Informatyka w Edukacji, XV
UMK Toruń, 2018

JAK MYŚLEĆ KOMPUTACYJNIE

Maciej M. Sysło
Wydział Matematyki i Informatyki, UMK w Toruniu
syslo@mat.umk.pl; syslo@ii.uni.wroc.pl, http://mmsyslo.pl

Abstract. This is a continuation of my talks at this conference focused on computational thinking (CT). First time I addressed this topic at the ISSEP 2008 in Toruń, however I introduced CT here during my talk in 2011 and in the years which followed I discussed the meaning and the use of CT in the new informatics curriculum. This year I am going to be more specific and go into details of CT since it seems that CT is used mostly as a new term for what existed before and no changes and improvements can be really observed. We also demonstrate relations between informatics (computer science), information technology, programming, and computational thinking.

Celem programowania jest abstrakcja,
a nie programy

The purpose of programming is abstraction
not programs

1. Wprowadzenie

Pojęcie myślenie komputacyjne (ang. *computational thinking* – **CT**) zrobiło w ostatniej dekadzie zawrotną karierę. Pojawia się niema w każdym dokumencie, pracy, prezentacji dotyczących zwłaszcza edukacji informatycznej i jej powiązań z innymi edukacjami. Pojawia się i nadal nie wiadomo, o co chodzi: jak myśleć komputacyjnie, jaki to ma związek z informatyką i innymi przedmiotami, co wnosi nowego. Można odnieść wrażenie, że to inna nazwa tego, co było i jest kontynuowane bez żadnych zmian. A jednak to "nowa jakość" w kształceniu, a zwłaszcza w dalszych losach uczniów. Postaramy się ponownie przybliżyć to pojęcie, odnieść do innych, z którymi jest w relacji, i bardziej szczegółowo zilustrować, w jakim kontekście i zakresie pojawia się w kształceniu. Ogólne rozważania zostaną zilustrowane dwoma przykładami – jednym z edukacji informatycznej w nauczaniu wczesnoszkolnym, a drugim – z edukacji polonistycznej w szkole ponadpodstawowej.

2. Co to jest myślenie komputacyjne, *again*

Zacznijmy od historii. Już w 1980 roku, a później w 1996, o myśleniu komputacyjnym pisał Seymour Papert, znacznie wyprzedzając swoimi ideami konstruktywistycznymi możliwości technologii, patrz [7-8]. Jeannette Wing wprowadzając myślenie komputacyjne w 2006 roku [19], określiła tym terminem „użyteczne postawy i umiejętności, jakie każdy, nie tylko informatyk, powinien starać się wykształcić i stosować" (ang. *a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use*). Jak sama przyznaje w [20], myślenie komputacyjne uznała wtedy za skrót od „myśleć jak informatyk" (ang. *thinking like a computer scientist*) i dopiero pod wpływem szerokie dyskusji na temat znaczenia tego terminu przyjęła, że:

> myślenie komputacyjne to procesy myślowe angażowane w formułowanie problemu i przedstawianie jego rozwiązań w taki sposób, aby komputer[1] – człowiek lub maszyna – mógł skutecznie wykonać.

Uzupełnijmy, te procesy myślowe towarzyszą cały czas procesowi formułowania i rozwiazywania problemu, nie tylko którejś z jego części, np. programowaniu.

Wing uznała w [19], że myślenie komputacyjne stanowi naturalne poszerzenie[2] kompetencji określanych jako 3R (Reading, wRiting, aRithmethic), o umiejętności stosowania metod pochodzących z informatyki i analitycznego myślenia przy rozwiązywaniu problemów pochodzących z różnych dziedzin. Dodała przy tym swoją wizję: tak jak prasa drukarska przyczyniła się do rozprzestrzeniania się tych 3R, tak komputery przyczynią się do rozpowszechniania się myślenia komputacyjnego.

Chociaż myślenie komputacyjne ma swoje korzenie w informatyce, uwzględniając chociażby profesję Wing, obecnie *Corporate Vice President, Microsoft Research*, pisze jednak w [20], że celem jej artykułu [19] było wywołanie pozytywnego myślenia w społeczności informatyków i zamiast narzekania na malejące zainteresowanie ich dziedziną (obserwowane w USA w latach 2000-2010), wykrzyczenie w świat o radości z zajmowania się informatyką (*joy of computing*) i ważności informatyki (*importance of computing*). Po latach trendy zostały odwrócone, co po części można przypisać jej inicjatywie i działalności (m.in. w ramach NSF).

---

[1] *computer* w języku angielskim to (na podstawie Webster's New World Dictionary, 1969): *a device used for computing* ale także *a person who computes*.

[2] Dopisywanie kolejnych umiejętności do 3R należy traktować z umiarem. Na przełomie wieków XX/XXI, 3R uzupełniono o TI, czyli zastosowania informatyki, głównie posługiwanie się gotowymi aplikacjami – przeżywaliśmy to w Polsce, chociaż informatyka nigdy nie zniknęła z podstaw programowych. Obecnie słychać o dopisaniu programowania do 3R, co jest znaczącym zawężeniem właściwego znaczenia myślenia komputacyjnego.

W pracy [19] Wing wymienia spectrum metod i podejść do modelowania i rozwiązywania problemów zaliczając je do myślenia komputacyjnego, m.in.:

- **rozpoznawanie wzorów i wzorców** – wzorce mogą umożliwiać modelowanie, algorytmizację i analizę a później – automatyzację obliczeń;
- **abstrakcja** umożliwiająca modelowanie najważniejszych cech badanej sytuacji problemowej po zaniedbaniu cech drugorzędnych;
- **redukcja i dekompozycja** złożonego problemu na mniejsze podproblemy, których rozwiązania są znane lub są łatwiejsze do rozwiązania;
- **aproksymacja**, czyli znajdowanie rozwiązania przybliżonego, gdy dokładne rozwiązanie jest poza zasięgiem nawet komputerów ze względu na niedokładny charakter danych lub złożoność problemu;
- **rekurencja** jako metoda indukcyjnego myślenia i zwięzłej, komputerowej implementacji rozwiązań – to typowy dla informatyki zwięzły sposób formułowania rozwiązań problemów;
- znajdowanie rozwiązań **metodami heurystycznymi**, czyli mało precyzyjnymi, ale bazującymi na trafnej intuicji.

Innym podejściem do uściślenia znaczenia myślenia komputacyjnego było zaproponowanie przez Stowarzyszenie Nauczycieli Informatyki w USA (CSTA, patrz [2]) operacyjnej definicji, której celem jest uporządkowanie aktywności uczniów i działań nauczycieli w trakcie rozwiązywania problemów z różnych dziedzin z wykorzystaniem metod rozumowania, zaliczanych do myślenia komputacyjnego. Definicja operacyjna przypomina kolejne etapy algorytmicznego rozwiazywania problemów i może stwarzać wrażenie, że myślenie komputacyjne jest inną nazwą podejścia algorytmicznego, musi więc być stosowana z pełnym zrozumieniem, czym jest myślenie komputacyjne.

3. Cechy myślenia komputacyjnego – cd

W tym rozdziale chcemy zwrócić uwagę na ważność abstrakcji, która pojawia się jako jedno z *mental tools* – narzędzi myślenia komputacyjnego, jako metoda i sposób rozumowania w trakcie rozwiązywania problemów. Wing przywołuje w tym kontekście opinię dwóch znanych informatyków Aho i Ullmana [1], że informatyka zajmuje się automatyzacją abstrakcji (ang. *automation of abstraction*; *mechanization of abstraction*). Jak to ujął Peter Denning [3] „Podstawowym pytaniem tej dziedziny [informatyki] jest: Co może być (efektywnie) zautomatyzowane?"

Skrótem myślowym jest jednak stwierdzenie, że ponieważ każdy komputerowy program jest abstrakcją sytuacji, do rozwiązania której może być użyty, to programowanie jest faktycznie esencją myślenia komputacyjnego. Zdecydowanie NIE, jeśli ten tok rozumowania prowadzi do zrównania programowania z informatyką

i dalej – z myśleniem komputacyjnym; rozdz. 5. Faktycznie (rozdz. 6), na drodze do programu – tego abstrakcyjnego efektu myślenia komputacyjnego, uczeń cały czas stosuje abstrakcyjne myślenie: analizując sytuację problemową, wydzielając z niej istotne dane, wybierając odpowiedni abstrakcyjny model reprezentowania danych w dalszych rozważaniach, dobierając algorytm – abstrakcyjny przepis obliczeń i wreszcie tworzą program do ewentualnego zautomatyzowania obliczeń.

Wróćmy na chwilę do przełomowej pracy Wing [19], by krótko odnieść się do terminów, które pojawiają się w tej pracy, jak i w różnych dyskusjach wokół informatyki i programowania. Pisze ona – co można uznać także za wykładnię znaczenia „myśleć jak informatyk":

> Myśleć jak informatyk to coś więcej niż umieć programować komputer; to wymaga myślenia na wielu poziomach abstrakcji.
>
> [*Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.*]

Przypomnijmy, abstrakcja umożliwiająca wydobycie i określenie najważniejszych cech rozważanej sytuacji problemowej po zaniedbaniu cech drugorzędnych. Stanowi często pierwszy krok w rozwiazywaniu problemu – abstrahując od mało istotnych cech sytuacji problemowej tworzymy jej model reprezentujący najistotniejsze cechy.

W parze z abstrakcja w rozwiązywaniu problemów postępuje redukcja i dekompozycja (rozkład) złożonego problemu na mniejsze podproblemy, których rozwiązania są znane lub są łatwiejsze do otrzymania.

Myślenie komputacyjne obejmuje całą gamę szczególnych metod takich, jak myślenie logarytmiczne [10], myślenie redukcyjne [11], czy myślenie rekurencyjne [12]. Jednym z najważniejszych podejść w myśleniu komputacyjnym, zwłaszcza na wczesnym etapie kształcenia informatycznego, jest rozumowanie heurystyczne, które, w sytuacji niepełnej lub braku wiedzy na dany temat, umożliwia uczniom odkrywanie i tworzenie własnych rozwiązań problemów bazując na intuicji.

Myślenie komputacyjne daleko wykracza poza stosowanie gotowych narzędzi czy aplikacji informatycznych w rozwiązywaniu problemów z różnych dziedzin, jego istotą jest bowiem tworzenie nowych narzędzi i nowych informacji. Gotowe rozwiązania bywają jednak przydatne w sytuacjach typowych lub w rutynowym postępowaniu, które często stanowią element bardziej złożonych metod rozwiazywania.

Myślenia komputacyjnego nie należy uznawać za wyróżnioną w szczególny sposób charakterystykę informatyki (*computer science*) [4] – jest to raczej zbiór praktyk, których źródło leży w informatyce, ale które są przeznaczone do stosowania w różnych dziedzinach aktywności człowieka, szeroko poza informatyką. Podstawowa wiedza informatyczna pomaga jednak systematycznie, poprawnie i efektywnie przetwarzać informacje i rozwiązywać problemy. Dlatego miejscem pierw-

szych kontaktów uczniów z myśleniem komputacyjnym są zajęcia z informatyki, które powinny kłaść podwaliny pod myślenie komputacyjne z jednoczesnym wytyczaniem obszarów poza informatyką, w których może znaleźć zastosowanie.

Kształtowanie myślenia komputacyjnego niezależnie od komputerów u każdego ucznia może być wkładem do edukacji ogólnej. Umiejętność myślenia jak ekonomista, fizyk czy artysta powinna obejmować również uzasadnione posłużenie się modelami obliczeniowymi i obliczeniami w rozwiązaniach problemów z tych dziedzin, a także formułowania nowych wyzwań, które mogą być owocnie rozważane. Z punktu widzenia ekonomii, nie tylko potrzeba coraz większej liczby informatyków, aby pozostać konkurencyjnym w świecie napędzanym technologią, ale należy również przygotować w zakresie informatyki specjalistów innych dziedzin dla wsparcie ich innowacyjności i rozwoju. Głęboko społeczne znaczenie ma kształcenie postaw twórczych w przeciwieństwie do prostej konsumpcji produktów technologicznych. W sferze kultury zaś, przygotowanie informatyczne umożliwia obywatelom przewodzenie transformacjom kulturowym, a nie tylko uleganie zmianom wywieranym przez rozwój samej technologii.

Nie wszyscy muszą myśleć jak informatyk, ale elementy myślenia wywodzące się z informatyki mogą znacznie wspomóc i wzbogacić arsenał metod innych dziedzin. Myślenie komputacyjne jest w tym istotnym elementem, ponieważ umożliwia rozpoznawanie aspektów obliczeniowych w otaczającym nas świecie i stosowaniu narzędzi i metod informatyki w celu rozumienia i rozwiązywania problemów związanych zarówno z naturalnymi, jak i sztucznymi systemami i procesami. Związek informatyki z myśleniem komputacyjnym jest więc nierozerwalny.

4. Programując – najpierw pomyśl komputacyjnie

W ostatnich latach obserwujemy boom zainteresowania programowaniem wśród najmłodszych, często bez związku z innymi umiejętnościami. W podstawie programowej, programowanie jest elementem kształcenia informatycznego, ważne więc jest powiązanie tej aktywności z rozwijaniem myślenia komputacyjnego.

Jak wspomnieliśmy, komputerowy program jest tylko jednym z wielu abstrakcyjnych efektów myślenia komputacyjnego, które jednak nie jest ściśle zależne od umiejętności programowania – kładzie bowiem raczej nacisk na procesy myślowe, prowadzące do formułowania problemów i rozwiązań, ale niekoniecznie by je realizować na komputerze, chociaż rozwiązania są na ogół reprezentowane w formie dogodnej do automatycznego otrzymania wyników za pomocą komputera. Niemniej jednak umiejętność implementowania wyników myślenia komputacyjnego poprzez programowanie umożliwia uczniom ocenę sposobów i efektów ich myślenia i zajęcia z informatyki są najlepszym ku temu miejscem. Z drugiej jednak strony, myślenie komputacyjne a w jego ramach – projektowanie rozwiązań i ich formułowanie

w postaci dogodnej dla komputera są niezbędnym przygotowaniem do programowania. W wielu dyscyplinach podejmowane decyzje bazują na obliczeniach komputerowych, a także pisanie programów jest użyteczną umiejętnością radzenia sobie z różnymi aspektami codziennego życia.

Podsumowaniem dyskusji dotyczącej roli programowania w rozwijaniu myślenia komputacyjnego może być przyjęcie wykładni, że programowanie jest nazwą całego procesu rozwiązywania problemu. Można pójść jeszcze dalej. Kształcenie informatyczne w nowej podstawie programowej jest nie tylko propozycją włączenia programowania do zajęć szkolnych, ale ma ambicje znacznie szersze – skierowanie zainteresowania uczniów, nauczycieli i społeczeństwa na te kompetencje związane z umiejętnością programowania komputerów, które mogą być przydatne, by uczestniczyć w zaprogramowaniu… swojej przyszłości [17].

Wspomnijmy jeszcze, że programowanie ma przynajmniej dwa znaczenia. Za pierwszą programistkę uznaje się Adę Augustę (1815-1852), córkę Byrona, która podała „program" na obliczanie liczb Bernoulliego w opisie analitycznej maszyny Babbage'a. Na próżno jednak szukać terminów 'program' czy 'programowanie' w jej notatkach. Jednak jako pierwsza podała przepis dla komputera, chociaż ani to nie był program, ani to nie był komputer w dzisiejszym sensie. Pierwsze komputery powstawały w okresie II Wojny Światowej. Nie miały one jednak wielkiego wpływu na losy wojny, z wyjątkiem Colossusa, pracującego na potrzeby łamania niemieckich szyfrogramów. Hitler – szczęśliwie dla nas wszystkich – nie wykorzystał żadnego z dość zaawansowanych komputerów Konrada Zuse.

Czasy ostatniej wojny były natomiast impulsem do pojawienia się terminu programowanie na określanie planowania działań wojennych i nie tylko, z wykorzystaniem zaawansowanych metod matematycznych i pochodzących z innych dziedzin. Narodziło się m.in. programowanie dynamiczne, o którym uczymy na informatyce, a które wcale nie jest uzależnione od możliwości programowania komputerów.

Jeśli dzisiaj docierają do uczniów wezwania „Zaprogramuj swoją przyszłość", to byłoby dobrze, gdyby nie ograniczały się głównie do nauki programowania komputerów, a tak niestety jest w wielu przypadkach, ale oznaczało uwzględnienie wielu aspektów i metod tak, aby ich przyszłość była w przyjętym przez nich sensie „rozwiązaniem optymalnym". Rozwijanie umiejętności programowania powinno być umieszczone w kontekście rozwoju kompetencji myślenia komputacyjnego, w szczególności: logicznego myślenia, kreatywności w poszukiwaniu rozwiązań, myślenia heurystycznego w znaczeniu dobrze umotywowanego myślenia 'na chłopski rozum', poszukiwania innowacyjnych rozwiązań, algorytmicznego myślenia w znaczeniu dobrze uporządkowanych kroków postępowania. W tym również, posługiwanie się 'językiem' komunikacji z komputerem – może to być język programowania, by nająć go do współpracy w rozwiązywaniu problemów.

Nowe podejście do kształcenia informatycznego otwiera również nowy rozdział dla posługiwania się aplikacjami komputerowymi. Korzystanie z oferowanych dzisiaj aplikacji biurowych jest w pewnym sensie również ich „programowaniem". Edytor tekstu służy do „programowania" tekstu, któremu możemy nadawać przeróżną formę, a najważniejsze – pracować nad jego treścią, a wypełniony arkusz kalkulacyjny jest niczym innym, jak „programem" zapisanych w nim obliczeń. Największą rewolucję czeka prezentacja – projekty w języku Scratch to prezentacje, które mogą oddać nieograniczoną wyobraźnię ucznia, stosującego animacje, interakcje, reakcje na zdarzenia i wszelkie media.

O językach programowania i środowiskach programistycznych, ich cechach i wyborze dla poszczególnych etapów edukacyjnych piszemy w innym miejscu [15]. Można zaobserwować olbrzymią ilość propozycji zajęć dla uczniów, głównie dla najmłodszych, dotyczących programowania, na ogół nazywanego kodowaniem. Można mieć obawy, czy te zajęcia nie sprowadzają się głównie do „pisania programów". W tym miejscu warto przytoczyć słowa obawy, wyrażonej ostatnio przez Mitchela Resnicka, twórcę Scratcha, ucznia Seymoura Paperta: *Today, millions of children are participating in learn-to-code initiatives, but Papert's dream remains unfulfilled. Papert saw* ***programming not as a set of technical skills but as a new form of fluency*** *– a new way for all children to explore, experiment, and express themselves.*

Język programowania ma wiele cech jakiegokolwiek języka, czyli medium porozumiewania się i poznania, co świetnie oddaje powiedzenie Ludwiga Wittgensteina:

Granice naszego języka są granicami naszego poznania (świata)

Dzisiaj, w kontekście technologii, to powiedzenie można sparafrazować:

Granice naszego języka programowania technologii
są granicami naszego poznania świata za pomocą technologii.

Zwróćmy jeszcze uwagę, że z każdym językiem jest związana sfera „kultury", obejmująca różne aspekty językowe, jak poprawność i styl, znaczenie, komunikację, aspekty społeczne. Odnosi się to nie tylko do języka mówionego, ale także do języków programowania. Kształcąc umiejętności programowania komputerów nie należy o tym zapominać, pomóc w tym może umieszczenie programowania w kontekście myślenia komputacyjnego.

5. Relacje miedzy obszarami

Relacje zobrazowane na ilustracji poniżej oddają sedno dyskusji o tych pojęciach w tym artykule. Nie wspominamy tylko o kodowaniu (*coding*), jest to bowiem termin, którego znaczenie zostało znacząco poszerzone przez programowanie.

Ostatnio pojawił się on w kontekście programowania w inicjatywie code.org (patrz http://godzinakodowania.pl), gdyż w języku angielskim, *code* jest krótszym słowem niż *programming*, jak to uzasadnił twórcy tej inicjatywy Hadi Portavi. W świetle naszej dyskusji zbędny jest jakikolwiek dodatkowy komentarz do tej ilustracji.

6. Proces rozwiązywania sytuacji problemowej

W tym rozdziale ilustrujemy na przykładach proces rozwiazywania danej sytuacji problemowej, wyróżniając i podkreślając w nim elementy myślenia komputacyjnego.

Poniżej w tabeli przedstawiamy ogólną specyfikację kolejnych kroków na drodze do rozwiązania konkretnej sytuacji problemowej – wyróżniono pismem półgrubym pojawiające się elementy myślenia komputacyjnego.

L.p.	Kolejne kroki w procesie rozwiązania sytuacji problemowej
1.	Dość luźny opis sytuacji problemowej, zadania do wykonania
2.	zebranie informacji i danych – **abstrakcja** na niskim poziomie, nie wszystkie dane są potrzebne
3.	wykrywanie w danych dla problemu: zasad, **wzorców**, reguł, teorii
4.	analiza danych i reprezentacja danych (jeszcze poza komputerem): lista, tabela, tabele powiązane – **abstrakcyjne struktury danych**
5.	**dekompozycja danych** i/lub **problemu** (na mniejsze znane)

6.	projektowanie algorytmu – modelu obliczeniowego: • podejście ad hoc – **heurystyka** • **metody/algorytmy informatyczne**: algorytm liniowy, pętle, warunki, rekurencja • współbieżność, interakcja – **zdarzenia** • automatyzacja rozwiązania – **program** – **abstrakcja** sytuacji zewnętrznej
7.	**symulacja komputerowego modelu problemu** • testowanie i poprawianie programu (debagowanie)
8.	głębsza analiza problemu oraz **modyfikowanie**, **poszerzanie**

Tabelę poniżej wypełniły zapisy związane z rozwiązaniem konkretnej sytuacji problemowej, przed którą stanęli uczniowie nauczania wczesnoszkolnego. Te zapisy są informacją dla nauczyciele, na jakie aspekty pracy uczniów powinien zwrócić uwagę, półgrubym pismem wyróżniono kształcone na tych zajęciach kompetencje uczniów, być może nieświadomie dla nich, ale z pełną świadomością nauczyciela.

L.p.	Kolejne kroki w procesie rozwiązania sytuacji problemowej
1.	Sytuacja: na podłodze leżą porozrzucane obrazki zwierząt Zadanie: znajdź najlżejszego ptaka
2.	Dane: selekcja/wybór ptaków – kura ptakiem? lata? – **abstrakcja**
3.	**Zasada** w danych: zwierzę lata
4.	**Reprezentacja danych**: ptaki w rzędzie, w jakiejś kolejności (lista, ciąg – **abstrakcyjne struktury danych**)
5.	**Dekompozycja**: np. na domowe i inne ptaki
6.	**Algorytm**: losowy wybór, systematyczny algorytm: przeglądanie liniowe – **abstrakcja**, bo liczy się tylko waga
7.	Komputer, **program**: projekt w Scratchu, w innym języku – **automatyzacja** sytuacji wyabstrahowanej – liczby
8.	**Modyfikacje**: • danych – inne zwierzęta, • metody: uporządkuj od najlżejszych

W kolejnej tabeli zamieszczamy opis sytuacji-projektu, postawionego przed uczniami szkoły ponadpodstawowej w ramach zajęć polonistycznych lub informatycznych (pochodzi z podręcznika [5]).

L.p.	Kolejne kroki w procesie rozwiązania sytuacji problemowej
1.	Sytuacja: Wypowiedzi Umberto 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 Umberto 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 Umberto Eco i realizatorami projektu

Na zakończenie poniżej zamieszczamy fragmenty z aplikacji dla klas 1-3. W tym konkretnym przypadku, przy rozwiazywaniu łamigłówki Sudoku pojawiają się elementy myślenia abstrakcyjnego (nie są bowiem istotne rodzaje obiektów: owoce, zwierzęta, liczby), ale zasada, zgodnie z którą są rozkładane, oraz dekompozycji (rozkładu) tabeli na mniejsze fragmenty (mniejsze kwadraty złożone z pojedynczych pól, wiersze i kolumny), w których ma obowiązywać ta sama zasada. Ilustracje pochodzą z przygotowywanej aplikacji „Informatyka dla Smyka”.

Epilog

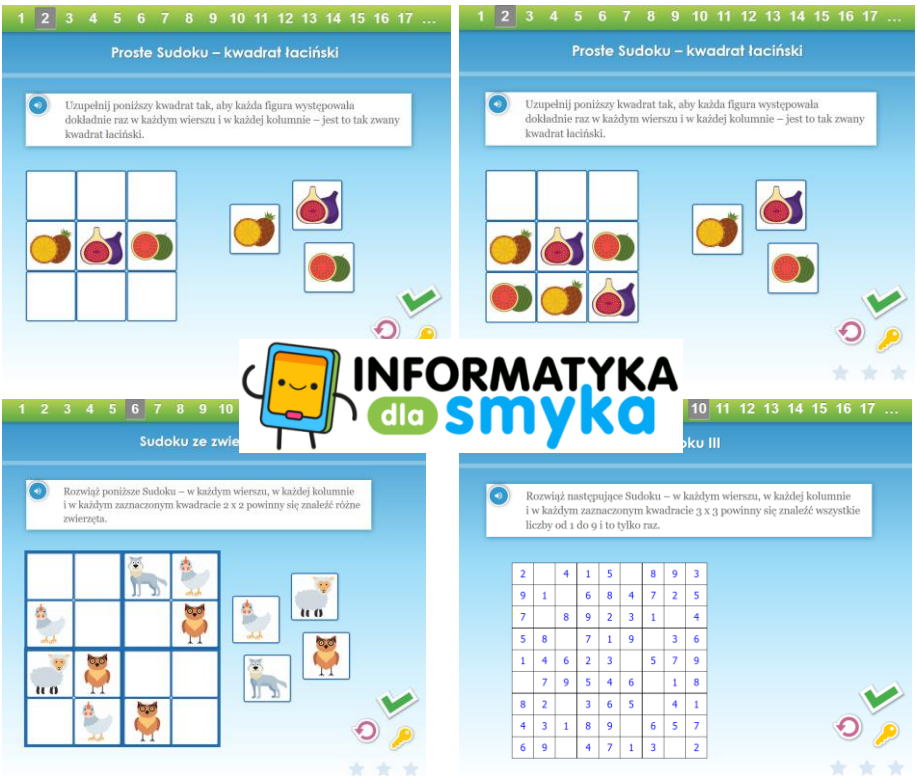
W kształceniu matematycznym, jak i w badaniach matematycznych znajduje potwierdzenie powiedzenie R.W. Hemminga z 1959 roku (zauważmy, że wtedy obliczenia komputerowe były jeszcze w powijakach):

Celem obliczeń jest wgląd, a nie liczby

The purpose of computing is insight not numbers

To powiedzenie sparafrazowaliśmy dla naszych celów na początku jako:

> Celem programowania jest abstrakcja, a nie programy
> *The purpose of programming is abstraction not programs*



Literatura

1. Aho Al, Ullman Jeff, *Foundations of Computer Science*, W.H. Freeman, 1992. Książka nie jest już drukowana, ale można ją pobrać ze strony: http://i.stanford.edu/~ullman/focs.html#pdfs
2. CSTA: Computational Thinking Task Force, http://csta.acm.org/Curriculum/sub/CompThinking.html
3. Denning P.J., *Report of the ACM Task Force on the Core of Computer Science*, ACM, 1989; w skróconej postaci został opublikowany jako Computing as a Discipline, *Comm. ACM* 32(1), 1989, 9-23.

4. Denning P.J., The Profession of IT Beyond Computational Thinking, *Comm. ACM* 52(6), 2009, 28-30.
5. Gurbiel E., Hard-Olejniczak G., Kołczyk E., Krupicka H., Sysło M.M., *Informatyka to podstawa*, WSiP, Warszawa 2012.
6. Hour of Code: http://csedweek.org/; http://godzinakodowania.pl
7. Papert S., *Burze mózgów*, WN PWN, Warszawa 1997 (oryginalne wydanie Basic Books 1980).
8. Papert S., An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning* 1(1996), No. 1, str. 95-123.
9. Sysło M.M., Myślenie komputacyjne: informatyka dla wszystkich, Materiały Konferencji „Informatyka w Edukacji, VIII", UMK Toruń, 2011.
10. Sysło M.M., Kwiatkowska A.B., Myśl logarytmicznie!, *Delta* nr 12/2014.
11. Sysło M.M., Kwiatkowska A.B., Learning Mathematics Supported by Computational Thinking, w: Materiały *Constructionism and Creativity*, Wiedeń 2014.
12. Sysło M.M., Kwiatkowska A.B., Introducing Students to Recursion: a Multi-Facet and Multi-Tool Approach, w: Materiały *ISSEP 2014*, Istanbul (Turkey), 2014.
13. Sysło M.M., Myślenie komputacyjne. Nowe spojrzenie na kompetencje informatyczne, w: Materiały „Informatyka w Edukacji, XI", UMK Toruń 2014, 15-32.
14. 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).
15. Sysło M.M., Kwiatkowska A.B., Informatyka dla najmłodszych. Pojęcia, algorytmy, programy, Materiały Konferencji „Informatyka w Edukacji, XII", UMK Toruń, 2015, str. 15-40
16. Sysło M.M., Wprowadzając… porządek, w: Materiały konferencji „Informatyka w Edukacji – IwE 2016", UMK, Toruń 2016; dostępne na: http://iwe.mat.umk.pl/archiwum/iwe2016/?q=node/20
17. Sysło M.M., Zaprogramuj swoją przyszłość, *Wprost* nr 7/2017.
18. Syslo M.M., Rozwój pojęć informatycznych od pierwszej klasy, Materiały Konferencji „Informatyka w Edukacji, XIV", UMK Toruń, 2017, str. 29-40.
19. Wing J., Computational thinking, *Comm. ACM* 49(3), 2006, 33–35
20. Wing J., Computational thinking benefits society, http://socialissues.cs.toronto.edu/index.html%3Fp=279.html