Informatyka w Edukacji, XVI
UMK Toruń, 2019

INFORMATYKA – FUNDAMENTY WDRAŻANIA

Maciej M. Sysło
Warszawska Wyższa Szkoła Informatyki
syslo@ii.uni.wroc.pl, http://mmsyslo.pl

Abstract. This is a continuation of my talks at this conference focused on a new informatics (computer science) curriculum just introduced to all levels of education in schools in Poland. In this paper we discuss the most important factors of implementation of the curriculum which are fundamental for the success of all learners.

1. Wprowadzenie

Nowa podstawa programowa kształcenia informatycznego, czyli przedmiotu informatyka, została wprowadzona do szkół podstawowych od 2017 roku zgodnie z [3], a od 2019 roku obejmie również szkoły ponadpodstawowe, a więc licea, technika iszkoły branżowe, zgodnie z [4]. Do szkół ponadpodstawowych przyjdą w 2019 roku absolwenci gimnazjów, po dziewięciu latach nauki w szkole, i uczniowie po ośmioletniej szkoły podstawowej. Ci pierwsi, przez sześć pierwszych lat mieli zajęcia komputerowe i przez trzy lata – informatykę z elementami algorytmiki i programowania. Ci drudzy zaś, po sześciu latach zajęć komputerowych mieli dwa lata informatyki według nowej podstawy programowej. W obu przypadkach liczba godzin informatyki była taka sama, gdyż w trzyletnim gimnazjum przewidziano na informatykę dwie godziny w cyklu kształcenia. Wyraźne różnice są jednak w podstawach programowych informatyki w tych ostatnich latach nauki przed szkołą ponadpodstawową.

Różnorodność przygotowania uczniów w zakresie informatyki na początku szkoły ponadpodstawowej, wynikająca, jak dotychczas, z różnic w realizacji tych samych podstaw programowych na wcześniejszym etapie edukacyjnym – dotyczy to nie tylko informatyki – będzie spowodowana także odmiennymi podstawami programowymi realizowanymi przed przejściem do szkoły ponadpodstawowej. Nie zatrzymujemy się w tej pracy nad tą kwestią, nie negując jednak jej ważności zwłaszcza dla uczniów. Będzie musiał zmagać się z tym nauczyciel, uwzględniając wcześniejsze, zróżnicowane przygotowanie uczniów i próbując wyrównać poziom zajęć oraz ostateczny ich efekt – nabyte przez uczniów wiedzę, umiejętności i kompetencje informatyczne określone w podstawie programowej.

W tej pracy skupiamy uwagę na realizacji podstawy programowej informatyki w całym procesie kształcenia, ze szczególnym uwzględnieniem ostatniego etapu edukacyjnego. Omówienie zakresu podstaw programowych informatyki zawarliśmy we wcześniejszych publikacjach konferencyjnych – na poziomie szkoły podstawowej w [9], [11-14], i na poziomie szkoły ponadpodstawowej w [2].

2. Podstawa programowa i jej fundamenty

Podstawa programowa informatyki jest oparta na kilku fundamentach, które powinny być uwzględnione w jej realizacji i stanowić jednocześnie fundamenty kształcenia informatycznego w polskich szkołach. Wynikają one m.in. z konstrukcji podstawy programowej, miejsca i roli programowania, powiązań informatyki z innymi dziedzinami oraz sugerowanej metodyki kształcenia. Oto te fundament:

- kolejność celów ogólnych kształcenia – kształcenie w zakresie logicznego, abstrakcyjnego i algorytmiczne myślenie zostało umieszczone w podstawie programowej w pierwszym punkcie, jako najważniejsze, przed programowaniem i korzystaniem z aplikacji komputerowych;
- spiralność – w podstawie programowej przyjęto identyczne ogólne cele kształcenia dla wszystkich etapów edukacyjnych sugerując w ten sposób spiralny rozwój uczniów wokół tych samych celów przez wszystkie lata w szkole od pierwszej po ostatnią klasę;
- myślenie komputacyjne – jednym z głównych celów edukacji informatycznej jest rozwój sposobów myślenia angażowanego w formułowanie problemu i przedstawianie jego rozwiązania w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać; jest to sedno podejścia informatycznego do rozwiązywania problemów;
- programowanie – etap kreatywnego rozwiazywania problemów – *learning by doing* – konstrukcjonizm – dialog z komputerem
- informatyka w swoich zastosowaniach – nauczanie przez rozwiązywanie problemów z różnych dziedzin
- metoda projektów – zalecana w podstawach wszystkich przedmiotów, praca w zespołach – informatyk nie pracuje dzisiaj sam

Fundamentalne założenia przyjęte w konstrukcji i realizacji podstawy programowej informatyki zostały dostrzeżone i docenione w międzynarodowym środowisku specjalistów (patrz np. prace grupy roboczej IFIP TC 3, [15]), jak również w wielu krajach, stanowiąc propozycję dla lokalnych rozwiązań.

W kolejnych rozdziałach szczegółowo odnosimy się do powyższych fundamentalnych założeń kształcenia informatycznego w polskich szkołach.

3. Kolejność celów ogólnych

Przypomnijmy, kolejność „Celów kształcenia – wymagań ogólnych", w jakiej występują w podstawie programowej informatyki na wszystkich poziomach kształcenia:

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Ta kolejność odzwierciedla faktyczną rolę komputerów w rozwiazywaniu problemów z ich pomocą. Komputer, języki programowania, przeróżne aplikacje komputerowe to tylko narzędzia, którymi możemy posłużyć w procesie rozwiązywaniu problemów na różnych jego etapach. Najważniejsze w tym to umiejętność doboru narzędzi, ale to zależy od celu, w jakim mają być użyte, a ten cel powinien być określony przed lub w trakcie procesu rozwiązywania problemu.

Warto pamiętać oczywisty ciąg zależności:

komputer wykonuje tylko programy
każdy program jest zapisem jakiegoś algorytmu
trzeba znać więc algorytmy, aby mieć coś do przekazania komputerowi,

Oczywistą więc konkluzją, której zapisem jest kolejność dwóch pierwszych celów ogólnych w podstawie jest: algorytm przed programowaniem; piszemy o tym więcej w rozdz. 6.

Przytoczę jeszcze dobitną wypowiedź na ten temat mojego nieżyjącego już kolegi, Jerzego Szczepkowicz, którą często cytuję:

N.N. nie miał nic do powiedzenia
N.N. nauczył się hiszpańskiego [języka Python]
teraz N.N. nie ma nic do powiedzenia po hiszpańsku [w Pythonie]

Ale ten pogląd płynie nie tylko od informatyków. Fermina Daza, bohaterka powieści *Miłość w czasach zarazy* Gabriela Garcí Márqueza mówi w pewnym momencie: „Języki trzeba znać wtedy, gdy się ma coś do powiedzenia …".

Nauki programowania nie trzeba zaczynać od napisania programu, który drukuje pozdrowienie dla świata: Hello World. Tym bardziej, nawet na początku drogi programowania w jakimkolwiek języku na poziomie szkoły ponadgimnazjalnej, nie zachęcimy i nie umotywujemy ucznia do zgłębiania informatyki, jeśli ma on napisać program dodawania dwóch liczby – „to wykonuję w pamięci", jak odpowiedział jeden z uczniów.

Powyższe uwagi odnoszą się nie tylko do programowania, ale także do posługiwania się aplikacjami komputerowymi, co także powinno być umotywowane głębszym celem ich użycia. Niestety, na przykład w e-podręcznik do klasy IV, w dziale nt. edytora tekstów, znajdujemy takie zadania: „Napisz jakieś zdanie i skopiuj je 5 razy" czy „Napisz jakieś 3 zdania i zmień ich kolejność". To i tak nieco lepiej, bo w pierwszej wersji tego podręcznika było zadanie wielokrotnego wyboru „Który klawisz na klawiaturze jest najdłuższy?" Uważam, że tego typu zadania, ćwiczenia, testy absolutnie nie rozwijają informatycznych kompetencji uczniów, a ponadto, moim zdaniem, uwłaczają inteligencji uczniów, jak i nauczycieli. To tylko pogarsza społeczny odbiór informatyki.

4. Spiralność kształcenia

Spiralność kształcenia była i jest obecna w wielu systemach i metodykach kształcenia w różnych dziedzinach i szkolnych przedmiotach. Najczęściej w tym kontekście powołuje się na prace Jeromego Brunera, zwłaszcza na jego *The proces of Education*, gdzie czytamy: „każde dziecko w każdym wieku może być nauczane każdego przedmiotu w sposób, który jest odpowiedni" (ang. *any subject could be taught to any child at any age in some form that is honest*).

W podstawie programowej przyjęto identyczne ogólne cele kształcenia dla wszystkich etapów edukacyjnych sugerując w ten sposób spiralny rozwój uczniów wokół tych samych celów ogólnych przez wszystkie lata w szkole od pierwszej po ostatnią klasę. Szczegółowe rozpisanie tych celów ogólnych, odpowiednio do etapu kształcenia, znalazło się w „Treściach nauczania – wymaganiach szczegółowych", które są spiralnym rozwinięciem celów ogólnych, odpowiednio do etapu kształcenia. `

Bruner komentuje, że spiralne podejście umożliwia personalizację kształcenia, a więc uwzględnienie przygotowania i potrzeb uczniów, pisząc: „zaczynamy tam, gdzie jest uczeń. I zaczynamy, gdy uczeń jest gotowy, by rozpocząć swoją karierę uczącego się (ang. *One starts somewhere – where the learner **is**. And one starts **whenever** the student arrives to begin his career as a learner.*) Bruner został zagadnięty, czy uważa, że sześciolatka można nauczyć analizy (ang. *calculus*). Odpowiedział, że nie o to chodzi, ale że można przybliżyć dziecku podstawowe dla analizy pojęcie nieskończoności, Richard Feynman zwrócił przy tym uwagę na niezbędne do tego kompetencje nauczycieli: „Jeśli nie potrafisz wyjaśnić czegoś sześciolatkowi, to naprawdę tego nie rozumiesz" (ang. *If you can't explain it to a six year old you don't really understand it*).

W wystąpieniu [12] zilustrowałem na przykładzie porządkowania, jak intuicje, pojęcia i metody wokół tego zagadnienia mogą być spiralnie rozwijane przez 12 lat na zajęciach informatycznych w szkole. Uzasadnieniem dla wyboru tego problemu może być (Donald E. Knuth poświęcił porządkowaniu ponad 700-stronicowy tom swojego

dzieła na temat Sztuki Programowania *The Art. Of Computer Programming. Vol. 3: Sorting and Searching*):

- poroblem porządkowania i wyszukiwania występuje w podstawach programowych informatyki w zdecydowanej większości krajów;
- problem porządkowania jest „nośnikiem" wielu pojęć i metod informatycznych, które mają szerokie zastosowania w wielu obszarach informatyki;
- sam problem porządkowania doczekała się, a często był źródłem bardzo wielu wersji, algorytmów i technik algorytmicznych o różnych własnościach

5. Myślenie komputacyjne

Myśleniu komputacyjnemu były już poświęcone dwa wystąpienia na tej konferencji, w 2014 roku [9] i w 2018 roku [14], w tym drugim uwaga była skupiona na roli myślenia komputacyjnego w rozwijaniu umiejętności programowania.

Obecnie, po wielu dyskusjach i propozycjach (patrz [16] i [17]), czym jest myślenie komputacyjne, przyjmuje się, że obejmuje ono procesy myślowe angażowane w formułowanie problemu i przedstawianie jego rozwiązania w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać.

Na pożytek edukacji szkolnej można przyjąć, że myślenie komputacyjne powinno towarzyszyć rozwiązywaniu problemów z wykorzystaniem podejścia i metod, które wywodzą się z informatyki, ale mają znacznie szersze zastosowania, niemal w każdej dziedzinie. Scharakteryzujmy je ponownie:

- to sposoby rozumowania (ang. *mental tools*), w tym analityczne sposoby myślenia, przydatne zwłaszcza w rozwiązywaniu problemów z różnych dziedzin, z pozoru dalekich od informatyki;
- według J. Wing [16], to nastawienie i umiejętności (ang. *attitiude and skill set*), które każdy może i powinien posiąść i stosować w obszarach swojej działalności i aktywności;
- poszerza ludzkie myślenie i integruje z możliwościami komputerów
- obok stosowania gotowych narzędzi i informacji kształtuje kreatywność w tworzeniu własnych narzędzi i informacji
- przygotowuje do wykorzystywania metod i narzędzi komputerowych oraz informatycznych w różnych dziedzinach
- wzbogaca i poszerza metodologię rozwiązywania problemów z wykorzystaniem komputerów: analityczne narzędzie w realizacji metody projektów
- odgrywa ważną rolę w kształtowaniu umiejętności programowani, ale jednak zgodnie z zasadą: programując – najpierw pomyśl komputacyjnie.

We wcześniejszych publikacjach [5, 9, 14] podaliśmy operacyjną definicję myślenia komputacyjnego, stosowaną przy rozwiązywaniu problemów z pomocą komputerów,

która przypomina etapy rozwiązywania problemów w podejściu algorytmicznym, zalecanym w poprzedniej podstawie programowej informatyki.

Przywołajmy jeszcze niektóre z *mental tools*, wymienione w oryginalnej pracy Wing [16], które powinny zostać uwzględnione w kształceniu informatycznym w szkole ponadpodstawowej ze względu na ich znaczenie w rozwiązywaniu rzeczywistych problemów:

- abstrakcja – wyszukanie w danych i problemie ich istoty i pozostawienie do dalszej analizy i rozważań, a pomijanie cech drugorzędnych;
- redukcja i dekompozycja złożonych problemów na problemy o prostszej strukturze, które są łatwiejsze do rozwiązania, lub których rozwiązanie może być już znane;
- tworzenie reprezentacji danych i ich modelowanie;
- stosowanie heurystyk, podejścia często intuicyjnego, które ma szansę być dobrym rozwiązaniem; myślenie heurystyczne wzmacnia intuicję, prowadzi do nowych pomysłów, jest motorem kreatywności;
- tworzenie przybliżonych rozwiązań (aproksymacji) – znajduje zastosowanie przynajmniej w dwóch sytuacjach: gdy obliczenia są prowadzone na liczbach typu rzeczywistego (czyli przybliżonych) lub na niedokładnych danych oraz w przypadkach, gdy dokładne rozwiązanie problemu nie jest możliwe;
- stosowanie rekurencji, a ogólniej – myślenia rekurencyjnego, jako metody indukcyjnego myślenia i zwięzłej, komputerowej implementacji rozwiązań – to typowy dla informatyki zwięzły sposób formułowania rozwiązań problemów;.

Dwa przykłady ilustrujące pojawianie się i kształtowanie myślenia komputacyjnego na najniższym i na najwyższym poziomie edukacyjnym, zaczerpnięte z [14]. Na rys. 1 są przedstawione zagadki typu Sudoku, rozwiązywane przez uczniów w klasach 1-3. W efekcie, uczniowie rozwijają takie pojęcia, jak:

- abstrakcja – nie ma znaczenia, co układamy: owoce, zwierzęta, liczby, ważna jest zasad obowiązująca ponad tymi obiektami – w wyróżnionych miejscach (wierszach, kolumnach, kwadratach) te obiekty mają być różne;
- dekompozycja – wyróżnienie w powyższej zasadzie miejsc prowadzi do rozkładu zagadki na podzadania poprawnego wypełnienia wyróżnionych fragmentów;
- algorytm, który jest krokowym sposobem wypełnienia planszy według podanej zasady; w kolejnych krokach podejmowane są decyzje w jednym z wybranych fragmentów planszy, przy czym ta kolejność nie jest z góry ustalona, dla różnych plansz może być różna, uczeń znajduje tę kolejność wybierając te miejsca, w których ma pewność, jaki powinien tam znaleźć się element.

Oczywiście, na poziomie klas 1-3 myślenie komputacyjne jest sposobem postępowania uczniów, które staramy się wywołać odpowiednimi zadaniami, bez specjalnej ich świadomości, jak faktycznie postępują. Przyjdzie na to czas po odpowiedniej licz-

bie takich aktywności. Nauczyciel, powinien jednak świadomie dobierać tego typu zadania, by wywołać odpowiedni efekt.

Rysunek 1 Zagadki typu Sudoku dla uczniów klas 1-3 jako ilustracja wystąpienia myślenia komputacyjnego przy ich rozwiązywaniu.

Nieco inne podejście powinno towarzyszyć stosowaniu myślenia komputacyjnego na etapie szkoły ponadpodstawowej, zwłaszcza w kształceniu informatycznym w zakresie podstawowym. W tabeli 1 przytaczamy przykład z wystąpienia w zeszłym roku [14], ilustrujący ogólne podejście do wykonania projektu nakreślone w podręczniku [1], który równie dobrze mógłby być wykonany na lekcjach języka polskiego. Jednak jako projekt w ramach kształcenia informatycznego, zidentyfikowano w nim elementy myślenia komputacyjnego, których znaczenie daleko wykracza poza ten projekt i poza typowy projekt informatyczny i może znaleźć zastosowanie w podobnych sytuacjach. Dodatkowo, w porównaniu z zajęciami nt. Sudoku w klasach 1-3, uczniowie powinni

świadomie identyfikować elementy myślenia komputacyjnego, którymi się będą posługiwać w trakcie jego realizacji..

Tabela 1 Rozwiązanie jednego zadania

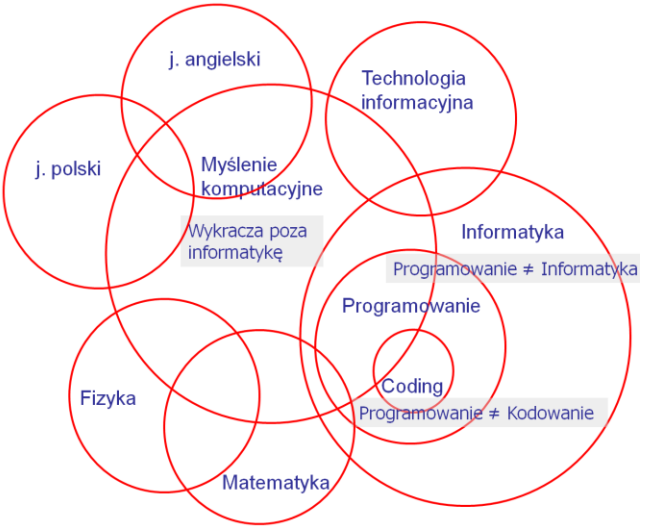
L.p.	Kolejne kroki w procesie rozwiązania sytuacji problemowej
1.	Sytuacja: Wypowiedzi Umberta Eco na temat książek, m.in.: „Jeśli ktoś myśli, że książki znikną, to się myli" Zadanie: Raport z projektu polegającego na dyskusji między dwoma grupami uczniów o przeciwnych poglądach.
2.	Dane: teksty drukowane i elektroniczne związane z tematem; selekcja/wybór fragmentów ze źródeł – **abstrakcja**
3.	**Zasada** w danych: fragmenty z tekstów dotyczą losu książek, w tym są teksty Umberta Eco
4.	**Reprezentacja danych**: szablon relacji z dyskusji z przeciwnymi argumentami – odpowiedni układ tabeli: osoby, poglądy, argumenty
5.	**Dekompozycja**: najpierw wydzielenie w grupach argumentów „za" i „przeciw", a następnie ich uporządkowanie i ostateczna postać
6.	**Algorytm**: metoda/tryb postępowania, organizacja dyskusji, uporządkowana relacja z dyskusji
7.	Komputer, **program**: struktura (automatyzacja) realizacji projektu, „programowanie" edytora – style tekstu, organizacja tekstu
8.	**Modyfikacje**: • uwzględnienie argumentów innych osób, poza Umbertem Eco i realizatorami projektu

Na rys, 2 ilustrujemy relacje między różnymi obszarami edukacji przedmiotowej oraz wykorzystania narzędzi i podejść informatycznych.

6. Programowanie

Programowanie stało się w ostatnich latach „ciepłym ciasteczkiem" (ang. *hot cake*), które, często nazywane kodowaniem, zwłaszcza na najniższych etapach edukacyjnych, jest „uprawiane" przez rzesz najmłodszych nawet uczniów, często bez związku z informatyką i jej podstawą programową, bez specjalnego umieszczenia tej aktywności uczniów w szerszym kontekście edukacyjnym kształcenia informatycznego, przewidzianego w podstawie programowej na 12 lat pobytu w szkole. Co więcej, przekonuje się, że umiejętność programowania będzie potrzebna każdemu obywatelowi. Nie-

zupełnie, bo nie każdy obywatel będzie programistą, ale kształcenie umiejętności programowania jednocześnie rozwija logiczne myślenie, kreatywność, myślenie komputacyjne, systematyczne podejście do rozwiązywania problemów.



Rysunek 2 Relacje między dziedzinami i pojęciami występującymi w tym artykule.

Skupiamy tutaj uwagę na programowaniu w takim sensie, w jakim występuje w podstawie programowej przedmiotu informatyka. Najważniejsze wnioski stąd płynące można krótko ująć:

- programowanie to etap rozwiązywania problemu mający na celu komputerową implementację rozwiązania rozważanego problemy, sytuacji problemowej, pewnego pomysłu lub idei;
- podobnie, jak tworzenie rozwiązania, także ten etap wiąże się kreatywnością uczącego się, gdyż zaprogramowanie rozwiązania nie jest czynnością automatyczną, a komputer jest dość wymagającym partnerem;
- programowanie to uczenie się przez działanie, wykonywanie (ang. *learning by doing*), polegające na tworzeniu (konstruowaniu) wytworów (podejście konstrukcjonistyczne) – programów komputerowych; to działanie powinno aktywizować ucznia, nie być odtwórcze, zbyt elementarne, bo może zniechęcić;

Podobnie jak myślenie komputacyjne powinno towarzyszyć całemu procesowi rozwiązywania sytuacji problemowej, od jej sformułowania po realizację rozwiązania w postaci programu, tak programowaniem można określić cały proces rozwiązywania problemu – finalną postać rozwiązania problemu w postaci programu należy „przewidywać" od samego początku rozwiązywania problemu.

Wybór języka programowania w realizacji podstawy programowej ma w pewnym sensie drugorzędne znaczenie (patrz poniższy dekalog). Przyjmuje się, że na najniższym, etapie kształcenia język programowania nie powinien stwarzać uczniom zbyt wielu problemów syntaktycznych (składniowych). Odpowiednim wyborem jest tutaj język popularnie zwany wizualno-blokowym, jak Scratch czy Blockly. Języki tego typu nie umożliwiają jednak programowania typowych i bardziej zaawansowanych algorytmów, dlatego w pewnym momencie powinien pojawić się na zajęciach informatyki tekstowy język programowania, jak C++ czy Python. To przejście między językami powinno mieć charakter spiralnego rozwoju umiejętności programowania, być naturalne dla uczniów, gdyż język gra tutaj drugorzędną rolę. Właściwym potraktowaniem języka programowania X w kształceniu informatycznym powinno być przyjęcie podejścia, które krótko można określić jako: „Programowanie i język X", a nie „Programowanie w języku X.

Rolę programowania w kształceniu informatycznym szeroko komentujemy w zeszłorocznym wystąpieniu [14]. Tutaj podsumujmy dyskusję tam i tutaj, jak również w innych miejscach dziesięcioma „przykazaniami" – stwierdzeniami cech języków programowania z punktu widzenia ich roli i wykorzystania w edukacji.

Dekalog języka programowania:

1. dobry język odzwierciedla ważne pojęcia, powinien być nośnikiem, a nie obiektem nauczania; jest narzędziem, a nie powinien być celem;
2. programując najpierw pomyśl komputacyjnie;
3. celem programowania jest abstrakcja, a nie programy;
4. to sposób myślenia – różne języki, to różne metodyki programowania, różne obszary aplikacji, różni „czytelnicy" – ludzie i maszyny;
5. poznawaj, gdy go potrzebujesz, używaj ciągle;
6. żaden szczególny – jest ich ponad 3000;
7. program – to komunikat, czytelny i zrozumiały przekaz dla innych osób, nie tylko dla maszyn – uczeń powinien umieć czytać programy;
8. są języki wizualne i tekstowe, jednak nie ma ani programowania wizualnego ani tekstowego – języki wizualne minimalizują techniczne wymagania składni (syntaktyki);
9. niemal każda aplikacja może być programowana;
10. programowanie to cały proces rozwiązania problemu.

Skomentujmy „przykazanie" nr 7. Złym zadaniem jest polecenie: „napisz program", gdyż na każde takie polecenie są dziesiątki, setki, tysiące gotowych odpowiedzi w Internecie, w każdym języku programowania. Program powinien być efektem, częścią rozwiązania problemu, który jest przedstawiony uczniowi. Jednak, napisano do-

tychczas programy dla każdego algorytmy w każdy języku. Z tego powodu uczeń powinien również nauczyć się „czytać" gotowe, nie swoje programy, napisane nawet w języku, którego się nie uczył. Może to dotyczyć, najlepiej, algorytmu, którego nie zna, by go rozpoznał z tego programu i umiał opisać. Obecnie to ważna umiejętność programistów, ale nie tylko, także tych osób, które chciałyby poznać, jak coś to działa na podstawie programu.

Jeszcze jeden komentarz związany z programowaniem. Narosło wiele nieporozumień wokół programowania dynamicznego (szerzej piszemy o tym w [?]). Programowanie dynamiczne to metoda optymalizacji, technika algorytmiczna, zaproponowana przez Richarda Bellmana w latach 1950' i jako taka nie ma nic wspólnego z programowaniem, poza tym, że rozwiązanie problemu otrzymane tą metodą można zaprogramować dla komputera.

7. Zastosowania informatyki

Zastosowania informatyki w szkole mogą przyjąć przynajmniej dwie formy. Sytuacje problemowe z różnych dziedzin (przedmiotów) mogą być tak dobierane (przez nauczyciela), by w trakcie ich rozwiązywania pojawiały się zamierzone przez tegoż nauczyciela pojęcia i metody informatyczne. Przykład takiego podejścia został zaprezentowany w rozdz. 5. Innym podejściem jest integracja kształcenia informatycznego z innymi przedmiotami i rozwój wybranych pojęć i metod informatycznych w ramach przedmiotów nieinformatycznych. Przykładem takie zagadnienia mogłyby być przybliżone obliczenia wybranych wielkości na zajęciach matematycznych, lub opracowanie z użyciem arkusza kalkulacyjnego wyników doświadczeń na lekcjach fizyki. Niestety, nie można liczyć na takie wsparcie ze strony przedmiotów, które mogłoby się wydawać, że są najbliżej informatyki – podstawy programowe tych przedmiotów nie zawierają żądnego odniesienia do informatyki czy wykorzystania komputerów.

W tej sytuacji można jedynie zalecić, co następuje:

- kształcenie informatyczne przez rozwiązywanie problemów z różnych dziedzin
- wykorzystanie źródeł sytuacji problemowych z innych przedmiotów, dziedzin, obszarów – ich wybór powinna dyktować nauczycielowi informatyczna „zawartość" sytuacji:;
- wykorzystywane zastosowania powinny być polem dla aktywacji i mobilizacji myślenia komputacyjnego na potrzeby rozwiązania problemów z różnych dziedzin;
- przykłady zastosowań powinny być również ilustracją użyteczności informatyki;
- w szczególności, takie przykłady z innych przedmiotów mogą wnieść dodatkowe elementy kształcenia i użyteczności informatyki w tych przedmiotach;
- integracja między przedmiotowa z inicjatywy informatyki powinna prowadzić do szerszej współpracy, również na szkolne potrzeby;

- zastosowania informatyki mogą przyczynić do popularyzacji tej dziedziny wśród uczniów, którzy wybrali ten przedmiot w zakresie podstawowym;

Można wymienić bardzo wiele problemów i sytuacji problemowych, na bazie których mogą być budowane rozważania informatyczne. Można do nich zaliczyć:

- rozpoznawanie wzorca, czyli fragmentów w większych obiektach, np. części DNA (odpowiedzialnej za chorobę) w całym DNA; fragmentów tekstów (w wykrywaniu plagiatów);
- opisywanie konstrukcji fraktalnych, występujących na przykład w różnych obiektach przyrody, na ogół za pomocą zależności rekurencyjnych;
- doskonałość w przyrodzie i w sztuce – liczby Fibonacciego a doskonała proporcja;

8. Metoda projektów

Praca uczniów metodą projektów jest zalecana w podstawach programowych wszystkich przedmiotów. Praca tą metodą:

- służy realizacji zadań, a jednocześnie uczy pracy tą metodą;
- umożliwia, formalizuje i usprawnia pracę w zespołach;
- może służyć realizacji zadań pochodzących z różnych dziedzin;
- umożliwia większą indywidualizację kształcenia – różne grupy uczniów w klasie mogą pracować nad różnymi tematami projektów; uczniowie mogą przyjmować różne role w realizacji projektów; projekty mogą być realizowane także przez pojedynczych uczniów;
- w przypadku informatyki – kształcone są umiejętności, które są niezbędne w zawodzie informatyka – dzisiaj praca informatyka to praca w zespole, informatyk nie pracuje w pojedynkę;

Podręcznik do informatyki w liceum [1] z 2012 roku był w pełni propozycją wykonywania projektów przez uczniów, z tego powodu zbędne były w tym podręczniku zadania i ćwiczenia do wykonania przez uczniów, poza projektami. Główne cechy zajęć z takim podręcznikiem to:

- faktycznie nauka nie polega na „przerabianiu" podręcznika;
- w miejsce lekcji uczniowie pracują metodą projektów i zajęcia w ten sposób prowadzone przeradzają się w odwrócone uczenie się, które przebiega nie tylko w klasie;
- powyższe jest propozycją zmiany kultury uczenia się;
- kształtowane jest myślenie komputacyjne na rozwiązywaniu sytuacji problemowych pochodzących z różnych dziedzin – realizacja jednego z takich projektów jest przedstawiona w tabeli 1.

Dla pełnego wykorzystania potencjału metody projektów niezbędny jest serwis internetowy, złożony m.in. z:

- repozytorium szczegółowych propozycji projektów dla uczniów i nauczycieli;
- strefy realizacji projektów przez grupy uczniów, nadzorowane przez nauczyciela;
- repozytorium raportów z wykonanych projektów wraz z ich oceną przez wykonawców i nauczycieli.

Trwają prace nad stworzeniem takiego systemu zarzadzania projektami.

Literatura

1. Gurbiel E., Hard-Olejniczak G., Kołczyk E., Krupicka H., Sysło M.M., *Informatyka to podstawa*, WSiP, Warszawa 2012.
2. Kwiatkowska A.B., Podstawa programowa informatyki w szkole ponadpodstawowej – algorytmika, programowanie i myślenie komputacyjne, w: *Myśl komputacyjnie!*, Materiały konferencji „Informatyka w Edukacji, XV", UMK Toruń 2018, 15-24.
3. Rozporządzenie Ministra Edukacji Narodowej z dnia 14 lutego 2017 r. w sprawie podstawy programowej wychowania przedszkolnego oraz podstawy programowej kształcenia ogólnego dla szkoły podstawowej, w tym dla uczniów z niepełnosprawnością intelektualną w stopniu umiarkowanym lub znacznym, kształcenia ogólnego dla branżowej szkoły I stopnia, kształcenia ogólnego dla szkoły specjalnej przysposabiającej do pracy oraz kształcenia ogólnego dla szkoły policealnej, http://www.dziennikustaw.gov.pl/DU/2017/356
4. Rozporządzenie Ministra Edukacji Narodowej z dnia 30 stycznia 2018 r. w sprawie podstawy programowej kształcenia ogólnego dla liceum ogólnokształcącego, technikum oraz branżowej szkoły II stopnia, http://dziennikustaw.gov.pl/du/2017/356/1.
5. Sysło M.M., Myślenie komputacyjne: informatyka dla wszystkich, Materiały Konferencji „Informatyka w Edukacji, VIII", UMK Toruń, 2011.
6. Sysło M.M., Kwiatkowska A.B., Myśl logarytmicznie!, *Delta* nr 12/2014.
7. Sysło M.M., Kwiatkowska A.B., Learning Mathematics Supported by Computational Thinking, w: Materiały *Constructionism and Creativity*, Wiedeń 2014.
8. Sysło M.M., Kwiatkowska A.B., Introducing Students to Recursion: a Multi-Facet and Multi-Tool Approach, w: Materiały *ISSEP 2014*, Istanbul (Turkey), 2014.
9. Sysło M.M., Myślenie komputacyjne. Nowe spojrzenie na kompetencje informatyczne, w: Materiały „Informatyka w Edukacji, XI", UMK Toruń 2014, 15-32.
10. Sysło M.M., Kwiatkowska A.B.: Introducing a new computer science curriculum for all school levels in Poland. w: Brodnik, A., Vahrenhold, J. (red.), *Informatics in*

Schools. Curricula, Competences, and Competitions. ISSEP 2015, LNCS 9378, Springer, 141-154 (2015).

11. Sysło M.M., Kwiatkowska A.B., Informatyka dla najmłodszych. Pojęcia, algorytmy, programy, Materiały Konferencji „Informatyka w Edukacji, XII", UMK Toruń, 2015, str. 15-40
12. Sysło M.M., Wprowadzając… porządek, w: Materiały konferencji „Informatyka w Edukacji, XIII", UMK, Toruń 2016; dostępne na: http://iwe.mat.umk.pl/archiwum/iwe2016/?q=node/20
13. Syslo M.M., Rozwój pojęć informatycznych od pierwszej klasy, Materiały Konferencji „Informatyka w Edukacji, XIV", UMK Toruń, 2017, str. 29-40.
14. Syslo M.M., Jak myśleć komputacyjnie, Materiały Konferencji „Informatyka w Edukacji, XV", UMK Toruń, 2018, str. 3-14.
15. Webb M., Bell T., Davis N., Katz Y. J., Reynolds N., Chambers D. P., Sysło M. M., Fluck A., Cox M., Angelivalanides C., Malynsmith J., Voogt J., Zagami J., Micheuz P., Chtouki Y. & Mori N. (2017) *Computer science in the school curriculum: Issues and challenges*. In: Tatnall A. & Webb M. (eds.) Tomorrow's learning: Involving everyone. Learning with and about technologies and computing. WCCE 2017. IFIP Advances in Information and Communication Technology 515. Springer, Cham: 421–431
16. Wing J., Computational thinking, *Comm. ACM* 49(3), 2006, 33–35
17. Wing J., Computational thinking benefits society, http://socialissues.cs.toronto.edu/index.html%3Fp=279.html