MYŚLENIE KOMPUTACYJNE W PRACY NAUCZYCIELA I SZKOŁY

Robert Porzak, Panagiotis Psomos

Wydawnictwo Innovatio Press

MYŚLENIE KOMPUTACYJNE W PRACY NAUCZYCIELA I SZKOŁY

Redakcja:
Robert Porzak
Panagiotis Psomos

MYŚLENIE KOMPUTACYJNE W PRACY NAUCZYCIELA I SZKOŁY

Redakcja:
Robert Porzak
Panagiotis Psomos

Lublin 2023

Niniejszy projekt został zrealizowany przy wsparciu finansowym Komisji Europejskiej (nr projektu: 2020-1-PL01-KA201-081924). Publikacja odzwierciedla jedynie stanowisko jej autora i Komisja Europejska nie ponosi odpowiedzialności za umieszczoną w niej zawartość merytoryczną.

Lubelska Akademia WSEI

Seria wydawnicza:
Monografie Wydziału Nauk o Człowieku

Myślenie komputacyjne w pracy nauczyciela i szkoły
Wydanie pierwsze

Redaktor naukowy:
Robert Porzak, Panagiotis Psomos

Recenzenci:
dr hab. Wiesław Kowalski, prof. WSEI
dr Paraskevi Poulogiannopoulou - European University Cyprus

Tłumaczenie:
Magda Janiak

Skład, łamanie:
Marta Krysińska-Kudlak

Korekta:
Teresa Markowska

Projekt okładki:
Wiktor Bogusz

Grafika okładki:
gpointstudio/ freepik.com

Niniejsza książka stanowi tłumaczenie monografii pt. Computational Thinking for Teachers and Classes – manual for teachers wydanej w języku angielskim o numerze ISBN electronic version: 978-83-67550-13-0

Niniejsza publikacja została sfinansowana przy wsparciu Komisji Europejskiej w ramach Programu Erasmus + (projekt nr: 2020-1-PL01-KA201-081924). Publikacja odzwierciedla jedynie stanowisko jej autora i Komisja Europejska nie ponosi odpowiedzialności za umieszczoną w niej zawartość merytoryczną.

Wydano w Polsce
Innovatio Press Wydawnictwo Naukowe
Lubelska Akademia WSEI
20-209 Lublin, Projektowa 4
tel.: +48 81 749 17 77, fax: +48 81 749 32 13
www.wsei.lublin.pl, e-mail: wydawnictwo@wsei.lublin.pl

ISBN - wersja elektroniczna: 978-83-67550-14-7

PUBLIKACJA BEZPŁATNA

Spis treści

Wprowadzenie

Myślenie komputacyjne (MK – ang.: CT – *Computational Thinking*) to umiejętność, która pozwala nam rozwiązywać złożone problemy poprzez dzielenie ich na mniejsze i prostsze etapy, znajdowanie wzorców i podobieństw, abstrahowanie od nieistotnych szczegółów do ogółów i projektowanie algorytmów, które mogą być wykonywane przez komputery lub ludzi. Jest to umiejętność niezbędna w XXI wieku, ponieważ technologia wymagająca algorytmizacji staje się coraz bardziej wszechobecna i wpływa na każdy aspekt naszego życia. Myślenie komputacyjne może również zwiększyć naszą kreatywność, krytyczne myślenie i umiejętności współpracy, ponieważ uczymy się stosować je w różnych dziedzinach i dyscyplinach, takich jak sztuka, języki, matematyka, nauki ścisłe i nauki społeczne.

Myślenie komputacyjne bywa określane myśleniem obliczeniowym, ale między tymi pojęciami występują istotne różnice. „Obliczenia" są częściej kojarzone z procesami arytmetycznymi, podczas gdy „komputacja" jest często kojarzona z procesami algorytmicznymi. Matematyka obliczeniowa koncentruje się na liczbach, myślenie komputacyjne koncentruje się na procesach. Uczniowie zaangażowani w praktykę myślenia komputacyjnego dzielą złożony problem lub proces na mniejsze kroki, aby lepiej go zrozumieć, opisać lub wyjaśnić, a ostatecznie zalgorytmizować rozwiązywanie podobnych problemów. Celem myślenia komputacyjnego jest więc algorytm możliwy do wielokrotnego zastosowania. Dla uczniów skoncentrowanych na obliczeniach, celem jest jedno, właściwe rozwiązanie lub jednorazowa odpowiedź.

Ta książka jest skierowana do nauczycieli, którzy jeszcze nie korzystali w szerokim zakresie z narzędzi wspierających rozwój myślenia komputacyjnego uczniów. Książka ma na celu pomóc nauczycielom zrozumieć, czym jest myślenie komputacyjne, dlaczego jest ważne i jak mogą je zintegrować z istniejącymi programami nauczania. Książka opiera się na przeglądzie literatury na temat aktualnych badań i najlepszych praktyk w zakresie wspierania uczniów w poznawaniu myślenia komputacyjnego, a także na doświadczeniach i spostrzeżeniach autorów i partnerów projektu CTApp. Książka zawiera szereg praktycznych wskazówek i przykładów dla nauczycieli, którzy chcą włączyć myślenie komputacyjne w programy zajęć swoich przedmiotów, wykorzystując ogólne narzędzia edukacyjne, a także metody specyficzne dla treści danego przedmiotu.

Książka składa się z siedmiu rozdziałów. Pierwszy rozdział wprowadza koncepcję myślenia komputacyjnego i jego komponenty: dekompozycję, rozpoznawanie wzorców, abstrakcję i projektowanie algorytmów. Wyjaśnia również korzyści i wyzwania

związane z uczeniem myślenia komputacyjnego oraz rolę edukatorów w jego wspieraniu. Rozdział drugi koncentruje się na nauczaniu dekompozycji, a trzeci na rozpoznawaniu wzorców, czyli umiejętności rozkładania problemu na mniejsze części i znajdowania podobieństw między nimi. Zawierają one strategie i ćwiczenia do nauczania tych umiejętności na różnych przedmiotach i poziomach zaawansowania. Czwarty rozdział obejmuje nauczanie abstrakcji, a piąty projektowania algorytmów, czyli umiejętności usuwania zbędnych szczegółów i tworzenia sekwencji kroków w celu rozwiązania problemu. Rozdziały te zawierają wskazówki i przykłady nauczania umiejętności abstrahowania i projektowania algorytmów w różnych kontekstach i scenariuszach. Szósty rozdział zawiera przegląd gry CTApp Game, która pomaga uczniom ćwiczyć umiejętności myślenia komputacyjnego w zabawny i angażujący sposób. Rozdział opisuje ideę, strukturę i funkcje gry oraz sposób, w jaki nauczyciele mogą ją wykorzystać w swoich klasach. Siódmy rozdział podsumowuje niektóre popularne strategie i zasoby internetowe służące integracji myślenia komputacyjnego z różnymi przedmiotami. Zawiera również linki do dodatkowych materiałów i wskazuje możliwości dalszego kształcenia w zakresie myślenia komputacyjnego.

Dodatek do rozdziału siódmego przedstawia trzy przykładowe scenariusze dla różnych tematów (po jednym scenariuszu dostarczonym przez partnerów projektu CTApp: Cypr, Polskę i Włochy), które ilustrują, w jaki sposób myślenie komputacyjne może być stosowane w różnych dziedzinach.

Cristina Fregonese
INNOVA S.r.l.

1. Myślenie komputacyjne na pierwszy rzut oka – co, dlaczego i jak

1.1. Definicja myślenia komputacyjnego (CT)

Prawda? Nie ma jeszcze powszechnie akceptowanej definicji „myślenia komputacyjnego".

Koncepcja *myślenia komputacyjnego* (CT) została po raz pierwszy wprowadzona przez pedagoga Seymoura Paperta w 1967 roku, mówiącego o LOGO, języku programowania, który opracował w MIT (Massachusetts Institute of Technology), aby uczyć dzieci programowania. Papert był przekonany, że korzystanie z komputerów może wspierać formalne myślenie u dzieci, a w szczególności może pozwolić dzieciom na autonomiczne „konstruowanie" ich uczenia się i myślenia.

Koncepcja CT została następnie zrewitalizowana w 2006 roku przez informatyczkę Jeannette M. Wing, która w artykule „Myślenie komputacyjne" argumentowała, że zajmuje się zagadką inteligencji maszyn, *pytając, co maszyny robią lepiej niż człowiek i co człowiek robi lepiej niż maszyny*. Wing argumentuje, że myślenie komputacyjne nie jest po prostu proceduralną czynnością kodowania, ale jest podstawową umiejętnością konceptualną, która wraz z czytaniem, pisaniem i arytmetyką powinna być udostępniana wszystkim dzieciom. Wydaje się, że myślenie komputacyjne ma być krytycznym myśleniem w ocenie sytuacji i zaawansowaną umiejętnością rozwiązywania problemów przy użyciu skomputeryzowanych narzędzi.

Jeśli informatyka jest nauką o tym, co można skomputeryzować i jak to skomputeryzować, myślenie komputacyjne nie jest umiejętnością unikalną dla informatyków. Pozwala ono rozwiązywać problemy, projektować systemy i rozumieć ludzkie zachowania w codziennym życiu, w alternatywny sposób, poprzez podstawowe koncepcje technologii informacyjnej.

1.2. Charakterystyka myślenia komputacyjnego

Niektóre badania powiązały CT z krytycznym myśleniem, definiując je jako nową metodę rozwiązywania problemów przy użyciu technik informatycznych. Wspomniani autorzy (Giacalone, 2020) definiują krytyczne myślenie jako „*umiejętność lub kompetencję, dzięki której jednostka wykracza, w sposób celowy poza dostarczone informacje, w celu osiągnięcia rozsądnych wniosków, które można potwierdzić za pomocą ważnych informacji*". Jest to sposób myślenia, który umożliwia znalezienie wielu rozwiązań. W 2016 r. prof. Leen-Kiat Soh wskazał, że CT uzupełnia krytyczne myślenie jako sposób rozumowania w celu rozwiązywania problemów,

podejmowania decyzji i interakcji ze światem. CT obejmuje zatem techniki takie jak: abstrakcja, dekompozycja, projektowanie algorytmów, generalizacja, ewaluacja i interakcja z komputerem.

Kluczowe umiejętności myślenia komputacyjnego

Istnieją cztery kluczowe umiejętności w myśleniu komputacyjnym:

1. *Dekompozycja*
2. *Rozpoznawanie wzorców*
3. *Abstrakcja wzorca*
4. *Projektowanie algorytmów*

1. *Dekompozycja*

Rozbijanie dużych i trudnych problemów na coś znacznie prostszego. Często duże problemy to po prostu wiele małych problemów razem wziętych. Dekompozycja jest ważną umiejętnością życiową w przyszłości, gdy uczniowie i dorośli będą musieli podejmować się większych zadań. Uczniowie poznają sposoby delegowania zadań w projektach grupowych i rozwijają umiejętności zarządzania czasem

2. *Rozpoznawanie wzorców*

Czasami, gdy problem składa się z wielu małych fragmentów, można zauważyć, że fragmenty te mają ze sobą coś wspólnego. Jeśli tego nie mają, mogą mieć silne podobieństwa do fragmentów innego problemu, który został już wcześniej rozwiązany. Jeśli będziesz w stanie znaleźć te prawidłowości, identyfikacja poszczególnych elementów stanie się o wiele łatwiejsza: rozpoznawanie wzorców to po prostu szukanie ich w łamigłówkach i określanie, czy którykolwiek z problemów lub rozwiązań, które napotkaliśmy w przeszłości, może mieć zastosowanie w obecnej sytuacji. Czego nauczyliśmy się w przeszłości, co może nam pomóc w rozwiązaniu aktualnego problemu? Jeśli kiedykolwiek zmontowałeś mebel z IKEA, zrozumiesz znaczenie wzorców. Kiedy składasz szafkę z szufladami IKEA, złożenie pierwszej szuflady prawdopodobnie zajmie ci znacznie więcej czasu niż czwartej lub piątej. Kiedy powtarzamy kroki w naszej budowie, uczymy się szybciej rozwiązywać instrukcje i uczymy się na błędach. Żmudny proces montażu pierwszej części uczy nas umiejętności, które pozwolą nam wykonywać ten proces bardziej efektywnie w przyszłości.

3. *Abstrahowanie*

Po znalezieniu wzorca, możliwe jest abstrahowanie (ignorowanie) szczegółów, które odróżniają różne rzeczy i używanie ogólnych technik do znajdowania rozwiązań, które działają dla więcej niż jednego problemu. Identyfikacja kluczowych informacji w problemie i ignorowanie nieistotnych informacji jest jedną z najtrudniejszych części uczenia komputacyjnego.

4. *Projektowanie algorytmów*

Gdy rozwiązanie jest gotowe, można je zapisać, aby można je było wykonać krok po kroku. Ułatwia to uzyskanie oczekiwanych rezultatów. Projektowanie algorytmów

polega na określeniu kroków i zasad, których należy przestrzegać, aby za każdym razem osiągnąć ten sam pożądany rezultat.

Źródło: https://code.org/curriculum/course3/1/Teacher.pdf

Istotą jest to, że CT tworzy procedury, które pozwalają stosującej je osobie osiągnąć postawione cele w uprzednio zdefiniowanym kontekście. Dlatego CT jest użytecznym narzędziem intelektualnym dla każdego, niezależnie od wykonywanej pracy.

1.3. Algorytmy i kodowanie - dlaczego myślenie komputacyjne jest tak cenne

Myślenie komputacyjne pomaga nam radzić sobie z problemami poprzez ich uogólnianie.

Problemy wynikające z braku myślenia komputacyjnego pojawiają się w irracjonalnych zachowaniach: nie jesteśmy robotami, ponieważ kierują nami emocje, ale racjonalne podejście do problemu upraszcza jego rozwiązanie. Ponadto rozwijanie takiego podejścia sprawia, że dzieci są bardziej skłonne do współpracy z rówieśnikami.

Zgodnie z myśleniem komputacyjnym, **rozwiązanie problemu znajduje się poprzez sformalizowanie go w sekwencji działań w celu przekazania go innym**: praktycznym przykładem jest przepis z książki kucharskiej. Musimy postępować zgodnie z precyzyjnymi krokami, aby upiec ciasto, a jeśli wiemy, jak jasno i jednoznacznie przekazać procedurę, ciasto może być również zrobione przez inne osoby.

Algorytmy i kodowanie są ściśle związane z myśleniem komputacyjnym, ale czym one są?

Algorytmy są jednym z największych osiągnięć ludzkości. Algorytm to rygorystyczny proces rozwiązywania problemu lub realizacji pomysłu. Algorytmy są podstawą większości naszych codziennych czynności. To dzięki ich nieświadomemu zastosowaniu wiemy, jak obliczyć sumę dwóch liczb, znaleźć nazwisko na liście, zdecydować, jaką drogą podążać, by dotrzeć do jakiegoś miejsca. CT to nic innego jak umiejętność rozumienia, stosowania i tworzenia algorytmów. Praktyka kodowania pozwala każdemu zapoznać się z algorytmami.

Kodowanie to słowo, które odpowiada słowu „programowanie". Oznacza oczywiście programowanie w sensie informatycznym, ale nie w bardziej tradycyjnym sensie. Kodowanie w szkole to niedawne odkrycie.

Jest to podejście, które stawia szeroko rozumiane programowanie w centrum ścieżki, w której nauka, począwszy od pierwszych lat życia, jest otwarta na nowe ścieżki i znajduje się w centrum perspektywy, która przełamuje bariery informatyczne i stymuluje komputacyjne podejście do rozwiązywania problemów. Należy stwierdzić, że CT to nietypowe podejście do problemów i ich rozwiązywania. **Dzięki kodowaniu dzieci i młodzież rozwijają CT i umiejętność rozwiązywania mniej lub bardziej złożonych problemów. Uczą się nie tylko kodować, ale także komputacyjnie podchodzić do uczenia się**.

Kodowanie można również ćwiczyć **bez urządzeń cyfrowych**, w analogicznych kontekstach: w tym przypadku mówimy o „**kodowaniu unplugged"**. Dyscyplina ta jest szczególnie **odpowiednia dla uczniów od przedszkola do szkoły średniej I stopnia** (Midoro, 2016).

Obecnie wiele szkół wskazuje wśród zajęć, które rezerwują dla dzieci, godziny kodowania i myślenia komputacyjnego. Jest to potrzebne, ponieważ uczniowie bardzo często przychodzą do szkoły średniej i na uniwersytet bez umiejętności zarządzania i radzenia sobie z problematycznymi sytuacjami.

Młodzi ludzie **mają trudności z interpretacją rzeczywistości**: być może dlatego, że są cyfrowymi tubylcami i odłożyli na bok te umiejętności, które kiedyś były używane do radzenia sobie z problemami. Do tego dochodzi fakt, że obecnie w społeczeństwie dominuje logika „***wszystko i natychmiast"***: jeśli problem jest trudny, odkłada się go na bok i stosuje strategię unikania.

Zaletą kodowania jest przybliżenie dzieciom technologii informacyjnej jako obiektu kulturowego, który jest coraz bardziej częścią naszego życia i świata pracy. Język ten musi być zatem zrozumiały i dzieci nie mogą go ignorować. Podobnie jak geografia, roboty również muszą wiedzieć: opanowanie kodowania w rzeczywistości pomaga zarządzać maszynami, wiedząc, jak pisać ich język, a nie tylko go czytać (Giacalone, 2020).

CT pomaga studentom rozwijać umiejętności, które są atrakcyjne dla przyszłych możliwości zatrudnienia. Informatyka jest najszybciej rozwijającym się rynkiem pracy, a studenci posiadający umiejętności kodowania są bardzo poszukiwanymi kandydatami do pracy. Podczas gdy twarde umiejętności techniczne są bardzo ważne, to bardziej miękkie umiejętności rozumowania i rozwiązywania problemów

są naprawdę atrakcyjne dla pracodawców. Te umiejętności są kluczem do zrozumienia, dlaczego myślenie komputacyjne jest tak cenne.

Naukowo-kulturowa strona informatyki, określana również jako „myślenie komputacyjne", pomaga rozwijać umiejętności logiczne i zdolność rozwiązywania problemów w kreatywny i skuteczny sposób, cechy, które są ważne dla przyszłych obywateli. Poprzez kodowanie rozwijane są zatem **umiejętności przekrojowe oraz uwaga, koncentracja, pamięć, kreatywność itp.** (Gabbari, Gagliardi, Gaetano, Sacchi, 2020).

Rozwiązywanie problemów kluczową umiejętnością w przyszłej pracy

Dzisiejszy rynek pracy wymaga pracowników zdolnych do rozwiązywania nierutynowych problemów (patrz Światowe Forum Ekonomiczne, 2016). Zgodnie z raportem na temat umiejętności osób dorosłych (PIACC Survey of Adult Skills) ludzie napotykają dziś coraz większe trudności w swojej pracy, co stwarza sytuacje, w których coraz częściej konieczne jest myślenie przed podjęciem działania.

Jednym z możliwych wyjaśnień przejścia akcentu z rutynowych na nierutynowe zadania w miejscach pracy jest to, że ze względu na powszechne wprowadzenie skomputeryzowanego sprzętu, pracownicy nie muszą już wykonywać rutynowych czynności ręcznie. Zamiast tego, muszą stawić czoła **nieoczekiwanym i nieznanym problemom** w zarządzaniu maszynami.

Nauczyciele odkrywają jednak często, że podczas gdy ich uczniowie wyróżniają się w rutynowych ćwiczeniach, nie radzą sobie z rozwiązywaniem problemów, z którymi nigdy wcześniej się nie zetknęli. Umiejętność rozwiązywania problemów jest istotnym elementem umiejętności wymaganych do analitycznego podejścia do zadań interpersonalnych i nierutynowych. W obu przypadkach ludzie muszą myśleć o tym, jak poradzić sobie z sytuacją, systematycznie monitorować efekt swoich działań i odpowiednio je modyfikować.

Począwszy od szkoły podstawowej, możemy stwierdzić, że myślenie komputacyjne uczy myślenia w sposób algorytmiczny, znajdowania rozwiązania i rozwijania go, a dzieje się tak dzięki programowaniu, kodowaniu. W rzeczywistości **kodowanie daje dzieciom sposób myślenia, który pozwoli im radzić sobie ze złożonymi problemami, gdy będą starsze.**

W ten sam sposób, w jaki nie uczymy muzyki w szkołach, aby uczniowie stali się profesjonalnymi skrzypkami, ani języka angielskiego, aby uczniowie zdobyli pracę w dziennikarstwie, nie powinniśmy uczyć dzieci programowania, aby zdobyły pracę programisty: powinniśmy to robić, aby **zdobyły nowy sposób myślenia i postrzegania świata.**

1.4. Zastosowanie myślenia komputacyjnego w klasie

Koncepcje myślenia komputacyjnego opisane powyżej (myślenie algorytmiczne, dekompozycja itp.) wiążą się z kilkoma zachowaniami ze strony uczniów, które można zaobserwować w klasie (Computing at School – CAS, 2015).

Myślenie algorytmiczne

Myślenie algorytmiczne reprezentuje zdolność do myślenia w kategoriach sekwencji i reguł, jako sposób rozwiązywania problemów. Jest to podstawowa kompetencja, którą uczniowie rozwijają, gdy uczą się pisać własne programy komputerowe. W klasie można zaobserwować:

- Formułowanie poleceń w celu uzyskania pożądanego efektu.
- Sformułowanie instrukcji, które należy wykonać w określonej kolejności (sekwencji).
- Formułowanie poleceń przy użyciu operacji arytmetycznych i logiki.
- Pisanie sekwencji poleceń, które zapamiętują, przenoszą i manipulują danymi (zmienne i przypisanie).
- Pisanie poleceń, które wybierają między kilkoma instrukcjami, które tworzą (wybór).

Źródło: AlgoBlocks - Suzuki, H., & Kato, H. (1993): https://www.edutech.it/education/blog-edu/item/41-coding-tangibile-la-sua-evoluzione-e-i-vantaggi-di-utilizzo.html

- Pisanie poleceń, które powtarzają grupy instrukcji, które tworzą (pętla/powtarzanie).
- Grupowanie i nazywanie zestawu poleceń, które wykonują określone zadanie w celu utworzenia nowej instrukcji (podprogramy, procedury, funkcje, metody).
- Pisanie poleceń, które zawierają podprogramy wykorzystujące repliki samych siebie (algorytm rekurencyjny).
- Pisanie serii poleceń, które mogą być wykonywane jednocześnie przez kilku wykonawców (komputery/ludzi, równoległe myślenie i procesy, konkurencja).
- Napisanie zestawu deklaratywnych reguł (kodowanie w Prolog lub w języku zapytań do bazy danych).

Oznacza to również:

- Użycie odpowiednich sformułowań do napisania kodu reprezentującego jedno z powyższych poleceń.
- Tworzenie algorytmów do testowania hipotez.
- Tworzenie algorytmów, które zapewniają rozwiązania oparte na doświadczeniu (heurystyka).
- Tworzenie algorytmicznych opisów rzeczywistych procesów zachodzących w świecie, w celu jego lepszego zrozumienia (modelowanie obliczeniowe).
- Projektowanie rozwiązań algorytmicznych, które uwzględniają możliwości, ograniczenia i potrzeby osób, które z nich korzystają, przynosi korzyści.

Dekompozycja

Dekompozycja to sposób myślenia o artefaktach w kategoriach części, które je tworzą. Poszczególne części mogą być rozumiane, rozwiązywane, rozwijane i oceniane oddzielnie. I

W klasie można obserwować:

- Podział artefaktów na części składowe w celu ułatwienia pracy.
- Rozbicie problemu na prostsze wersje, które można rozwiązać w ten sam sposób (metoda *divide et impera* i metoda rekurencyjna).

Uogólnienie

Generalizacja to sposób rozwiązywania nowych problemów na podstawie rozwiązań poprzednich. Chodzi o identyfikowanie i wykorzystywanie modeli. W klasie można zaobserwować:

- Identyfikację wzorców i wspólnych cech artefaktów.
- Dostosowanie rozwiązań lub ich części w celu zastosowania ich do całej klasy podobnych problemów.
- Transfer pomysłów i rozwiązań z jednego obszaru problemowego do innego.

Abstrahowanie

Abstrahowanie to proces tworzenia szerszego zrozumienia problemu poprzez pomijanie szczegółów. W klasie można przeprowadzić:

- Zmniejszenie złożoności poprzez wyeliminowanie niepotrzebnych szczegółów.
- Wybór sposobu reprezentowania szczegółowych obiektów (artefaktów), aby można było nimi manipulować w użyteczny sposób.
- Zmniejszenie złożoności skomplikowanych obiektów (ukrycie złożoności funkcjonalnej).
- Zmniejszenie złożoności danych, np. przy użyciu struktur danych.
- Identyfikację relacji między uogólnieniami (abstrakcjami).
- Filtrowanie informacji podczas opracowywania rozwiązań.

Ocena

Ewaluacja to proces sprawdzania, czy rozwiązanie jest prawidłowe, czyli odpowiednie do celu. Podczas oceny opartej na myśleniu komputacyjnym, istnieje szczególna i często ekstremalna dbałość o szczegóły. W klasie warto zwrócić uwagę na:

- Ocenę, czy obiekt (artefakt) jest odpowiedni do celu.
- Określenie, który artefakt wykonuje właściwą funkcję (poprawność funkcjonalna).
- Projektowanie i wykonywanie zaplanowanych testów oraz interpretacja wyników (testów).
- Ocenę, czy działanie artefaktu jest wystarczająco dobre (użyteczność: skuteczność i wydajność).
- Porównanie wydajności artefaktów uruchamiających tę samą funkcję.
- Kompromis między sprzecznymi potrzebami.

- Ocenę, czy artefakt jest łatwy w użyciu dla ludzi (użyteczność).
- Określenie, czy artefakt oferuje odpowiednio pozytywne doświadczenie podczas użytkowania (doświadczenie użytkownika).
- Ocenę któregokolwiek z powyższych pod kątem specyfikacji i ustalonych kryteriów.
- Przejście przez procesy lub algorytmy/algorytmy krok po kroku, aby przetworzyć to, co robią (dowody/śledzenie).
- Użycie precyzyjnego argumentu do uzasadnienia, że algorytm działa (test).
- Używanie precyzyjnej argumentacji do testowania użyteczności lub wydajności artefaktu (ocena analityczna).
- Korzystanie z metod, które obejmują obserwację artefaktów wykorzystywanych do oceny ich użyteczności (ocena empiryczna).
- Ocenę, czy produkt spełnia ogólne kryteria wydajności (heurystyka).

1.5. Podejścia edukacyjne

Zauważono, że w szkole można pracować nad rozwojem CT, na przykład prosząc uczniów o mentalne „rozmontowanie" niektórych czynności, które wykonują automatycznie, i zapisanie poprawnej, rygorystycznej procedury, która pozwoliłaby robotowi („kosmicie", jeśli uczniowie tak wolą) odtworzyć tę samą czynność bez błędów: na przykład narysować prostokąt bez odrywania długopisu od papieru lub wyszukać słowo w słowniku (Gabbari, Gagliardi, Gaetano, Sacchi, 2020).

Istnieje nieskończona liczba ćwiczeń bez podłączenia do sieci, które można wykonywać w klasie stopniowo i z dużym zaangażowaniem. Przykładowo, we Włoszech, w zaktualizowanych podstawach programowych (*Nowe Scenariusze* 2018) podkreśla się wszechobecną funkcję CT: „*Jest to kreatywny proces logiczny, który, mniej lub bardziej świadomie, jest wprowadzany w życie codzienne, aby stawiać czoła problemom i je rozwiązywać. Edukacja do świadomego działania: ta strategia pozwala nauczyć się radzić sobie z sytuacjami w sposób analityczny, rozkładając je na różne aspekty które je charakteryzują, planując najbardziej odpowiednie rozwiązania dla każdego z nich*".

Od myślenia komputacyjnego do kodowania

Ale to przede wszystkim w działaniach związanych z kodowaniem, a zatem w pisaniu języków przeznaczonych dla maszyny, CT może znaleźć dużo miejsca na rozwój. Komputery są idealnymi wykonawcami, nie są obdarzone inteligencją. Dlatego też pisanie instrukcji, które maszyna będzie musiała w pewnym stopniu wykonać, wymaga większego stopnia formalności i rygoru niż w przypadku komunikacji między ludźmi.

Programowanie sprawia, że koncepcje CT stają się konkretne i stają się narzędziem do nauki.

Istnieją systemy programowania *tekstowego* i systemy programowania *wizualnego*: w tych pierwszych instrukcje (dla maszyny) muszą być pisane w sekwencji za pomocą edytora tekstu; z drugiej strony w systemach wizualnych poszczególne instrukcje są reprezentowane przez kolorowe bloki, które można przeciągać do obszaru roboczego

(przeciągnij i upuść). Bloki można łączyć ze sobą w celu utworzenia sekwencji instrukcji, która stanowi program (Gabbari, Gagliardi, Gaetano, Sacchi, 2020).

Systemy wizualne są często preferowane przez nauczycieli, ponieważ programowanie blokowe pozwala skupić się wyłącznie na procedurze, nie biorąc pod uwagę poprawności języka: bloki są już poprawnie połączone z punktu widzenia składni (co najwyżej można popełnić pewne błędy semantyczne) i można skupić się na rozumowaniu. Zadaniem nauczyciela jest zaoferowanie odpowiedniego pretekstu i kontekstu.

Jak wybrać język programowania do kodowania edukacyjnego? Większość nauczycieli wydaje się być zorientowana na edukacyjny język oprogramowania, który oferuje „niską podłogę i wysoki sufit", jak mówi S. Papert, czyli język, który ułatwia pierwsze kroki w jak największym stopniu, ale jednocześnie pozwala na realizację coraz bardziej złożonych projektów (Gabbari, Gagliardi, Gaetano, Sacchi, 2020).

Jak poważne gry wspierają szkolenia

Dzięki poważnym grom szkolenia stają się potężnym i wszechstronnym narzędziem. Zwłaszcza w erze, w której duża ilość informacji powoduje, że użytkownicy łatwo się rozpraszają, kursy szkoleniowe tworzone przez firmy, podmioty lub instytucje odpowiadają na główną potrzebę, a mianowicie uczynienia szkolenia tak skutecznym, szybkim i angażującym, jak to tylko możliwe. W tym kontekście trendem stało się wzbogacanie szkoleń o filmy wideo, kreskówki, gry online i odgrywanie ról.

Wszystkie te treści multimedialne są wykorzystywane przez użytkowników w każdym wieku do nauki i bycia na bieżąco w kontekście zabawy i rozrywki. W ten sposób zarówno milenialsi i pokolenie Z, lubiący gry wideo, jak i bardziej dorośli mogą bawić się i uczyć, grając w poważne gry. W końcu nauka przez zabawę zawsze była jednym z najgłębiej zakorzenionych mechanizmów u ludzi.

Będąc interaktywnym rodzajem nauki, poważne gry pozwalają graczowi **uczyć się poprzez działanie** i tworzyć własne treści. Dzięki uczeniu się przez działanie w poważnych grach, uczący się znajdzie się przed wieloma scenariuszami decyzyjnymi, interakcjami z obiektami dydaktycznymi napotkanymi podczas gry i informacjami zwrotnymi otrzymanymi od innych. Uczący się otrzyma wynik oparty na poprawności decyzji, którą ma podjąć. Możliwe jest również uzyskanie odznak lub medali w oparciu o osiągnięte cele **(Luciano i Fabio, 2017).**

Kodowanie i myślenie komputacyjne w szkole podstawowej i przedszkolu: narzędzia

Jak odbywa się kodowanie w szkole, jakie narzędzia są dostępne? Mogą to być zabawne narzędzia, takie jak Scratch lub Scratch Jr. dla najmłodszych i Kodi (Luciano i Fabio 2017*)* lub korzystanie z ćwiczeń **code.org.** Bardziej niż ćwiczenia wyglądają jak gry. I faktycznie z pewnego punktu widzenia nimi są. Dzieci grają, a wygranie każdego wyzwania oznacza rozwiązywanie problemów. Małych problemów, takich jak ominięcie przeszkody lub złapanie przez jednego z bohaterów, złoczyńców z historii, by podać tylko kilka przykładów. Aby rozwiązać problem, muszą pracować,

aby zrozumieć, jakie jest możliwe rozwiązanie, a jeśli osiągną cel, nauczą się, jak to zrobić. W międzyczasie nieświadomie napisali linijki kodu komputerowego, nawet jeśli materialnie nie napisali żadnego, nawet jednego, a jedynie przesunęli prostokątne bloki, z których każdemu odpowiada funkcja i kod.

Jednak strategią, na której większość nauczycieli chce się skupić, jest wykorzystanie problemu matematycznego jako sytuacji doświadczalnej, w celu stymulowania dzieci do angażowania się w identyfikowanie strategii rozwiązania, być może budowanej wspólnie z grupą towarzyszy i dzielonej z całą klasą. Aby to zrobić, potrzebne są złożone działania, które nie są ćwiczeniami mającymi na celu po prostu zastosowanie wzoru, ale propozycjami mającymi na celu szkolenie krytycznego myślenia, opracowywania, dzielenia się i refleksji nad własną pracą. I tutaj bardzo dobrą pomocą mogą być problemy, które mają wiele możliwych rozwiązań, lub te „otwarte", gdzie końcowy rezultat nie jest unikalny.

Istnieje wiele możliwych propozycji, ale jak zawsze nauczyciele muszą radzić sobie z czasem, który chcą poświęcić na aktywność i rodzajem klasy, w której pracują. Mogą to być **punkty krytyczne, ale istnieje** wiele możliwych propozycji operacyjnych do wyboru i dostosowania do konkretnego celu i środowiska.

Pomysłem może być zaproponowanie dzieciom, podzielonym na grupy, aby zidentyfikowały ścieżkę rozwiązania problemu, a następnie udzieliły wskazówek dotyczących rozwiązania robotowi-towarzyszowi, który nie zna tekstu: uczniowie z pewnością będą się dobrze bawić, a w międzyczasie będą zmotywowani do poprawienia siebie w przypadku błędu. Nieco bardziej złożonym ćwiczeniem może być stworzenie „sztafety" problemów: jedna grupa je wymyśla, druga rozwiązuje, a trzecia poprawia. Końcowe porównanie i zachęta do zidentyfikowania innych możliwych strategii będą przydatne do skupienia uwagi na różnych etapach rozwiązywania problemów i na możliwych alternatywach (Luciano i Fabio, 2017*)*.

Innym pomysłem może być poproszenie każdej grupy uczniów o wybranie problemu spośród tych zaproponowanych przez nas, prosząc ich o napisanie kroków niezbędnych do jego rozwiązania (Luciano i Fabio 2017*)*.

Przydatnym narzędziem do wykorzystania w tym przypadku może być stary dobry schemat blokowy, liniowy lub nie, który trudno znaleźć w podręcznikach i zeszytach szkoły podstawowej, ale który może być doskonałym źródłem wizualizacji obranej ścieżki (Luciano i Fabio 2017*)*. W końcu, jak zdefiniować pasek poleceń tworzony za pomocą programu Scratch, jeśli nie za pomocą cyfrowego schematu blokowego?

Georginia Boyd
Aspire-Igen Group Ltd.

2. Dekompozycja

2.1. Definicja dekompozycji

Dekompozycja to kolejna z podstawowych umiejętności w myśleniu komputacyjnym, będąca jednym z czterech kluczowych elementów kursu Google „Computational Thinking for Educators" (Yihaun, 2019). Jeśli myślenie komputacyjne jest „procesem zaangażowanym w formułowanie problemów, dzięki czemu rozwiązania mogą być reprezentowane jako kroki obliczeniowe i algorytmy", to dekompozycja jest istotnym krokiem w myśleniu komputacyjnym, polegającym na rozbijaniu danych, procesów lub problemów na mniejsze, łatwiejsze do zarządzania kroki lub części, aż użytkownik będzie pewny swojej zdolności do wykonania rozwiązania (Alder i in., 2022; Rich i in., 2020; Mannila, 2014, Aho, 2012; Csizmadia i in., 2015; Sengupta i in., 2013; Yihaun, 2019). Biorąc pod uwagę taką definicję, prawdopodobnie każdy zda sobie sprawę, że stosuje dekompozycję w swoich procesach myślowych i działaniach w życiu codziennym. Za każdym razem, gdy dzielimy zadania na możliwe do zarządzania części, z powodzeniem ćwiczymy umiejętności dekompozycji.

2.2. Korzyści z dekompozycji w środowisku edukacyjnym

Jak można przypuszczać, umiejętności dekompozycji mają wiele zalet w środowisku edukacyjnym. Trzy z nich, omówione już częściowo przy okazji opisu rozpoznawania wzorców, odnoszą się także do dekompozycji. Są to: to poprawa funkcji poznawczych uczniów, zwiększenie zdolności nauczycieli do przekazywania złożonych tematów oraz możliwość przewidywania osiągnięć i trudności edukacyjnych uczniów .

Włączenie umiejętności dekompozycji do planu lekcji może pomóc poprawić wydajność poznawczą uczniów, zwiększając ich pewność siebie podczas radzenia sobie z większymi i bardziej złożonymi teoriami (Weintrop et al., 2016). W ten sam

sposób, w jaki przeżuwamy jedzenie, dzielenie złożonych zadań na łatwiejsze do opanowania porcje pomaga zapewnić, że uczniowie zaangażują się w lekcje i „strawią" lub zaoamiętają umiejętności w nich przekazywane. Jak stwierdził nauczyciel w badaniu przeprowadzonym przez Alder: „[Dekompozycja] jest świetna, gdy próbuję nauczyć skomplikowanego tematu i mogę podzielić go na znacznie mniejsze części, aby uczniowie mogli go zrozumieć" (Alder i in., 2022).

Umiejętności dekompozycji przynoszą korzyści nie tylko uczniom, ale także nauczycielom. We wspomnianym wyżej badaniu Uzumcu (2021), nauczyciele biorący udział w badaniu byli zgodni co do tego, że dekompozycję można łatwo włączyć do istniejących planów lekcji i że odgrywa ona istotną rolę w rozwiązywaniu problemów i planowaniu lekcji (Uzumcu, 2021). Wynikało to po części z faktu, że dekompozycja ułatwia nauczycielom podzielenie kurs na moduły, którymi można łatwiej zarządzać. Oprócz pomocy nauczycielom w planowaniu i prowadzeniu lekcji, dekompozycja może stanowić papierek lakmusowy skuteczności kursu, ponieważ zdolność ucznia do zastosowania dekompozycji jest dobrym predyktorem sukcesu szkolnego ucznia (Haddad & Kalaani, 2015 ... na podstawie danych od 982 studentów w ciągu dwóch lat). Podobnie nauczyciele badani przez Uzumcu deklarowali zwiększoną zdolność przewidywania i rozumienia poziomu zrozumienia tematów przez ucznia dzięki skutecznemu badaniu umiejętności dekompozycji (Uzumcu, 2021). To z kolei daje nauczycielom czas na wspieranie klasy w szerokim zakresie umiejętności.

2.3. Pedagogiczne korzyści płynące z dekompozycji

Jednym z powodów, dla których umiejętności dekompozycji są chętnie włączane do lekcji, jest to, że są one w dużej mierze stosowane w nauczaniu przedszkolnym i wczesnoszkolnym (Calderon, 2015), czyniąc przedmioty i ich tematy „strawnymi".

2.4. Długoterminowe korzyści z nauki umiejętności dekompozycji

Należy zauważyć, że umiejętności dekompozycji, nie są niezależnym przedmiotem ani niezależnym tematem nauczania (Hsu, 2018). Jest ono wykorzystywane w życiu codziennym, a korzyści płynące z wdrożenia umiejętności dekompozycji zostały powszechnie uznane przez naukowców i nauczycieli (Hsu, 2018; Lockwood, 2017) Na przykład uczestnicy badania Uzumcu stwierdzili, że byli w stanie wykorzystać nabyte umiejętności w życiu codziennym i zawodowym, jak prawidłowo zintegrować dekompozycję z obowiązkami zawodowymi (Uzumcu, 2021). Innym przykładem korzyści pozaszkolnych jest umiejętność rozpoznania luk w zrozumieniu zagadnienia. Umiejętności dekompozycji pozwalają uczniom samodzielnie określić, co rozumieją, a nad czym muszą popracować. Zwiększa to pewność siebie i daje uczniom wskazówki do właściwego rozplanowania czasu i zasobów.

2.5. Jak zintegrować dekompozycję z lekcjami?

Podstawowym sposobem włączania dekompozycji do lekcji jest „podpowiadanie" uczniom, na czym polega ta strategia, a następnie zachęcanie do korzystania z niej podczas rozwiązywania problemów (Rich i in., 2020). Można to zrobić, prosząc uczniów o wskazanie, gdzie i w jaki sposób wykorzystali daną technikę oraz w jaki sposób przyczyniło się to do rozwiązania problemów (Rich i in., 2020). Na lekcji astronomii nauczyciel może np. poprosić ucznia o zidentyfikowanie planety na podstawie szczegółowych parametrów, zamiast ogólnych właściwości. Następnie pyta uczniów, jak przeprowadzili identyfikację, „rozbijając" analizę na mniejsze czynności składowe, łatwiejsze do wykorzystania w innych sytuacjach (Rich i in., 2020). Zasadniczo podejście to opiera się na wykorzystaniu praktyk dekompozycji jako ogólnych strategii problemowych, które można zastosować do różnych kwestii (Rich i in., 2020). Kluczem jest tutaj unikanie opisywania uczniom, w jaki sposób problem został zdekomponowany, a zamiast tego zapewnienie możliwości samodzielnego poszukiwania wiedzy i rozwiązań (Alder i in., 2022).

Przykładowo, nauczyciel może włączyć dekompozycję do lekcji matematyki. Najpierw dzieli uczniów na grupy, a następnie pyta, jak rozłożą dużeą liczbę na czynniki pierwsze. Następnie prosi uczniów, aby rozwiązali problem, analizując równanie po jednej cyfrze naraz. Po wykonaniu tej czynności wyjaśnij uczniom, że był to przykład dekompozycji w praktyce (Rich et al., 2020).

Inną metodą stosowaną w celu zintegrowania dekompozycji do programu nauczania jest użycie bodźców wizualnych i dotykowych, takich jak schematy blokowe w formie kloców lub plansz do zobrazowania dekompozycji (Krist i in., 2017). Inną techniką jest „kadrowanie" dekompozycji i rozpoznawania wzorców w ramach lekcji. Zasadniczo technika ta, choć bardzo podobna do podpowiedzi, polega na informowaniu uczniów o technice dekompozycji, podawaniu przykładów, jak ją zastosować (przygotowanie uczniów do myślenia o zastosowaniu dekompozycji), zanim uczniowie faktycznie zaangażują się w ćwiczenie dekompozycji w dalszej części lekcji (Rich i in., 2020). Kluczowa różnica polega na tym, że podczas gdy podpowiadanie polega na mówieniu uczniom, kiedy użyć techniki, kadrowanie daje uczniom szansę na samodzielne jej zastosowanie, bez podpowiedzi. Kadrowanie może być również wykorzystywane jako narzędzie refleksji po lekcji.

Zajęcia uwzględniające dekompozycję w sposób w pełni zintegrowany mogą wyglądać następująco: zaczynamy od odkrycia różnic między danymi, które chcemy, aby uczniowie przeanalizowali, a informacjami, które obecnie mają pod ręką. Zachęcamy do aktywnej, werbalnej refleksji nad kryteriami wydzielania przez uczniów pewnych aspektów zadania, być może poprzez poproszenie ich o wyjaśnienie tego swoim partnerom. Na przykład na zajęciach z języka angielskiego nauczyciele mogą poprosić uczniów o zaplanowanie eseju poprzez podzielenie tematu na podtytuły, a następnie wyjaśnienie, dlaczego dokonali tych rozróżnień. Podczas zajęć plastycznych nauczyciele mogą wyjaśnić i pokazać korzyści płynące z podejścia do przedstawiania obiektów przez pryzmat gry świateł i cieni.

Niezależnie od tego, w jaki sposób nauczyciele zdecydują się zająć dekompozycją, ważne jest, aby wyjaśnić, dlaczego dekompozycja jest ważna: ćwiczenia na odkrywanie i uczenie się krok po kroku pomagają w nauce i zrozumieniu złożonych tematów (Lockwood, 2017). Ponownie, łączenie uczniów w pary może być tutaj przydatne, ponieważ sprawi, że odkrywanie nowych umiejętności będzie wygodniejsze i pozwoli parom pomagać, odświeżać, dyskutować i sprawdzać nawzajem swoją wiedzę (Isbell et l., 2010). Podobnie, spróbuj stworzyć plany lekcji zaprojektowane tak, aby pokazać, w jaki sposób umiejętności rozkładu już istnieją w życiu uczniów (Burgett 2016).

2.6. Włączanie dekompozycji do lekcji

Badania pokazują, że nauczycielom łatwiej jest włączać strategie myślenia komputacyjnego do bogatych, otwartych zadań, które zostały celowo zaprojektowane tak, aby angażować uczniów w myślenie na wyższym poziomie (Rich et al., 2020.) Niestety, wiele podstawowych programów nauczania nie zapewnia nauczycielom dostępu do takich zadań (Rich et al., 2020; Banilower et al., 2013; van Zanten & van den Heuvel-Panhuizen, 2018). Pierwszym krokiem jest zatem upewnienie się, że każdy wystandaryzowany program nauczania pozwala na eksplorację myślenia teoretycznego na wyższym poziomie, przede wszystkim poprzez zapewnienie uczniom czasu na refleksję nad wyuczonymi lekcjami, zamiast skupiania się na czystych wynikach z ocen szkolnych.

W związku z tym prawdopodobnie będzie trzeba przeprojektować lekcje w celu wprowadzenia możliwości prowadzenia dekompozycji. Choć na początku może to być zniechęcające, da to nauczycielom szansę na dalsze dopracowanie lekcji, umożliwiając im dogłębne przemyślenie instrukcji i możliwości zaangażowania, które zapewniają (Rich i in., 2020). Zarówno uczniowie, jak i nauczyciele uczą się tylko poprzez zaangażowanie i stosowanie tych technik (McCormick, 2022).

Georginia Boyd
Aspire-Igen Group Ltd.

3. Rozpoznawanie wzorców

3.1. Definicja rozpoznawania wzorców

Jednym z kluczowych elementów myślenia komputacyjnego jest rozpoznawanie wzorców, czyli formalnie rzecz ujmując, identyfikacja danych z jednym lub wieloma podobieństwami w sekwencji (Howard, W.R., 2007). Inne robocze definicje obejmują szukanie podobieństw między nowymi problemami i problemami, które zostały już rozwiązane, szukanie podobieństw i wzorców między rzeczami lub identyfikację danych z jednym lub większą liczbą podobieństw w sekwencji (Rich i in., 2020; Howard, W.R., 2007).

W praktyce, rozpoznawanie wzorców pozwala użytkownikowi na szybkie rozwiązywanie problemów poprzez zastosowanie rozwiązań, które sprawdziły się w przeszłości, pod warunkiem, że problem, przed którym obecnie stoi dana osoba, jest podobny do tych, które napotkała w przeszłości. Wymaga to oczywiście od użytkownika obserwowania i identyfikowania wzorców, trendów i prawidłowości w danych, procesach lub problemach (Yihaun, 2019). Większość nauczycieli i wychowawców prawdopodobnie zna już rozpoznawanie wzorców. Praktyczne przykłady obejmują pytanie uczniów: „Czego nauczyliśmy się w przeszłości, co może pomóc nam znaleźć rozwiązanie obecnego problemu?".

Chociaż istnieją proste zastosowania rozpoznawania wzorców w głównym nurcie uczenia się – głównie w matematyce, naukach ścisłych, sztuce – istnieje wiele zastosowań w szkoleniu zawodowym i w świecie pracy. W szkoleniu zawodowym rozpoznawanie wzorców jest nieodłącznym elementem, umożliwiającym uczniom osobiste wykonywanie prac manualnych w celu rozwijania ich umiejętności. Rozwijając umiejętności rozpoznawania wzorców, uczniowie mogą odnieść bieżące lekcje do swojej przyszłej kariery i zyskać perspektywę tego, co jest potrzebne, aby odnieść sukces w danej dziedzinie.

3.2. Rozpoznawanie wzorców - korzyści dla nauczycieli

Korzyści specyficzne dla przedmiotu

Podczas gdy znaczenie rozpoznawania wzorców można dostrzec w tematach takich jak matematyka, informatyka i nauki ścisłe, nauczanie rozpoznawania wzorców może wspierać uczniów we wszystkich przedmiotach. Rozpoznawanie wzorców to po prostu metoda myślenia, a jej zastosowania są szerokie i wszechstronne.

Na przykład w badaniu Uzumcu, w którym grupa nauczycieli stosowała różne techniki myślenia komputacyjnego, w tym rozpoznawanie wzorców w swoich

planach lekcji, wszyscy uczestnicy uważali, że ćwiczenia były pomocne w rozwijaniu umiejętności rozwiązywania problemów (Uzumcu, 2021).

Aby dać kontekst i jasne przykłady, podkreśliliśmy kilka przedmiotów, które korzystają z rozpoznawania wzorców. Należą do nich matematyka, nauki ścisłe, sztuka i kreatywność oraz języki – choć należy zauważyć, że rozpoznawanie wzorców ma zastosowanie do wszystkich przedmiotów.

W matematyce rozpoznawanie wzorców jest często wykorzystywane i bez wysiłku łączy się z innymi umiejętnościami myślenia komputacyjnego, takimi jak myślenie algorytmiczne. Badania pokazują, że zdolność młodych ludzi do rozpoznawania wzorców stanowi podstawę wczesnej teorii matematycznej. Kiedy młodzi ludzie nauczą się identyfikować podobieństwa, będą w stanie zastosować tą umiejętność we wszystkich przedmiotach. W związku z tym korzystna jest ekstrapolacja lekcji wyniesionych z matematyki, tj. jak rozpoznawać i łączyć wzorce, dostosowywać się i znajdować nieregularności, do innych przedsięwzięć (Departament Edukacji, 2014).

W naukach ścisłych szeroko zakrojone badania wykazały, że rozpoznawanie wzorców jest skuteczną metodą uczenia się umiejętności naukowych w dziedzinach takich jak medycyna, chemia i inżynieria (Samarasinghe, 2006). W rzeczywistości rozpoznawanie wzorców odgrywa kluczową rolę w teorii genetycznej i pomaga uczniom w konceptualizacji abstrakcyjnych i trudnych pojęć (Hsu, 2018).

W sztuce rozpoznawanie wzorców pomaga w przekazywaniu abstrakcyjnych i rozwojowych teorii. Na przykład rozpoznawanie wzorców pomaga w zrozumieniu rozwoju i analizy. (Shen, 2013). To zrozumienie może również pomóc w tworzeniu sztuki.

Ale być może najbardziej powszechnym zastosowaniem rozpoznawania wzorców poza muzyką i matematyką jest lingwistyka. Ku wielkiemu przerażeniu przeciętnego ucznia, język jest nauczany prawie wyłącznie przy użyciu wzorców gramatycznych, przynajmniej w typowej szkole. Teoria głosi, że nauka języka przy użyciu rozpoznawania wzorców wspiera bardziej szczegółowe zrozumienie sposobu, w jaki działa język. Na przykład, podstawowe zrozumienie języków może być rozwijane i rozszerzane przy użyciu identyfikacji wzorców do definiowania struktury zdań, kategorii pisania i słownictwa (Larson, 2010). Oczywiście rozpoznawanie wzorców nie ogranicza się tylko do powyższego. Później szczegółowo opiszemy konkretne przykłady, które inni nauczyciele wykorzystali do włączenia rozpoznawania wzorców do swoich przedmiotów.

Korzyści pedagogiczne

Podobnie jak wiele umiejętności w myśleniu komputacyjnym, rozpoznawanie wzorców jest umiejętnością podstawową dla skutecznej edukacji. Rozpoznawanie wzorców jest istotne w prawie każdym aspekcie naszego życia i wczesnej edukacji, a działania pedagogiczne dotyczące dzieci we wczesnych latach życia zazwyczaj integrują rozpoznawanie wzorców z tradycyjnymi dydaktycznymi praktykami nauczania, takimi jak uczenie się eksploracyjne, zadawanie pytań, umiejętności budowania ogólnego schematu i poszukiwania szczegółowych wiadomości (Calderon, 2015).

Jak wspomniano wcześniej, siłą rozpoznawania wzorców jest to, że pozwala uczniom rozwiązywać problemy przy użyciu uniwersalnych metod. Dlatego też,

zamiast uczyć uczniów odpowiedzi na każdy problem z osobna, nauczyciele powinni uczyć rozpoznawania wzorców i odpowiedniego rozwiązania. Pomaga to uczniom rozwijać umiejętności szybszego rozumienia złożonych pojęć poprzez demonstrowanie podobnych wzorców, występujących w różnych przedmiotach. Klasycznym przykładem jest nauczanie tabliczki mnożenia, w szczególności mnożenia przez dziewięć. Tam, gdzie nauczyciel mógłby nauczyć uczniów każdej pojedynczej wartości w tabeli, może zamiast tego nauczyć wzorca: odpowiedź można uzyskać, zwiększając pierwszą cyfrę o wartość jeden, jednocześnie zmniejszając drugą cyfrę o wartość jeden, tj. dziewięć (09), osiemnaście (18), dwadzieścia siedem (27) itd.

Jak zasugerowano wcześniej, rozpoznawanie wzorców jest szczególnie przydatne w nauczaniu języków: istnieje wiele podobieństw (a zatem wzorców) między językami romańskimi: hiszpańskim, włoskim i francuskim, a także te podobieństwa w językach germańskich, w tym niemieckim, angielskim i holenderskim. Jeśli uczeń jest potrafi rozpoznawać wzorce i zna jeden z języków w rodzinie, np. francuski, może mu być łatwiej nauczyć się hiszpańskiego niż niderlandzkiego – ponieważ więcej wzorców i podobieństw (w słownictwie, gramatyce lub strukturze zdań) jest wspólnych w rodzinach językowych. Wiedza ta może pomóc uczniom przyspieszyć edukację, a nauczycielom dokładniej zaplanować lekcje.

Rozpoznawanie wzorców może również pomóc w kształceniu uczniów w zakresie bardziej abstrakcyjnych pomysłów, takich jak rozwój koncepcji. Opisywanie koncepcji jako unikalnej teorii oderwanej od innych może zniechęcać uczniów, ale nauczyciel może pokazać powiązanie poznawcze jednej teorii z innymi, wcześniej poznanymi. Lekcje mogą być łatwiejsze do zrozumienia, gdy pokazują proste postępy znanych koncepcji, przyspieszając w ten sposób przyswajanie wiedzy poprzez ekstrapolację znanych koncepcji.

Rozpoznawanie wzorców może również pomóc nauczycielom. Badania wskazują, że opanowanie przez ucznia podstawowych bloków konstrukcyjnych jest „unikalnym predyktorem umiejętności wyższego poziomu, wykraczającym poza wpływ umiejętności niższego poziomu" i działa jako prognostyk sukcesu akademickiego ucznia, uzyskania pozytywnej oceny z przedmiotu, prawdopodobieństwa postępów w edukacji i wskaźnika ukończenia szkoły (Lockwood, 2017; Haddad & Kalaani, 2015). Wiedza, na temat umiejętności rozpoznawania wzorców przez uczniów pozwala nauczycielom szybko identyfikować uczniów, którzy mogą potrzebować dodatkowego wsparcia.

Umiejętność ta nie tylko pozwala nauczycielom zidentyfikować tych uczniów, ale także pomaga im ich wspierać, ponieważ nauczyciele mogą szybko powielać strategie nauczania dla podobnych uczniów. Tam, gdzie jeden uczeń z pewnymi zdolnościami i stylami uczenia się może mieć trudności z konkretnym przedmiotem, nauczyciel będzie w stanie rozpoznać podobne cechy i wykorzystać wcześniejsze doświadczenia ze skutecznymi metodami z podobnymi uczniami, aby ich wspierać.

3.3. Rozpoznawanie wzorców - korzyści dla uczniów

Oprócz zwiększania pewności siebie uczniów i działania jako czynnik zwiększający przyswajanie przedmiotów, umiejętności nabyte w myśleniu obliczeniowym – w tym rozpoznawanie wzorców – funkcjonują jako podstawowe elementy składowe edukacji dziecka. Nauczanie tej umiejętności zapewnia uczniom umiejętności obliczeniowe i krytycznego myślenia niezbędne w XXI wieku, niezależnie od ich ostatecznego kierunku studiów lub zawodu (Mannila, 2014). W rzeczywistości powszechnie uznaje się, że zdolność do identyfikowania i analizowania struktur pod kątem wzorców jest podstawą zrozumienia sposobu przetwarzania informacji (Mannila, 2014). Nauczanie rozpoznawania wzorców jest podobne do uczenia uczniów, jak uczyć się i studiować samodzielnie – jest to niezbędna umiejętność w każdym aspekcie życia.

3.4. Wyzwania w nauczaniu rozpoznawania wzorców

Chociaż rozpoznawanie wzorców jest uwzględnione w istniejących systemach edukacyjnych, nie zawsze jest wyraźnie zidentyfikowane. Może być zatem trudno wyraźnie włączyć je do lekcji, zwłaszcza jeśli nauczyciel nie wie, jak wyeksponować istotę tej kompetencji (Hsu, 2018). Biorąc pod uwagę obecne wymagania wobec nauczycieli, zmiana materiałów dydaktycznych w krótkim czasie może być szczególnie trudna dla nauczycieli (Hsu, 2018).

W związku z tym edukatorom mogą przydać się sprawdzone przykłady integracji materiału edukacyjnego. Jeden z takich przykładów można znaleźć we Francji. W badaniu Chiprianova (2016) zintegrowano umiejętności myślenia komputacyjnego, w tym rozpoznawania wzorców, z francuską edukacją podstawową za pomocą gier i robotyki, wykorzystując dyskusje i demonstracje w całej klasie. Po nich następowało wspólne lub indywidualne uczenie się, a na koniec uczniowie byli zachęcani do refleksji nad rozwiązaniami, umiejętnościami i technikami, których się nauczyli (Chiprianov, 2016). Niektóre z korzyści płynących z tego wielopoziomowego podejścia to zwiększony wskaźnik promocji, a także zdolność stosowania przez uczniów umiejętności nabytych w klasie do alternatywnych wyzwań (Chiprianov, 2016).

Pewne udoskonalenia zostały przedstawione przez Burgetta (2016), który zasugerował tworzenie scenariuszy lekcji, mających na celu pokazanie, w jaki sposób umiejętności myślenia komputacyjnego są już wykorzystywane przez uczniów w codziennym życiu, a komputacyjne podejście do edukacji pomaga w przyswajaniu złożonych pojęć. To „praktyczne podejście" jest istotną przewagą, jeśli napotkasz problemy podobne do tych odnotowanych przez Mooney i in. (2014), którzy stwierdzili w swoim badaniu, że uczniowie uznali ideę myślenia komputacyjnego za trudną, jeśli przedstawiono ją jedynie jako teoretyczne i abstrakcyjne podejście. Nauczyciele mogą również „podpowiadać" uczniom techniki CT, zachęcając ich do korzystania ze konkretnej strategii podczas rozwiązywania problemu (Rich i in., 2020). Można to zrobić, prosząc ich o wskazanie, gdzie i w jaki sposób z nich korzystali oraz jak przyczyniły się one do rozwiązywania problemów. (Rich i in., 2020). Zasadniczo

podejście to opiera się na zdefiniowaniu rozpoznawania wzorców jako ogólnej strategii rozwiązywania problemów, którą można zastosować do różnych problemów (Rich i in., 2020).

Przykład praktycznego podejścia można znaleźć w badaniu Mannila (2014). Wykorzystano tam rymowanki, aby zachęcić uczniów szkół średnich do rozpoznawania wzorców. Rymowanki zostały wybrane, aby spróbować nauczyć uczniów rozpoznawania wzorców za pomocą medium, które już znali. Uczniom polecono przygotować znane rymowanki w domu, przeanalizować je, a następnie zidentyfikować w nich wzorce strukturalne (tj. prologi, powtarzające się refreny / frazy i epilogi). Następnie kazano uczniom wykorzystać te umiejętności rozpoznawania wzorców w zupełnie inny sposób, budując „maszyny-zabawki" do konstruowania rymowanek z kartonu i innych tanich materiałów (w rzeczywistości karty pracy), z zapisanymi na nich wyżej wymienionymi elementami wzorców. Następnie uczniowie byli zachęcani do „mieszania i dopasowywania" rymowanek zgodnie z wzorcami, które zidentyfikowali wcześniej, aby stworzyć zupełnie nowe rymowanki (Mannila, 2014). Pozwoliło to uczniom zacząć stosować empirycznie i świadomie to, co wcześniej było podświadomą, niewypowiedzianą teorią.

3.5. Włączanie rozpoznawania wzorców do lekcji

Pomimo wspomnianych wyżej potencjalnych niedociągnięć, związanych z szybką integracją nowej dla uczniów techniki, rozpoznawanie wzorców zostało z powodzeniem zintegrowane z edukacją szkolną na całym świecie (Yihaun, 2019). Ta udana integracja nie ogranicza się do tradycyjnej informatyki lub kursów STEM, ponieważ rozpoznawanie wzorców ma nie tylko kluczowe znaczenie dla myślenia komputacyjnego i programowania, ale też ułatwia rozwiązywanie problemów na lekacjach ze wszystkich przedmiotów szkolnych **(Yihaun, 2019).**

Jedno z badań przeprowadzonych na ponad 116 nauczycielach matematyki, przedmiotów ścisłych, nauk społecznych i języka angielskiego w szkołach średnich i ponadgimnazjalnych dostarcza bardzo przydatnych przykładów rozpoznawania wzorców, które z powodzeniem zintegrowano z istniejącymi zasobami w klasie (Yihaun, 2019). Nauczyciele historii i języka angielskiego w szkołach średnich przygotowali historyczne puzzle i gry fabularne dla swoich uczniów, aby ułatwić im poznanie życia w średniowieczu. Uczniowie zostali poinstruowani, aby zidentyfikować wzorce w grach, w taki sam sposób, jak w historycznych przekazach i wydarzeniach. (Yihaun, 2019). Istnieje możliwość dalszego rozwoju tego modelu poprzez zachęcanie uczniów do korzystania z wzorców, które rozpoznali w poprzednich okresach, aby przewidzieć, co napotkają w następnym okresie historycznym.

W sektorze nauk ścisłych uczniowie klas od szóstej do dziesiątej mogli zintegrować rozpoznawanie wzorców bezpośrednio z egzaminem za pomocą quizu Rock & Mineral Quiz. W ramach ćwiczeń laboratoryjnych uczniowie mieli zastosować rozpoznawanie wzorców podczas pomiaru różnych właściwości skał i minerałów. Następnie zapisywali te wyniki i wykorzystywali je do tworzenia prognoz dotyczących

przyszłych właściwości różnych próbek. Przewidywania te zostały następnie przetestowane, co pozwoliło uczniom pogłębić wiedzę na temat właściwości minerałów (Yihaun, 2019).

Na Uniwersytecie Północnej Florydy w USA, studenci byli zachęcani do wykorzystywania umiejętności rozpoznawania wzorców do tworzenia „map pojęć" w języku angielskim: pomagając uczniom rozpoznawać wzorce w strukturach pisania i dając im wzory, które można wykorzystać, aby pomóc im „pisać klarownie" (Howell et al., 2011). Ta umiejętność rozpoznawania wzorców została również zastosowana do tekstów piosenek, tworzenia scenariuszy dramatów i identyfikowania powtarzającej się symboliki w literaturze i mediach. Opierając się na tych pomysłach, uczniowie mogą być również zachęcani do identyfikowania powtórzeń w regułach gier, przekazach medialnych, zagadkach, wierszach, wykresach, historiach indywidualnych i grupowych.

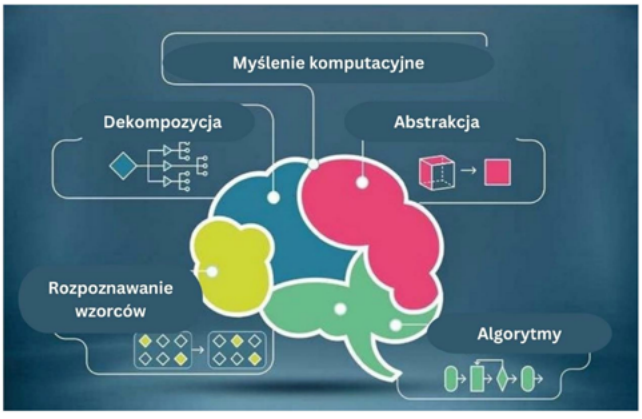
Wzorce odgrywają istotną rolę także w amerykańskim egzaminie adwokackim. Kandydaci uczą się z doświadczenia, że z czterech potencjalnych opcji w danym pytaniu, dwie zwykle zawierają prawnie nieprecyzyjne twierdzenia, a pozostałe dwie służą do przetestowania wiedzy prawnej i umiejętności rozumowania.

Niezależnie od kontekstu, wykazano, że ćwiczenia służące odkrywaniu i uczeniu się CT krok po kroku pomagają w nauce i zrozumieniu złożonych tematów (Lockwood, 2017). Warto też pamiętać, że łączenie uczniów w pary w trakcie uczenia się opartego na współpracy może sprawić, że nowe umiejętności będą bardziej zrozumiałe, a pary będą mogły pomagać sobie nawzajem, odświeżać, dyskutować i sprawdzać swoją wiedzę (Isbell et al., 2010).

Chrysanthi Konstanti
Eftychia Xerou
Centre for Advancement of Research
and Development in Educational TechnologyLtd.

4. Abstrahowanie

Abstrahowanie to umiejętność upraszczania i uogólniania złożonych problemów lub danych, aby znaleźć ich istotną istotę. Jest to jedna ze składowych myślenia komputacyjnego, która łączy rezultaty dwóch poprzedzających ją procesów: dekompozycji i rozpoznawania wzorów w nową jakościowo całość, ukazującą esencję problemu, którym się zajmujemy. Abstrakcje problemów, będące rezultatem abstrahowania są definicjami owej esencji i pomagają uporządkować i uprościć problem. Abstrahowanie jest ważne nie tylko w informatyce, ale także w innych dziedzinach wiedzy, takich jak matematyka, fizyka, biologia czy sztuka. Znakomitym przykładem jest malarstwo abstrakcyjne. Abstrakcja pomaga nam odkrywać wzorce, reguły i zasady, które rządzą światem. Abstrakcja jest także częścią naszej codzienności, gdy używamy symboli, metafor, schematów czy modeli, aby lepiej zrozumieć i komunikować się z innymi. Pojedyncze słowa, takie, jak np. dom, koń, czy czas są abstrakcjami dużych grup obiektów lub pojęć podstawowych. Umiejętność wydobywania i werbalizowania abstrakcji oraz posługiwania się nimi jest więc kluczową umiejętnością, ważną w nauczaniu różnych przedmiotów szkolnych, ponieważ pomaga tłumaczyć i wyjaśniać trudne pojęcia w sposób prosty i zrozumiały dla uczniów. Abstrakcja jest także sposobem na rozwijanie kreatywności i wyobraźni, ponieważ zachęca nas do tworzenia nowych i oryginalnych pomysłów. Abstrakcja jest nie tylko centralnym pojęciem w myśleniu komputacyjnym, ale także ważnym elementem edukacji i kultury.



Rysunek 1. Podstawowe elementy myślenia komputacyjnego.
Źródło: Elementy myślenia komputacyjnego. | Pobierz schemat naukowy (researchgate.net)

Abstrahowanie jest szczególnie przydatne w rozwiązywaniu złożonych problemów, ponieważ pozwala skoncentrować się na istotnych cechach obiektu i zignorować nieistotne szczegóły, które mogą zostać ukryte, umożliwiając osobie rozwiązującej problem lepsze zrozumienie tego, co się dzieje (Koppelman & Van Dijk, 2010). Zdobywanie doświadczenia w rozwiązywaniu problemów oznacza umiejętność określenia, jaki poziom abstrakcji jest odpowiedni dla danego etapu analizy problemu, projektowania rozwiązania i implementacji jego rozwiązania (Haberman & Muller, 2008). Abstrahowanie kładzie nacisk na proces usuwania szczegółów w celu uproszczenia i skupienia uwagi oraz podkreśla proces uogólniania w celu zidentyfikowania wspólnego rdzenia lub istoty (Kramer, 2007). Według Koppelmana i Van Dijka (2010) abstrahowanie wiąże się z następującymi cechami: (a) uogólnienie konkretnych przykładów; (b) identyfikacja, ekstrakcja i izolacja istotnych komponentów oraz (c) ignorowanie lub wykluczanie nieistotnych szczegółów.

Jedna z międzynarodowych inicjatyw, zmierzających do popularyzacji myślenia komputacyjnego i kompetencji informatycznych, Bebras (https://www.bebras.org) uznaje abstrahowanie za najważniejszą, choć niedocenioną kompetencję. Bebras oferuje tzw. „wyzwanie Bebras", które polega na tym, że nauczyciele wprowadzają uczestników do korzystania z komputerów lub urządzeń mobilnych. Od listopada 2022 r. do kwietnia 2023 r. w wyzwaniu wzięło udział ponad 3 miliony uczestników z 59 krajów i jest ono kontynuowane. Dominującymi tematami myślenia komputacyjnego w zadaniach Bebras w latach 2010–2014 były algorytmy (66%) i reprezentacja danych (38%). Abstrahowanie zostało zidentyfikowane jako istotny temat, występujący w 16% zadań (Barendsen i in., 2015). Chociaż badacze zgodzili się, że abstrakcja jest centralnym pojęciem w myśleniu komputacyjnym, nie ma zgody co do sposobu jej wdrażania o kształtowania w edukacji (Cetin & Dubinsky, 2017), choć nie ma wątpliwości, że jest ważnym elementem kultury i nauki.

4.1. Znaczenie nauczania abstrahowania

Wing (2006) stwierdził, że myślenie jak informatyk wymaga korzystania z abstrahowania na wielu etapach myślenia. Abstrakcja jest kluczową koncepcją i jedną z najbardziej fundamentalnych idei leżących u podstaw informatyki i jej praktyki (Armoni, 2013). Pojęcie abstrakcji jest często używane w różny sposób, w zależności od przedmiotu nauczania. Abstrakcje w informatyce i analizie danych są powszechne i oczywiste (Dorodchi et. al., 2021). Uczniowie powinni być jednak świadomi istnienia różnych poziomów abstrakcji, czyli stopnia uogólniania informacji i dostrzegać zalety świadomego przechodzenia między poziomami abstrakcji, gdy jest to konieczne, np. z powodu potrzeby uwzględnienia niektórych szczegółów (Hazzan, 2008).

4.2. Wyzwania w nauczaniu abstrakcji

Wiek, w którym dzieci mogą zrozumieć pojęcie abstrakcji i uczyć się abstrahowania bywa określany na podstawie klasycznych prac Piageta, często przywoływanych

w literaturze dotyczącej edukacji. Piaget sugeruje, że dzieci nie mogą uczyć się abstrahowania, dopóki nie osiągną czwartego etapu rozwoju poznawczego, czyli stadium operacji formalnych. Dzieje się to zwykle około dwunastego roku życia (Cetin i Dubinsky, 2017). Cechą charakterystyczną stadium operacji formalnych jest myślenie abstrakcyjne, które pozwala przekraczać granicę czasu i przestrzeni, ze zrozumieniem stałości niektórych ogólnych właściwości obiektów, np. masy czy objętości, zachowywanych pomimo zmiany kształtu obiektu, np., kulki z plasteliny rozwałkowanej na wałek lub placek. Nauczanie abstrakcji nowicjuszy jest więc bardzo trudnym zadaniem, o czym mówi wielu ekspertów (Armoni, 2013). Nawet studenci informatyki mają tendencję do obniżania poziomu abstrakcji, a dokładniej nieświadomie stosują mechanizmy poznawcze, które pozwalają im nadać konkretny sens abstrakcyjnym koncepcjom (Hazzan, 2008).

Bardzo ważne jest, aby uczyć pojęć abstrakcyjnych ogólnych, a nie abstrakcyjnych częściowych, bo może to nie wystarczyć dla zrozumienia mechanizmów abstrahowania (White & Mitchelmore, 2010). Szczególnie ważne jest, aby umiejętności abstrahowania rozwijali nowicjusze, co jest dla nich wyzwaniem. Spontaniczne korzystanie z abstrakcji jest trudne dla nowicjuszy ze względu na naturalną skłonność do polegania na znanej procedurze w sytuacji, w której musimy rozwiązać nowy problem (Koppelman & Van Dijk, 2010).

Kwestią dotyczącą nauczania abstrahowania jest to, czy należy poświęcić cały kurs idei abstrakcji, czy też wspominać o abstrahowaniu przy różnych okazjach w ramach innych kursów, gdy jest to właściwe (Hazzan, 2008). Wiąże się to z zapewnieniem właściwego wdrażania abstrahowania w edukacji jest jej dobre rozumienie przez nauczycieli. W szczególności bardzo ważne jest, aby nauczyciele potrafili określić, czy algorytm rozwiązywania problemu jest sam w sobie abstrakcją, czy aby stworzyć projekt, trzeba w ogóle abstrahować, czy też ogólny program postępowania znajduje się na innym poziomie abstrakcji niż algorytm rozwiązywania problemu (Waite i in., 2018).

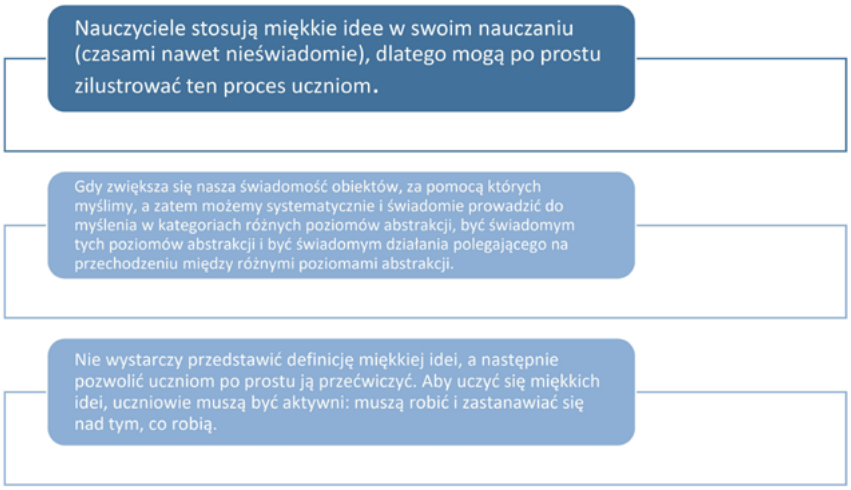
4.3. Potrzeba nauczania abstrahowania

Skuteczne opanowanie abstrahowania wymaga po pierwsze wiedzy na temat samej abstrakcji, a po drugie, nauczenia procesu abstrahowania i świadomego poruszania się między poziomami abstrakcji (Hazzan, 2008). Świadomość uczniów na temat natury abstrakcji, istnienia różnych poziomów abstrakcji oraz umiejętności i procesów umysłowych, które umożliwiają abstrahowanie i poruszanie się między poziomami abstrakcji, może zwiększyć ich umiejętności edukacyjne, a w przyszłości zawodowe (Hazzan, 2008). Badania wskazują, że zrozumienie poziomów abstrakcji (*Levels Of Abstaction - LOA*) i umiejętność poruszania się między poziomami jest też niezbędna do osiągnięcia sukcesu w programowaniu (Waite i in., 2018).

4.4. Nauczanie abstrahowania

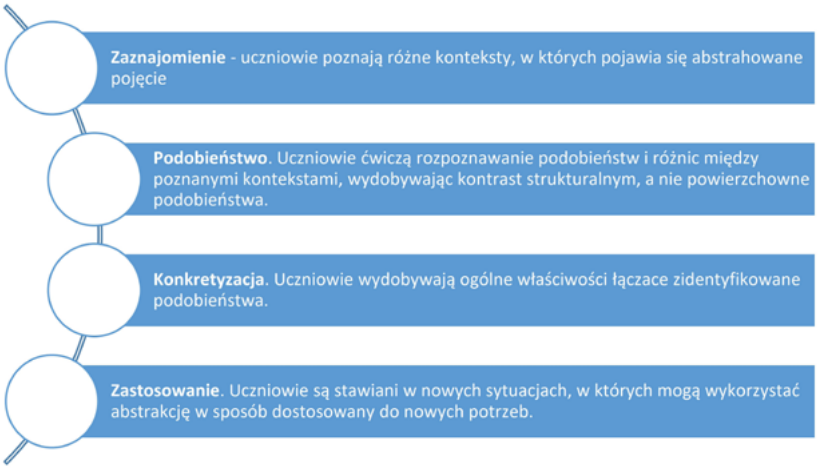
Według Hazzana (2008), uczniowie muszą najpierw zobaczyć i doświadczyć abstrahowania, a dopiero potem mogą wyabstrahować pierwsze ogólne pojęcia ze

zbioru elementów uzyskanych dzięki dekompozycji i poddanych grupowaniu w trakcie rozpoznawaniu wzorów. Opierając się na takim podejściu, autorzy sugerują następujące trzy sposoby, w jakie ogólne idee abstrahowania mogą być prezentowane uczniom (rysunek 2).



Rysunek 2. Nauczanie abstrahowania.

Wyniki badań przedstawionych w artykule White & Mitchelmore (2010) wykazały, że model nauczania abstrahowania w czterech krokach zapewnia skuteczne opanowanie przez uczniów abstrakcyjnych pojęć ogólnych dotyczących popularnych tematów z programu nauczania matematyki w gimnazjum. Badania rzucają również światło na to, co składa się na cztery fazy przedstawione na rysunku 3 i wyzwania związane z ich wdrażaniem w praktyce.



Rysunek 3. Nauczanie abstrahowania.

Haberman i Muller (2008) przedstawili dwa podejścia do nauczania abstrahowania.

1. Nauczanie zorientowane na wzorce ukazuje wzorce abstrahowania na początku zajęć. Zagadnienia wprowadzane później są zorganizowane wokół wzorców, ale ich treści nie odnoszą się do abstrahowania, tylko do tematyki przedmiotu.
2. Wykorzystanie abstrakcyjnych typów danych (ADT) w rozwiązywaniu problemów i reprezentacji wiedzy jest dominującym elementem programu nauczania.

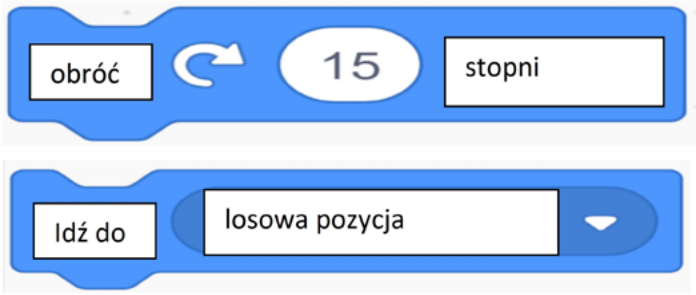
Pomocne może być też zachęcenie do stworzenia modelu omawianego zjawiska. Przygotowanie modelu wymaga uzyskania całościowego obrazu modelowanego pojęcia abstrakcyjnego krok po kroku i wymaga wcześniej wydobycia jego istoty przy jednoczesnym zrozumieniu, co należy zrobić (Dorodchi et al., 2021).

4.5. Abstrahowanie w praktyce

Abstrakcja, to w rozumieniu informatyki uproszczony zestaw działań lub cech programu, który wyjaśnia jego istotę, pomijając niepotrzebne szczegóły i zawiłości techniczne. Abstrahowanie pozwala na wydajne przechodzenie od wielu szczegółów do ogółu (abstrakcji) Przykłady abstrakcji:

1. **Pieczenie ciasta:** Podziel proces pieczenia na etapy, zidentyfikuj wspólne składniki i procedury oraz stwórz uogólniony zestaw instrukcji, których inni mogą przestrzegać. Przykładowa abstrakcja (istota) pieczenia ciasta, to: odmierzenie składników do miski, wymieszanie ich, umieszczenie wypełnionej ciastem formy w piekarniku na określony czas. Za pomocą takiej abstrakcji, po dodaniu szczegółów przepisu, można z powodzeniem upiec ciasto.
2. **Jazda przy użyciu map:** Mapy transportu publicznego są abstrakcyjne. Mapy przedstawiają tylko ważne informacje, takie jak przystanki i ogólny kierunek, a pomijają drobniejsze szczegóły. Dzięki pominięciu szczegółów, można intuicyjnie określić preferowaną trasę spośród wielu dostępnych opcji, zamiast oceniać każdy zakręt lub potencjalną zmianę trasy.
3. **Opisywanie roweru**: Kiedy rozmawiamy o rowerze, mamy tendencję do stosowania wybiórczych informacji, aby go opisać. Zazwyczaj wspominamy o jego typie, takim jak rower górski lub szosowy, i podajemy podstawowy opis jego koloru. Jeśli jednak rozmawiamy z kimś, kto naprawdę interesuje się rowerami, możemy zagłębić się w dalsze szczegóły. Możemy omówić materiał ramy, rozmiar opon, przełożenie, typ zawieszenia, a nawet markę i model („Przykłady abstrakcji w życiu codziennym", 2022).

Abstrahowanie warto pokazać także na przykładzie języków programowania. Większość języków programowania to zestawy abstrakcyjnych poleceń, ukrywających złożone instrukcje binarne wykorzystywane przez komputery. Języki programowania oferują również polecenia i słowa kluczowe, takie jak „drukuj" i „wczytaj", które abstrahują z ciągu skomplikowanych zadań ich cel lub oczekiwany rezultat. W popularnym języku programowania dla dzieci *Scratch*, istnieją różne „cegiełki" – bloki graficzne, które są abstrakcjami operacji zachodzących w tle. Bloki te (rysunek 4) wizualnie przedstawiają kroki do wykonania, umożliwiając użytkownikom intuicyjne zrozumienie przebiegu i celu procesu (Parry, 2021).



Rysunek 4. Bloki *Scratch* przedstawiające kroki, które są ukryte przed użytkownikiem w ich podpisach.

Chrysanthi Konstanti
Eftychia Xerou
Centre for Advancement of Research
nd Development in Educational Technology Ltd.

5. Algorytmizacja

5.1. Definicja myślenia algorytmicznego

Myślenie algorytmiczne to umiejętność konstruowania rozwiązań problemów w sposób, który daje powtarzalne rezultaty w różnych dziedzinach, nie tylko w dziedzinie nauk ścisłych, matematyki i logiki (Mezak & Papak, 2018). Algorytm jest jak przepis na przygotowanie potrawy, gdzie każdy krok jest precyzyjnie opisany i musi być wykonany w określonej kolejności (Peel & Friedrichsen, 2018).

Myślenie algorytmiczne to umiejętność wskazania podstawowych działań zmierzających do rozwiązania danego problemu w sposób adekwatny do jego specyfiki i uwzględniający możliwe sytuacje szczególne i typowe w celu zapewnienia najlepszej wydajności algorytmu (Hromkovic i in., 2017).

5.2. Rola nauczania algorytmizacji

Myślenie algorytmiczne jest zarówno jednym z filarów informatyki, jak i kluczową kompetencją (Malik i in., 2019). Jest to umiejętność, która wymaga elastycznego stosowania wiedzy z różnych dyscyplin w celu rozwiązywania codziennych, rzeczywistych problemów (Sarı, et al., 2022). W obecnej erze cyfrowej algorytmiczne rozwiązywanie problemów jest uważane za ważną umiejętność zarówno dla społeczeństwa, jak i w każdym środowisku pracy, co czyni ją kompetencją kluczową (Evripidou i in., 2021).

5.3. Wyzwania w nauczaniu algorytmizacji

Wielu uczniów uważa algorytmy za trudny i niezbyt atrakcyjny temat. Często tradycyjne kursy koncentrują się na nauce konkretnych algorytmów, które są uważane za ważne w edukacji lub w praktyce (Futschek & Moschitz, 2010). Dugan (2020) podkreśla znaczenie szkolenia nauczycieli w zakresie myślenia algorytmicznego i stwierdza, że bardzo ważne jest, aby nauczyciele, którzy mają niepodważalne miejsce w systemie edukacji, byli dobrze wyszkoleni, ponieważ mają przygotować uczniów, którzy stanowią drugi podstawowy element systemu, na przyszłość.

5.4. Potrzeba nauczania algorytmizacji

Istnieje potrzeba przeprowadzenia praktycznych badań nad tym, jak rozwijać myślenie algorytmiczne oraz jakiego rodzaju działania i treści edukacyjne mogą być wykorzystywane w klasach (Sarı i in., 2022). Umiejętności algorytmizacji stają się coraz bardziej kluczowe dla różnych aspektów naszych codziennych czynności, zwłaszcza po wzbogaceniu ich o moc komputerów i robotów (Evripidou i in., 2021).

5.5. Nauczanie algorytmizacji

Wprowadzenie uczniów do algorytmizacji powinno obywać się na bardzo wczesnym etapie nauczania myślenia komputacyjnego, z wykorzystaniem spiralnego programu nauczania Hromkovic et al. (2017). Uczniowie powinni od samego początku ćwiczyć projektowanie algorytmów w ustrukturyzowany sposób, doskonaląc szczegółowe kompetencje w późniejszych etapach edukacji. W spiralnym programie nauczania, algorytmy są wprowadzane w szkole podstawowej, a ich doskonalenie jest kontynuowane w edukacji ponadpodstawowej. Autorzy wskazują, że problemy przekazywane uczniom jako zadania do rozwiązania nie powinny być zbyt proste, ale ich opis powinien być łatwy do zrozumienia.

Metodą nauczania, która jest bardzo przydatna i skuteczna, są też wizualizacje algorytmów (Futschek & Moschitz, 2010). Jak twierdzą autorzy, uczenie się zasad i koncepcji algorytmizacji z wykorzystanie wizualizacji jest znacznie łatwiejsze do zrozumienia przez uczniów i sprawia, a zajęcia z wykorzystaniem wizualizacji można przygotować tak, że będą dla uczniów zabawne. Zadaniem nauczyciela w tej metodzie jest formułowanie problemów odpowiednich do poziomu kompetencji ucznia i zadawanie pytań, które zachęcają uczniów do myślenia zmierzającego do stworzenia poprawnie działających algorytmów rozwiązujących te problemy. Nauczyciel motywuje też uczniów do ulepszania swoich algorytmów w celu znalezienia jeszcze bardziej efektywnych rozwiązań. Futschek (2006) proponuje szeroką gamę tematów informatycznych, które mogą zachęcać uczniów do stosowania myślenia algorytmicznego, zwłaszcza z wykorzystaniem wizualizacji algorytmów w formie programu-narzędzia lub gry wykonywanej przez samych uczniów.

Warto podkreślić, że nauczyciele odgrywają bardzo ważną rolę jako wzory osobowe myślenia algorytmicznego. Nauczyciele uczestniczący w jednym z badań (Dugan, 2020) potwierdzili, że posiadając dobrze rozwinięte umiejętności myślenia algorytmicznego mają tendencję do nauczania spiralnego, podążania za procesem ułatwiającym naukę, zachęcania uczniów do dobrego rozplanowania i schludnego opracowania algorytmów oraz pomagania im w rozwijaniu/doskonaleniu umiejętności myślenia algorytmicznego. Zasugerowali również wykorzystanie takich technik, jak uczenie się przez odkrywanie, rozwiązywanie problemów, indukcja, burza mózgów, mapowanie koncepcji, gry, dyskusja, rybia ość i studium przypadku, które wymagają aktywnego zaangażowania uczniów w proces uczenia się w celu poprawy ich umiejętności algorytmizacji. W nauczaniu algorytmizacji można wykorzystać również wyobrażenie pudełka z cegiełkami, w którym dostępnych jest tylko

kilka podstawowych elementów, z których dzieci mogą tworzyć zarówno proste, jak i skomplikowane budowle (Milkova, 2012).

5.6. Zasoby wspierające kształcenie umiejętności abstrahowania i algorytmizacji

Aktywność dydaktyczna dotycząca myślenia komputacyjnego, w tym abstrahowania i algorytmizacji może być prowadzona z użyciem komputerów lub bez sprzętu, oprogramowania lub narzędzi cyfrowych. Działania bez komputerów („unplugged" – niepodłączone) koncentrują się na rozwijaniu umiejętności myślenia komputacyjnego poprzez ćwiczenia namacalne i interaktywne, wykorzystując instrukcje papierowe, wizualizacje, figurki i ruch fizyczny jako reprezentacje operacji dokonywanych przez programy komputerowe (Sigayret i in., 2022). Najczęściej wykorzystywane jest jednak łącznie podejść „podłączonych" i „niepodłączonych".

Narzędzia edukacyjne wspierające kształtowanie kompetencji pokrewnych do myślenia komputacyjnego i omawianego w tym rozdziale algotymizowania, to m.in. cMinds. Program cMinds jest interwencją edukacyjną dostarczającą wyzwań logicznych służących rozwijaniu umiejętności rozwiązywania problemów i zdolności analitycznego myślenia poprzez naukę opartą na grach komputerowych. cMinds Learning Suite zachęca uczniów do analizowania problemów, identyfikowania podstawowych komponentów rozwiązania, krytycznego łączenia różnych komponentów, optymalizowania swoich rozwiązań i zastanawiania się nad procesem myślenia (Tsalapatas et al., 2012).

TeaEdu4CT to jeden z projektów łączących podejście bez komputerów i z ich wykorzystaniem, który koncentruje się na innowacyjnych metodach edukacyjnych w zakresie myślenia komputacyjnego (CT) w kontekście transdyscyplinarnych i holistycznych perspektyw STEM (przedmiotów ścisłych i technicznych) w kształceniu nauczycieli (https://www.fsf.vu.lt/ct4teachers#about-the-project). Projekt ten zapewnia zbiór narzędzi, technik i podejść, które ułatwiają płynne przejście od działań bez komputerów i Internetu, odpowiednich dla małych dzieci, aż do zaawansowanego modelowania i symulacji komputerowych dla uczniów szkół ponadpodstawowych i studentów wczesnych lat studiów.

The Tech Interactive, organizacja kierująca się misją propagowania myślenia komputacyjnego (CT), tworzy i rozpowszechnia atrakcyjne dla uczniów zasoby edukacyjne STEM wysokiej jakości, pozwalające szybko włączyć elementy myślenia komputacyjnego do nauczania. Na przykład abstrahowanie, które opisuje zjawiska naturalne za pomocą zwięzłych stwierdzeń, jest ćwiczone w trakcie wykonywania szerokiej gamy eksperymentów. Algorytmizacja jest z kolei zestawem instrukcji, czyli „procedurą" zmierzającą do wykonania jednego z owych eksperymentów laboratoryjnych w klasie. Przykłady zasobów edukacyjnych, planów lekcji i zajęć dla nauczycieli i uczniów można znaleźć na stronie internetowej Tech Interactive, https://www.thetech.org/educators-students/resources/lessons-activities/computational-thinking/.

5.7. Algorytmizacja w praktyce

Na co dzień spotykamy różne rodzaje algorytmów, które wykonują określone kroki (procesy liniowe), podejmują decyzje (algorytmy warunkowe) czy powtarzają działania określoną liczbę razy (algorytmy zapętlone). Algorytmizacja wydaje się być złożonym procesem, ale algorytmizowanie wielu codziennych czynności przychodzi nam naturalnie. Oto trzy przykłady, które można przytoczyć:

1. **Wiązanie butów:** Czynności wykonywane krok po kroku, nierzadko automatycznie, które są konsekwentnie przeprowadzane w ten sam sposób wiele razy, np. wiązanie butów, mogą być uznane za algorytm. Proces uzyskiwania tradycyjnego węzła sznurowadła, często określanego jako pętelka lub kokardka, można wykonać za pomocą bardzo prostego, ograniczonego zestawu kroków: „pętelka, przeciągnięcie przez oczko, zaciśnięcie".

2. **Algorytm przepisu:** Każdy prosty przepis na gotowanie lub pieczenie może być uznany za algorytm. W przepisie występuje kolejność operacji, opis warunków, powtórzenia potrzebnych do pomyślnego ukończenia przepisu czynności. Algorytmizowanie przepisów kulinarnych, to pożyteczne ćwiczenie, promujące myślenie algorytmiczne i pokazujące wszechobecność algorytmizacji.

3. **Jazda do lub z jakiegoś miejsca:** Czynność prowadzenia samochodu, na przykład z domu do szkoły, może być algorytmem. Podobnie jak każdy inny algorytm, ta rutyna może zderzyć się z przeszkodami, takimi jak roboty drogowe lub duże natężenie ruchu, co może skłaniać do podejmowania decyzji w oparciu o napotkane warunki. Jeśli na ulicy X są roboty drogowe, skręć w prawo) („7 przykładów algorytmów w życiu codziennym dla uczniów", 2023).

Panagiotis Psomos
Innovation Frontiers

6. Gra CTApp. Pomysł, struktura i jej funkcje

Idea stojąca za CTApp Game

Opracowanie nowej aplikacji mobilnej, mającej na celu wzmocnienie umiejętności myślenia komputacyjnego, może wesprzeć edukację i wzmocnić potencjał uczniów. Aplikacja CTApp ma na celu zaangażowanie i zmotywowanie uczniów, szczególnie tych, którzy mogą znajdować się w niekorzystnej sytuacji, do zdobywania podstawowych umiejętności myślenia komputacyjnego. Rozbudzając zamiłowanie do rozwiązywania problemów w oparciu o podejście komputacyjne i przygotowując uczniów do świata w coraz większym stopniu napędzanego przez technologię, aplikacja ma na celu wywarcie trwałego wpływu na ich życie i przyszłość. Co więcej, to innowacyjne podejście do edukacji ma na celu poprawę rozwoju zawodowego nauczycieli i wyposażenie ich w narzędzia niezbędne do wprowadzenia innowacyjnych praktyk w klasie we współpracy z uczniami.

Umiejętności myślenia komputacyjnego stały się ważne dla uczniów, aby mogli rozwijać się edukacyjnie, zawodowo i osobiście w dzisiejszym szybko zmieniającym się świecie. Zdolność do uchwycenia i zastosowania idei myślenia komputacyjnego stała się kluczowym zestawem umiejętności, ponieważ cyfrowy świat nadal zmienia naszą cywilizację.

Myślenie komputacyjne to metodyczne podejście do rozwiązywania problemów, które opiera się na zasadach informatyki i logiki. Pozwala ludziom dekonstruować trudne kwestie na łatwe do zarządzania komponenty, analizować dane i opracowywać nowe rozwiązania. Uczniowie zdobywają kompetencje radzenia sobie ze złożonymi wyzwaniami obecnego świata poprzez rozwijanie umiejętności myślenia komputacyjnego.

Wszechobecna integracja technologii w wielu obszarach jest jednym z podstawowych powodów, dla których rozwijanie i wzmacnianie umiejętności myślenia komputacyjnego ma kluczowe znaczenie. Technologia przenika praktycznie każdy biznes, od sztucznej inteligencji i robotów po analizę danych i cyberbezpieczeństwo. Umiejętności myślenia komputacyjnego zapewniają uczniom podstawy dla skutecznego zrozumienia i wykorzystania technologii, umożliwiając im dostosowanie się do stale zmieniających się wymagań ery cyfrowej.

Co więcej, myślenie komputacyjne zachęca do krytycznego myślenia, kreatywności i logicznego rozumowania. Uczy studentów metodycznego podejścia do zagadnień, dostrzegania wzorców i tworzenia wydajnych algorytmów. Te zdolności poznawcze wykraczają poza informatykę i mogą być stosowane w wielu dziedzinach, w tym w matematyce, fizyce, biznesie i naukach humanistycznych. Studenci z umiejętnościami myślenia komputacyjnego są lepiej przygotowani do radzenia sobie

z trudnymi kwestiami, podejmowania przemyślanych decyzji i wnoszenia ważnego wkładu w wybrane przez siebie obszary.

Jednym z głównych celów aplikacji jest poprawa rozwoju zawodowego nauczycieli. Zapewniając kompleksowe narzędzie, wspierające nauczanie myślenia komputacyjnego, nauczyciele mogą uzyskać cenny wgląd w temat i rozwijać własną wiedzę. Aplikacja oferuje nauczycielom platformę do poszerzania wiedzy, uczenia się nowych strategii dydaktycznych i bycia na bieżąco z najnowszymi osiągnięciami w nauczaniu myślenia komputacyjnego. Ten rozwój zawodowy przynosi korzyści nie tylko samym nauczycielom, ale ma również bezpośredni wpływ na jakość edukacji, którą zapewniają swoim uczniom.

Ciągły rozwój zawodowy jest niezbędny, aby nauczyciele mogli pozostać skuteczni i dostosowywać się do zmieniających się potrzeb swoich uczniów. W związku z tym aplikacja służy jako pomoc dla nauczycieli. Nauczyciele mogą pogłębiać swoje zrozumienie koncepcji myślenia obliczeniowego i doskonalić swoje praktyki dydaktyczne. Nacisk na ciągły rozwój zawodowy pozwala nauczycielom stawać się liderami i mentorami edukacyjnymi, co ostatecznie przynosi korzyści zarówno im samym, jak i ich uczniom.

Oprócz symulowania rozwoju zawodowego, aplikacja CTApp ma na celu wspieranie nauczycieli w podejmowaniu innowacyjnych i opartych na współpracy praktyk w klasie. Ułatwiając wdrażanie innowacyjnych metod nauczania, aplikacja zachęca nauczycieli do kreatywnego myślenia i dostosowywania swoich metod nauczania w celu zaspokojenia różnorodnych potrzeb uczniów. To wsparcie dla współpracy i innowacji przyczynia się do rozwijania dynamicznego i angażującego środowiska uczenia się, które skutecznie wzmacnia umiejętności myślenia komputacyjnego.

Opracowanie aplikacji mobilnej, poświęconej wzmacnianiu umiejętności myślenia komputacyjnego, jest znaczącym krokiem w kierunku wyposażenia uczniów w narzędzia niezbędne do odniesienia sukcesu w erze cyfrowej. Aktywizując rozwój zawodowy nauczycieli, wspierając ich ciągłe uczenie się oraz promując współpracę i innowacyjne praktyki, aplikacja umożliwia nauczycielom stanie się katalizatorami zmian w systemie edukacji. Ostatecznie inicjatywa ta ma na celu wypełnienie luki w edukacji w zakresie myślenia komputacyjnego, szczególnie w przypadku uczniów znajdujących się w niekorzystnej sytuacji, i zapewnienie, że wszyscy uczniowie są przygotowani do rozwoju w świecie zdominowanym przez komputery i technologię. Zapewniając solidne narzędzie, wspierające szkolne programy nauczania i wzbogacając zestaw narzędzi dla nauczycieli, aplikacja ta może usprawnić edukację i kształtować lepsze środowisko nauczania.

Struktura aplikacji CTApp Game i jej funkcje

Wejście do gry CTApp

Gra CTApp jest ma charakter pokoju zagadek *(escape room)* i jest łatwo dostępna zarówno dla użytkowników Androida, jak i iPhone'a za pośrednictwem odpowiednio Sklepu Play i App Store. Kiedy użytkownicy otwierają grę, wita ich charakterystyczne

logo projektu CTApp, które jest uzupełnione tłem przedstawiającym obszar do nauki. Ponadto widoczny jest zestaw logicznie rozmieszczonych przycisków, które pomagają w płynnej nawigacji po aplikacji.

Kluczowym z przycisków jest przycisk Start, który działa jako punkt wejścia do świata wyzwań opartych na pytaniach. Jeśli użytkownicy zdecydują się opuścić aplikację, przycisk zakończenia jest umieszczony w widocznym miejscu, aby zapewnić płynne wyjście z gry. Aplikacja posiada też przycisk ustawień, który pozwala użytkownikom spersonalizować swoje doświadczenia. Użytkownicy mogą łatwo zmieniać muzykę i dźwięki za pomocą tego przycisku, aby dopasować dźwięki do swoich upodobań. Gdy użytkownicy uruchomią aplikację, muzyka towarzyszy im podczas gry, poprawiając wrażenia i zaangażowanie. Użytkownik podaje też na początku swoją nazwę i odpowiadający mu awatar, który można później zmienić.

Rozpoczynając grę, można też wybrać rozgrywkę z druga osobą. W tym celu należy nacisnąć przycisk „Tryb wieloosobowy". Do rozgrywki można wybrać dwie postaci, których personalizację przeprowadza się niezależnie. Gracze grają na jednym urządzeniu. Po ukończeniu poziomu przez jednego z uczestników, grę może rozpocząć drugi uczeń. Wyniki są obliczane niezależnie dla każdego z uczestników.

Użytkownicy mogą łatwo korzystać z aplikacji dzięki dostępności CTApp Game na platformach Android i iPhone, przyjaznemu dla użytkownika układowi i wygodnemu zestawowi przycisków do rozpoczynania, wychodzenia i zmiany ustawień. Intrygujące tło obszaru nauki nadaje ton doświadczeniu, a muzyka ambientowa dodaje odrobinę spokoju i pozytywnych emocji. CTApp Game gwarantuje, że każda osoba może dostosować swoje zaangażowanie w aplikację do własnego gustu i preferencji, dając użytkownikom możliwość wyboru.

CTApp Game jest przeznaczona dla zróżnicowanej grupy odbiorców, z wersjami dostępnymi w języku angielskim, włoskim, polskim i greckim. Użytkownicy posługujący się tymi językami będą mogli uzyskać dostęp do aplikacji i jej zawartości w swoim ojczystym języku. Aplikacja jest przyjazna dla użytkownika i oferuje przetłumaczone wersje pytań, zapewniając pozytywne doświadczenie gamie zróżnicowanych użytkowników.

Poruszanie się po menu

Uczniowie są witani intuicyjnym interfejsem menu, który pojawia się na ekranie po kliknięciu przycisku Start, aby rozpocząć główną grę. Menu to działa jako portal do czterech odrębnych etapów, z których każdy reprezentuje jeden poziom w grze. Pierwszy poziom składa się głównie z pytań teoretycznych na temat myślenia komputacyjnego i jego kluczowych etapów, które wymagają od użytkowników wykazania się wiedzą specjalistyczną poprzez udzielanie prawidłowych odpowiedzi. W miarę postępów użytkowników, kolejne trzy poziomy zapewniają serię praktycznych problemów, które należy rozwiązać, aby pomyślnie ukończyć grę.

Pytania praktyczne są przypisane do tych trzech poziomów w zależności od ich zróżnicowanego poziomu trudności. W rezultacie każdy kolejny poziom zapewnia nieco trudniejsze pytania niż poprzedni, zapewniając stopniową naukę. Aby przejść

przez grę, użytkownicy muszą pomyślnie ukończyć każdy poziom, zanim przejdą do następnego. Gdy użytkownicy rozpoczynają korzystanie z aplikację, mają dostęp tylko do pierwszego poziomu, który składa się z pytań teoretycznych i wymaga przygotowania informacyjnego użytkowników w trakcie zajęć wprowadzających do myślenia komputacyjnego.

Użytkownicy otrzymują jasną i chronologiczną ścieżkę, którą mogą podążać. Przechodzenie przez grę w ten sposób pozwala na uzyskanie poczucia sukcesu podczas pokonywania każdego poziomu. System progresji gwarantuje, że użytkownicy stale poprawiają swoje zrozumienie myślenia komputacyjnego i umiejętności korzystania z nabywanych kompetencji, co pozwala im radzić sobie z coraz bardziej skomplikowanymi problemami w miarę przechodzenia gry.

Podsumowując, gdy użytkownicy uruchamiają główną grę, naciskając przycisk Start, są witani przez przyjazny dla użytkownika ekran menu z czterema oddzielnymi poziomami. Pierwszy poziom zawiera pytania teoretyczne, podczas gdy kolejne poziomy zawierają coraz trudniejsze pytania praktyczne. Użytkownicy mają możliwość przejścia do następnego poziomu po pomyślnym ukończeniu poprzedniego. Ta sekwencyjna struktura promuje rozwój umiejętności i zapewnia satysfakcjonujące doświadczenie dla użytkowników, którzy chcą poprawić swoją wiedzę i kompetencje w grze. Dzięki systemowi podpowiedzi, gra może być dla wnikliwych użytkowników samodzielnym źródłem informacji, ale może być też wplatana w proces edukacyjny jako narzędzie wspierające, stosowane w ramach zadań domowych, czy nawet na lekcjach, dzięki trybowi multiplayer.

Gra w CTApp Game

Kiedy użytkownicy wchodzą na pierwszy poziom, wita ich obszar do nauki, wyposażony w zwykłe przedmioty, takie jak biurka, fotele, regały, globusy, lampy i inne. Każdy obszar w grze ma określony motyw i jest ozdobiony różnymi zwykłymi przedmiotami. Jeden z obszarów zawiera symbole matematyczne, np. znak plus i symbol pi. Zróżnicowane projekty obszarów zapewniają użytkownikom stymulujące wrażenia podczas przechodzenia gry.

Pomoc w nawigacji zapewnia przycisk w lewym dolnym rogu ekranu, który umożliwia użytkownikom poruszanie awatarem w prawo, w lewo, do tyłu lub do przodu. Dotknięcie i przesuwanie ekranu w innych miejscach pozwala użytkownikom na zmianę perspektywy, umożliwiając im patrzenie w górę lub w dół pomieszczenia i obracanie widoku.

Po wejściu do pokoju nauki zegar zaczyna odliczać pięć minut. W tym czasie użytkownicy muszą ukończyć odpowiadanie na zestaw pytań przygotowanych wyłącznie dla tego poziomu. Niektóre przedmioty w pokoju nauki są wyróżnione, ponieważ działają jako „przedmioty-klucze", które zawierają zestawy pytań. Gdy przedmiot jest oznaczony jako „kluczowy", pojawia się nad nim zielone pole, odzwierciedlające liczbę pytań, na które udzielono prawidłowej odpowiedzi w tym zestawie we wcześniejszych podejściach. Na przykład, jeśli nie udzielono odpowiedzi na żadne z pytań w zestawie, nad odpowiednim „przedmiotem-kluczem" pojawi się

komunikat z nazwą przedmiotu, np.: „Projektor 0/16". Po dotknięciu nazwy przedmiotu wyświetla się seria pytań lub zadań.

Niektóre „przedmioty-klucze" są natychmiast widoczne i podświetlone, gdy tylko użytkownicy wchodzą do obszaru. Inne „kluczowe elementy" uzyskują oznaczenie tylko wtedy, gdy użytkownicy zbliżają się do nich i wtedy pojawia się zielone pole reprezentujące liczbę pytań bez odpowiedzi.

Użytkownicy mogą zacząć odpowiadać na pytania z łatwo widocznych „kluczowych elementów" zaraz po wejściu do pokoju. Po udzieleniu odpowiedzi na serię pytań użytkownicy są zachęcani do dalszego eksplorowania obszaru badania w celu znalezienia następnego zestawu pytań, które są przechowywane w innych „kluczowych elementach". Zielone pole nad tymi dodatkowymi „kluczowymi elementami" identyfikuje je i staje się widoczne tylko wtedy, gdy użytkownicy się do nich zbliżają. Dzięki powyższym właściwościom interaktywnego pokoju nauki użytkownicy zyskują kontrolę nad grą i zwiększają zaangażowanie. Gra zachęca do eksploracji i zaangażowania, motywując użytkowników do intensywnego korzystania z przestrzeni nauki i aktywnego poszukiwania kolejnego zestawu pytań.

Podsumowując, gdy uczniowie docierają do pierwszego pokoju, wita ich przestrzeń do nauki zawierająca różnorodne przedmioty. Mogą swobodnie przemierzać pokój, naciskając przyciski ekranowe, aby podróżować w różnych kierunkach. Rozpoczyna się odliczanie czasu, oznaczające konieczność ukończenia zestawów pytań w określonym czasie. Niektóre przedmioty w pokoju do nauki są „kluczowymi przedmiotami" i zawierają zestawy pytań, o czym informuje zielona ramka nad nimi. Część „kluczowych przedmiotów" jest natychmiast widoczna, inne uzyskują oznaczenie „kluczowego przedmiotu" tylko wtedy, gdy użytkownicy zbliżą się do nich bardziej. Użytkownicy mogą zacząć odpowiadać na pytania z widocznych „kluczowych elementów", a następnie poruszać się po przestrzeni do nauki, aby znaleźć kolejne zestawy pytań umieszczone w dodatkowych „kluczowych elementach". Ta interaktywna koncepcja pokoju do nauki kładzie nacisk na autonomię, zaangażowanie i szerokie badanie otoczenia.

Pytania

Gdy użytkownik kliknie na jeden ze zidentyfikowanych „kluczowych elementów", zostanie mu wyświetlone pierwsze pytanie w wyskakującym okienku. Każde pytanie ma taką samą strukturę. Pytanie jest wyraźnie wyświetlane w górnej części wyskakującego okienka, a użytkownicy mają do wyboru cztery możliwe odpowiedzi. Po dokonaniu wyboru przez użytkownika, wyskakujące okienko zapewnia szybką informację zwrotną, pokazując, czy wybrana odpowiedź jest poprawna, czy błędna. Dodatkowo, wraz z informacją zwrotną wyświetlane jest pole z wyjaśnieniem, wskazujące użytkownikom sposób myślenia, który należy przyjąć, aby uzyskać prawidłową odpowiedź.

Po udzieleniu odpowiedzi na wszystkie pytania w zestawie na wystarczającym poziomie poprawności, pojawia się wyskakujące okienko z komunikatem „poziom

ukończony". Gdy użytkownik poda zbyt dużo błędnych odpowiedzi, pojawia się wyskakujące okienko z frazą „spróbuj ponownie". To wyskakujące okienko zawiera dwie opcje: przycisk restartu, który pozwala odwiedzającym spróbować ponownie odpowiedzieć na pytania, oraz przycisk „przejdź do strony głównej", który wyświetla użytkownikom menu główne.

Wykorzystanie wyskakujących okienek zapewnia użytkownikom dynamikę i angażujące doświadczenie. Wyświetlanie pytań, opcji odpowiedzi, informacji zwrotnych i pól wyjaśnień w wyskakujących oknach pozwala na sprawne i dokładne przekazywanie informacji oraz angażuje uczniów w zdobywanie nowych kompetencji. Niezależnie od tego, czy użytkownicy odpowiedzą poprawnie, czy źle, gra zapewnia szybką informację zwrotną i alternatywy dla dalszych kroków, promując wytrwałość i zachęcając do nauki.

Podsumowując, po wybraniu „kluczowego elementu" odwiedzający otrzymują pierwsze pytanie w wyskakującym okienku. Układ jest identyczny we wszystkich pytaniach, z pytaniem po lewej i czterema opcjami odpowiedzi poniżej. Użytkownicy otrzymują szybką informację zwrotną na temat swojej odpowiedzi, wraz z polem wyjaśnienia przedstawiającym sugerowany proces myślenia. Ukończenie wszystkich pytań w określonym czasie skutkuje pojawieniem się wyskakującego okienka upamiętniającego „ukończony poziom". W przypadku udzielenia dużej liczby błędnych odpowiedzi pojawia się drugie wyskakujące okienko, dając użytkownikowi wybór ponownego uruchomienia poziomu lub powrotu do strony głównej. Wykorzystanie wyskakujących okienek w całej grze zapewnia użytkownikom wrażenie dynamiki.

Oto przykład pytania teoretycznego należącego do **poziomu 1** (pytania teoretyczne):

Które z tych zjawisk jest przykładem dekompozycji?

A. *Wypisanie z przepisu kulinarnego poszczególnych składników potrawy.*
B. *Sprawdzenie, czy ma się wszystkie naczynia, które będą potrzebne do wykonania potrawy z przepisu.*
C. *Zastanawianie się nad różnymi sposobami organizacji gotowania według przepisu.*
D. Żadne z powyższych.

W tym przykładzie poprawną odpowiedzią jest A. Wyjaśnienie brzmi: „Dekompozycja polega na podzieleniu problemu na mniejsze części".

Oto przykład pytania należącego do **poziomu 2** (pytania o poziomie trudności 1).

Aby polecieć na Marsa, należy wykonać poniższe kroki w podanej kolejności:

A. *zbudować rakietę, określić trajektorię lotu, uruchomić pełną moc silników, wgrać dane o celu lotu do komputera, obliczyć koszty operacji, opracować szczegółowy plan operacji, zebrać informacje o kluczowych wyzwaniach i zagrożeniach lotu, zebrać załogę.*

B. *opracować szczegółowy plan operacji, zebrać informacje o kluczowych wyzwaniach i zagrożeniach lotu, zebrać załogę, zbudować rakietę, określić trajektorię lotu, uruchomić pełną moc silników, wgrać dane o celu lotu do komputera, obliczyć koszty operacji.*
C. *zebrać informacje o kluczowych wyzwaniach i zagrożeniach lotu, opracować szczegółowy plan operacji, obliczyć koszty operacji, zebrać załogę, zbudować rakietę, określić trajektorię lotu, wgrać dane o celu lotu do komputera, uruchomić pełną moc silników.*
D. *zebrać załogę, zebrać informacje o kluczowych wyzwaniach i zagrożeniach lotu, zbudować rakietę, określić trajektorię lotu, uruchomić pełną moc silników, wgrać dane o celu lotu do komputera, obliczyć koszty operacji, opracować szczegółowy plan operacji, .*

W tym pytaniu poprawną odpowiedzią jest C. Oto wyjaśnienie: „Myślenie komputacyjne polega na dzieleniu złożonych zadań na mniejsze, łatwiejsze w zarządzaniu części i stosowaniu logicznego podejścia do ich rozwiązywania. Taką kolejność działania można zastosować do rozwiązywania prawdziwych problemów w uczeniu się lub w codziennym życiu".

UWAGA – powyższe przykłady są fikcyjne, takie pytania nie występują w grze!

Etapy sporządzania listy wszystkich elementów, rozpoznawania zasad łączących elementy, definiowania reguł do zastosowania i przygotowywania instrukcji krok po kroku opierają się na komputacyjnym podejściu do rozwiązywania problemów, a każdy krok wynika z poprzednich.

Ukończenie gry

Gdy użytkownicy pomyślnie odpowiedzą na pytania na danym poziomie, pojawi się wyskakujące okienko z gratulacjami i informacją, że następny pokój został otwarty. Użytkownicy mogą przejść dalej, powracając do menu głównego i wybierając kolejny poziom. Dostępny jest też przycisk pauzy w prawym górnym rogu ekranu, umożliwiając graczom zatrzymanie gry w dowolnym momencie i ponowne jej wznowienie.

Dzięki wielopoziomowej strukturze i dynamicznej akcji, CTApp Game jest narzędziem przydatnym dla młodych użytkowników, którzy chcą poprawić swoje umiejętności myślenia komputacyjnego. Oprogramowanie jest też wsparciem dla nauczycieli, którzy chcą rozwijać myślenie komputacyjne w klasie. Gra zachęca użytkowników do krytycznego myślenia, analizowania zagadnień i wykorzystywania logicznego myślenia w celu uzyskania odpowiedzi, oferując szereg trudnych pytań i zagadek. Użytkownicy są zachęcani do doskonalenia swoich umiejętności rozwiązywania problemów.

Robert Porzak
Lubelska Akademia WSEI

7. Integracja zajęć CT z programem nauczania i scenariuszem lekcji

Myślenie komputacyjne w szkołach

Poprzednie rozdziały opisywały myślenie komputacyjne jako umiejętność, która pomaga uczniom uczyć się logiki i zasad stojących za wyodrębnianiem podstawowych elementów, rozpoznawaniem ich powiązań i struktur, uogólnianiem i tworzeniem algorytmów postępowania. W podejściu komputacyjnym nie chodzi o zapamiętywanie faktów czy poleceń, ale o krytyczne i kreatywne myślenie. Uczniowie mogą ćwiczyć myślenie komputacyjne bez korzystania z pomocy czy pracowni, co pozwala włączyć je do każdej lekcji, nawet dla młodszych uczniów. Myślenie komputacyjne staje się obecnie niezbędną umiejętnością uczniów, ponieważ przygotowuje ich do zrozumienia i korzystania z technologii przyszłości. Nie jest to dodatkowe, czasochłonne zadanie nauczyciela, a wręcz przeciwnie – droga zaoszczędzenia czasu, którą można łatwo zintegrować z istniejącymi procedurami i programami nauczania (Yadav et al., 2017).

Myślenie komputacyjne może zaoszczędzić czas w szkolnym programie nauczania dzięki temu, że analizowanie i rozwiązywanie problemów pomaga uczniom uczyć się wszelakich umiejętności bardziej wydajnie i skutecznie, niż poprzez zapamiętywanie i powtarzanie. Myślenie komputacyjne może na przykład znakomicie przyspieszyć zrozumienie przez uczniów logiki, reguł matematyki, gramatyki języka obcego i własnego, czy praw fizyki. Stosując myślenie komputacyjne w nauczaniu różnych przedmiotów, uczniowie mogą łatwiej zrozumieć podstawowe koncepcje teoretyczne leżące u podstaw danej dziedziny wiedzy, unikać typowych błędów i szybciej rozwiązywać problemy (Flórez et al., 2017).

Myślenie komputacyjne może też ułatwić uczniom rozwijanie złożonych kompetencji podstawowych, które są niezbędne w XXI wieku. Kompetencje te obejmują krytyczne myślenie, kreatywność, współpracę i komunikację. Ucząc się myślenia komputacyjnego, uczniowie mogą poprawić swoją zdolność do analizowania informacji, generowania nowych pomysłów, współpracy z innymi i jasnego wyrażania siebie. Takie umiejętności znakomicie zwiększają szanse odniesienia sukcesu w różnych dziedzinach akademickich i zawodowych, a także w życiu osobistym (Pérez-Marín et al., 2020).

Myślenie komputacyjne jest też doskonałym, naturalnym narzędziem nauczania w szkołach o profilu zawodowym wszystkich szczebli. Stosując myślenie komputacyjne w szkole o profilu zawodowym, nauczyciele mogą zachęcać uczniów do myślenia o intencjach poszczególnych kroków postępowania, zamiast przekonywać ich do podążania za owymi krokami. „W przeciwieństwie do tradycyjnych metod nauczania,

metoda nauczania oparta na myśleniu komputacyjnym nie tylko kształtuje u ucznia rozumienie zasad postępowania wspieranego wykorzystaniem automatyzacji i komputeryzacji, ale także pomaga opanować metody rozwiązywania typowych problemów w sposób zorientowany na przyspieszenie i zarazem zmniejszenie uciążliwości pracy dzięki zaprzęgnięciu do pracy komputerów i automatów, a także AI. Warto przy tym zwrócić uwagę, że jeśli nauczyciele w szkołach o profilu zawodowym chcą prowadzić zajęcia z dowolnego przedmiotu przy użyciu metody nauczania opartej na myśleniu komputacyjnym, powinni:

a) rozłożyć program przedmiotu/kursu na kluczowe punkty wiedzy,
b) określić ważność i trudność każdego z zadań, ustalić kluczowe treści, które uczniowie koniecznie muszą zrozumieć,
c) zaprojektować modele eksperymentalne/symulacyjne odpowiadające kluczowym tematom i zadaniom, skonstruować środowiska obliczeniowe lub symulacyjne,
d) zoptymalizować proces rozwiązywania problemów,
e) zaplanować weryfikację wyników eksperymentów i zasady sprawdzenia następstw wypracowanych rozwiązań" (Shuiyan He et al., 2014, p. 819).

Przy zastosowaniu powyższych zaleceń, myślenie komputacyjne może być cennym rozszerzeniem szkolnego programu nauczania, ponieważ może dać uczniom czas i umiejętności potrzebne do szybszego i skuteczniejszego uczenia się innych umiejętności, a także rozwijać kompetencje podstawowe, które są kluczowe dla przyszłości. Myślenie komputacyjne może być również zabawne i angażujące dla uczniów, ponieważ mogą go używać do tworzenia gier, animacji, opowiadań, sztuki, muzyki i nie tylko. Myślenie komputacyjne można zintegrować w ścieżki międzyprzedmiotowe z różnymi przedmiotami i zajęciami w szkolnym programie nauczania, np. z matematyką, naukami ścisłymi, językami obcymi i językiem ojczystym, naukami społecznymi, sztuką, muzyką, wychowaniem fizycznym, a nawet zajęciami pozalekcyjnymi. Ucząc myślenia komputacyjnego, nauczyciele mogą w naturalny i płynny sposób przygotować uczniów na wyzwania i możliwości ery cyfrowej (Grover & Pea, 2018).

Może się wydawać, że myślenie komputacyjne jest zbyt zaawansowane dla młodszych uczniów, ale w rzeczywistości dobrze komponuje się z podejściem aktywizującym i uczeniem przez działanie, wykorzystywanym często w szkołach podstawowych, jak i młodszych klasach szkół średnich. Dzieci uwielbiają się bawić i odkrywać. Nie boją się próbować nowych rzeczy. Wykorzystując ich naturalną ciekawość i potrzebę rozwiązywania problemów, możemy pomóc im rozwinąć myślenie komputacyjne. Myślenie komputacyjne nie tylko czyni uczenie się przyjemną i ekscytującą zabawą, ale także nadaje strukturę i porządek procesowi zmagania się z problemami, dzięki czemu umiejętności zdobyte przez uczniów można później zastosować do bardziej złożonych zadań. Pasja wynalazcza młodszych uczniów bywa nawet zaskakująca, zawsze chętnie dołączają do zespołu wynalazców rozwiązujących problemy i zwykle z radością zanurzają się w odkrywanie świata myślenia komputacyjnego. Dobrym rozwiązaniem jest również połączenie myślenia komputacyjnego z uczeniem opartym na projektach, zwłaszcza w połączeniu z materiałem dydaktycznym w postaci

narzędzi robotycznych lub programów wizualizujących przebiegi zdarzeń w czasie lub w zależności od zmiany parametrów wejściowych. Takie mieszane podejście wykazuje znacznie lepszą skuteczność w poprawianiu długofalowych, głęboko utrwalonych i uwewnętrznionych osiągnięć kompetencyjnych uczniów, zwłaszcza w porównaniu z tradycyjną metodą nauczania, opartą na papierowych materiałach dydaktycznych (Hsieh et al., 2022).

Żyjemy w świecie smartfonów i smartdomów, a zrozumienie, jak działają te urządzenia, pozwala nam wykorzystywać technologię jako narzędzie pomocne w rozwiązywaniu problemów. Myślenie komputacyjne pozwala uczniom być aktywnymi, a nie biernymi użytkownikami technologii. Sposób, w jaki rozumiemy otaczającą nas technologię i sposób, w jaki zadajemy pytania dotyczące tych urządzeń i programów, będzie kluczowym czynnikiem kształtującym potencjał wytwórczy gospodarki XXI wieku. Uczniowie, którzy opanują w szkole umiejętność skutecznego i efektywnego rozwiązywania problemów, będą mieli większe szanse na sukces zawodowy i osobisty. Przygotowanie do rozwiązywania problemów z wykorzystaniem myślenia komputacyjnego może i powinno rozpocząć się już od najwcześniejszych etapów edukacji (Ung et al., 2022).

Strategie nauczania myślenia komputacyjnego na lekcjach

Jakie są skuteczne sposoby nauczania myślenia komputacyjnego? Przegląd literatury wskazuje, że autorzy sugerują kilka popularnych strategii uczenia kompetencji komputacyjnych, powiązanych z podejściem teoretycznym, w którym się poruszają (Hsu et al., 2018). Do najszerzej spopularyzowanych możemy zaliczyć zalecenia płynące z klasycznych podejść, takich jak teoria konstruktywistyczna sformułowana przez Deweya Piageta, Wygotskiego, Brunera i Glasersfelda(Csizmadia et al., 2019; Oktan &Vural, 2021) oraz strategie takie, jak uczenie się oparte na problemach, uczenie się oparte na projektach, uczenie się oparte na współpracy i uczenie się oparte na grach(Ghani et al., 2022; Yadav & Berthelsen, 2021).

Uczenie się oparte na problemach

Uczenie się poprzez rozwiązywanie problemów jest najpopularniejszą strategią doskonalenia umiejętności komputacyjnych. W tej metodzie nauczyciele przedstawiają rzeczywisty problem, który uczniowie próbują rozwiązać, wykorzystując swoją wcześniejszą wiedzę i doświadczenie. Podobnie jak w przypadku edukacji opartej na kompetencjach, uczenie się ma na celu rozwinięcie zestawu umiejętności, które uczniowie mogą zastosować w rzeczywistych sytuacjach.

Rozwiązywanie problemów z wykorzystaniem myślenia komputacyjnego może pomóc uczniom w krytycznym myśleniu, zadawaniu właściwych pytań i samodzielnym generowaniu wielu rozwiązań. Może również pomóc nauczycielom w prowadzeniu dyskusji na temat danego problemu (Tekdal, 2021).

Uczenie się oparte na projektach

W uczeniu się opartym na projektach uczniowie również otrzymują problem do rozwiązania. Główną różnicą w porównaniu z uczeniem opartym na rozwiązywaniu problemów jest to, że oczekuje się od nich wytworzenia produktu, który jest rezultatem opracowanego przez zespół rozwiązania problemu.

John Dewey, zwolennik nauczania opartego na projektach, opowiadał się za „uczeniem się przez działanie". Gdy uczniowie badają możliwe rozwiązania problemu i na końcu widzą rezultat swojego projektu, rozwijają nie tylko umiejętności komunikacyjne i umiejętności współpracy, ale również krytyczne myślenie.

Nauczyciele mogą wykorzystywać naukę opartą na projektach do rozwijania myślenia komputacyjnego wśród uczniów. Myślenie komputacyjne może być ważnym narzędziem, pomagającym uczniom w nauce systematyczności w projektowaniu rozwiązań problemów i planowaniu wytwarzania rezultatów (Belmar, 2022).

Uczenie się oparte na współpracy

W przeciwieństwie do strategii przedstawionych wyżej, uczenie się oparte na współpracy kładzie większy nacisk na ideę wspólnej pracy nad rozwiązaniem problemu. Uczniowie muszą rozwijać się nie tylko intelektualnie, ale także społecznie i emocjonalnie. W trakcie tego procesu uczą się umiejętności, które mogą wykorzystać w rzeczywistych scenariuszach pracy, gdzie umiejętności komunikacji i współpracy często dają jednostce lepsze możliwości osiągania celów.

Promowanie myślenia komputacyjnego w połączeniu z uczeniem się opartym na współpracy oznacza, że uczniowie będą mierzyli się z punktami widzenia innych osób i rozwiązywali kolizje opierając się na meritum, wypracowanym na podstawie reguł myślenia komputacyjnego. Będą też mieli szansę współpracy z rówieśnikami w celu rozwiązania problemów złożonych na tyle, że już trudnych do rozwiązania samodzielnie w zaproponowanym czasie bez wykorzystania reguł myślenia komputacyjnego (Neumann et al., 2021).

Uczenie się oparte na grach

Idea uczenia myślenia komputacyjnego w oparciu o gry może być początkowo zaskakująca. Nauczyciele projektują gry z myślą o zestawie konkretnych celów edukacyjnych. Gry mogą przybierać różne formy, takie jak gry planszowe, gry karciane, gry fabularne i łamigłówki. Jedną z popularnych obecnie opcji jest cyfrowa nauka oparta na grach i kierowana samodzielnie przez ucznia, która według niektórych jest przyszłością nauki. Algorytm pracy ucznia może zostać oparty na regułach myślenia komputacyjnego, co podwaja korzyści z tego miksu metod edukacyjnych.

Nauczyciele mogą projektować lub pozyskiwać gry specjalnie zaprojektowane w celu poprawy umiejętności myślenia komputacyjnego. W nauczaniu opartym na grach uczniowie będą bardziej zaangażowani niż w innych strategiach, a jednocześnie będą mogli popełniać błędy bez ryzyka lub poważnych konsekwencji szkolnych (Hsu et al., 2018).

Techniki uczenia myślenia komputacyjnego na lekcjach

Wybór metody nauczania myślenia komputacyjnego zależy od konkretnych umiejętności, które chcemy w danym etapie pracy ucznia rozwinąć. Wykorzystując naturalne skłonności dzieci do odkrywania i zabawy oraz zachęcając je do rozwiązywania problemów, możemy rozwijać różne aspekty myślenia uczniów. Kilka praktycznych wskazówek, dotyczących podstaw edukacji przygotowującej do posługiwania się myśleniem komputacyjnym, przedstawia KristenThorson i kilkoro innych autorów (Ezeamuzie et al., 2022; Lee et al., 2023; Munn, 2021; Thorson, 2018).

Uczenie dekompozycji

Uczenie się dekompozycji może rozpocząć się od zaproszenia uczniów do dołączenia do nauczyciela rozwiązującego problem. Nauczyciel dzieli się z uczniami złożonym, wieloetapowym problemem i pomaga uczniom podzielić go na mniejsze części, stosując przede wszystkim pytania, jak by to zadanie wykonali uczniowie. Nawet, jeśli są to młodsi uczniowie i nie są gotowi na wieloelementowe problemy lub wieloetapowe wskazówki, są w stanie zobaczyć, jak myślą dorośli. W ten sposób nauczyciel pomaga uczniom zbudować ramy strategiczne myślenia komputacyjnego.

Pomysły do wypróbowania: możesz użyć gotowego scenariusza, dotyczącego znanych uczniom sytuacji, np. planowanie przyjęcia urodzinowego, które wymaga wykonania wielu kroków. Tego rodzaju zadanie może być trudne do wykonania bez zestawu mniejszych, łatwiejszych do wykonania działań. Uczniowie mogą pomóc ci podzielić duże zadanie na mniejsze składowe, a ty możesz pomóc uczniom narysować lub zapisać przebieg ich myślenia. Notatki umożliwią uczniom powrócenie do reguł wskazujących sposób myślenia i rozwiązywania podobnych problemów w przyszłości (Munn, 2021; Thorson, 2018).

Uczenie rozpoznawania wzorów

Rozpoznawanie wzorów jest jednym z kluczowych etapów myślenia komputacyjnego. Uczenie dekompozycji można zacząć w młodszych klasach szkół podstawowych od tworzenia wzorów typu „szlaczki" i sprawdzania, z czego składa się „szlaczek". W kolejnych etapach przechodzi się do rozpoznawania bardziej złożonych struktur. Rozpoznawanie wzorców wymaga od uczniów odkrycia analogii do podobnychwzorów, obiektów lub doświadczeń i znalezienia tego, co je łączy. Znajdując to, co łączy rzeczy lub doświadczenia, uczniowie mogą zacząć rozumieć istotę trendu i uczyć się przewidywania, co będzie dalej.

Pomysły do wypróbowania: w celu przygotowania uczniów do rozpoznawania wzorów, możesz ich zachęcić do przyjrzenia się drzewom. Co łączy wszystkie drzewa? Wszystkie mają pień. Wszystkie mają korzenie. Wszystkie mają gałęzie. Chociaż istnieje wiele różnic między rodzajami drzew, te części są we wszystkich drzewach.

Następnie, wspólnie z uczniami, przygotuj rysunek schematu drzewa. Zwróć uwagę, że wszystkie drzewa mają pnie, korzenie i gałęzie. Następnie porozmawiaj o tym, jak pnie różnią się od siebie. Niektóre są grube, a inne cienkie. Niektóre są brązowe, a inne białe. Porozmawiaj o tym, jak różnią się korzenie i gałęzie. Aby pogłębić to myślenie, poproś uczniów o narysowanie obrazu drzewa, oznaczając pień, korzenie i gałęzie. Podkreśl, że choć drzewa narysowane przez uczniów mogą się różnić, to są podobne w swoich głównych częściach. Znajdowanie wzorców ułatwia wykonywanie zadań, ponieważ można wykorzystać to, co już się wie. Uczenie się rozpoznawania wzorów zwiększa świadomość otaczającego uczniów świata. Pomaga im też wykorzystywać znalezione wzory do rozwiązywania przyszłych problemów i snucia domysłów na temat świata (Ezeamuzie et al., 2022; Thorson, 2018).

Uczenie abstrakcji

Abstrahowanie polega na skupieniu się na istotnych informacjach i pominięciu tych, które nie mają znaczenia. Abstrahowanie wymaga oddzielenia głównych informacji od dodatkowych szczegółów.

Pomysły do wypróbowania: w klasach początkowych nauczyciele zazwyczaj uczą dzieci abstrahowania w czasie lektury książek, gdy poszukują głównej myśli i kluczowych szczegółów. Idąc o krok dalej, nauczyciele mogą zachęcić uczniów do poszukiwania informacji, wskazówek lub ważnych informacji, wyznaczając uczniom zadania związane z abstrahowaniem, do zrealizowania podczas czytania książki lub wykonywania doświadczenia. Gdy uczniowie wysłuchają prelegenta podczas szkolnej prezentacji na temat higieny jamy ustnej, nauczyciel może później szukać wspólnie z uczniami ogólnych zasad i szczegółów dotyczących mycia zębów. Uczniowie potrafiący abstrahować są w stanie uporządkować dostępne im informacje, a następnie znaleźć konkretne, najważniejsze dane, których potrzebują, odrzucając zbędne szczegóły. Jest to bardzo przydatna umiejętność, gdy uczniowie czytają obszerniejsze teksty, otrzymując coraz bardziej złożone informacje (Lee et al., 2023; Thorson, 2018).

Uczenie algorytmizacji

Myślenie algorytmiczne oznacza tworzenie reguł rozwiązywania problemów. Jest to tworzenie listy zasad, których należy przestrzegać w celu rozwiązania problemu. W klasach młodszych, dzieci mogą nauczyć się, że kolejność wykonywania czynności może mieć znaczenie.

Pomysły do wypróbowania: możesz poprosić uczniów, aby pomyśleli o zrobieniu kanapki. Co musimy zrobić najpierw? Co następnie? Co się stanie, jeśli nałożę ketchup na kanapkę przed położeniem sera i sałaty? Rozmowa o sekwencji i kolejności pomoże uczniom zbudować podstawy myślenia algorytmicznego.

Aby zachęcić uczniów do myślenia algorytmicznego, poproś ich o zaplanowanie drogi z klasy do sali gimnastycznej, zapisując liczbę kroków. Następnie pozwól im to

wypróbować! Poproś ich również, aby pomyśleli o swoich porannych czynnościach. Jakie działania podejmują uczniowie z rana przed wyruszeniem do szkoły każdego dnia? Jak ich kolejność wpłynęłaby na przygotowanie i dotarcie do szkoły? Poproś uczniów o zastanowienie się nad tym, w jaki sposób zmiana kolejności czynności zmienia wynik. Taka dyskusja pomaga myśleć w sposób algorytmiczny i wprowadzać zmiany w swoim planie, ukierunkowane na polepszenie rezultatów (Thorson, 2018; Whitney-Smith, 2023).

Narzędzia cyfrowe przydatne w nauczaniu myślenia komputacyjnego

Nauczanie myślenia komputacyjnego może być wspierane przez platformy, które pozwalają na samodzielne ćwiczenie umiejętności myślenia komputacyjnego, jak i umożliwiają włączenie myślenia komputacyjnego do programu nauczania w zabawny i angażujący uczniów sposób. Platformy przykładowe to:

- Ozobot: https://ozobot.com/ Ozobot to innowacyjna platforma, która pozwala uczyć myślenia komputacyjnego za pomocą robotów. Ozoboty to inteligentne roboty, które mogą podążać za liniami i poleceniami narysowanymi przez uczniów przy użyciu różnych kolorów. Ozobot oferuje również lekcje wideo z różnych przedmiotów, możliwe do udostępniania uczniom w klasach 3–5. Lekcje te pomagają uczniom nauczyć się kodować roboty za pomocą kodów wizualnych i stosować myślenie komputacyjne w różnych dziedzinach.
- Scratch: https://scratch.mit.edu/ Scratch to darmowa i łatwa w użyciu platforma, która umożliwia tworzenie interaktywnych gier i animacji przy użyciu kolorowych bloków kodu. Scratch obsługuje ponad 70 języków i oferuje różnorodne zasoby dla nauczycieli, takie jak samouczki, karty pracy, scenariusze lekcji i strategie wykorzystania. Można także skorzystać z programu nauczania CS First firmy Google lub modułów projektowych Code Club, aby korzystać ze Scratcha w swojej klasie. Dzięki Scratch można zainspirować uczniów do wyrażania swojej kreatywności i współpracy z innymi podczas nauki myślenia komputacyjnego.
- CTApp: https://ctapp.eu/ CTApp to nowa gra mobilna w formacie escape room. Celem gracza jest udzielenie odpowiedzi na serię pytań i rozwiązanie problemów. Aplikacja łączy w sobie koncepcje myślenia komputacyjnego i gry poważnej. Gra może być używana przez uczniów szkół podstawowych w wieku od 12 do 18 lat, którzy mówią po angielsku, grecku, włosku lub polsku. Z aplikacji można też korzystać na zajęciach szkolnych.
- Kodable: https://www.kodable.com/ Kodable to kompleksowa platforma do kodowania, która zawiera kompletny program nauczania myślenia komputacyjnego. Kodable pomaga uczniom w nauce kodowania przy użyciu JavaScript, Python, HTML, CSS, Java i nie tylko. Kodable pomaga również uczniom rozwijać inne umiejętności, takie jak praca zespołowa, odporność, komunikacja i wspólne projektowanie. Kodable jest dostępny na urządzenia z systemem iOS i komputery stacjonarne, a każda lekcja zawiera wskazówki dotyczące instrukcji, słowniczki i materiały, które pomagają skutecznie uczyć.

Ocenianie myślenia komputacyjnego w szkole

Myślenie komputacyjne (CT) jest postrzegane jako kluczowa kompetencja XXI wieku, ale nie ma uniwersalnego modelu oceny kompetencji CT w edukacji. Kompetencje CT można oceniać za pomocą: kwestionariuszy, testów/zadań, obserwacji, wywiadów i analizy produktów. Istnieją również narzędzia niezwiązane bezpośrednio z oceną CT, ale oceniające wymiary związane z CT, takie jak testy badające logiczne myślenie/rozumowanie, umiejętności analizy danych, myślenie krytyczne, automatyzację, planowanie poznawcze, koordynację, kompetencje obliczeniowe i wiele innych. Systematyczny przegląd narzędzi pokazuje, że nie odnoszą się one do jednolitego modelu kompetencji CT, zróżnicowanego ze względu na wiek. Przeprowadzono różnorodne badania oceniające CT u uczniów w różnym wieku, jednak nie jest jasne, jakie kompetencje uczniowie powinni osiągnąć w jakim wieku, ponieważ te same wymiary i narzędzia są stosowane dla uczniów w różnym wieku (Babazadeh&Negrini, 2022).

Przydatną pomocą dla wdrożenia myślenia komputacyjnego w praktykę codziennej pracy nauczyciela z klasą może być zestaw przykładowych scenariuszy lekcji. Zamieszczone niżej scenariusze zostały przetestowane przez nauczycieli i zawierają wskazania do nauczania przedmiotów informatycznych, a także literatury i języka ojczystego. Propozycje wskazują możliwość rozłożenia zajęć na klika jednostek lekcyjnych lub wdrożenia ich na jednej lekcji. Zachęcamy też wszystkich nauczycieli zainteresowanych myśleniem komputacyjnym do dzielenia się opracowanymi przez siebie pomysłami lekcji.

Robert Porzak
Lubelska Akademia WSEI

7.1. Scenariusz lekcji 1

Autor / Nauczyciel: Robert Porzak

Przedmiot: Zajęcia informatyczne, przedmioty komputerowe itp.

Poziom (ISCED, trudność)**:** Poziom ISCED 3 – wykształcenie średnie II stopnia

Temat: Zostań szpiegiem (podstawy) – ukryj wiadomość na obrazku

Wymagane umiejętności lub wiedza (powiązanie z poprzednią lekcją):

a) podstawowa znajomość programów graficznych (GIMP lub podobny)
b) zrozumienie, czym jest autostereogram (aby dowiedzieć się więcej, przeczytaj więcej o autostereogramach: https://en.wikipedia.org/wiki/Autostereogram)

Czas wymagany na aktywność przed zajęciami: 2 godziny
Czas wymagany na aktywność w klasie: 1 godzina
Czas wymagany na aktywność po zajęciach: 2 godziny

Historia, kanwa, wyzwania dla ucznia (opcjonalnie, motywacyjne)

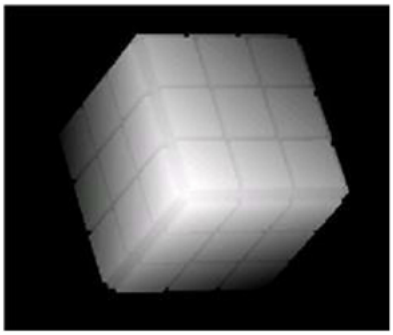
Wszyscy wiemy, że trudno jest być szpiegiem. Jedną z podstawowych kompetencji dobrego szpiega jest utajnianie komunikacji. Można próbować używać nowoczesnych kodów komputerowych lub specjalnych narzędzi elektronicznych do kodowania komunikacji, ale odbiorca będzie musiał polegać na komputerze lub elektronice, aby odszyfrować wiadomość. Istnieją jednak metody kodowania wizualnego, które nie są łatwe do odszyfrowania przez osoby nieupoważnione, które nie znają mechanizmu procesu. Twoja tajna wiadomość zakodowana w ten sposób może być widoczna dla wszystkich, ale tylko poinformowane osoby mogą ją odszyfrować. Czy jesteś gotowy na lekcję o tym, jak zostać szpiegiem wysyłającym ukryte wiadomości w obrazach?

1. **Nowy materiał ucznia (przed zajęciami)**
   a) dowiedz się, jak przygotować w programie graficznym płaską (dwuwymiarową) mapę wzorów (obraz złożony z małych, kolorowych kropek/obrazków/wzorów). Przykładowe dwuwymiarowe wzory powinny wyglądać tak, jak pokazano poniżej:

Źródło: https://www.easystereogrambuilder.com/

Dowiedz się, jak to zrobić w darmowym programie GIMP https://www.gimp.org/ lub innym oprogramowaniu. Możesz zobaczyć, jak to zrobić na przykład tutaj: https://www.youtube.com/watch?v=TKhs7F0hAik

b) dowiedz się, jak przygotować mapę głębi rozumianą jako prosty czarno-biały obraz wyglądający jak negatyw, rozmazany. Przykładowa mapa głębi może wyglądać, jak ta pokazana poniżej:



Źródło: https://www.easystereogrambuilder.com/

Dowiedz się, jak to zrobić w darmowym Blenderze https://www.blender.org/ lub innym oprogramowaniu. Możesz zobaczyć, jak to zrobić na przykład tutaj: https://www.youtube.com/watch?v=3P6DVDy6DG0

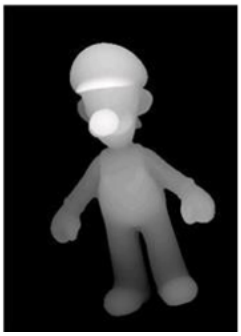
2. Zajęcia w klasie

a) przygotuj płaską (dwuwymiarową) mapę wzorów (obraz złożony z małych, kolorowych kropek/obrazków/wzorów). Wzór powinien być obrazem z wariacjami kolorów. Najważniejszą rzeczą jest to, że nie powinien mieć poziomych obszarów (pasków) o tym samym kolorze:

Źródło: https://www.easystereogrambuilder.com/

Możesz użyć darmowego GIMP https://www.gimp.org/ lub innego oprogramowania,

b) przygotuj mapę głębi o odpowiednim rozmiarze jako mapę kolorów (prosty czarno-biały obraz wyglądający jak negatyw, rozmazany). Im bielsze punkty, tym bliżej ciebie się znajdują. Tak więc czerń reprezentuje tło, a różne odcienie bieli reprezentują punkty unoszące się nad tłem:



Źródło: https://www.easystereogrambuilder.com/

Użyj darmowego Blendera https://www.blender.org/ lub innego oprogramowania,

c) pokryj oba obrazy nadpróbkowaniem (dodaniem głębi) we wspomnianych programach. Możesz również użyć wtyczki GIMP, jeśli chcesz spróbować: https://gimplearn.net/viewtopic.php?t=2974

d) napisz algorytm blokowy pokazujący kroki niezbędne do wykonania wiadomości tekstowej ukrytej w grafice.

3. Działania po zajęciach

a) zweryfikuj swój algorytm, przygotowując dwie wiadomości tekstowe ukryte w autostereogramach. Postępuj zgodnie z algorytmem krok po kroku, sprawdzając, czy jest on wystarczająco uniwersalny,

b) napisz swoje wnioski na temat możliwości kodowania algorytmów ukrywających wiadomości tekstowe w obrazach jako akapit o długości około 50 słów.

4. Ocena i ewaluacja

Efekt uczenia się – uczeń opracowuje poprawne algorytmy adekwatne do zadania:

√ E – dla prostych problemów z podaną techniką algorytmiczną
√ D – dla nietrywialnych problemów z daną podpowiedzią
√ C – dla nietrywialnych problemów
√ B – dla trudniejszych problemów przy użyciu najlepszej metody z daną podpowiedzią
√ A – dla trudniejszych problemów przy użyciu najlepszej metody.

Fahimeh Mousavi
INNOVA S.r.l.

7.2. Scenariusz lekcji 2

Autor / Nauczyciel: Fahimeh Mousavi / Cristina Fregonese

Przedmiot: Sztuka, sztuki językowe i studia medialne oraz edukacja postaci

Poziom (ISCED, trudność)**:** Poziom ISCED 3 – wykształcenie średnie II stopnia

Temat: Storyboardy w grach wideo

Wymagane umiejętności lub wiedza (powiązanie z poprzednią lekcją):

a) zrozumienie, czym jest storyboard (https://en.wikipedia.org/wiki/Storyboard)
b) podstawowa wiedza w zakresie umiejętności pisania

Czas wymagany na aktywność przed zajęciami: 2 godziny
Czas wymagany na aktywność w klasie: 1 godzina
Czas wymagany na aktywność po zajęciach: 2 godziny

Historia, kanwa, wyzwania dla ucznia (opcjonalnie, motywacyjne)

Ponieważ gry wideo są medium wizualnym, storyboardy są świetnym sposobem na przedstawienie historii opowiadanej przez grę wideo. Uczniowie będą stosować i dzielić się swoimi pomysłami, osobistymi perspektywami i kreatywnymi przemyśleniami poprzez ćwiczenie storyboardów. W szczególności uczniowie będą mieli okazję stworzyć mapę historii za pomocą storyboardów i zademonstrować, w jaki sposób wskazówki wizualne mogą pomóc w przekazywaniu uczuć, pomysłów i zrozumienia podczas opowiadania historii. Każdy uczeń użyje storyboardu, aby podzielić się stworzoną przez siebie historią.

5. Nowy materiał ucznia (przed zajęciami)

Używane słownictwo:

POV (Point of View): POV, czyli ujęcie z punktu widzenia, jest właśnie tym: pozwala widzom zobaczyć, co dzieje się oczami postaci.

Ujęcie główne: Termin odnoszący się do ujęcia, które trwa przez całą długość sceny i pokazuje wszystkie widoczne postacie.

Zoom: Zoomowanie to ruch obiektywu kamery, a nie ruch samej kamery. Przybliżanie oznacza regulację obiektywu w celu przybliżenia obiektu, podczas gdy oddalanie oznacza coś przeciwnego: regulację obiektywu w celu objęcia większej części sceny.

Śledzenie: Ujęcie śledzące to kolejny sposób na podążanie za obiektem. Ten rodzaj ujęcia obejmuje przesuwanie całej kamery z jednego miejsca do drugiego i często podąża za poruszającym się obiektem. Śledzenie może obejmować przesuwanie kamery za pomocą szyn lub na wózku, lub może być wykonywane z ręki.

Pytania przewodnie:

1. W jaki sposób wybór ujęć wpływa na ogólny nastrój sceny?
2. Jakie rodzaje ujęć (zbliżenie vs. ujęcie szerokie) przekazują różne znaczenia i emocje?

Materiały:

a) Aplikacja Scratch do tworzenia projektów – Pobierz
b) Adobe AIR (wymagane do uruchomienia Scratch) – Pobierz
c) Szablony scenorysów – Szablon pierwszy, Szablon drugi
d) Film lub gra wideo, którą uczniowie wykorzystają do stworzenia scenorysu swojej ulubionej sceny.

6. Zajęcia w klasie

Aktywność niekomputerowa

Część A: Wyświetl scenę z ulubionej gry wideo lub filmu.

a. Czy potrafisz znaleźć moment, w którym zmiana rodzaju ujęcia (ujęcie szerokie, ujęcie średnie, zbliżenie) pomaga podkreślić punkt fabuły?
b. W jaki sposób inscenizacja i kadrowanie zwiększają emocjonalny wpływ sceny?

Część B: Wybierz scenę lub sekwencję z ulubionej gry wideo lub filmu i stwórz storyboard dla krótkiej sceny (o długości 3–5 minut).

a. Naszkicuj kadr dla każdego ujęcia w wybranej scenie. Wskaż w opisie, czy jest to ujęcie szerokie, średnie czy zbliżenie.

Aktywność komputerowa

Scratch pozwala programować własne interaktywne historie, gry i animacje.

Storyboarding to świetne ćwiczenie wprowadzające, które można wykonać z uczniami przed użyciem Scratcha, aby pomóc zaplanować i zmapować fabułę gry lub interaktywnej historii. To ćwiczenie poprowadzi cię przez ten proces.

Po pierwsze, wybierz kilka wspólnych wytycznych dla uczniów, których będą przestrzegać podczas tworzenia interaktywnej historii lub gry. Mogą one obejmować temat, który każda interaktywna historia lub gra powinna poruszać (na przykład nękanie, społeczność, tożsamość), a także miejsce, w którym rozgrywa się historia. Poniżej przedstawiono ramy dla sposobu, w jaki można ustrukturyzować historię:

- Akt 1 to „początek". Niech uczniowie odpowiedzą na pytania Gdzie, Kiedy, Kto, Co i Dlaczego.
- Akt 2 to „środek", w którym postacie próbują osiągnąć cele i napotykają konflikty.
- Akt 3 to „koniec", w którym konflikt zostaje rozwiązany, a prawdziwy charakter bohatera ujawniony.

7. Działania po zajęciach

Po tym, jak uczniowie zilustrują swoje storyboardy, wytnij każdą komórkę i zmień kolejność wydarzeń. Poproś uczniów, aby ułożyli komórki w różnych kolejnościach i zastanowili się, jak zmienia to historię, zanim zdecydują się na ostateczny wynik. Proces ten wprowadza koncepcję mediów doświadczalnych i demonstruje podstawy struktury opowieści, pokazując, jak historia może się zmieniać w zależności od kolejności wydarzeń.

Po ukończeniu storyboardów każdy uczeń ma mapę do rozpoczęcia projektu w Scratch.

8. Ocena i ewaluacja

Efekt uczenia się – student rozwija umiejętności

a. Powtarzanie podobieństw między przedmiotami, pomysłami i problemami.
b. Informacje ilościowe lub jakościowe (np. tekst, symbole itp.) zebrane do analizy.
c. Znajdowanie wzorców wśród zdekomponowanych problemów, które mogą pomóc nam w rozwiązywaniu bardziej złożonych problemów.
d. Wyizoluj kluczowe szczegóły, ignorując pozostałe elementy.
e. Proces dzielenia złożonych problemów na mniejsze, łatwiejsze w zarządzaniu części.
f. Twórz schematy blokowe, storyboardy lub inne formy reprezentacyjne, aby przygotować się do pisania.
g. Identyfikowanie wzorców w piśmie poprzez tworzenie i udoskonalanie pisemnych algorytmów, pseudokodu lub schematów blokowych.

Eftychia Xerou
Centre for Advancement of Research
and Development in Educational Technology Ltd.

7.3. Scenariusz lekcji 3

Autor / Nauczyciel: Eftychia Xerou

Przedmiot: Zajęcia informatyczne, przedmioty komputerowe itp.

Poziom (ISCED, trudność)**:** Szkolnictwo średnie – gimnazjum

Temat: Stworzenie interaktywnej historii przedstawiającej społeczne zjawisko wpływu człowieka na środowisko.

Wymagane umiejętności lub wiedza (powiązanie z poprzednią lekcją):

a. Programowanie blokowe (Scratch)
b. Wiedza dotycząca wpływu człowieka na środowisko, a dokładniej niszczenia środowiska (zjawisko społeczne, w którym działanie człowieka degraduje i zanieczyszcza różne zasoby naturalne, takie jak ziemia, woda, powietrze, minerały i lasy).

Czas wymagany na aktywność przed zajęciami: 2 godziny
Czas wymagany na aktywność w klasie: 1 godzina
Czas wymagany na aktywność po zajęciach: 2 godziny

Historia, kanwa, wyzwania dla ucznia (opcjonalnie, motywacyjne)

Jak wszyscy rozumiemy, niszczenie środowiska naturalnego, w którym żyjemy, jest niezwykle ważnym zjawiskiem społecznym, które wpływa bezpośrednio na każdego człowieka w ciągu jego życia. Degradacja środowiska to pogorszenie jego stanu poprzez wyczerpywanie się zasobów, takich jak powietrze, woda i gleba; niszczenie ekosystemów i wymieranie dzikiej przyrody. Podczas tej lekcji uczniowie będą musieli przedstawić degradację środowiska i jeden z jej wielu różnych aspektów wraz z konsekwencjami dla przyrody. Czy jesteśmy gotowi zrozumieć, jak prowadzić badania, dowiedzieć się więcej, zrozumieć konsekwencje i być proaktywnym na przyszłość?

1. **Nowy materiał ucznia (przed zajęciami)**
   a) przeprowadź badania dotyczące jednego konkretnego problemu: zanieczyszczenia, zanieczyszczenia powietrza, smogu, globalnego ocieplenia, wymierania gatunków i zdecyduj, na którym problemie skupi się interaktywna kreacja. Postaraj się przedstawić problem, jego przyczyny, konsekwencje i rozwiązania. Zbierz swoje wnioski w dokumencie Word,

b) dowiedz się, jak przygotować prototyp storyboardu dla scenariusza w programie online, aby przedstawić 4 wyżej wymienione obszary w fabule. Napisz historię i stwórz swój storyboard tutaj: Free Storyboard Creator – Easily Make Storyboards Online | Canva. Pomoc dla kreatora scenorysów Canva: Design Storyboard Activities | Canva for Education – YouTube.

2. Zajęcia w klasie

a) po rozbiciu złożonego problemu na 4 różne części (przyczyny, problem, konsekwencje i rozwiązania) i przedstawieniu problemu w interaktywnej historii, poprzez stworzenie prototypu, następnym krokiem jest stworzenie historii w Scratchu. Scratch – Imagine, Program, Share (mit.edu) Samouczek oprogramowania: How to Make a Story in Scratch | Tutorial – YouTube,

b) interaktywna historia musi mieć co najmniej dwóch głównych bohaterów (spritey), 4 różne tła, dźwięki i słowa dla każdej postaci, a interaktywna część historii musi przedstawiać wybór. Wybór może dotyczyć różnych konsekwencji (na przykład, jeśli główny bohater zdecyduje się chodzić po mieście, liczba samochodów może się zmniejszyć, a poziom zanieczyszczenia powietrza może się zmniejszyć). Ta część musi być przedstawiona jako jeden wybór dla uczniów, a drugi wybór może być odwrotny.

3. Działania po zajęciach

a) sprawdź, czy interaktywna historia działa, wybierając opcję uruchomienia i przetestowania obu podanych opcji,
b) napisz swoje wnioski na temat możliwości zmiany drobnych codziennych nawyków, aby wpłynąć na środowisko (100 słów).

4. Ocena i ewaluacja

Efekty uczenia się:

1. Studenci dowiedzieli się, jak prowadzić badania.,
2. Uczniowie dowiedzieli się o niszczeniu środowiska naturalnego.
3. Uczniowie przedstawiają historie na storyboardach i w interaktywnych reprezentacjach.

Odniesienia: https://archive.unescwa.org/environmental=-degradation#:~:text-Environmental%20degradation%20is%20the%20deterioration,and%20the%20extinction%20of%20wildlife.

Bibliografia

7 examples of algorithms in everyday life for students. (2023, February 13). Learning.com. https://www.learning.com/blog/7-examples-of-algorithms-in-everyday-life-for-students/

Ackermann, E., (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication, 5*(3), p.438.

Adler, R. F., Hibdon, J., Kim, H., Mayle, S., Pines, B., & Srinivas, S. (2023). Assessing computational thinking across a STEM curriculum for pre-service teachers. *Education and Information Technologies, 28*(7), 8051–8073. https://doi.org/10.1007/s10639-022-11508-4

Ahamed, S.I., Brylow, D., Ge, R., Madiraju, P., Merrill, S.J., Struble, C.A. and Early, J.P. (2010). Computational thinking for the sciences: a three day workshop for high school science teachers. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). ACM.

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal, 55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Armoni, M. (2013). On teaching abstraction in CS to novices. *Journal of Computers in Mathematics and Science Teaching, 32*(3), 265-284.

Babazadeh, M., & Negrini, L. (2022). How Is Computational Thinking Assessed in European K-12 Education? A Systematic Review. *International Journal of Computer Science Education in Schools, 5*(4). ERIC. https://search.ebscohost.com/login.aspx?direct=true&db=eric&AN=EJ1366667&lang=pl&site=ehost-live

Banilower, E. R., Smith, P. S., Weiss, I., Malzahn, K. A., Campbell, K. M., & Weis, A. M. (2013). *Report of the 2012 national survey of science and mathematics education*. Chapel Hill: Horizon Research, Inc

Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., ... & Stupurienė, G. (2015). Concepts in K-9 computer science education. *Proceedings of the 2015 ITiCSE on working group reports* (pp. 85-116).

Belmar, H. (2022). Review on the teaching of programming and computational thinking in the world. *Frontiers in Computer Science, 4*. https://www.frontiersin.org/articles/10.3389/fcomp.2022.997222

Bocconi S., Chioccariello A., Dettori G., Ferrari A., Engelhardt K. (2016) Developing computational Thinking in Compulsory Education. Implication for policy and practice. *JRC Working Papers* JRC104188, Joint Research Centre

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice. Resource document* (EUR 28295 EN). https://doi.org/10.2791/792158.

Burgett, T., Folk, R., Fulton, J., Peel, A., Pontelli, E. and Szczepanski, V. (2015). DISSECT: Analysis of pedagogical techniques to integrate computational thinking into K-12 curricula. *Frontiers in Education Conference (FIE)*, 2015. 32614 2015. IEEE (pp. 1-9). IEEE.

Calderon, A., Crick, T., & Tryfona, C. (2015). *Developing computational thinking through pattern recognition in early years education (Version 1).* Cardiff Metropolitan University. https://hdl.handle.net/10779/cardiffmet.21262746.v1 ([])

Cetin, I., & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *The Journal of Mathematical Behavior, 47*, 70-80.

Chiprianov, V. and Gallon, L. (2016), July. Introducing Computational Thinking to K-5 in a French Context. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 112-117). ACM.

Costa, E. J. F., Campos, L. M. R. S., & Guerrero, D. D. S. (2017). Computational thinking in mathematics education: A joint approach to encourage problem-solving ability. *2017 IEEE Frontiers in Education Conference (FIE)* (pp. 1–8). IEEE.

Csizmadia A., Curzon P., Dorling M., Humphreys S. (2015). *Computational thinking. A guide for teachers*. Computing At School (CAS).

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: a guide for teachers.

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the Constructionist Learning Theory with Computational Thinking Classroom Activities. Informatics in Education, 18(1), 41–67. ERIC.

Cui, Z., & Ng, O. L. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research, 59*(5), 988–1012.

Department of Education, UK. (2023). *Early Years Foundation Stage Profile 2024 handbook*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1109972/Early_Years_Foundation_Stage_profile_2023_handbook.pdf

Dogan, A. (2020). Algorithmic Thinking in Primary Education. International *Journal of Progressive Education, 16*(4), 286-301.

Dominici P. (2016) *Il grande equivoco. Ripensare l'educazione (#digitale) per la Società Ipercomplessa.* Nòva.

Dorodchi, M., Dehbozorgi, N., Fallahian, M., & Pouriyeh, S. (2021). Teaching Software Engineering using Abstraction through Modeling. *Informatics in Education, 20*(4).

Evripidou, S., Amanatiadis, A., Christodoulou, K., & Chatzichristofis, S. A. (2021). Introducing algorithmic thinking and sequencing using tangible robots. *IEEE Transactions on Learning Technologies, 14*(1), 93-105.

Examples of Abstraction in Everyday Life. (2022). *Learning.com.* https://www.learning.com/blog/examples-of-abstraction-in-everyday-life/

Ezeamuzie, N. O., Leung, J. S. C., Garcia, R. C. C., & Ting, F. S. T. (2022). Discovering computational thinking in everyday problem solving: A multiple case study of route planning. *Journal of Computer Assisted Learning, 38*(6), 1779–1796. Academic Search Ultimate.

Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research, 87*(4), 834–860.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. *International conference on informatics in secondary schools-evolution and perspectives* (pp. 159-168). Springer, Berlin, Heidelberg.

Futschek, G., & Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. *Proceedings of the 2010 constructionist approaches to creative learning, thinking and education: Lessons for the 21st century (constructionism 2010)*, 1-10.

Gabbari M., Gagliardi R., Gaetano A., Sacchi D. (2020). Integrare Coding e Pensiero Computazionale nella didattica. *Bricks Magazine* http://www.rivistabricks.it/2020/03/03/integrare-coding-e-pensiero-computazionale-nella-didattica/

Ghani, A., Griffiths, D., Salha, S., Affouneh, S., Khalili, F., Khlaif, Z. N., & Burgos, D. (2022). Developing Teaching Practice in Computational Thinking in Palestine. *Frontiers in Psychology, 13*. https://www.frontiersin.org/articles/10.3389/fpsyg.2022.870090

Giacalone S. (2020). Che cos'è il pensiero computazionale? Training Course on Computational Thinking, Problem solving e Coding. *Cessaniti, 16/20.*

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., & Pea, R. (2018). Computational Thinking: A Competency Whose Time Has Come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science*

Education: Perspectives on Teaching and Learning in School (pp. 19–38). Bloomsbury Academic. https://doi.org/10.5040/9781350057142.ch-003

Haberman, B., & Muller, O. (2008). Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes. *2008 38th Annual Frontiers in Education Conference* (pp. F1C-7). IEEE.

Haddad, R.J. and Kalaani, Y. (2015). Can computational thinking predict academic performance?. *Integrated STEM Education Conference (ISEC), 2015 IEEE* (pp. 225-229).

Hazzan, O. (2008). Reflections on teaching abstraction and other soft ideas. *ACM SIGCSE Bulletin, 40*(2), 40-43.

Howard, W.R. (2007), Pattern Recognition and Machine Learning, *Kybernetes, Vol. 36* No. 2, pp. 275-275. https://doi.org/10.1108/03684920710743466)

Howell, L., Jamba, L., Kimball, A.S. and Sanchez-Ruiz, A. (2011, March). Computational thinking: modeling applied to the teaching and learning of English. *Proceedings of the 49th Annual Southeast Regional Conference* (pp. 48-53). ACM.

Hromkovic, J., Kohn, T., Komm, D., & Serafini, G. (2017). Algorithmic thinking from the start. *Bulletin of EATCS, 1*(121).

Hsieh, M.-C., Pan, H.-C., Hsieh, S.-W., Hsu, M.-J., & Chou, S.-W. (2022). Teaching the Concept of Computational Thinking: A STEM-Based Program With Tangible Robots on Project-Based Learning Courses. *Frontiers in Psychology, 12*. https://www.frontiersin.org/articles/10.3389/fpsyg.2021.828568

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

https://archive.unescwa.org/environmental-degradation#:~:text=Environmental%20degradation%20is%20the%20deterioration,and%20the%20extinction%20of%20wildlife.

Isbell, C.L., Stein, L.A., Cutler, R., Forbes, J., Fraser, L., Impagliazzo, J., Proulx, V., Russ, S., Thomas, R. and Xu, Y. (2010). (Re) defining computing curricula by (re) defining computing. *ACM SIGCSE Bulletin, 41*(4), pp.195-207.

Kiss, G., & Arki, Z. (2017). The influence of game-based programming education on the algorithmic thinking. *Procedia-Social and Behavioral Sciences, 237*, 613-617.

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189. doi: 10.1016/j.compedu.2018.08.026

Koppelman, H., & Van Dijk, B. (2010). Teaching abstraction in introductory courses. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 174-178).

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM, 50*(4), 36-42.

Krist, C., Elby, A., Good, J., Gupta, A., Sohr, E. R., & Yadav, A. (2017). Integrating computational thinking strategies that support science inquiry: A case study from a summer PD. *Paper presented at the Annual Meeting of American Educational Research Association*, San Antonio, TX

Larsen, M. (2010). How to recognize word patterns in a foreign language: Autolingual – learn a foreign language by yourself. *AutoLingua*l. https://autolingual.com/word-pattern-recognition/

Lee, J., Joswick, C., & Pole, K. (2023). Classroom Play and Activities to Support Computational Thinking Development in Early Childhood. *Early Childhood Education Journal, 51*(3), 457–468. Academic Search Ultimate.

Lockwood, James & Mooney, Aidan. (2017). Computational Thinking in Education: Where does it Fit? A systematic literary review. *International Journal of Computer Science Education in Schools.2*.

Lv, L., Zhong, B. & Liu, X. (2022). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Educ Inf Technol*. https://doi.org/10.1007/s10639-022-11518-2

Malik, S. I., Mathew, R., Tawafak, R. M., & Khan, I. (2019). Gender difference in perceiving algorithmic thinking in an introductory programming course. *EDULEARN19 Proceedings of the International Conference on Education and New Learning Technologies (Vol. 18*, pp. 8246-8254).

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 1-29). ACM Press Digital Library. https://doi.org/10.1145/2713609.2713610.

McCormick, K. & Hall, J. (2022). Computational thinking learning experiences, outcomes, and research in preschool settings: a scoping review of literature. *Education and Information Technologies. 27*. 1-36. 10.1007/s10639-021-10765-z.

Mezak, J., & Papak, P. P. (2018, May). Learning scenarios and encouraging algorithmic thinking. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 0760-0765). IEEE.

Midoro V. (2015). *La Scuola ai tempi del Digitale – Istruzioni per costruire una scuola nuova*. FrancoAngeli.

Milková, E. (2012). Development of algorithmic thinking and imagination: base of programming skills. *Proceedings of the 16th WSEAS International Conference on Computers*.

Mooney, A., Duffin, J., Naughton, T., Monahan, R., Power, J. and Maguire, P. (2014). *PACT: An initiative to introduce computational thinking to second-level education in Ireland*. Conference: International Conference on Engaging Pedagogy.

Munn, C. (2021). A Qualitative Study Exploring Robots as a Potential Classroom Tool for Teaching Computational Thinking within a Sixth-Grade Class. *Journal of Computers in Mathematics and Science Teaching, 40*(3), 229–264. ERIC.

Neumann, M. D., Dion, L., & Snapp, R. (2021). *Teaching Computational Thinking: An Integrative Approach for Middle and High School Learning.* The MIT Press. https://doi.org/10.7551/mitpress/11209.001.0001

Oktan, S., & Vural, S. (2021). A teaching strategies model experiment for computational design thinking. *TECHNE: Journal of Technology for Architecture & Environment*, 154–158. Academic Search Ultimate.

Parry, M. (2021). Abstraction: The important bits. Hello World. https://helloworld.raspberrypi.org/articles/hw16-abstraction-the-important-bits

Peel, A., & Friedrichsen, P. (2018). Algorithms, abstractions, and iterations: Teaching computational thinking using protein synthesis translation. *The American Biology Teacher, 80*(1), 21-28.

Pensiero Computazionale - Una guida per insegnanti CNR (2016). Computing At School. https://pensierocomputazionale.itd.cnr.it/pluginfile.php/957/mod_page/content/7/Guida%20al%20Pensiero%20Computazionale.pdf

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior, 105*, 105849. https://doi.org/10.1016/j.chb.2018.12.027

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272–283.

Rich, K.M., Yadav, A. & Larimore, R.A. Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. *Educ Inf Technol 25*, 3161–3188 (2020). https://doi.org/10.1007/s10639-020-10115-5

Samarasinghe, S. (2006). *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition* (1st ed.). Auerbach Publications. https://doi.org/10.1201/9780849333750

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: a theoretical framework. *Education and Information Technologies, 18*(2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Shen, X., Efros, A. A., & Aubry, M. (2019). Discovering visual patterns in art collections with spatially-consistent feature learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9278-9287).

Shuiyan He, Yongmin Hang, & Yi Ding. (2014). Teaching method based on computational thinking a case research. *2014 9th International Conference on Computer Science & Education*, 817–820. https://doi.org/10.1109/ICCSE.2014.6926576

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Sigayret, K., Tricot, A., & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education, 184*, 104505.

Sung, W., & Black, J. B. (2020). Factors to consider when designing effective learning: infusing computational thinking in mathematics to support thinking-doing. *Journal of Research on Technology in Education, 53*(4), 404–426.

Symposium on Computer Science Education (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 906–912. https://doi.org/10.1145/3287324.3287431

Tekdal, M. (2021). Trends and development in research on computational thinking. *Education and Information Technologies, 26*(5), 6499–6529. https://doi.org/10.1007/s10639-021-10617-w

The Bowers Institute. (n.d.). *TECH TIP: Computational Thinking*. https://www.thetech.org/media/l0nasrfv/techtip_computationalthinking.pdf

Thorson, K. (2018). *Early Learning Strategies for Developing Computational Thinking Skills. Getting Smart.* https://www.gettingsmart.com/2018/03/18/early-learning-strategies-for-developing-computational-thinking-skills/

Tsalapatas, H., Heidmann, O., Alimisi, R., & Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the Petru Maior University of Targu Mures, 9*(1), 56-63.

Ung, L.-L., Labadin, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education, 177*, 104379. https://doi.org/10.1016/j.compedu.2021.104379

Uzumcu, O., Bay, E. The effect of computational thinking skill program design developed according to interest driven creator theory on prospective teachers. *Educ Inf Technol 26*, 565–583 (2021). https://doi.org/10.1007/s10639-020-10268-3

van Zanten, M., & van den Heuvel-Panhuizen, M. (2018). Opportunity to learn problem solving in dutch primary school mathematics textbooks. *ZDM: The International Journal on Mathematics Education, 50*(5), 827–838.

Waite, J. L., Curzon, P., Marsh, W., Sentance, S., & Hadwen-Bennett, A. (2018). Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education in Schools, 2*(1), 14-40.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147

White, P., & Mitchelmore, M. C. (2010). Teaching for abstraction: A model. *Mathematical Thinking and Learning, 12*(3), 205-226.

Whitney-Smith, R. M. (2023). The Emergence of Computational Thinking in National Mathematics Curricula: An Australian Example. *Journal of Pedagogical Research, 7*(2), 41–55. ERIC.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35. Business Source Ultimate.

Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why. *The link Magazine, 6*, 20-23. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf Wing, J. M. (2006). Computational Thinking, CACM 49(3), pp. 33-35.

Wing J. M. (2016). *Computational thinking, 10 years later*. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/

Wray S. (2012). *Not a Tool, but a Philosophy of Knowledge*. http://www.stuartwray.net/philosophy-of-knowledge.pdf

Yadav, A., & Berthelsen, U. (2021). *Computational Thinking in Education: A Pedagogical Perspective*. Routledge. https://doi.org/10.4324/9781003102991

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*, 55–62. https://doi.org/10.1145/2994591

Yihuan D., Catete V., Jocius R., Lytle N., Barnes T., Albert J., Joshi D., Robinson R., and Andrews A. (2019). PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. *Proceedings of the 50th ACM Technical.*

Myślenie komputacyjne to umiejętność rozumienia, analizowania oraz rozwiązywania problemów przy wykorzystaniu narzędzi i koncepcji informatycznych.
Wartość praktyczna opracowania jest niezaprzeczalna, zwłaszcza w kontekście współczesnych wyzwań edukacyjnych. Przedstawione koncepcje nie są jedynie teoretycznymi rozważaniami, lecz są ukierunkowane na praktyczne zastosowanie w codziennej pracy nauczyciela. Konkretne przykłady i wskazówki dotyczące implementacji myślenia komputacyjnego w procesie nauczania dostarczają czytelnikowi konkretnych narzędzi do skutecznego działania.
Opracowanie wyróżnia się również poprzez uwzględnienie aplikacji edukacyjnej - CTApp, co stanowi cenny dodatek do teoretycznych rozważań.

dr hab. Wiesław Kowalski, prof. Akademii WSEI

Jest to aktualna książka obejmująca wszystkie ostatnie osiągnięcia w tej dziedzinie. Kluczowym elementem jest poważna gra CTApp, która jest doskonałym sposobem na poznanie myślenia obliczeniowego za pomocą najnowszego trendu w technologii edukacyjnej, poważnych gier. Poważne gry mogą być niesamowitymi narzędziami do nauczania, uczenia się i edukacji.

Paraskevi Poulogiannopoulou, Phd, European University Cyprus

Wydawnictwo Innovatio Press
Lubelska Akademia WSEI
20-209 Lublin, Projektowa 4
tel.: +48 81 749 17 77, fax: +48 81 749 32 13
www.wsei.lublin.pl
e-mail: wydawnictwo@wsei.lublin.pl

ISBN wersja elektroniczna: 978-83-67550-14-7

Co-funded by the Erasmus+ Programme of the European Union

Publikacja jest współfinansowana przy wsparciu Komisji Europejskiej (numer umowy: 2020-1-PL01-KA201-081924). Niniejsza publikacja odzwierciedla jedynie stanowisko jej autora, Komisja Europejska nie ponosi odpowiedzialności za wykorzystanie zawartych w niej informacji.