » (Scherer, Siddiq et Sánchez Viveros, 2018, notre traduction).

évident. Il y aurait, malgré tout, un effet de transfert des compétences acquises par la programmation dans d'autres aptitudes cognitives ou compétences transversales d'ordre intellectuel. Le transfert d'apprentissage dans des situations qui requièrent un processus créatif, des compétences en mathématiques, un raisonnement logique et spatial (*spatial skills*) et la capacité métacognitive seraient les éléments qui bénéficieraient le plus de l'apprentissage et de la pratique de la programmation. Les bénéfices seraient toutefois moindres pour d'autres compétences, la littératie, par exemple (compréhension de texte et écriture). Dans tous les cas, l'apprentissage de la programmation « ne nuit pas » (« *does not do harm* ») aux autres aptitudes cognitives (Scherer, Siddiq et Sánchez Viveros, 2018).

Perspectives et Conclusion

Les travaux récents portant sur les apports de l'apprentissage et la pratique de la programmation en contexte scolaire concluent à des effets bénéfiques à plusieurs niveaux, quoique ces conclusions ne reposent pas encore sur des preuves empiriques solides. Les résultats escomptés de l'apprentissage de la programmation dépendent toutefois largement du design pédagogique et de la manière dont est intégrée la programmation dans le curriculum scolaire.

> [L]intégration des TIC dans l'enseignement ne va pas de soi. En effet, comme plusieurs études l'ont déjà souligné, une technologie elle-même ne peut pas améliorer automatiquement, seulement par son utilisation, la qualité de l'enseignement ou de l'apprentissage; cette utilisation doit être « orchestrée » (Trouche, 2005) ou « apprivoisée » (Charlier et Péraya, 2002) par une approche pédagogique appropriée. Pédagogie et didactique sont souvent à la remorque des technologies, ce qui engendre une utilisation opportuniste de ces outils et pratiques; un placage plutôt qu'une réelle réflexion (Lapierre et Fournier, 2018, p. 142-143).

L'accompagnement du personnel enseignant et la formation aux approches pédagogiques pour appuyer l'intention pédagogique et didactique des formations à l'enseignement de la programmation informatique, recommandés par le CRIRES (Barma, 2018), sont donc cruciaux.

Toutefois, par-delà la démonstration de la pertinence de l'apprentissage de la programmation pour le développement de compétences transversales et de la pensée informatique, des voix s'élèvent pour nuancer l'optimisme ambiant. Les technologies « loin d'être neutres, incorporent les valeurs issues des contextes dans lesquels elles émergent. Ce faisant, le numérique représente une forme de pouvoir généralement annexé par les groupes dominants pour assurer leurs intérêts, aux dépens d'autres développements sociaux possibles, de nature plus démocratique notamment » (Collin, Guichon et Gabin Ntébutsé, 2015). Le rôle prépondérant de l'industrie des TIC dans l'introduction de l'apprentissage de la programmation dans de nombreux curriculums et leur influence toujours croissante auprès des décideurs illustrent cette forme de pouvoir.

La programmation comme solution au chômage des jeunes et aux problèmes du marché du travail est un autre argument critiqué. En fait, le ministère du travail américain prévoit une diminution du nombre de programmeuses et de programmeurs en informatique en raison de l'externalisation des contrats (U.S. Bureau of Labor Statistics, 2019). À ce propos, pour certaines chercheuses et certains chercheurs, les compétences en programmation ne seront plus nécessaires dans un avenir rapproché, puisque les méthodes de l'intelligence artificielle pourront remplacer les codeurs. Ce qui rejoint l'avis exprimé par Andreas Schleicher, directeur de la Direction de l'éducation et des compétences de l'OCDE, dans un entretien qu'il a accordé à ***The Telegraph***. Il affirme que : « la programmation est simplement "une technique de notre temps" et qu'elle ne sera plus pertinente à l'avenir ». Schleicher plaide cependant pour l'enseignement plus large de la pensée informatique et de ses sous-composantes, qui sont favorisées par l'apprentissage de la programmation. En fait, c'est l'apprentissage de la programmation sans intégration dans un curriculum plus large et sans être liée à d'autres compétences, qui serait inutile (Turner, 2019).

Enfin, l'apprentissage de la programmation dans le cadre des compétences du 21e siècle est également remis en question par certains penseurs de l'éducation, qui invitent à une réflexion sur des enjeux qui dépassent les simples compétences :

> *While 21st century learning may make perfect sense as an efficient strategy to meet the pre-given end of strengthening a globalized, knowledge economy and providing a highly-qualified workforce for the future, it can be ethically, morally, historically and culturally invalid if its aim is to simply recreate the hyper-consumptive patterns of beliefs and behaviors that threaten the viability of life on the planet through a transnational globalization guided by a money code of value. By not engaging meaningfully with the end of education, proponents of 21st century learning miss the opportunity to open doors to rational debate, communication, mediation* (Howard, 2018, p.10)[9].

Malgré les critiques envers l'enseignement de la programmation, de son côté utilitariste, instrumental, technique et de son caractère restrictif, il reste que les personnes qui maîtrisent les bases de la programmation et ses langages ont davantage de pouvoir d'agir et d'autonomie dans leurs usages des technologies numériques que ceux qui ne disposent pas de connaissances dans ce domaine. En ce sens, l'apprentissage de la programmation, et ultimement le développement de la pensée informatique et des compétences transversales qu'elle peut favoriser, est une option qui peut comporter certains avantages. Comme l'avancent Desjardins, Tran et Girard :

> […] il devient de plus en plus important que les jeunes développent une compréhension des outils qu'ils utilisent quotidiennement pour en comprendre le fonctionnement, en maximiser l'utilisation et développer un regard critique face à cette même utilisation. En quelque sorte, il s'agit d'un niveau élevé de littératie numérique qui s'opère dans le développement de la pensée informatique. […] les cours de programmation pourront

9 « [L] e mouvement vers les nouveaux apprentissages du 21e siècle est parfaitement logique en tant que stratégie visant à renforcer l'économie du savoir et à fournir une main-d'œuvre hautement qualifiée pour l'avenir. Cette intention est toutefois discutable du point de vue éthique, moral, historique et culturel, si elle conduit à soutenir et à multiplier les comportements d'hyperconsommation et la mondialisation qui menacent la planète. En ne réfléchissant pas aux finalités de l'éducation, les tenants des compétences du 21e siècle manquent l'occasion d'ouvrir un débat rationnel avec les acteurs du milieu de l'éducation qui partagent d'autres valeurs » (Howard, 2018, p. 10, notre traduction).

> diminuer les effets d'un potentiel « fétichisme technologique » (attrait pour le « gadget ») et permettre de garder une distance critique face aux outils utilisés quotidiennement (Desjardins, Tran et Girard, 2018).

Au terme des délibérations sur le sujet, le Comité du rapport sur l'état et les besoins de l'éducation retenait que même si le discours ambiant accorde une grande importance à la programmation, il y aurait lieu de recontextualiser cette pratique dans l'ensemble des autres façons possibles de développer les compétences liées au numérique (notamment collaboratives, expressives et créatives). Par ailleurs, même si elle n'est pas une panacée, la programmation informatique (ou robotique), par son caractère concret, mène certains élèves vers le succès (notamment ceux dont les capacités rédactionnelles sont moins grandes). En ce sens, elle permet de démocratiser les apprentissages. Elle peut aussi devenir un prétexte à la collaboration et à la résolution de problèmes. Elle aurait donc sa place en éducation, mais ne devrait pas être présentée comme la seule façon de développer les compétences visées et ne devrait pas faire l'objet de pression sur les enseignantes et les enseignants pour qu'ils adoptent ce moyen plutôt que d'autres.

Bibliographie

Barma, Sylvie (2018). *Réaliser une étude de cas multiple qui vise à affiner les connaissances sur l'usage pédagogique ou didactique de la programmation dans les écoles du Québec*, Québec, Université Laval et Centre de recherche et d'intervention sur la réussite scolaire, 212 p., réf. de septembre 2020, **https://lel.crires.ulaval.ca/sites/lel/files/barma_mees_rapport_final_-_lel.pdf**.

Collin, Simon, Nicolas Guichon et Jean Gabin Ntébutsé (2015). « Une approche sociocritique des usages numériques en éducation », *STICEF*, n° 22, réf. de septembre 2020, **https://hal.archives-ouvertes.fr/hal-01218240/document**.

Conseil supérieur de l'éducation (1984). *Le développement de la micro-informatique dans les écoles primaires et les écoles secondaires*, Sainte-Foy, Le Conseil, 42 p., réf. de septembre 2020, **http://www1.cse.gouv.qc.ca/fichiers/documents/publications/Avis/50-0327.pdf**.

Grover, Shuchi et Roy Pea (2013). « Computational Thinking in K-12: a Review of the State of the Field », *Educational Researcher*, vol. 42, n° 1, p. 38-43.

Howard, Patrick G. (2018). « *Twenty-First Century Learning as a Radical Re-Thinking of Education in the Service of Life* », *Education Sciences*, vol. 8, n° 89, p. 1-13.

Lamb, Stephen et Quentin Maire et Esther Doeck (2017). *Key Skills for the 21st Century: an Evidence-Based Review*, Future Frontiers Analytical Report, Melbourne (Australia), Centre for International Research on Education Systems, 70 p.

Lapierre, Hugo G. et Frédéric Fournier (2018). « Enseigner l'informatique par le codage et la robotique pédagogique! », dans Frédéric Fournier, Martin Riopel, Patrick Charland et Patrice Potvin (dir.), *Utiliser les TIC dans le contexte de l'enseignement de la science et de la technologie*, Montréal, EREST, p. 139-153.

Lye, Sze Yee et Joyce Hwee Ling Koh (2016). « Review on Teaching and Learning of Computational Thinking Through Programming: What is Next for K-12? », *Computers in Human Behavior*, vol. 41, p. 51-61.

Ministère de l'Éducation, Bureau de la mise en œuvre du plan d'action numérique (2020). L'usage pédagogique de la programmation informatique, Québec, Le Ministère, 12 p., réf. de septembre 2020, **http://www.education.gouv.qc.ca/fileadmin/site_web/documents/ministere/Usage-pedagogique-programmation-informatique.pdf**.

Ministère de l'Éducation et de l'Enseignement supérieur (2018a). Plan d'action numérique en éducation et en enseignement supérieur, Québec, Le Ministère, 84 p., réf. de septembre 2020, **http://www.education.gouv.qc.ca/fileadmin/site_web/documents/ministere/PAN_Plan_action_VF.pdf**.

Ministère de l'Éducation et de l'Enseignement supérieur (2018b). Programme de formation de l'école québécoise, réf. de juin 2018, **http://www.education.gouv.qc.ca/enseignants/pfeq/**.

Ministère de l'Éducation et de l'Enseignement supérieur (2017). Instruction annuelle 2017-2018 : la formation générale des jeunes : l'éducation préscolaire, l'enseignement primaire et l'enseignement secondaire, Québec, Le Ministère, 18 p., réf. de septembre 2020, http://www.education.gouv.qc.ca/fileadmin/site_web/documents/dpse/formation_jeunes/Instruction_annuelle_2017-2018.pdf.

Moreno-León, Jesús, Gregorio Robles et Marcos Román-González (2016). « Code to learn: Where Does it Belong in the K-12 Curriculum? », *Journal of Information Technology Education: Research*, vol. 15, p. 283-303, réf. de septembre 2020, https://www.informingscience.org/Publications/3521.

Papert, Seymour (1980). *Mindstorms : Children, Computers and Powerful Ideas*, New York (N.Y.), Basic Books, 230 p.

Pea, Roy D. (1983). « Logo Programming and Problem Solving » dans *American Educational Research Association Symposium* (Montréal, Canada, April 1983).

Popat, Shaira et Louise Starkey (2019). « Learning to Code or Coding to Learn ?: a Systematic Review », *Computers & Education*, vol. 128, p. 365- 376.

Sawchuk, Stephen (2009). « Motives of 21st-Century-Skills Group Questioned », *Education Week*, Decembre 4, réf. de septembre 2020, https://www.edweek.org/ew/articles/2009/12/09/14partnership_ep.h29.html.

Scherer, Ronny (2016). « Learning from the Past–the Need for Empirical Evidence on theTransfer Effects of Computer Programming Skills », *Frontiers in Psychology*, vol. 7, n° 1390.

Scherer, Ronny, Fazilat Siddiq et Bárbara Sánchez Viveros (2018). « The Cognitive Benefits of Learning Computer Programming: a Meta-Analysis of Transfer Effects », *Journal of Educational Psychology*, vol. 111, n° 5, p. 764- 792, réf. de septembre 2020, https://doi.apa.org/doiLanding?doi=10.1037%2Fedu0000314.

Turner, Camilla (2019). « Teaching Children Coding is a Waste of Time, OECD Chief Says », *The Telegraph*, February 21.

U.S. Bureau of Labor Statistics (2020). *Occupational Outlook Handbook: Computer Programmers*, réf. de septembre 2020, https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm.

Webb, Mary, Niki Davis, Tim Bell, Yaacov J. Katz, Nicholas Reynolds, Dianne P. Chambers et Maciej M. Sysko (2016). « Computer Science in K-12 School Curricula of the 21st Century: Why, What and When? », *Education and Information Technologies*, vol. 22, n° 2, p. 445-468.

Wing, Jeannette M. (2006). « Computational Thinking », *Communications of the ACM*, vol. 49, n° 3, p. 33-35, réf. de septembre 2020, http://www.cs.cmu.edu/~./15110-s13/Wing06-ct.pdf.

50-2112

Conseil supérieur de l'éducation
Québec

@csequebec
cse.gouv.qc.ca