Od redakcji

Rok szkolny 2017/2018 jest pierwszym, w którym część uczniów uczy się w nowym systemie 8-klasowej szkoły podstawowej oraz 4-letniego liceum lub 5-letniego technikum (lub dwustopniowej szkoły branżowej). W tym roku szkolnym dotyczy to I, IV i VII klasy szkoły podstawowej oraz I klasy szkoły branżowej. Zmiana systemu spowodowała także opracowanie nowych podstaw programowych. W przypadku informatyki te zmiany są bardzo poważne, były postulowane przez środowiska informatyczne związane z edukacją od kilku lat. Już sama zmiana nazwy przedmiotu w szkole podstawowej z „zajęcia komputerowe" na „informatyka" świadczy o tym, że nowa podstawa kładzie nacisk na inne treści i umiejętności.

Zmieniają się także ramowe plany nauczania, informatyka będzie nauczana przez prawie wszystkie lata edukacji w wymiarze 1 godziny tygodniowo (w klasach I-III edukacja informatyczna w ramach nauczania zintegrowanego). W szkole ponadpodstawowej oznacza to trzy razy więcej godzin względem aktualnie jeszcze obowiązującej podstawy!

Nowa podstawa programowa wprowadza powszechną naukę programowania na wszystkich etapach edukacyjnych. Ponieważ terminy „kodowanie" i „programowanie" niesłusznie są często używane zamiennie, warto się zastanowić, jak je rozumiemy. Kodowanie kojarzy się raczej z odtwórczym działaniem, zakodowaniem czegoś według gotowej już instrukcji, np. zakodowaniem ruchów robota według przepisu dostarczonego przez producenta, a nie własnego pomysłu. Programowanie jest rozumiane znacznie szerzej – jako twórcza działalność od projektowania do finalnego uruchomienia.

Czy wprowadzenie powszechnej nauki programowania oznacza, że każdy uczeń w przyszłości ma zostać programistą? Oczywiście nie, choć kształcenie informatyczne (w tym programowanie) powinno ułatwić w przyszłości wybór dalszej drogi kształcenia.

W tym numerze znajdą Państwo wiele artykułów związanych zarówno z dotychczasowymi doświadczeniami z nauką programowania w polskich szkołach, jak i przede wszystkim z nowym podejściem do powszechnego kształcenia informatycznego.

Zapraszamy do lektury.

Teorie i badania

Prof. dr hab. Maciej M. SYSŁO
Zajęcia informatyczne w nowej odsłonie 2

Małgorzata ROSTKOWSKA
Wspomaganie powszechnej nauki programowania – czy można je znaleźć w podstawach programowych innych przedmiotów niż informatyka? 8

Tomasz ŁUKAWSKI
Konstruktywistyczne spojrzenie na naukę programowania w szkole podstawowej 16

Nauczanie i uczenie się

dr Jan A. WIERZBICKI
Programujemy już dziesiątki lat... 22

Agnieszka BOROWIECKA
Scratch, matematyka i klocki LEGO 26

Maciej BOROWIECKI, Krzysztof CHECHŁACZ
Od programowania wizualnego do tekstowego 34

Witold KRANAS
W poszukiwaniu środowiska do nauki programowania – poza Scratchem 39

Wanda JOCHEMCZYK, dr Katarzyna OLĘDZKA
Czy można zaprzyjaźnić się z Pythonem? 43

Dobra praktyka

Prof. dr hab. Krzysztof DIKS
25 lat Polskiej Olimpiady Informatycznej. Następny krok 48

Agnieszka BOROWIECKA
Programowanie (nie tylko) dla artystów 53

Janusz S. WIERZBICKI
Nauka programowania na iPadzie – od zera do bohatera 60

Zyta CZECHOWSKA, Jolanta MAJKOWSKA
Programowanie na specjalne zamówienie 69

Samokształcenie

Witold KRANAS
Programowanie w Akademii Khana 79

Zajęcia informatyczne w nowej odsłonie

Prof. dr hab. Maciej M. SYSŁO

1. Wstęp

Nowa podstawa programowa informatyki jest rzeczywiście nową odsłoną kształcenia informatycznego. Przede wszystkim kształcenie to przebiega przez wszystkie lata w szkole i chociaż ma pewne cechy rewolucji, uwzględniając dotychczasowe doświadczenia i osiągnięcia na tym polu, jej wdrożenie może przebiegać ewolucyjnie. Powierzchowne spojrzenie na sytuację zdaje się przekonywać, że szkoły są przygotowane na kolejną odsłonę w tym ewolucyjnym rozwoju kształcenia informatycznego, który teraz następuje. Przyglądając się głębiej, można dostrzec jednak wiele obszarów, które wymagają poważniejszych przemyśleń i zmian. Wśród nich są m.in. sposób prowadzenia zajęć informatycznych w ramach nauczania wczesnoszkolnego oraz wdrożenie spiralnego rozwoju pojęć i umiejętności informatycznych przez wszystkie lata w szkole – 12 lat kształcenia informatycznego nie wystarczy zaplanować jako realizację programów czy też „przerobienie" podręczników dla poszczególnych lat w szkole. Komentujemy tutaj tak kwestie ogólne, jak i szczegółowe, ważne dla sukcesu nowego kształcenia informatycznego.

2. Podstawa, tło zmian, wyzwania

Nowa podstawa programowa informatyki

W porównaniu z poprzednią podstawą w przypadku informatyki zaszły najpoważniejsze zmiany: (1) informatyka jest obecnie obowiązkowym przedmiotem na każdym etapie edukacyjnym i (2) dla zapewnienia harmonijnego rozwoju uczniów określono identyczne Cele kształcenia – wymagania ogólne dla wszystkich etapów edukacyjnych. Taka budowa podstawy została podyktowana przez sugerowaną metodykę kształcenia – **spiralny** rozwój pojęć i kształcenia umiejętności informatycznych na kolejnych etapach edukacyjnych określonych w Treściach nauczania – wymaganiach szczegółowych.

Należy zwrócić uwagę, że wśród wymagań ogólnych, **Programowanie i rozwiazywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych** stanowi drugą grupę wymagań, poprzedzoną przez grupę pierwszą **Rozumienie, analizowanie i rozwiązywanie problemów**. Odzwierciedla to właściwe miejsce dla kształcenia umiejętności programowania – programowanie jest narzędziem w rozwiązywaniu problemów za pomocą komputera, a nie celem samym w sobie (patrz rozdz. 4).

Kształcenie w zakresie **posługiwania się aplikacjami** i innymi gotowymi programami nie zniknęło z podstawy – znajduje się w drugiej grupie Celów kształcenia – wymagań ogólnych – jako posługiwanie się aplikacjami komputerowymi i jest szczegółowo rozpisane w Treściach nauczania – wymaganiach szczegółowych. Nowe podejście do kształcenia informatycznego otwiera nowy rozdział dla posługiwania się aplikacjami komputerowymi, by korzystanie z nich było 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 „programem" zapisanych w nim obliczeń. Największą rewolucję czeka prezentacja – gdyż projekty w języku Scratch to prezentacje, które mogą oddać nieograniczoną wyobraźnię ucznia stosującego animacje, interakcje, reakcje na zdarzenia i wszelkie media.

Tło i kierunki zmian

Kształcenie informatyczne w nowej podstawie uwzględnia zmiany, jakie zachodzą w społeczeństwie i w technologii. Żadna inna dziedzina nie stwarza takich możliwości (np. zatrudnienia), jak informatyka, a w ogólności – technologia, bez względu na obrany kierunek kształcenia i zawodowe zainteresowania uczniów. Formułowane są ważkie argumenty ekonomiczne, społeczne i kulturowe za umieszczeniem informatyki w podstawie programowej na każdym etapie edukacyjnym. Przygotowanie informatyczne umożliwi młodym Polakom przewodzenie transformacjom w tych obszarach, a nie tylko uleganie zmianom wywieranym przez rozwój technologii. Dotychczas w edukacji dużą uwagę przywiązywano do kształcenia umiejętności korzystania z aplikacji komputerowych oraz zasobów i komunikacji w sieci. Te umiejętności są nadal potrzebne, ale nie są już wystarczające. Podstawowe zadanie szkoły – alfabetyzacja w zakresie czytania, pisania i rachowania wymaga dzisiaj poszerzenia o alfabetyzację w zakresie **myślenia komputacyjnego**, czyli o umiejętności rozwiązywania problemów z różnych dziedzin z wykorzystaniem metod oraz narzędzi wywodzących się z informatyki oraz o lepsze zrozumienie, jakie są możliwości komputerów, ich zastosowań i technologii. Potrzeby rynku pracy podnoszą rangę zawodów informatycznych, na co szybko zareagowały rządy wielu państw. Nasz system edukacji stara się nie przeoczyć tego trendu, zwłaszcza że nauczanie informatyki ma u nas w kraju długą tradycję.

Najważniejsze wyzwania

Rewolucyjnych cech zmian w kształceniu informatycznym można upatrywać zarówno w „szerzeniu" myślenia komputacyjnego, jak i w propozycji nauki programowania od najmłodszych lat. To drugie jednak z solidnym przygotowaniem uczniów do „dialogu" z komputerem, by mieli o czym z nim „rozmawiać".

Nowa podstawa programowa weszła w życie od września 2017 jednocześnie w klasach I, IV i VII, a w przypadku informatyki jest znacząco inna niż dotychczasowa. Uczniowie przychodzący do tych klas, przynajmniej w pierwszych latach, nie będą więc wcześniej przygotowani zgodnie z nową podstawą, z wyjątkiem uczniów rozpoczynających naukę. Tych innych uczniów nie można jednak „stracić" – nauczyciele powinni odpowiednio rozpoczynać realizację nowej podstawy z uczniami, którzy „przystępują" do niej z marszu. Wielu uczniów zapewne wcześniej miało styczność z programowaniem.

Dzisiejszy entuzjazm najmłodszych, widoczny głównie przy programowaniu, jeszcze nie gwarantuje, że informatyka będzie ich interesować i angażować przez 12 lat w szkole. Cele ogólne – takie same na kolejnych etapach lat pobytu w szkole, każą myśleć o wszystkich etapach jednocześnie, a na poszczególnych etapach – uwzględniać wcześniejsze i późniejsze etapy. To jedno z wyzwań stojących przed nauczycielami, jak również przed autorami podręczników i innych pomocy[1].

Dla sukcesu proponowanych zmian na przestrzeni 12 lat edukacji szkolnej decydujące będą dwa pierwsze etapy szkoły podstawowej (klasy I-III i IV-VI). To etapy dotychczasowych zajęć komputerowych, w czasie których nauczyciele na ogół rzadko sięgali po programowanie, a główna uwaga, ich i uczniów, była skupiona na korzystaniu z gotowych aplikacji (patrz punkt 3.1).

Innym wyzwaniem będzie, jak nie „przegapić" w szkole momentu głębszego zainteresowania uczniów informatyką i programowaniem, by w rezultacie obrali i podążali ścieżką kształcenia informatycznego na dalszych etapach, kierując się ku karierze zawodowej informatyka czy programisty. Wydaje się, że takim momentem są klasy V-VII szkolnej

[1] W [2] został zamieszczony przykład rozwijania pojęć i metod informatycznych przez 12 lat za pomocą posługiwania się zagadnieniami z obszarów poszukiwania i porządkowania informacji.

edukacji, czyli wiek 12-15 lat. Jednym z właściwych objawów ciągłości (spiralności) w kształceniu informatycznym będzie przemyślane przejście między środowiskami programowania wizualno-blokowego (np. w języku Scratch) i tekstowego (np. w języku Python).

Jeśli chodzi o miejsce programowania na zajęciach (patrz również rozdz. 4), to warto pamiętać, że skupienie głównej uwagi na programowaniu, a nie na rozwiązywanych problemach, może doprowadzić do podobnego znużenia uczniów, jakie obserwujemy przy posługiwaniu się (bez celu) biurowymi aplikacjami komputerowymi. Zniechęcanie do pisania programów może łatwo przerodzić się w zniechęcenie do informatyki. **Informatyka to coś więcej niż kodowanie** (tutaj w znaczeniu programowanie) – to hasło serwisu *Godzina Kodowania* (`http://godzinakodowania.pl`). Dla przykładu, przedmiot, którym objęci są wszyscy uczniowie w Wielkiej Brytanii, to *computing* (ma szersze znaczenie niż *computer science*, informatyka), a w Stanach Zjednoczonych prezydent Barack Obama ogłosił narodowy program „Computer Science for ALL". Nauka programowania jest tylko elementem tych inicjatyw.

3. Czy jesteśmy przygotowani?

Przy tak szerokim froncie zmian i oczekiwań na ich efekty w kształceniu informatycznym pojawia się naturalne pytanie, czy jesteśmy – nauczyciele, szkoły, system edukacji – przygotowani na udźwignięcie pojawiających się wyzwań. Warto zauważyć i wypada docenić, że obecne propozycje zmian w edukacji informatycznej są konsekwencją jej rozwoju od połowy lat 60. XX wieku. Na pytanie, czy polska szkoła jest przygotowana na proponowane zmiany można więc przekonywać, że TAK, jednak wdrożenie zmian będzie wymagać znacznego wysiłku wszystkich aktorów w teatrze szkoły. Dzisiejsze propozycje, gotowe już w połowie 2014 roku, nie byłyby możliwe bez wcześniejszych decyzji i nagromadzonych doświadczeń. Zazdroszczą nam tego inne kraje: (1) na każdym etapie edukacyjnym są wydzielone zajęcia informatyczne – a według nowej podstawy programowej będzie to przedmiot informatyka na wszystkich etapach kształcenia; (2) zajęcia te prowadzą przygotowani do tego nauczyciele – przed którymi teraz olbrzymie wyzwanie podniesienia poziomu wiedzy, umiejętności i kompetencji do poziomu wymagań nowej podstawy przedmiotu informatyka, dotyczy to zwłaszcza nauczycieli nauczania wczesnoszkolnego; (3) szkoły są w miarę dobrze wyposażone – wśród nowych inwestycji powinny się jednak znaleźć tablety i roboty oraz dostęp do szerokopasmowego Internetu. Najważniejsze jednak (4), że mamy rozbudzoną szkolną młodzież, która jest poważnie zainteresowana programowaniem i rozwojem swoich kompetencji informatycznych. Ten ostatni argument jest najistotniejszy – musimy wyjść naprzeciw oczekiwaniom głównych beneficjentów edukacji, nawet jeśli nie do końca rozumieją i są świadomi, czym jest informatyka – to im mamy przybliżyć – która nie sprowadza się tylko do zabawy w kodowanie i programowanie robotów. Poniżej komentujemy stan przygotowania i podejmowane działania.

3.1. Wydzielone przedmioty informatyczne i ich nauczyciele

Na informatykę przewidziano jedną godzinę w każdej klasie. Może mało, ale niech ten czas będzie właściwie wykorzystany przez zastosowanie informatyki również w innych przedmiotach. Niestety lektura podstaw programowych innych przedmiotów jest smutna i niepokojąca – na przykład w podstawach matematyki i fizyki nie pojawia się komputer ani metody informatyczne, ani też technologia w roli technologii kształcenia. Rolą kształcenia informatycznego jest więc również wypełnić tę przepaść – między klasyką a potrzebami młodego pokolenia, by przygotować je na wyzwania technologiczne i zawodowe, tak osobiste, jak i społeczne.

Olbrzymim skrótem jest powiedzenie, że jeśli w szkole są wydzielone przedmioty informatyczne, to w szkołach są nauczyciele przygotowani do ich prowadzenia. Nowa podstawa programowa informatyki jest jednak całkowicie nowa, zwłaszcza na najniższym etapie kształcenia, bo nie było na nim dotychczas w podstawie programowej niemal żadnych elementów informatyki jako dziedziny.

Od poprzedniej reformy systemu edukacji cieszymy się, że zajęcia komputerowe objęły również nauczanie wczesnoszkolne w klasach I-III. Gorzej z realizacją tych zajęć – często były one oddawane nauczycielom wyższych etapów edukacyjnych. W takich przypadkach, ze względów organizacyjnych, najmłodsi uczniowie mieli tygodniowo jedną godzinę informatyki oderwaną od ich zintegrowanych edukacji prowadzonych przez „naszą Panią". Dla sukcesu zmian w kształceniu informatycznym to musi ulec zmianie za wszelką cenę, a wtedy zajęcia informatyczne w klasach I-III zostaną „rozciągnięte" na cały tydzień – może to być także antidotum na braki w wyposażeniu, nie wszyscy uczniowie muszą bowiem jednocześnie korzystać z komputerów czy tabletów. Można również zaplanować wyjście do pracowni komputerowej, ale decyzję o tym powinien podejmować i uczestniczyć w tych zajęciach nauczyciel nauczania wczesnoszkolnego.

Zakres przygotowania nauczycieli informatyki do realizacji nowej podstawy programowej tego przedmiotu na wszystkich etapach edukacyjnych został szczegółowo rozpisany w standardach przygotowania nauczycieli informatyki (patrz [1, 5]). Standardy dla nauczycieli nauczania wczesnoszkolnego wydzielono jako osobną grupę. Wszystkie standardy zapisano w formie operacyjnej, czyli „co nauczyciel powinien wiedzieć i umieć, by prowadzić zajęcia z informatyki". Dla spiralnego kształcenia jest niezbędne, by nauczyciel na każdym etapie edukacyjnym wiedział, z czym przychodzą do niego uczniowie i jak wygląda kontynuacja zajęć informatycznych na dalszych etapach.

3.2. Wyposażenie uczniów, nauczycieli i szkół

Podobnie jak z godzinami zajęć, sprzętu nigdy nie jest dość w szkole. Ponadto z czasem sprzęt ulega degradacji. Pojawiają się też nowe rozwiązania technologiczne, z którymi warto zapoznać uczniów. Ostatnio takimi urządzeniami są drukarki 3D, którymi pasjonują się uczniowie, chociaż nie bardzo wiadomo, jak miałyby one uzupełnić ofertę zajęć informatycznych i wspomóc inne przedmioty.

Jeśli chodzi o doposażenie szkół w sprzęt, który mógłby wspomóc realizację nowej podstawy programowej informatyki, to nie ma dobrych wiadomości. Planowany jest rządowy program „Aktywna tablica", ale propozycja, by oprócz interaktywnych tablic lub ekranów szkoła mogła zakupić tablety czy roboty – wykorzystywane na zajęciach informatycznych, zwłaszcza z najmłodszymi uczniami – nie została uwzględniona. Szkoda, że jeden program MEN – nowe kształcenie informatyczne – nie będzie wspierany przez inny program MEN.

Rzut oka na nowe podręczniki do informatyki nie napawa optymizmem. Krótki czas ich przygotowania spowodował, że głównie są to modyfikacje dotychczasowych podręczników. Mając jednak na uwadze spiralny rozwój informatycznych umiejętności uczniów przez 12 lat, trzeba bardzo poważnie pochylić się nad propozycją podręczników i materiałów edukacyjnych dla całego obszaru kształcenia informatycznego, od pierwszej po ostatnią klasę. To jest olbrzymie wyzwanie. W szczególności dla klas I-III i IV-VI nie wystarczy dostosować jedynie dotychczasowego materiału, jaki był przeznaczony dla zajęć komputerowych w tych klasach. Obecnie, ucząc posługiwania się aplikacjami komputerowymi, należy brać pod uwagę również realizację zapisów związanych z informatyką i programowaniem. Dla klas I-III nie powinna to być propozycja rozpisana na godziny, bo to sugeruje wyjście klasy do pracowni komputerowej i ograniczenie informatyki do jednej godziny w tygodniu w miejsce jej integracji z innymi edukacjami. Niestety ten tryb prowadzenia zajęć z informatyki w klasach I-III utwierdzają ukazujące się na rynku podręczniki.

W propozycjach podręczników nie ma odniesień do zajęć z programowalnymi robotami, szczególnie przydatnymi przy realizacji zapisów podstawy na najniższym etapie edukacyjnym. Nie korzysta się również z tak znakomitych serwisów edukacyjnych, jak *Godzina Kodowania* (GK – `http://www.godzinakodowania.pl`) czy konkurs Bóbr (`http://bobr.edu.pl`), chociaż oba są bardzo popularne w szkołach, a metodycznie mogą być znakomitym, angażującym uczniów wprowadzeniem do programowania. Ostatnia informacja zza Wielkiej Wody ilustruje olbrzymi potencjał tkwiący w GK – zdaniem analityków ten serwis przyczynił się w ostatnim roku do wzrostu o około 150% liczby dziewcząt i przedstawicieli mniejszości narodowych w USA podejmujących kurs informatyczny

AP (ang. *advance placement*). To tylko potwierdza, że jeśli w klasach I-III uczeń pozna programy dla łamigłówek w GK, to nie będzie trzeba tłumaczyć mu później (w klasie IV czy VIII), co to jest program, algorytm, programowanie, jak to się często robi w podręcznikach oferowanych naszym uczniom.

W starszych klasach wybór tekstowego języka programowania powinien być ukierunkowany na zawodowe i profesjonalne kształcenie w kierunkach informatycznych. Żaden z języków, Baltie, Scratch czy Logo, tego nie spełnia.

Generalnie, należy porzucić nauczanie informatyki metodą „przerabiania podręcznika", polegającą na wykonywaniu przez uczniów poleceń autora lub programu nauczania. Kształcenie informatyczne nie ma polegać na pokazywaniu uczniom możliwości komputerów i ich oprogramowania, ale na ujawnianiu i wydobywaniu możliwości uczniów (A. Walat).

Powinna zostać porzucona XIX/XX-wieczna idea podręcznika, a podpatrując, jak pracują uczniowie oraz uwzględniając, jak funkcjonuje sieć i jej społeczności, powinno się zaprojektować środowisko kształcenia informatycznego na miarę uczących się i ich czasów. W [3] zapowiedziano taki „podręcznik" do informatyki dla nauczania wczesnoszkolnego (dostępny od września 2017), będący środowiskiem do zajęć z informatyki zintegrowanych z innymi edukacjami. W perspektywie ma to być podręcznik-środowisko na wszystkie następne lata w szkole.

3.3. Uczniowie

Wreszcie najważniejsi beneficjenci zmian – uczniowie – ale przecież faktycznie cokolwiek pisząc na temat zmian w kształceniu informatycznym, stale mamy ich na uwadze: w podstawie określono obszar ich zainteresowań, zaangażowania i potrzeb, obecnych i przyszłych, informatycznych i innych; jeśli troszczymy się o przygotowanie nauczycieli, to głównie, by uczniowie znaleźli w nich swoich tutorów i doradców, którzy wskażą im drogę, pozwolą rozwinąć indywidualne zainteresowania i możliwości; jeśli chodzi o sprzęt, to liczymy na wyrozumiałość uczniów – szkoła nie ma szans w konkurencji z najlepszymi firmami czy nawet z wyposażeniem w domach uczniów, a często także w kieszeniach, ale też rola sprzętu w szkole jest inna, ma być tylko interfejsem między pomysłami i technologią, a te pomysły, sprawdzone w szkolnych warunkach, znakomicie da się zrealizować również w innych, doskonalszych – bo myśl w tym wszystkim jest najważniejsza.

4. Rola programowania

Elementem powszechnego kształcenia informatycznego staje się umiejętność programowania, uważana za jedną z podstawowych kompetencji XXI wieku, tradycyjnie określana jako „pisanie programów dla komputerów". Programowanie ma jeszcze jedno znaczenie – to również planowanie działań z wykorzystaniem metod matematycznych, informatycznych i z innych dziedzin, przynoszących najlepsze efekty. To znaczenie pochodzi z czasów II wojny światowej i dzisiaj na informatyce spotyka się np. jako programowanie dynamiczne, które można stosować niekoniecznie... programując je dla komputera.

Łącząc oba te znaczenia, można zwrócić się do młodego pokolenia z apelem: **zaprogramuj swoją przyszłość**, czyli planuj ją z uwzględnieniem wielu aspektów oraz z wykorzystaniem wielu metod tak, aby Twoja przyszłość była w przyjętym przez Ciebie sensie „rozwiązaniem optymalnym", najlepszym. Rozwijanie umiejętności programowania (komputerów) przyczynia się do kształcenie takich kompetencji, jak: logiczne myślenie, kreatywność w poszukiwaniu rozwiązań, myślenie heurystyczne w znaczeniu dobrze umotywowanego myślenia „na chłopski rozum", poszukiwanie innowacyjnych rozwiązań, algorytmiczne myślenie w znaczeniu dobrze uporządkowanych kroków postępowania, myślenie komputacyjne jako zespół *mental tools*, służących rozwiązywaniu problemów, i wreszcie posługiwanie się „językiem" komunikacji z komputerem – może to być język programowania, by nająć go do współpracy w rozwiązywaniu problemów.

Zatem kształcenie informatyczne w nowej podstawie programowej jest nie tylko propozycją włączenia programowania do zajęć szkolnych, ale ma ambicje znacznie szersze – skierowanie zainteresowania uczniów i nauczycieli na te kompetencje związane z umiejętnością programowania

komputerów, które mogą być przydatne, by uczestniczyć w zaprogramowaniu... swojej przyszłości[2].

Język programowania

Komunikacja z komputerem wymaga języka do jej prowadzenia, a więc języka programowania. Przy wyborze języka należy uwzględnić wiele czynników. Zauważmy, że ważniejsze od języka jest to, co chcemy w nim wyrazić, powiedzieć, przekazać komputerowi, skomunikować się z nim, nawiązać dialog. Ważniejsze więc jest to, co uczeń wyraża i w jakim stylu za pomocą języka programowania niż w jakim robi to języku.

Na ten temat napisano tomy i tony publikacji. Kilka ważniejszych cech doboru i obcowania z językiem programowania to: (1) dobry język odzwierciedla ważne pojęcia; (2) powinien być nośnikiem, a nie obiektem nauczania; jest narzędziem, a nie celem; (3) kolejne konstrukcje powinny być poznawane przez uczniów wtedy, gdy są im potrzebne; (4) program to komunikat zrozumiały dla innych osób, nie tylko dla komputera; (5) na potrzeby edukacji rozróżniamy języki blokowo-wizualne i tekstowe; te pierwsze umożliwiają łagodne wprowadzenie do programowania od najmłodszych lat.

Jako konkluzję z tego wyliczenia przyjmijmy, że **programowanie jest określeniem całego procesu rozwiązywania problemu**. Komputer może, ale nie musi pojawić się w tym procesie, na ogół pojawia się na ostatnim etapie, gdy mamy skrystalizowany pomysł jego użycia do rozwiązania czy też otrzymania realizacji naszego pomysłu (algorytmu).

Obserwujemy olbrzymią ilość propozycji zajęć dla uczniów, głównie dla najmłodszych, zajęć dotyczących programowania na ogół nazywanego kodowaniem. Można się obawiać, czy te zajęcia nie sprowadzają się głównie do „pisania programów". W tym miejscu warto przytoczyć słowa obawy wyrażonej ostatnio przez Mitchela Resnicka, twórcy Scratcha, ucznia Seymoura Paperta, twórcy Logo:

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.*

Bibliografia

1. Sysło M.M. *Standardy przygotowania nauczycieli informatyki*, Wrocław, Toruń 2014.

2. 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

3. Sysło M.M. *Rozwój pojęć informatycznych od pierwszej klasy* [w:] Materiały konferencji „Informatyka w Edukacji – IwE 2017", UMK, Toruń 2017; dostępne na http://iwe.mat.umk.pl/iwe2017/node/21

4. Sysło M.M. *Standardy przygotowania nauczycieli informatyki* [w:] Materiały konferencji „Informatyka w Edukacji – IwE 2016", UMK, Toruń 2016, dostępne na http://iwe.mat.umk.pl/archiwum/iwe2016/?q=node/20

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

[2] Ten wątek autor rozwija w wielu swoich wypowiedziach i publikacjach.

Wspomaganie powszechnej nauki programowania – czy można je znaleźć w podstawach programowych innych przedmiotów niż informatyka?

Małgorzata ROSTKOWSKA

1. Wstęp

Wprowadzenie do etapu wczesnoszkolnego edukacji informatycznej, a w niej programowania i myślenia komputacyjnego[1] **jest jedną z fundamentalnych zmian całej obecnej reformy szkolnej**. Takie jest moje zdanie i dlatego chcę przyjrzeć się wybranym podstawom programowym innych przedmiotów (pp) i zobaczyć, czy w nich znajdują się zapisy, które mogą być „wspomaganiem" powszechnej nauki programowania zapisanej w pp do informatyki. A może są tam tylko zapisy dotyczące stosowania technologii informacyjno-komunikacyjnych (TIK)?

Chcę również zachęcić nauczycieli innych przedmiotów niż informatyka do przeczytania z uwagą ogólnych celów zapisanych w pp oraz poznania niektórych zapisów z pp informatyki.

Zwrócę również uwagę nauczycieli, że w pp przedszkola czy w klasach I-III zmiany wiążące się z nauką programowania można powiązać z szeregiem działań dotychczas podejmowanych przez nauczycieli, a wiążących się z logicznym, algorytmicznym myśleniem dzieci. Tu chodzi o nazywanie codziennych ćwiczeń z dziećmi słowami, które wnikają w ich myślenie i są przygotowaniem do późniejszego pisania programów.

2. Jakie korzyści będą mieli uczniowie i nauczyciele wspomagający powszechną naukę programowania?

Myślenie komputacyjne, które będą rozwijać nauczyciele u swoich uczniów, jest zyskiem właśnie dla uczniów, a przykłady programów, które zaproponują im do zrobienia, będą też miały potrójną dla uczniów korzyść. **Wzmocnią naukę treści** z przedmiotu, **z którego uczniowie wykonają program; uczniowie będą tworzyli rzeczy dla siebie ważne**, nauczyciel **łatwiej wprowadzi** do swojego arsenału środków i narzędzi dydaktycznych **nowoczesne technologie**. To będzie prawdziwe **uczenie się przez tworzenie**, a przecież to jest najwyższa strategia zdobywania wiedzy i umiejętności.

3. Nauczyciel – potrzebujący nowej podstawy programowej lub radzący sobie bez niej

Nauczyciele w polskiej szkole to grupa ludzi dobrze wykształconych. W większości widzą oni, jak zmienia się świat, jak ten świat wpływa na ich uczniów

[1] Terminem **myślenie komputacyjne** (ang. *computational thinking*) określa się **procesy myślowe towarzyszące formułowaniu problemów i ich rozwiązań w postaci umożliwiającej ich efektywną realizację z wykorzystaniem komputera**. Obejmuje szeroki zakres intelektualnych metod i narzędzi, przydatnych przy rozwiązywaniu problemów z różnych dziedzin z wykorzystaniem przy tym komputera i metod mających swoje źródło w informatyce, wywodzących się z komputerowego przetwarzania informacji i rozwiązywania problemów z pomocą komputerów w różnych dziedzinach. Integruje ludzkie myślenie z możliwościami komputerów. Według Jeannette Wing, która ukuła ten termin (2006), myślenie komputacyjne określa użyteczne postawy i umiejętności, jakie każdy, nie tylko informatyk, powinien starać się wykształcić i stosować. Dzięki takiemu **szerokiemu spojrzeniu na kompetencje informatyczne**, informatyka nie jest ograniczana do nauki o komputerach, ale dostarcza metod dla działalności umysłowej, które mogą być użyte z korzyścią dla innych dziedzin i w codziennym życiu.

i czego ci uczniowie potrzebują. Tacy nauczyciele na bieżąco starają się tak pracować z uczniami, aby mogli oni jak najlepiej się rozwijać. Jeśli sami mają jakieś wątpliwości, to próbują się z innymi nauczycielami porozumieć i wypracowywać najlepsze strategie. Oni sami wiedzą, jak mądrze wprowadzać TIK do edukacji, prawdopodobnie też znają zalety wprowadzania programowania dla każdego ucznia i starają się tak pracować. Można powiedzieć, że podstawa programowa jest dla nich potwierdzeniem tego, co robią i jak pracują od dawna z uczniami. Ale są też zapewne nauczyciele, którzy główną uwagę skupiają na przestrzeganiu prawa oświatowego i dopiero wówczas wprowadzają zmiany, gdy są one zapisane wprost w odpowiednich dokumentach. Mogą oni być też rozliczani przez urzędników-dyrektorów ze zgodności z zapisami w podstawie programowej, poza tym nie bardzo chce się im samodzielnie poszukiwać rozwiązań i dopiero nacisk z góry pomoże im się zająć sprawą wprowadzania myślenia komputacyjnego i programowania.

Właściwie zapis w pp **kreatywne rozwiązywanie problemów z różnych dziedzin ze świadomym wykorzystaniem metod i narzędzi wywodzących się z informatyki, w tym programowanie,** mógłby nauczycielom wystarczyć do wszystkich aktywności, które wymyślą dla uczniów (uczniom niektórych nauczycieli ten zapis służy już od dawna, zanim pojawił się w pp).

Nauczycielom poszukującym, jeśli jeszcze sami czegoś nie wymyślili, wystarczy np. pokazać, jak to robią inni nauczyciele. Takich przykładów jest wiele. W naszym Ośrodku nauczycielom nauczania przedszkolnego i wczesnoszkolnego już od dawna pokazujemy, jak uczyć programowania w zależności od tego, co się ma, czego się potrzebuje i co się lubi. W grach, zabawach, z tabletem, z komputerem, wykorzystując rozszerzoną rzeczywistość itp.

W Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie na potrzeby nauczycieli poszukujących wsparcia w nauce programowania na wszystkich etapach edukacyjnych utworzono portal `http://programowanie.oeiizk.edu.pl`. Proszę popatrzeć chociażby na same zakładki, np. *Informatyka (prawie) bez komputera, Szkolenia dla nauczycieli, Warszawa programuje!, Scratch i Processing, Mistrzowie kodowania* czy *Uczymy z e-podręcznikiem*. A w menu: *Scratch, Logo, Python, Processing, C* i *C++*. Bogactwo materiałów i zachęty do szkoleń.

Warto też popatrzeć np. na blogi, strony, którymi dzielą się nauczyciele z grupy SuperBelfrzy, np. na blogu: `http://www.superbelfrzy.edu.pl`, gdzie każdy nauczyciel może znaleźć *Pomysłodajnię, Projektowanie, Edu-granie, Edu-refleksje, Narzędziownię* – już same nazwy tych zakładek dają wiele nadziei na znalezienie tego, czego się szuka. Stamtąd można przejść na inne blogi czy kanały dokładniej wprowadzające w interesujące nauczyciela zagadnienia.

Bogactwo pomysłów, inspiracji znalezionych w sieci jest nie do ogarnięcia, żaden nauczyciel, który chciałby coś znaleźć, podzielić się, nie może czuć się osamotniony, tylko musi sam poczuć chęć i potrzebę. Proszę np. popatrzeć na jeden z blogów p. Jolanty Okuniewskiej pn. „Tableszyt w okładce motyla": `http://tableciaki.blogspot.com`.

Nauczyciel potrzebujący zapisów pp może ją ze zrozumieniem przeczytać, gdyż podstawa programowa dla najmłodszych uczniów jest już ogłoszona i można się jej przyjrzeć, próbując znaleźć odpowiedź na pytanie, czy nauczyciele przedmiotów innych niż informatyka mają szansę na wprowadzenie zmian w swoim przedmiocie, aby po pierwsze **stosować właściwie TIK**, po drugie **wspomóc powszechną naukę programowania**, a po trzecie **wprowadzać myślenie komputacyjne na innych przedmiotach**.

4. Rozporządzenie Ministra Edukacji Narodowej z 14 lutego 2017 roku (w Dz. U. z 24 lutego 2017 r. poz. 356) w sprawie podstawy programowej wychowania przedszkolnego oraz podstawy programowej kształcenia ogólnego dla szkoły podstawowej

Prześledźmy to rozporządzenie pod kątem wspomagania powszechnej nauki programowania

w innych niż informatyka przedmiotach, wprowadzania myślenia komputacyjnego oraz wprowadzania TIK do pp danego przedmiotu czy etapu edukacyjnego. Podstawa programowa dla przedszkoli zawarta jest w załączniku nr 1, zaś dla szkoły podstawowej w załączniku nr 2.

4.1. Przedszkole

Dla przedszkolaka, który poznaje mnóstwo pojęć, zjawisk, uczy się nazw roślin, zwierząt, zjawisk przyrodniczych zapisano zadania w 17 punktach, zaś osiągnięcia dziecka na koniec wychowania przedszkolnego zapisano w IV obszarach rozwoju dziecka.

W III społecznym obszarze rozwoju dziecka pp opisuje, co przedszkolak wymienia, określa, przelicza, klasyfikuje, rozpoznaje, nazywa, odróżnia, wykonuje itp. W jednym punkcie podstawa programowa zauważa:

19)[2] **podejmuje samodzielną aktywność** poznawczą np. oglądanie książek, zagospodarowywanie przestrzeni własnymi pomysłami konstrukcyjnymi, **korzystanie z nowoczesnej technologii** itd.

Niezbyt obszerny zapis, ale jest się już do czego odwołać. Osoba bliska przedszkolakowi powinna dopilnować, aby ta samodzielna aktywność poznawcza w kwestii korzystania z nowoczesnej technologii odbywała się w sposób bezpieczny i rozwijający przedszkolaka właśnie w tym, czego ma się nauczyć, poznać i opisać.

4.2. Szkoła podstawowa – zapisy ogólne

Załącznik 2 obejmuje podstawę programową kształcenia ogólnego dla szkoły podstawowej. Kształcenie w szkole podstawowej trwa osiem lat i jest podzielone na dwa etapy edukacyjne: I etap – klasy I-III, edukacja wczesnoszkolna i II etap obejmujący klasy IV-VIII szkoły podstawowej. I tu już znajdujemy dużo więcej zapisów. Spójrzmy chociażby na **Cele kształcenia ogólnego w szkole podstawowej** (str. 11, pp[3]):

4) rozwijanie kompetencji, takich jak: kreatywność, innowacyjność i przedsiębiorczość;

5) rozwijanie umiejętności krytycznego i logicznego myślenia, rozumowania, argumentowania i wnioskowania;

...

9) wspieranie ucznia w rozpoznawaniu własnych predyspozycji i określaniu drogi dalszej edukacji;

Wiele jest argumentów do wprowadzenia myślenia komputacyjnego czy programowania.

Albo **Najważniejsze umiejętności rozwijane w ramach kształcenia ogólnego w szkole podstawowej** (str. 12):

1) sprawne komunikowanie się w języku polskim oraz w językach obcych nowożytnych;

2) sprawne wykorzystywanie narzędzi matematyki w życiu codziennym, a także kształcenie myślenia matematycznego;

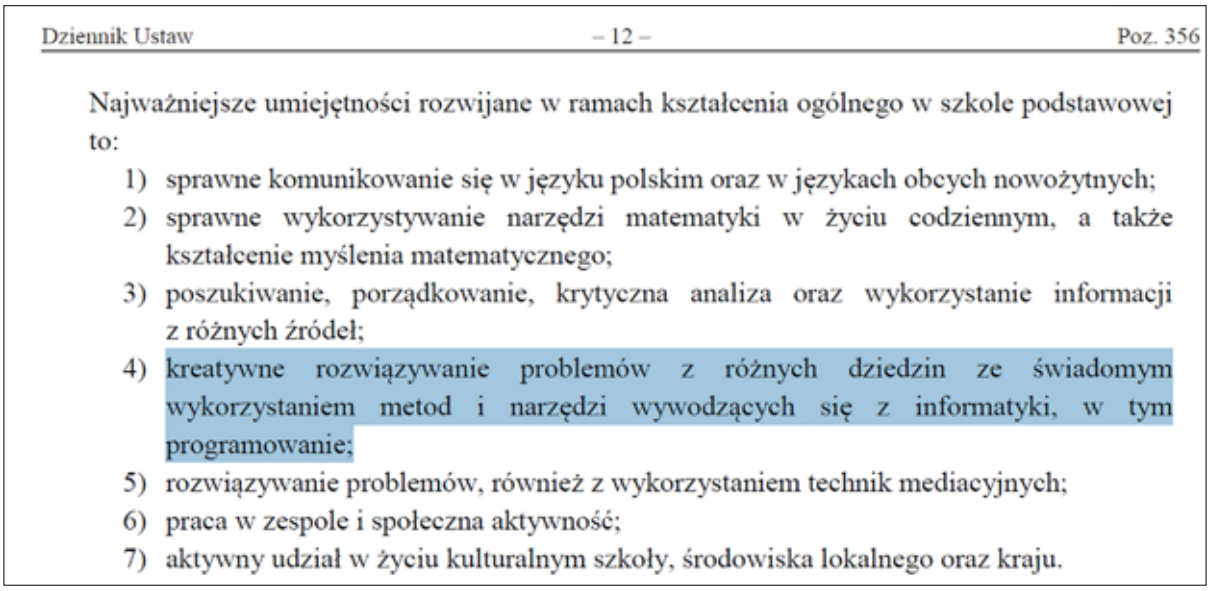
3) poszukiwanie, porządkowanie, krytyczna analiza oraz wykorzystanie informacji z różnych źródeł;

4) **kreatywne rozwiązywanie problemów z różnych dziedzin ze świadomym wykorzystaniem metod i narzędzi wywodzących się z informatyki, w tym programowanie;**

5) rozwiązywanie problemów, również z wykorzystaniem technik mediacyjnych;

[2] Czcionką Courier wpisano teksty skopiowane z zapisów nowej podstawy programowej.

[3] Podane strony odnoszą się do stron omawianego rozporządzenia opublikowanego jako plik PDF pod adresem http://www.dziennikustaw.gov.pl/DU/2017/356.



Dziennik Ustaw – 12 – Poz. 356

Najważniejsze umiejętności rozwijane w ramach kształcenia ogólnego w szkole podstawowej to:

1) sprawne komunikowanie się w języku polskim oraz w językach obcych nowożytnych;
2) sprawne wykorzystywanie narzędzi matematyki w życiu codziennym, a także kształcenie myślenia matematycznego;
3) poszukiwanie, porządkowanie, krytyczna analiza oraz wykorzystanie informacji z różnych źródeł;
4) kreatywne rozwiązywanie problemów z różnych dziedzin ze świadomym wykorzystaniem metod i narzędzi wywodzących się z informatyki, w tym programowanie;
5) rozwiązywanie problemów, również z wykorzystaniem technik mediacyjnych;
6) praca w zespole i społeczna aktywność;
7) aktywny udział w życiu kulturalnym szkoły, środowiska lokalnego oraz kraju.

6) praca w zespole i społeczna aktywność;

7) aktywny udział w życiu kulturalnym szkoły, środowiska lokalnego oraz kraju.

Patrząc na powyższe umiejętności, najważniejsze w ramach kształcenia ogólnego w szkole podstawowej, widzimy, że miejsce programowania możemy znaleźć w każdym punkcie. Również dla rozwijania myślenia komputacyjnego każdy nauczyciel znajduje uzasadnienie w tych najważniejszych umiejętnościach. Po to zostały one zapisane na początku, aby wszyscy mogli je przeczytać i zastosować nim rozpoczną zapoznanie się z podstawą programową swojego przedmiotu.

Dalej rozwinięto zapis czwartej umiejętności następująco (str. 13):

Szkoła ma stwarzać uczniom warunki do nabywania wiedzy i umiejętności potrzebnych do rozwiązywania problemów z wykorzystaniem metod i technik wywodzących się z informatyki, w tym logicznego i algorytmicznego myślenia, programowania, posługiwania się aplikacjami komputerowymi, wyszukiwania i wykorzystywania informacji z różnych źródeł, posługiwania się komputerem i podstawowymi urządzeniami cyfrowymi oraz stosowania tych umiejętności na zajęciach z różnych przedmiotów m.in. do pracy nad tekstem, wykonywania obliczeń, przetwarzania informacji i jej prezentacji w różnych postaciach.

Szkoła ma również przygotowywać ich do dokonywania świadomych i odpowiedzialnych wyborów w trakcie korzystania z zasobów dostępnych w Internecie, krytycznej analizy informacji, bezpiecznego poruszania się w przestrzeni cyfrowej, w tym nawiązywania i utrzymywania opartych na wzajemnym szacunku relacji z innymi użytkownikami sieci.

W tym zapisie widać już formalną zachętę do wskazywania uczniom na różnych przedmiotach ewentualnych problemów i rozwiązywania ich z wykorzystaniem metod i technik wywodzących się z informatyki, czyli do wykonania przez nich programów oraz trenowania myślenia komputacyjnego.

Należy przy okazji zwrócić uwagę na podkreślanie w pp roli edukacji zdrowotnej. Zapisy są ogólne i sugerują skupianie się na odżywianiu, aktywności fizycznej, bezpieczeństwie i profilaktyce. Może wszystko, co odnosi się do świata wirtualnego, jest w tym zawarte, ale nauczyciel powinien mieć świadomość, że większość zagrożeń dla ucznia przeniosła się i znajduje w świecie, w którym uczeń żyje (świat Internetu – str. 14).

Podstawa programowa kieruje nauczycieli w stronę stosowania metody projektów, podkreślając jej zalety, przy czym zaleca (str. 15):

Przy realizacji projektu wskazane jest wykorzystywanie technologii informacyjno-komunikacyjnych.

4.3. I etap edukacyjny: edukacja wczesnoszkolna

Najważniejszy zapis znajduje się na początku: celem edukacji wczesnoszkolnej jest wspieranie całościowego rozwoju dziecka. Potem zadania szkoły zapisane są w 11 punktach. Między innymi:

6) zapewnienie dostępu do wartościowych, w kontekście rozwoju ucznia, źródeł informacji i nowoczesnych technologii;

A gdzie można znaleźć „najwięcej" myślenia komputacyjnego i wprowadzania pojęć związanych z programowaniem. Myślę, że w punkcie 11 (str. 18):

11) systematyczne wspieranie rozwoju mechanizmów uczenia się dziecka, prowadzące do osiągnięcia przez nie umiejętności samodzielnego uczenia się.

Dla dzieci na tym etapie szkolnym też przedstawiono wymagania w odniesieniu do czterech obszarów rozwojowych: fizycznego, emocjonalnego, społecznego i poznawczego. Podstawa programowa kilkakrotne odwołuje się do wykorzystania technologii w odniesieniu do tych obszarów: na str. 32, 33, 39, 41, 42, 43, 49.

Dużo jest też o umiejętności samodzielnego, refleksyjnego, logicznego, krytycznego i twórczego myślenia. Komputacyjnego? A także: na str. 44 jest zapis na temat całej edukacji informatycznej, gdzie wprost zapisane są osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Ponieważ edukacja wczesnoszkolna jest edukacją zintegrowaną więc nauczyciel po prostu połączy wszystkie zapisy w całość i będzie holistycznie uczył uczniów we wszystkich obszarach edukacyjnych. Wprowadzenie do etapu wczesnoszkolnego edukacji informatycznej, a w niej programowania i myślenia komputacyjnego jest jedną z fundamentalnych zmian całej obecnej reformy szkolnej.

W zapisach warunków i sposobu realizacji podano kilka zachęt do korzystania z narzędzi TIK, zasobów Internetu (str. 57), i najważniejsze:

Przygotowując uczniów do myślenia abstrakcyjnego w przyszłości i rozwiązywania problemów, w tym programowania, nauczyciel wykorzystuje treści wszystkich edukacji.

Teraz tylko potrzeba nauczycieli nauczania wczesnoszkolnego, którzy to wszystko dobrze zrozumieją i wprowadzą do swoich zajęć z uczniami. Edukacją informatyczną w klasach I-III nie powinni zajmować się nauczyciele informatyki, którzy przyjdą do klasy na 1 godzinę w tygodniu i wprowadzą uczniów do pracowni komputerowej.

Oby tych zapisów nikt nie zmarnował. Nauczycielu, masz okazję stać się mistrzem, jeśli dobrze wszystko zaplanujesz i wykonasz zgodnie z zapisami podstawy programowej dla edukacji wczesnoszkolnej.

4.4. II etap edukacyjny: klasy IV-VIII

Na tym etapie musimy już przeanalizować kolejne przedmioty, przy czym w opisach zadań ogólnych poszczególnych przedmiotów wprost o technologii napisano jedynie przy fizyce, że w zadania szkoły i jej funkcję wychowawczą wpisują się (str. 26):

7) uświadamianie roli fizyki jako naukowej podstawy współczesnej techniki i technologii, w tym również technologii informacyjno-komunikacyjnej.

Ważny zapis ogólny znajduje się przy informatyce, ważny dla wszystkich nauczycieli. W Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie od dawna mówimy, że programowanie to trzeci najważniejszy język współczesnego świata, po języku ojczystym i językach obcych oraz matematyce. Dlatego ten zapis w pp jest dla wszystkich nauczycieli bardzo ważny (str. 27):

Podstawowe zadanie szkoły – alfabetyzacja w zakresie czytania, pisania i rachowania – wymaga poszerzenia **o alfabetyzację w zakresie umiejętności rozwiązywania problemów z różnych dziedzin ze świadomym wykorzystaniem metod i narzędzi wywodzących się z informatyki** oraz na lepsze zrozumienie, jakie są obecne możliwości technologii, komputerów i ich zastosowań.

Elementem powszechnego kształcenia staje się również umiejętność programowania.

Programowanie jest tu rozumiane znacznie szerzej niż tylko samo napisanie programu w języku programowania. To cały proces, informatyczne podejście do rozwiązywania problemu: od specyfikacji problemu (określenie danych i wyników, a ogólniej – celów rozwiązania problemu), przez znalezienie i opracowanie rozwiązania, do zaprogramowania rozwiązania, przetestowania jego poprawności i ewentualnej korekty przy użyciu odpowiednio dobranej aplikacji lub języka programowania. Tak rozumiane programowanie jest częścią zajęć informatycznych od najmłodszych lat, **wpływa na sposób nauczania innych przedmiotów**, służy właściwemu rozumieniu pojęć informatycznych i metod informatyki. **Wspomaga kształcenie takich umiejętności, jak: logiczne myślenie, precyzyjne prezentowanie myśli i pomysłów, sprzyja dobrej organizacji pracy, buduje kompetencje potrzebne do pracy zespołowej i efektywnej realizacji projektów.**

Umiejętności nabyte podczas programowania są przydatne na zajęciach z innych przedmiotów, jak i później w różnych zawodach, niekoniecznie informatycznych.

Te ogólne zapisy powinny wystarczyć wszystkim nauczycielom do wprowadzania na swoich przedmiotach podstaw myślenia komputacyjnego i zachęcać uczniów do tworzenia programów z ich przedmiotów, ale popatrzmy jeszcze na zapisy w przedmiotach, które przede wszystkim kształtują **najważniejsze umiejętności rozwijane w ramach kształcenia ogólnego w szkole podstawowej.**

4.4.1. Język polski

W celach kształcenia – wymaganiach ogólnych podzielonych na cztery obszary, tj. kształcenia literackiego i kulturowego, kształcenia językowego, tworzenia wypowiedzi i samokształcenia – w IV obszarze samokształcenia jest zapis:

Elementem powszechnego kształcenia staje się również umiejętność programowania. Programowanie jest tu rozumiane znacznie szerzej niż tylko samo napisanie programu w języku programowania. To cały proces, informatyczne podejście do rozwiązywania problemu: od specyfikacji problemu (określenie danych i wyników, a ogólniej – celów rozwiązania problemu), przez znalezienie i opracowanie rozwiązania, do zaprogramowania rozwiązania, przetestowania jego poprawności i ewentualnej korekty przy użyciu odpowiednio dobranej aplikacji lub języka programowania. Tak rozumiane programowanie jest częścią zajęć informatycznych od najmłodszych lat, wpływa na sposób nauczania innych przedmiotów, służy właściwemu rozumieniu pojęć informatycznych i metod informatyki. Wspomaga kształcenie takich umiejętności jak: logiczne myślenie, precyzyjne prezentowanie myśli i pomysłów, sprzyja dobrej organizacji pracy, buduje kompetencje potrzebne do pracy zespołowej i efektywnej realizacji projektów.

Umiejętności nabyte podczas programowania są przydatne na zajęciach z innych przedmiotów, jak i później w różnych zawodach, niekoniecznie informatycznych.

6. Rozwijanie umiejętności efektywnego posługiwania się technologią informacyjną w poszukiwaniu, porządkowaniu i wykorzystywaniu pozyskanych informacji.

W wymaganiach szczegółowych też w tym IV obszarze:

Uczeń:

9) rozwija umiejętności efektywnego posługiwania się technologią informacyjną oraz zasobami internetowymi i wykorzystuje te umiejętności do prezentowania własnych zainteresowań.

Doskonały zapis, aby uczniom podpowiadać ważne dla nich tematy, które mogą pogłębiać, pisząc własne programy komputerowe.

W klasach VII-VIII pp dodatkowo mówi, że uczeń:

4) uczestniczy w projektach edukacyjnych (np. tworzy różnorodne prezentacje, projekty wystaw, realizuje krótkie filmy z wykorzystaniem technologii multimedialnych).

Nauczycieli języka polskiego pp (str. 70) zobowiązuje także do:

Zadaniem nauczyciela języka polskiego na II etapie edukacyjnym jest przede wszystkim:

8) kształtowanie samodzielności w docieraniu do informacji, rozwijanie umiejętności ich selekcjonowania, krytycznej oceny oraz wykorzystania we własnym rozwoju.

Wystarczy ten powyższy zapis, aby nauczyciel języka polskiego także przyczynił się do formowania myślenia komputacyjnego u swoich uczniów.

Na języku polskim i językach obcych wymaga się także od ucznia, aby potrafił zredagować sms, mail czy umiał wypowiedzieć się na czacie.

4.4.2. Historia i WOS

W warunkach i sposobie realizacji (str. 103) wprost zachęca się do tworzenia programów multimedialnych.

Zastosowane w procesie dydaktycznym różnorodne metody nauczania i środki dydaktyczne powinny być dostosowane do możliwości wiekowych uczniów oraz ich indywidualnych potrzeb. Powinny to być zarówno klasyczne metody, jak: opis, pogadanka czy wykład, jak i metody aktywizujące, oparte na działaniu, np. przygotowanie prezentacji komputerowych, zajęcia z tablicą interaktywną, **tworzenie programów multimedialnych**, filmy, praca z mapą, gry dydaktyczne, inscenizacje, przedstawienia.

Zaś dla WOS (str. 110) w warunkach i sposobach realizacji pp zaleca:

W kształceniu kompetencji pozyskiwania, gromadzenia, porządkowania, analizy i prezentacji informacji o życiu społecznym, w tym publicznym, powinna być wykorzystywana technologia informacyjno-komunikacyjna. Istotne jest korzystanie ze stron internetowych instytucji publicznych, w tym organów samorządowych, organów władzy publicznej czy organizacji społecznych. Niezbędna jest również praca z różnymi typami przekazu (np. interaktywnymi).

4.4.3. Matematyka i informatyka

W matematyce, przedmiocie najbardziej powiązanym z informatyką i programowaniem, nie ma żadnej wzmianki na temat stosowania technologii, dopiero w przykładach podanych w warunkach i sposobie realizacji (od str. 171) widać, że stosowanie technologii uważa się za coś oczywistego i naturalnego. Np. pokazując zadanie do zliczania liter w tekście, obliczenia częstości ich występowania i stworzenia diagramu.

Na II etapie edukacyjnym jest podział na przedmioty i uczniowie mają regularnie prowadzone lekcje informatyki. W pp tego przedmiotu nauczyciel jest zobowiązany do rozwiązywania problemów i pisania programów z życia codziennego i z różnych przedmiotów. Wystarczy, że nauczyciel zajrzy do pp z matematyki i właściwie w każdym punkcie treści nauczania – wymagania szczegółowe – odnajdzie pomysł, temat na napisanie programu. Mogą to być programy dotyczące liczb, ułamków, obliczania NWD, geometrii na płaszczyźnie czy przestrzennej. Na początek te zagadnienia, które zna nauczyciel i wie, jakie programy może stworzyć uczeń. Potem tych tematów będzie przybywać, zresztą nauczyciel matematyki też może sporo podpowiedzieć. Wystarczy, że nauczyciele będą ze sobą współpracować i dzielić się pomysłami na to, co mogą pod ich kierunkami zrobić uczniowie.

Podsumowanie

Zapisy dotyczące używania TIK można odnaleźć w pp. Także w małym zakresie propozycję pisania programów.

Najważniejszy wydaje mi się jest zapis w pp nauczania wczesnoszkolnego wyodrębniający edukację informatyczną. W podstawie tej zapisane są wprost osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Jeśli będą uczyć uczniów dobrze przygotowani nauczyciele, to sami uczniowie „przeniosą" tę edukację (informatyczną) do następnych etapów. Oni będą cieszyć się z tworzenia własnych programów i sami będą zachęcać swoich nauczycieli do zadawania im prac z tym związanych. A nauczyciele tylko muszą realizować prawidłowo zapisy nowej podstawy programowej i wiedzieć, że `programowanie, które jest częścią zajęć informatycznych od najmłodszych lat`**, wpływa na sposób nauczania innych przedmiotów.**

Myślę, że przed systemem edukacji w Polsce stoi **zadanie przygotowania wszystkich nauczycieli do zmierzenia się z edukacją 2.0** (na razie), a resztę „wymuszą" sami uczniowie i upływający czas. Dlatego jestem w tej sprawie optymistką.

Bibliografia

1. Kędracka-Feldman E., Rostkowska M. *Nowa podstawa programowa z informatyki szansą na zmianę dydaktyki wszystkich szkolnych przedmiotów*, Informatyka w Edukacji, Toruń 2016.

2. Rostkowska M. *Poszukiwanie programowania i myślenia komputacyjnego w innych niż informatyka przedmiotach w nowej podstawie programowej.* Wystąpienie podczas konferencji Informatyka w Edukacji, Toruń 2017.

3. Rozporządzenie Ministra Edukacji Narodowej w sprawie podstawy programowej wychowania przedszkolnego oraz podstawy programowej kształcenia ogólnego dla szkoły podstawowej..., Dziennik Ustaw Rzeczypospolitej Polskiej, 24 lutego 2017 roku, http://www.dziennikustaw.gov.pl/DU/2017/356.

Małgorzata ROSTKOWSKA jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Konstruktywistyczne spojrzenie na naukę programowania w szkole podstawowej

Tomasz ŁUKAWSKI

Wrzesień 2017 to dla polskiej szkoły nie tylko zmiany związane z reformą oświaty. Środowiska od lat zajmujące się promowaniem rozwoju technologii informacyjnych w szkołach z nadzieją oczekują trwałej zmiany, którą ma przynieść nowa podstawa programowa przedmiotu informatyka. Wbrew rozpowszechnianym opiniom o szybkim tempie powstawania podstaw programowych trzeba podkreślić, że tekst podstawy do informatyki tworzono przez około dwa lata. Proces był opatrzony szeregiem często burzliwych dyskusji i wywoływał wiele emocji w środowisku informatyków, związanych z tematem edukatorów i profesorów akademickich. Niewątpliwie najważniejszą oczekiwaną zmianą jest powszechne wprowadzenie programowania już na wczesnym etapie szkoły podstawowej. Spotkałem się z wieloma często sprzecznymi opiniami na temat nauki programowania w klasach początkowych. Najbardziej skrajne dotyczyły wręcz krytyki pomysłu i oddalenia procesu do obecnej klasy VII szkoły podstawowej. W niniejszym artykule będę wyrażał oraz bronił idei nauczania programowania od jak najwcześniejszych lat z zastrzeżeniem użycia przy tym odpowiednich metod uczenia się oraz właściwej postawy nauczyciela, który nie będzie przekazywał wiedzy, lecz stwarzał sytuacje pedagogiczne skłaniające do interdyscyplinarnego modelu nauczania, włączając w to naukę programowania. W celu poparcia tej tezy przytoczę idee konstruktywizmu krytycznego na podstawie jednego z pierwszych w Polsce opracowań tej teorii opisanej przez profesora Stanisława Dylaka oraz nowego modelu szkoły Taylora. Odwołam się do psychologów Jerome'a Brunera i tworzenia kultury klasy Douglasa Barnesa. Opiszę praktykę zastosowaną podczas pilotażu programowania w Szkole Podstawowej Nr 3 im. Małego Powstańca w Ząbkach, gdzie koordynowałem ten proces.

Nauczyciel edukacji wczesnoszkolnej czy nauczyciel informatyki?

W dyskusjach, często burzliwych, o których wspomniałem we wstępie, padało zawsze pytanie: kto ma uczyć programowania w klasach I-III? Moim zdaniem na pewno nie nauczyciel informatyki uczący w klasach starszych. Wielokrotnie broniłem zdania, że nauczyciel edukacji wczesnoszkolnej najlepiej zna metodykę nauczania zintegrowanego, posiada niezbędne kompetencje do pracy z maluchami. Oczywiście spotkałem się z oporem samych nauczycieli, że „przecież my nie potrafimy programować". Wówczas zaprosiłem ich na portal `www.code.org`, mówiąc, że tam jest wszystko, czego potrzebują. Platforma jest zorganizowana w sposób, który pozwala na założenie kont uczniom i utworzenia klasy, gdzie nauczyciel może w czasie rzeczywistym śledzić postępy uczniów. Jednak tym, co skłoniło mnie i „moich" nauczycieli edukacji wczesnoszkolnej do szerokiego wykorzystania tego

internetowego portalu, jest interaktywne sterowanie procesem przyswajania wiedzy ucznia. Jednym z początkowych zadań, które ma wykonać uczeń, jest np. doprowadzenie do celu postaci z ekranu komputera za pomocą wydawanych poleceń przez ucznia. Polecenia są blokami do wyboru typu: „idź krok do przodu", „skręć w prawo". Jeżeli uczeń użyje bloku, który nie pasuje do algorytmu, komputer zasugeruje podpowiedź. Pierwsze kursy nie używają nawet napisów, tylko polegają na łączeniu obrazków jak w grze puzzle. Nauka staje się więc wspaniałą zabawą z komputerem, a nauczyciel jest wciągany w zabawę razem z uczniem, co wytwarza specyficzną miłą atmosferę.

Profesor Sugata Mitra, który prowadził badania w ubogich dzielnicach Indii, montując w slumsach komputery, w swoich podsumowaniach doszedł do wniosku, że rola nauczyciela w takim procesie powinna ograniczyć się do funkcji motywacyjnej i inspirującej ucznia. Sugata wręcz sugerował, aby na pytanie ucznia, jak to zrobić, nie odpowiadać, nawet jeśli znamy odpowiedź. Powinniśmy pierwszakowi odpowiedzieć, że jest tak zdolny, że na pewno sam znajdzie rozwiązanie po sugestii programu komputerowego. Wydaje się, że problemem nie jest nieumiejętność programowania przez nauczyciela, lecz nieumiejętność przyznania się do tego uczniowi. To jest bariera, którą najtrudniej zlikwidować.

Douglas Barnes, badając kulturę porozumiewania się nauczyciela z klasą, już w latach 70. zwracał uwagę, że nauczyciele pragną wykładać wiedzę i to daje im poczucie spełnienia i wypełniania obowiązku wobec ucznia i pracodawcy. Poza tym posiadanie specjalistycznej wiedzy i posługiwanie się terminologią czerpaną z danej dyscypliny daje nauczycielowi poczucie podniesienia własnej wartości. Nie to jednak jest istotne w pracy pedagoga, aby górować kompetencjami nad dziećmi. Proponuję odstąpić od tego archaizmu i pobawić się z klasą w programowanie. Podczas pracy na portalu `code.org` wkład nauczyciela powinien ograniczyć się do pomocy przy zalogowaniu na konto. Nauczyciele w Szkole Podstawowej nr 3 w Ząbkach rozwiązali problem logowania się pierwszaków na swoje konto w `code.org` przez utworzenie kodów QR. Uczniowie otrzymują karteczki z kodem i po najechaniu tabletem ustawionym na skanowanie QR automatycznie są przekierowywani do wybranego kursu. Wspomniałem o kulturze porozumiewania się nauczyciela z uczniami. Nauczyciel klas starszych wprowadzony do edukacji wczesnoszkolnej może być narażony na brak wykonywania przez uczniów jego poleceń, co jest związane ze specyfiką wieku uczniów, którzy na tym etapie chcą być w centrum zainteresowania i swoimi pytaniami odbiegają od stosowania się do reguł narzucanych przez nauczyciela, do których stosowania pedagog jest przyzwyczajony. Rodzi to ryzyko próby zaprowadzenia przez nauczyciela dyscypliny w klasie poprzez bardziej autorytarny i przedmiotowy sposób potraktowania ucznia. Postawienie nauczyciela w roli wykładającego wiedzę, a ucznia w roli wyposażanego w nią, którego obowiązkiem jest tę wiedzę przyswoić, buduje generalną zasadę autorytarnego modelu nauczania-uczenia się. Stawianie nauczyciela na pozycji dominującej generuje swoisty sposób komunikacji nauczyciel-uczeń-model porozumiewania się, który ma duży wpływ na uczenie się.

Douglas Barnes w książce „Nauczyciel i uczniowie-od porozumiewania się do kształcenia"[1] przedstawia pewne sposoby posługiwania się mową stosowane przez dzieci w trakcie uczenia się oraz wskazuje, jak jest to uzależnione od systemów porozumiewania się, które nauczyciele wprowadzają w swych klasach. Po pierwsze zachowanie nauczycieli (porozumiewanie się) wpływa na naukę ich uczniów, po drugie przedmiotem zainteresowania powinien być **proces kształcenia** oraz właściwe sposoby rozpatrywania go. Dlatego metoda nauki programowania w sposób podający wiedzę, która ma do dzisiaj wielu wyznawców wśród nauczycieli szkolnych, nie sprawdzi się w klasach początkowych szkoły podstawowej. Ważniejszy jest proces kształcenia niż treści. Nie tyle jest ważne, czego się naucza, ale w jaki sposób się to robi. Proponuję więc podejście interdyscyplinarne do procesu nauczania programowania. Wraz z nauczycielkami edukacji wczesnoszkolnej Szkoły Podstawowej nr 3 w Ząbkach prezentowaliśmy szereg otwartych lekcji na konferencjach w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie,

[1] Barnes Douglas, *Nauczyciel i uczniowie – Od porozumiewania się do kształcenia*, ang. From communication to curriculum, WSiP 1975; Warszawa 1988, wyd. 1, s. 3.

na seminarium dla studentów pedagogiki Uniwersytetu Warszawskiego oraz na konferencji Model Nowoczesnej Szkoły w Ząbkach. Jednym z przykładów takich zajęć jest lekcja z planszą, na której uczniowie klasy pierwszej wcielają się w postać pszczółki zbierającej nektar na miodek. Taka pszczółka występuje w jednym z kursów na platformie code.org. Zanim więc młodzi programiści wykonają zadanie na swoim tablecie, są podzieleni na dwie drużyny siedzące po dwu stronach planszy. Przed sobą mają rozłożone wydrukowane przez nauczyciela bloki do programowania ze strzałkami „idź do przodu", „skręć w prawo" itd. Jedno z dzieci z nakryciem głowy imitujące owada staje na środku planszy, a nauczyciel umieszcza kwiatki zgodnie z ekranem komputera wyświetlonym na tablicy multimedialnej. Zadaniem grup jest jak najszybsze, w formie rywalizacji, ułożenie schematu blokowego z wydrukowanych kartek – algorytmu, który doprowadzi ucznia przebranego za pszczółkę do kwiatków i zebrania nektaru. Następnym etapem jest zeskanowanie kodu QR w celu zalogowania się do kursu i wykonanie zadania w internetowej aplikacji. Zajęcia, o których mowa, są wkomponowane w blok edukacji przyrodniczej, gdzie już odbyły się lekcje o roli pszczół, wartościach miodu i jego przetworach, ekologii i ochronie środowiska, są więc elementem szerszego interdyscyplinarnego procesu kształcenia.

W teorii konstruktywistycznego krytycyzmu, zdaniem Taylora, nauczyciel powinien uczyć się razem z uczniem. Prowadzenie nauki programowania przez nauczyciela uczącego w klasach I-III wychodzi naprzeciw konstruktywistycznemu modelowi nowej szkoły, który różni się znacznie od tradycyjnego. Największe różnice dotyczą kwestii rezygnacji z przekazywania wiedzy przez nauczyciela na rzecz konstruowania wiedzy przez każdego ucznia z osobna. *Procesy nauczania zejdą na drugi plan wobec procesów uczenia się. Nauczyciel będzie musiał zrezygnować z dotychczasowej władzy w klasie szkolnej na rzecz objęcia nowych funkcji: mentora, doradcy, partnera ucznia, a nawet ucznia. Szansy odgrywania przez ucznia aktywnej roli w procesach uczenia się, uzależnionej od własnych potrzeb i zainteresowań. Jego intelektualna aktywność będzie się przejawiała w poszukiwaniu nowych informacji, tworzeniu wiedzy, pracowaniu nad projektami badawczymi, nabywaniu niezbędnych umiejętności intelektualnych, by móc z powodzeniem stosować wiedzę w nowych sytuacjach i kontekstach. Będzie odpowiedzialny za swoje uczenie się, aczkolwiek kierunek nauki może wytyczać nauczyciel*[2]. Profesor Stanisław Dylak w opracowaniu „Konstruktywizm jako obiecująca perspektywa kształcenia nauczycieli"[3] powołuje się na Jerome'a Brunera i Wygotskiego: *(...) w kontekście właśnie twierdzeń konstruktywizmu. Według Piageta, jeżeli świat nie może być poznawany bezpośrednio, ale tylko przez pośrednictwo operacji logicznych, nasza wiedza jest konstrukcją, konstrukcją, która ma być testowana w działaniu – zarówno przez skuteczność działania, jak i rozumienie świata* (Bruner, 1996)[4]. Dylak tłumaczy, że *Ludzie uczą się w interakcji z otoczeniem, aktywnie konstruują własną wiedzę, wykorzystując wiedzę już posiadaną. Nie rejestrują informacji, ale budują struktury wiedzy z dostępnych informacji (...). W konsekwencji konstruktywizm akcentuje proces, w wyniku którego uczący się tworzą i rozwijają własną wiedzę. Jedną z najpoważniejszych konsekwencji praktycznych takich założeń jest dyrektywa, że w tworzeniu programów nauczania bardziej niż na dostosowanie ich do możliwości uczniów należy dbać o to, aby były one wyzwaniem dla dotychczasowego rozumienia świata*[5]. Profesor Dylak podaje dwa zasadnicze twierdzenia epistemologiczne konstruktywizmu. Po pierwsze, że wiedza jest aktywnie konstruowana przez podmiot poznający; po drugie, że dochodzenie do wiedzy jest procesem adaptacyjnym, w którym następuje organizacja doświadczanego świata. Idąc tym tropem – uczenie programowania w klasach początkowych prowadzone w sposób znany z tradycyjnego systemu szkolnego skupionego na przekazywaniu wiedzy stwarza zagrożenie zwłaszcza u maluchów, związane z brakiem odniesienia przekazywanej wiedzy do wiedzy czynnej uczniów, o której piszą Douglas Barnes oraz psycholog Jerome Bruner. *Każdy uczeń interpretuje słowa nauczyciela w kategoriach posiadanej już wiedzy*[6].

[2] http://www.ciccum.pl/technologia-a-oswiata/konstruktywizm-krytyczny

[3] Kwiatkowska H., Lewowicki T., Dylak S. [red.] *Współczesność a kształcenie nauczycieli*, WSP ZNP, Warszawa 2000.

[4] Op. cit., s. 1.

[5] Op. cit., s. 3.

[6] Barnes Douglas, op. cit., s. 21.

Uczniowie wynoszą z lekcji część wspólnej wiedzy oraz część interpretowaną na bazie swoich dotychczasowych doświadczeń spoza szkoły. Uczenie to nie nowe cegiełki w murze. Wg Piageta i Brunera wiedza to systemy służące do interpretacji świata, więc uczenie się programowania to zmiana systemu interpretowania. Percepcja rzeczywistości-wiedza-to oddziaływanie schematów interpretacyjnych na dane odbierane zmysłami. Dzieci w wieku 7-8 lat nie posiadają umiejętności przyswajania wiedzy tak abstrakcyjnej, jak algorytm czy program. Stąd przebieranki w pszczółki, chodzenie po macie, rywalizacja między grupami o to, kto stworzy prawidłową drogę dojścia owada do kwiatka. Pominięcie tej fazy u dzieci w tym wieku nie pozwoli na skompilowanie nowej podawanej przez nauczyciela wiedzy z już posiadaną, gdyż tej drugiej brak. Ogólnie przekazywanie wiedzy uczniom nawet z klas starszych to teoria archaiczna, niemająca miejsca we współczesnej szkole oraz epoce informacji, gdzie informacja i wiedza (teoretyczna) są dostępne „na żądanie". Zasoby wiedzy i umiejętności nabytych przez uczniów poza szkołą i w Internecie są konfrontowane z wiedzą dostarczaną przez nauczycieli. To daje wyraz twierdzeniu, że współczesna szkoła z metodą wykładową wydaje się być przeżytkiem. Dostęp do wiedzy z sieci, która jest już przetworzona przez uczniów, bardzo często stawia nauczyciela wykładającego w niezręcznej sytuacji, jeżeli będzie on sądził, że tylko on zna wyjaśnienia dla omawianych zjawisk. Należy postawić pytanie: Jak uczyć, gdy wiedza nie jest już tylko w głowach nauczycieli, lecz gdy jest w Internecie na kliknięcie? Coraz częściej używając określenia „wiedza", myślimy również o umiejętnościach. Myślę, że czas już coraz częściej zastępować w szkole twierdzenie o „wyposażaniu w wiedzę" na „wyposażanie w umiejętności".

Moim zdaniem, co sprawdziło się w SP3 w Ząbkach, metodami wychodzącymi naprzeciw oczekiwaniom dzisiejszych wyzwań są **aktywne zadania prowadzące do zdobywania nowej wiedzy i umiejętności, bazujące na wiedzy i umiejętnościach posiadanych już przez uczniów. Jest to charakterystyczne „odwrócenie". Proces zaczyna się od tyłu, tzn. od zadań bazujących na samokształceniu, poprzez poszukiwanie informacji i sposobów rozwiązań postawionych przez uczniów problemów; dalej przez działanie w zespole, grupie zadaniowej, aż do przekodowania wiedzy czynnej ucznia do nowej jakości zalokowanej w intelekcie, przekonaniach i wartościach ucznia, ukształtowanych po zakończeniu projektu czy bloku tematycznego**.

Rolą nauczyciela jest zaplanowanie procesu czasami metodami zaczerpniętymi z improwizacji i ustanowienie w klasie takiej kultury systemu porozumiewania się, która będzie sprzyjała spontanicznemu odkrywaniu zjawisk przez uczniów oraz rozbudzaniu ich ciekawości świata. Tu szczególną rolę pełnią projekty interdyscyplinarne, do których włączenia uczenia programowania chciałbym zachęcić. W projekcie „Zakazane piosenki", zaprezentowanym na ogólnopolskiej konferencji Model Nowoczesnej Szkoły, który dotyczył w większości wiedzy z zakresu historii, sztuki, literatury, uczniowie pod kierunkiem nauczycieli Szkoły Podstawowej nr 3 im. Małego Powstańca w Ząbkach zaprogramowali małego robota, który przebrany za turystę jeździł po makiecie Starego Miasta, wskazując ukryte znaczniki rozszerzonej rzeczywistości. Projekt można zobaczyć na stronie konferencji `www.modelnowoczesnejszkoly.pl` w zakładce „materiały". Stawiam nacisk, przedstawiając tę ideę, na społeczne zaangażowanie programujących robota uczniów, którzy uczestniczą w procesie, mając przed sobą wyznaczony cel, do którego zmierzają. Jest to szczególnie istotne w motywowaniu dochodzenia do wiedzy. Interdyscyplinarne podejście do lekcji programowania może być na przykładzie szkolnych projektów przeniesione z klas początkowych na starsze, gdzie uczniowie więcej czynności wykonują samodzielnie.

Wracając do roli nauczyciela klas młodszych, dodatkowo sugeruję zwrócić uwagę na możliwość stwarzania sytuacji uczenia się bez obecności nauczyciela i samodzielną działalność uczniów, co jest jak najbardziej możliwe w pracy na platformie `code.org`. Jest to tym bardziej istotne w określaniu nowej roli nauczyciela jako stojącego w roli obserwatora. W tym miejscu warto przytoczyć badania Sugata Mitra dotyczące roli nauczyciela jako tzw. instytucji babci, motywującej do samodzielnego działania, poszukiwania, interpretowania. „Babcia" mówi: „Ja nie potrafię tego zrobić, ale Ty na pewno dasz radę, gdyż jesteś zdolny i fantastyczny. Uwierz

kochanie, że możesz wszystko. Może poszukaj w Internecie instrukcji lub podpowiedzi?". W klasach III szkoły podstawowej jest to jak najbardziej możliwe do wykonania. Demokratyczny i swobodny sposób komunikowania się z nauczycielem jest podstawą do rozwoju u uczniów kreatywności, innowacyjności rozwiązań i krytycznego myślenia, określanych jako tak istotne kompetencje XXI wieku. Jeden z prekursorów polskiej myśli konstruktywistycznej, cytowany wcześniej profesor Stanisław Dylak, tłumaczy, że *(...) taka perspektywa ujmowania nauczania i uczenia się akcentuje aktywność uczącego się, w wyniku której podmiot buduje swą rzeczywistość (...). Uczący się aktywnie konstruują własną wiedzę, a nie przyswajają ją jako przekazaną przez nauczycieli, gdyż ludzie nie są rejestratorami informacji, ale budowniczymi struktur własnej wiedzy (...). Taki jak wyżej sposób myślenia o uczeniu się korzeniami swymi sięga myśli Deweya, Wygotskiego, Bartletta i Kelly'ego teorii konstruktów osobistych. Nurt ten jest wyzwaniem dla dominującego obecnie w uczeniu się i nauczaniu „osiągania wiedzy" oraz dla związanego z tym obiektywizmu*[7], tj. przekazywania czy wykładania informacji.

Jak oceniać z programowania na lekcjach informatyki?

Zanim programowanie wprowadzono jako obowiązkowe, było ono domeną najlepszych uczniów, którym stawialiśmy ocenę celującą. Od września 2017, kiedy stało się obowiązkowe dla wszystkich uczniów, nauczyciele będą wymagali od uczniów wiedzy z programowania, a dyrektorzy od nauczycieli realizacji podstawy programowej. Wiąże się z tym niebezpieczeństwo utraty zainteresowania i pasji do tej dyscypliny i w ogóle przedmiotu informatyka. Model szkoły, która ma uczyć kreatywności, innowacyjności, a przede wszystkim pasji do programowania i ciekawości rozwiązań algorytmicznych, powinien opierać się w szczególności na:

1. Roli i znaczeniu funkcji nauczyciela.
2. Twórczej, samodzielnej pracy uczniów.

Bardziej niż wyniki pracy własnej ucznia odniesione do programu nauczania należy doceniać i podkreślać proces dochodzenia do tych wyników. Twórczy, oryginalny, kreatywny niczym nieskrępowany innowacyjny pomysł nie może być z góry oceniony jako zły. Stawianie ocen niedostatecznych bez możliwości ich poprawienia również uczy, że nie wolno uczniowi popełniać błędów. Współczesna szkoła, jeżeli ma dostarczać na rynek pracy absolwenta kreatywnego, myślącego krytycznie, innowacyjnego lidera i przywódcę, musi odejść od rygorystycznego i podsumowującego systemu oceniania, jaki mamy dzisiaj w szkołach. Systemy oceniania, klasówki, egzaminy i sprawdziany wprowadzają rygor i dyscyplinę, równając wszystkich uczniów w dół. Systemy oceniania w szkołach budowane przez nauczycieli często stwarzają uczniowi sytuację, w której nie warto się starać, gdyż i tak nie otrzyma się lepszej oceny. Mam tu na myśli wewnątrzszkolne systemy oceniania punktowego czy wyliczania średniej arytmetycznej, które nadal funkcjonują w przedmiotowych systemach oceniania, mimo wytycznych w rozporządzeniu o ocenianiu, aby tego nie czynić i **oceniać postępy uczniów**. Nie jest to egzekwowane przez dyrektorów albo z powodu milczącej zgody albo z braku ich kompetencji. To dotyczy również oceniania opisowego w klasach młodszych, w którym w wielu przypadkach nauczyciele stosują rozmaite symbole, buźki, słoneczka oraz cyferki, co jest wbrew ogólnej zasadzie opisywania zachowania i postępów dziecka. W wyniku takiego podejścia uczniowie uczą się dla oceny, a rodzice oczekują najlepszych ocen, nie zważając na to, w jaki sposób zostały one wystawione i jak je osiągnęło ich dziecko. Często prowadzi to do patologii szkolnych i „milczących zgód" na społeczne nieprawidłowości.

Pozostawienie uczniów samych sobie bez roli nauczyciela jako nakierowującego na zawartość celów z programu nauczania otwiera uczniów na rozwiązania, na które nauczyciel nie jest przygotowany. Rozwiązania te, często kreatywne, nowe, innowacyjne powinny być oceniane (a raczej doceniane: **proponuję zmienić nazwę z „systemu oceniania" na „system doceniania"**) w szkole, w której większą wagę przywiązuje się do tzw. umiejętności XXI wieku nauczyciel powinien obserwować i wspierać kształtowanie podziału ról w grupach dokonywanego przez samych uczniów. Powinniśmy jako nauczyciele doceniać przywództwo, relacje społeczne w grupie roboczej, prezentację

[7] Op. cit., s. 4.

przed audytorium wyników pracy grupy. Proponuję nie oceniać uczniów z wyników realizacji programu nauczania ani wyników wyuczonej wiedzy szkolnej. Jeśli już, to oceniać (czytaj: doceniać) tło społeczne realizacji celów oraz praktyczne zastosowanie wiedzy już przez ucznia przetworzonej i przyjętej jako „wiedza czynna". Wiedzę czynną uczniowie prezentują, wyjaśniając zjawiska. Jeżeli uczeń potrafi wyjaśnić, znaczy, że przyswoił i jest to już wiedza ucznia, tak zwana przez Brunera „mowa ostateczna". Należy stwarzać uczniom zadania do wykonania i projekty do wykonywania wspólnie. Nauczyciele powinni obserwować, jak uczniowie planują pracę nad projektem, rozdzielają role, jak zaangażowanymi wolontariuszami są oraz jak się samorealizują podczas pracy. Poprzez wykonywanie umiejętnie zaplanowanych zadań kształtujemy kulturę komunikacji, porozumiewania się z nauczycielem-obserwatorem udzielającym czasami instrukcji oraz dostarczającym uczniom motywacji. Poprzez zadania uczniowie nabywają umiejętności, a poprzez umiejętności – wiedzę, na zasadzie przekodowywania wiedzy, którą już posiadają.

Jest to model dynamicznego przyswajania informacji, gdyż zasób wiedzy i umiejętności jest budowany na bazie istniejących predyspozycji i na bieżąco modyfikowanych i przekodowywanych pod wpływem umiejętności nabywanych wykonywaniem zadań. Zadania powinny zawierać „zaangażowanie społeczne" i być oparte na relacjach grupy uczniów je wykonujących.

Jak znaleźć złoty środek?

Dotychczas uczenie programowania było pasją, dotyczyło chętnych uczniów i często odbywało się na zajęciach pozalekcyjnych. Od tego roku z nadzieją oczekuję, że wprowadzenie obowiązkowej nauki programowania nie zdusi pasji nauczycieli do uczenia, a uczniów do programowania. Nie jest możliwe znalezienie złotego środka, dlatego duży nacisk należy położyć na **improwizację i artyzm**, rozbudzanie ciekawości świata. Posłużę się analogią do produkcji filmu. Nauczyciel powinien przyjmować różne role „planu filmowego", w zależności od zaplanowanych zadań dla uczniów zawsze prowadzących do określonego w długoterminowym planie celu. Nauczyciel-aktor jednego dnia będzie odgrywał główną rolę, a innego rolę drugo lub trzecioplanową. Jednego dnia będzie reżyserem, innego operatorem lub montażystą scen odegranych i wyreżyserowanych przez uczniów.

1. Dlatego też planowanie uczniowi przez nauczyciela następnego etapu zadań należy rozpocząć od ewaluacji (nie mylić ze sprawdzianami i klasówkami) wiedzy czynnej ucznia, tej przekodowanej z wiedzą szkolną z ostatnich zadań, aby ustalić za każdym razem poziom startowy do dalszego etapu.

2. Dlatego również nie używam słowa „nauczanie", lecz „uczenie się razem z uczniem", gdyż nie wiem, co przyniesie następna lekcja, jakie inspiracje, jakie nastroje uczniów, jakimi wydarzeniami będzie żył świat, do których będę się mógł odnosić, inspirując uczniów.

3. Dlatego jeszcze muszę dbać o kulturę porozumiewania się z moimi uczniami, aby panowała atmosfera wzajemnego zaufania.

4. Dlatego w końcu nie chcę „przyłapać swoich uczniów na czymś złym lub na błędzie" – **pragnę „przyłapać ucznia na czymś dobrym i się tym zachwycić na jego oczach" – również jako dyrektor w stosunku do nauczycieli**.

Tomasz ŁUKAWSKI jest dyrektorem Szkoły Podstawowej nr 3 im. Małego Powstańca w Ząbkach.

Programujemy już dziesiątki lat...

dr Jan A. WIERZBICKI

W ostatnich kilku latach można zauważyć silny trend związany z rozpowszechnianiem nauki programowania. Ta sytuacja skłoniła mnie do kilku refleksji. Nasuwa się naturalne pytanie, czy programowanie to jakaś nowość, czy dopiero teraz zaczynamy uczyć programowania? Te pytania można rozszerzyć do pytania ogólniejszego, dotyczącego informatyki.

Często na wielu kursach i zajęciach, jakie prowadzę ze studentami, uczniami i nauczycielami zadaję pytanie: Ile lat ma informatyka? Prawie zawsze udzielane odpowiedzi to: dwadzieścia lat, trzydzieści lat, około czterdziestu lat itp. Nikt nawet nie pomyśli o większej liczbie. Informatyka wiązana jest przez ogół osób z bezpośrednim użyciem komputerów i technologii. Czy słusznie? Moim zdaniem – zdecydowanie nie! Dlatego, że informatyka to nauka ścisła, korzeniami sięgająca starożytności. W wielu zapiskach starogreckich, starorzymskich mamy do czynienia z jednym z wyróżników informatyki, jakim jest algorytm. Od algorytmów opisujących na przykład proste sposoby obliczania do klasycznych algorytmów, takich jak algorytm Euklidesa czy sito Eratostenesa.

Informatyka jest związana z przetwarzaniem różnych danych i informacji. Komputery i obecna technologia są narzędziami opierającymi się na informatyce. Informatyki nie można, co niestety bardzo często się dzieje, mylić z technologiami informacyjno-komunikacyjnymi (TIK). Technologie informacyjno-komunikacyjne to „produkty" informatyki do wykorzystania zarówno w pracy zawodowej, jak i w życiu codziennym.

Jeśli informatyka jest tak starą nauką, to czy programowanie jest też tak stare?

Musimy znowu zadać pytanie – co to jest programowanie?

Programując, wydajemy komputerowi lub innemu urządzeniu stosowne polecenia, które ma on wykonać. Aby skutecznie wykonać zadane polecania, muszą być przemyślane i wykonane w odpowiedniej kolejności. Ten sposób pracy to nic innego jak algorytm.

Algorytmy, tak jak już wspomniałem, znane są od starożytności, a właściwie od początku istnienia ludzkości, bowiem nawet ludy pierwotne miały swoje metody polowań czy upraw.

Sam algorytm możemy tworzyć i testować bez komputera. Programowanie samo w sobie wymaga już zastosowania jakiegoś urządzenia, w którym opracowany program ma działać. Programy mogą wyglądać różnie. Sięgając w przeszłość – na przykład mechaniczne maszyny liczące też miały wbudowany program, tyle że oparty na mechanizmie różnych zębatek.

Po tym krótkim przeglądzie możemy już stwierdzić, że programowanie nie jest elementem nowym. Może więc jest czymś nowym w obecnej dydaktyce? Rzeczywiście, w nauczanym realnie materiale na lekcjach, nawet tych, które w nazwie miały informatykę, programowanie nie było wprowadzane dość powszechnie. Raczej uczono TIK.

Można zauważyć, że w bardzo wielu przypadkach osoby świetne w pracy z TIK mają małe doświadczenie w programowaniu. Wiążą się z tym problemy z myśleniem abstrakcyjnym oraz znajomością podstaw matematyki czy innych dziedzin. Narysowanie kwadratu, które wymaga podania jego właściwości, takich jak liczba boków i miara kąta wewnętrznego to zadanie czasami prawie niewykonalne przez ucznia. Dlatego jest tak ważne, aby od samego początku edukacji uczyć elementów programowania, które mogą i zazwyczaj wykorzystują wiele pojęć z różnych przedmiotów. Programowanie uczy logicznego myślenia, kreatywności, co jest niezwykle istotne w kształceniu każdego ucznia.

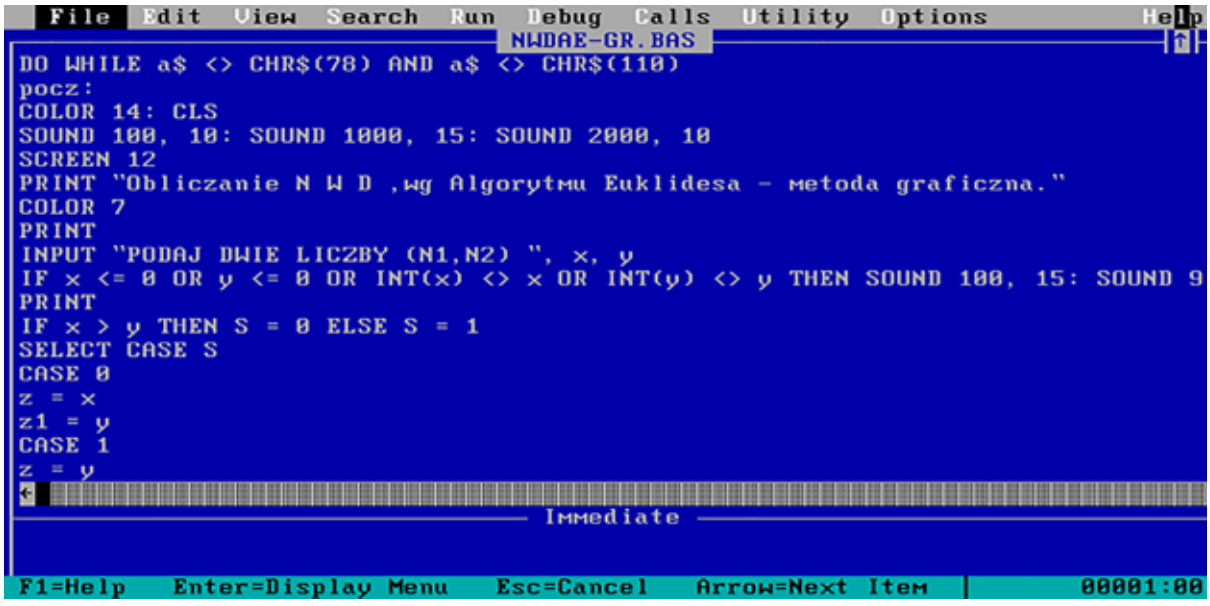
Opiszę teraz moje bezpośrednie doświadczenie z programowaniem wskazujące na to, że czasami potrzeba bardzo mało, aby nauczyć się bardzo wiele. Mój pierwszy komputer, który miałem w połowie lat 80. ubiegłego wieku, to był Commodore 16 (nie popularny wówczas Commodore 64). Komputer ten nie był zatem zbyt popularny, w szczególności w Polsce, więc praktycznie nie było do niego żadnego oprogramowania, posiadał tylko wbudowany interpreter języka programowania Basic. Można zadać pytanie, czy w takiej sytuacji czegoś się nauczyłem, pracując na tym komputerze?

Odpowiedź brzmi – bardzo dużo! Wszystko, co chciało się mieć, trzeba było samemu zaprogramować. Programowanie było nierozerwalnie związane z nauką i ugruntowywaniem wiedzy z różnych przedmiotów szkolnych. Budując program, uczymy się symulacji procesów i pojęć z różnych dziedzin. Programując nawet zabawkę-grę, uczymy się na przykład geometrii czy trygonometrii.

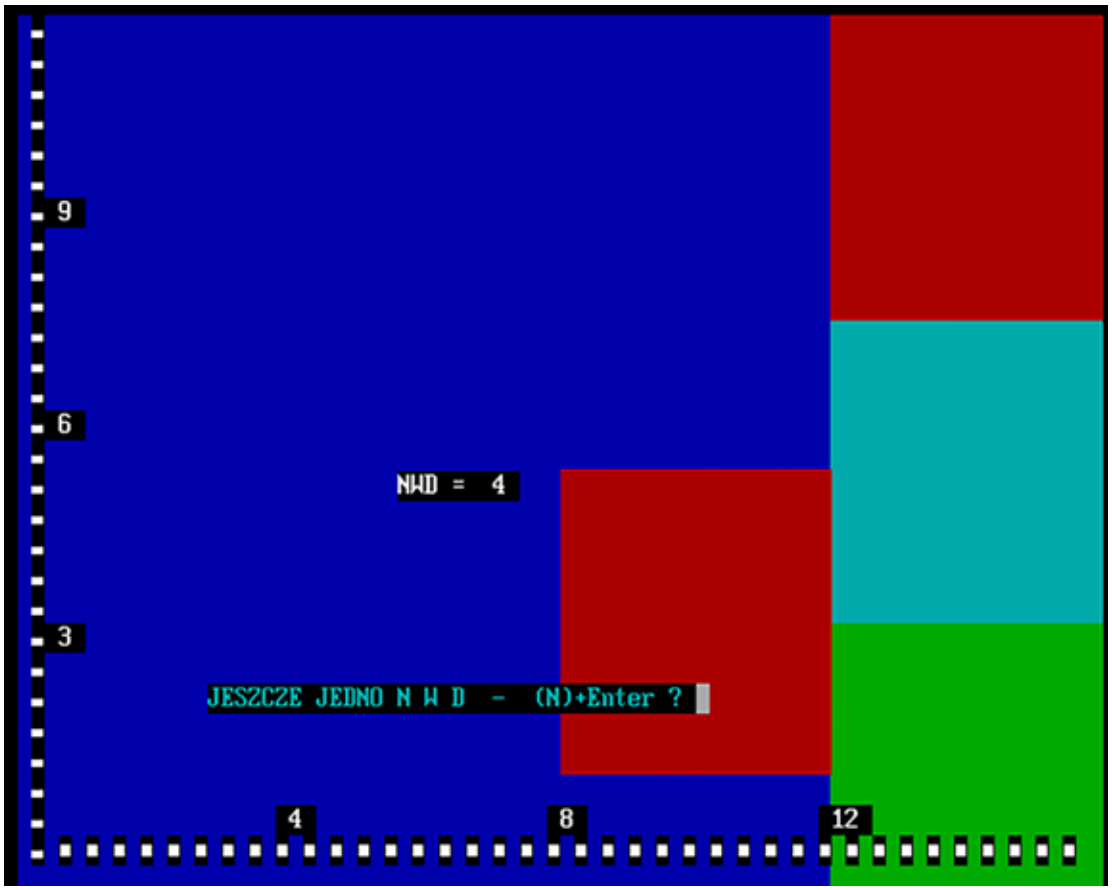
Napisałem mnóstwo programów – od gier do baz danych – co pozwoliło mi się nauczyć i ugruntować bardzo wiele zagadnień. Frajdę miałem przy tworzeniu programu, np. gry, a samo granie nie było już takie atrakcyjne.

Kolejne programy dydaktyczne napisane już na innych komputerach przedstawiały i obrazowały tradycyjne algorytmy, np. algorytm Euklidesa, badanie wykresów funkcji, metody sortowania, badanie ciągów, wizualizacje pojęć symetrii, jednokładności, działania na wektorach. Programy analizowały też teksty literackie, szukając palindromu lub anagramu.

Dzisiaj programy napisane pod dostępnym ponad 20 lat temu systemem operacyjnym DOS da się nawet uruchomić na najnowszych systemach operacyjnych stosując oprogramowanie do tworzenia maszyn wirtualnych (rys. 1, 2).



Rysunek 1. Fragment kodu programu algorytmu Euklidesa w języku Basic.



Rysunek 2. Interpretacja graficzna algorytmu Euklidesa – efekt działania programu w języku Basic.

Opisane przykłady świadczą, że nauka informatyki zawsze obejmowała jako jeden z kluczowych elementów naukę programowania. Jak można łatwo zauważyć, wiele pojęć przywoływanych w ramach uczenia algorytmiki i programowania jest ponadczasowych. Algorytmy i programy napisane kilkadziesiąt lat temu nadal obowiązują i będą obowiązywać w przyszłości, wskazane jako konieczne dla kształcenia uczniów.

Analiza i zrozumienie klasycznych algorytmów daje uczniom niezwykłe korzyści nie tylko w zakresie nauki informatyki, ale też ich ogólnego rozwoju. Uczy analizy i rozwiązywania problemów, kreatywności, systematyczności pracy oraz samokształcenia.

Obecnie naukę programowania można już wdrażać na początkowych etapach edukacyjnych. Opiera się ona w dużej mierze na idei geometrii żółwia, na której oparty był język programowania Logo – wspaniałe narzędzie dydaktyczne.

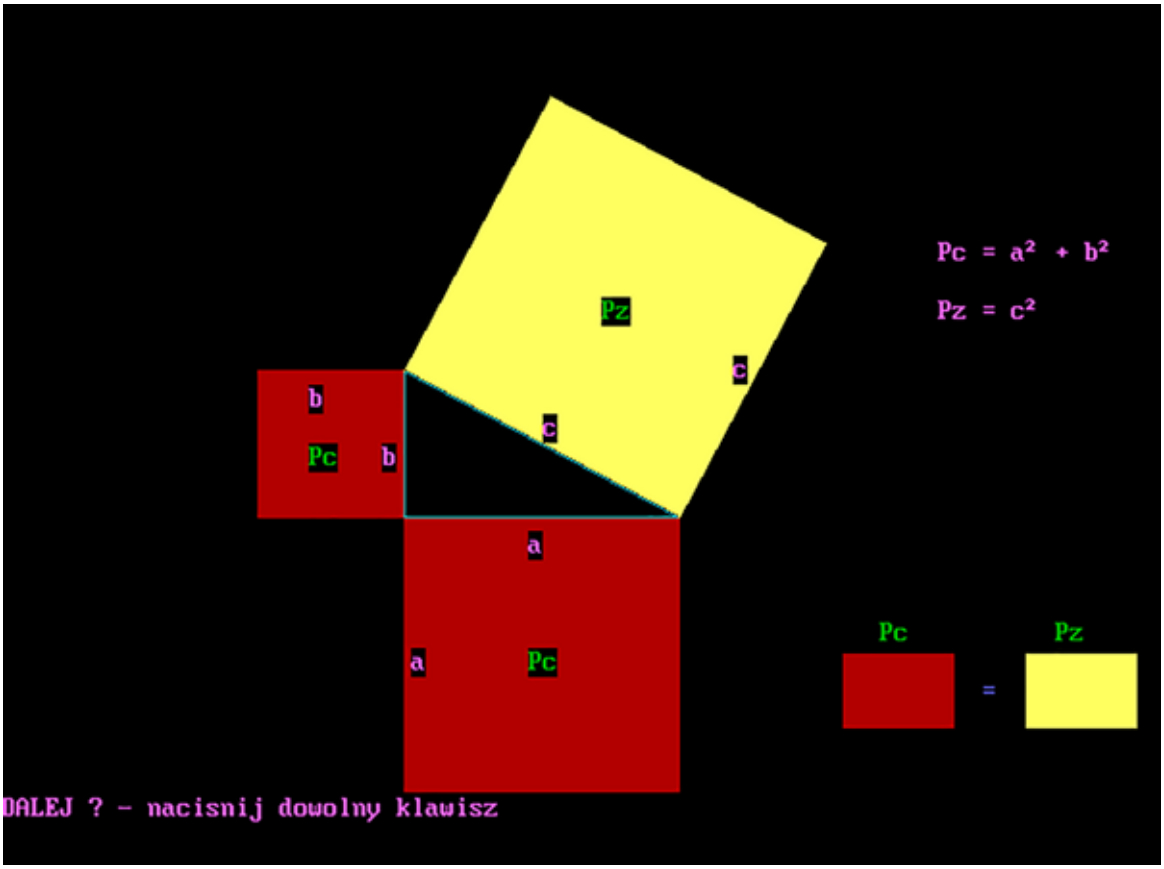
Język ten został stworzony przez Seymoura Paperta na potrzeby nauczania. Papert doskonale rozumiał, czym jest istota zastosowań informatyki w nauczaniu. Odwrócił popularne powiedzenie – „nauczanie wspomagane komputerowo" na „dziecko programuje komputer". Był to ważnym przełom, bowiem stwierdzenie „nauczanie wspomagane komputerowo" sugeruje, że chcemy programować dzieci i innych uczących się, a to jest absurd.

Tworzenie historii obrazkowych czy grafiki, to w tle nauka różnych pojęć z wielu przedmiotów szkolnych, na przykład matematyki czy języka polskiego. Chcąc coś stworzyć, uczymy się. Jak to określił Papert – „Uczenie przez tworzenie", chęć tworzenia wymusza naukę. Poruszając obiektem, uczeń musi znać podstawowe pojęcia geometrii, jeśli w programie ma zostać wypowiedziany jakiś wierszyk, uczeń musi go poznać i poprawnie wpisać. Tło historyjki obrazkowej może wymagać przedstawienia odpowiedniej scenerii związanej z czasem historycznym czy miejscem geograficznym. Mamy więc naukę historii, geografii, przyrody.

Jak widać, pomysłów na pisanie programów może być bardzo wiele, w zależności od tego, czego chcemy nauczyć ucznia. Zaczynamy programować na zasadzie zabawy, nie przytaczamy bezpośrednio definicji pojęć informatycznych, ale je stosujemy w tle, poprzez pewne wymuszenie spowodowane chęcią rozwiązania danego problemu. Ważne jest, aby uczeń starał się samodzielnie rozwiązywać zadania, testował rozwiązania i poprawiał ewentualne błędy. Te stwierdzenia są jawnie zapisane w podstawie programowanej przedmiotu informatyka już w szkole podstawowej.

Zagadnienie programowania jest istotne, umożliwia tworzenie czegoś nowego od podstaw i pełnię realizacji własnych wizji. Niestety rozwój technologii, najróżniejszych narzucanych programów i serwisów powoduje, że mało osób programuje i uczy się wartościowych elementów programowania. Prościej jest napisać różne bardziej lub mnie mądre wpisy na forach internetowych, niż programować. Obawiam się, czy paradoksalnie z użyciem technologii nie cofamy się do lat 50., 60. ubiegłego wieku. W owym czasie były też komputery, ale ich rozwojem, programowaniem zajmowały się tylko nieliczne grupy specjalistów. Ogół społeczeństw nie znał elementów informatyki i jej pozytywnego wpływu na rozwój myślenia, rozwiązywania problemów oraz nauki. W latach 80. fascynacja komputerami wymuszała poniekąd zajmowanie się programowaniem. Niestety, dalszy rozwój TIK i narzucanych form serwisów oraz aplikacji komputerowych nie promował globalnie idei programowania, co było wielką szkodą dla edukacji każdego człowieka. Na szczęście trend ten się teraz zmienia, czego dowodem są też zapisy w nowej podstawie programowanej przedmiotu informatyka zarówno dla szkoły podstawowej, jak i ponadpodstawowej. Zachęcając dzieci do programowania, nie mamy na celu, aby stały się one w przyszłości programistami, lecz dbamy o ich ogólny rozwój intelektualny. Programowanie uczy różnej wiedzy i wielu umiejętności, co jest wartościowe dla każdego, niezależnie od tego, jaką w przyszłości będzie miał profesję.

dr Jan Aleksander WIERZBICKI jest kierownikiem ds. nauki w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.



Rysunek 3. Efekt działania programu napisanego w pierwszej połowie lat 90., dowodzącego twierdzenia Pitagorasa.

Scratch, matematyka i klocki LEGO

Agnieszka BOROWIECKA

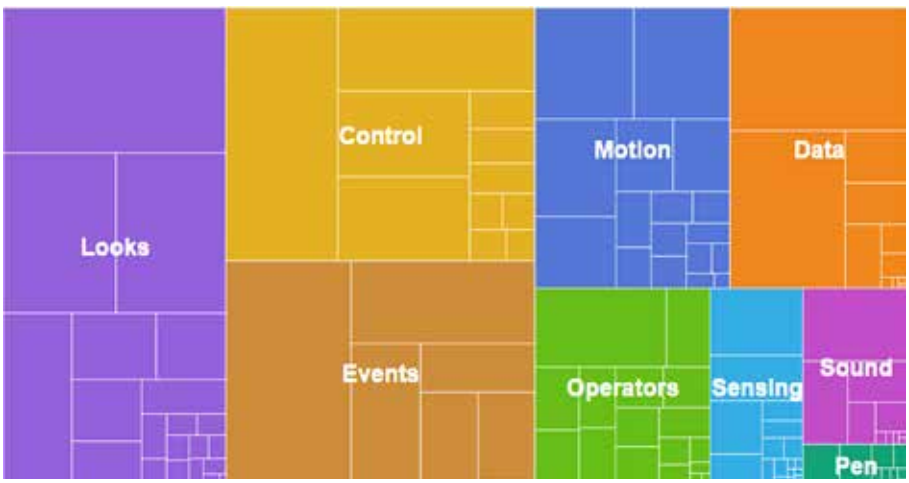
Wprowadzając uczniów w świat programowania, zwykle zaczynamy od środowiska wizualnego. Składanie programów z gotowych bloczków przyspiesza pracę szczególnie młodszych dzieci, mało jeszcze sprawnych w pisaniu i czytaniu. Bloczki łatwo ze sobą łączyć, zmieniać ich kolejność, zastępować innymi bez konieczności uczenia się skomplikowanej składni poleceń czy wpisywania ich z klawiatury.

Najbardziej popularnym środowiskiem do nauki programowania wizualnego jest Scratch. Został on zaprojektowany przez Mitchela Resnicka, twórcę języka StarLogo i pomysłodawcę serii zabawek Lego Mindstroms. Obecnie Scratch jest rozwijany przez grupę Lifelong Kindergarten z MIT Media Lab.

Scratch nie jest jedynym środowiskiem umożliwiającym budowanie programów z bloczków, ale inne podobne środowiska są zwykle mniej atrakcyjne wizualnie lub mają mniejsze możliwości.

Scratch i statystyki

We wrześniu 2017 roku społeczność Scratcha liczyła ponad dwadzieścia milionów zarejestrowanych użytkowników. Większość osób podczas zakładania konta deklaruje wiek poniżej 20 lat, najwięcej jest jedenasto- (prawie dwa miliony), dwunasto- (dwa miliony sto tysięcy) i trzynastolatków (prawie dwa miliony). Na platformie `https://scratch.mit.edu` można znaleźć ponad dwadzieścia cztery miliony udostępnionych projektów. Dużo z nich utworzyli samodzielnie członkowie społeczności Scratcha, ale wiele to wynik współpracy lub modyfikacji projektów udostępnionych przez innych (tzw. remiksy). Najczęściej wykorzystywane w projektach bloczki należą do kategorii *Wygląd* (22%), *Zdarzenia i Kontrola* (po 19%). Najrzadziej używane są bloczki z kategorii *Dźwięk* (3%) oraz *Pisak* (1%).



Rysunek 1. Wykorzystanie bloczków w projektach, strona ze statystykami Scratch.

Według założeń twórców Scratch miał być środowiskiem do nauki programowania, tymczasem większość przygotowanych w tym środowisku projektów to bardzo proste animacje lub gry, w których elementy programistyczne stosowane są w minimalnym zakresie. Może to wynikać ze struktury wiekowej użytkowników Scratcha lub z tego, że wiele pojęć, takich jak np. elementy logiki lub losowość, jest trudnych do samodzielnego zrozumienia bez wsparcia nauczyciela. Jednak mimo ograniczeń języka Scratch można przy jego użyciu tworzyć ciekawe i wartościowe projekty, rozwiązując konkretne problemy z różnych dziedzin, m.in. z matematyki.

Matematyka ze Scratchem

W środowisku Scratch można programować już w klasach I-III szkoły podstawowej. Warto zacząć zabawę od wersji dostępnej na urządzenia mobilne – Scratcha Juniora. Z jego pomocą uczniowie tworzą różne historyjki i animacje. Projekty mogą składać się z kilku scen i wielu duszków, dostępne bloczki podzielone są na sześć grup i nie wymagają od uczniów umiejętności czytania. Tworząc projekty w „dorosłej" wersji Scratcha, warto pamiętać, że możemy wykorzystać go na kilka sposobów. Prostsze projekty można tworzyć w całości razem z uczniami, przechodząc stopniowo do coraz trudniejszych konstrukcji i pojęć. Bardziej rozbudowane projekty możemy przygotować w formie szablonu zawierającego częściowe rozwiązanie problemu. Uczniowie uzupełniają brakujące elementy, dodają własną grafikę, testują działanie skryptów. Możliwe jest także przygotowanie projektów w całości przez nauczyciela, rola uczniów sprowadza się wtedy tylko do badania działania projektu lub poznawania w atrakcyjny sposób ważnych pojęć. Młodsi uczniowie mogą na przykład porównywać ze sobą przedmioty lub liczby, sprawdzać umiejętność wykonywania w pamięci różnych działań, rozpoznawać figury geometryczne itp. Inną propozycją jest wyszukanie przydatnych projektów na stronie `https://scratch.mit.edu` i po ewentualnej korekcie wykorzystanie ich w swojej pracy. Spróbujmy przyjrzeć się bliżej kilku przykładowym projektom łączącym matematykę i programowanie w środowisku Scratch.

Tabliczka mnożenia

Jednym z pierwszych projektów, jaki realizujemy z uczniami, jest zwykle dialog między postaciami. Może być to fragment wiersza, np. „Zoo" Jana Brzechwy, rozmowa o patronie szkoły lub wydarzeniach historycznych. Podobny schemat możemy wykorzystać do odpytywania z działań na liczbach: dodawania, odejmowania i tabliczki mnożenia. W najprostszej wersji projektu wystarczy dodać tło i jednego duszka, który będzie zadawał pytanie. Po udzieleniu odpowiedzi wyświetlany będzie odpowiedni komentarz, np. „Brawo" lub „Niestety nie, spróbuj ponownie". Liczby, na których wykonywane jest działanie, losujemy z podanego zakresu i zapamiętujemy w zmiennych. Możemy także sprawdzać umiejętność mnożenia przez konkretną wartość, wtedy losowana jest tylko jedna liczba.

Rysunek 2. Sprawdzamy umiejętność mnożenia

By zabawa była ciekawsza, można dodać drugiego duszka. Jego zadaniem będzie wyświetlenie liczby podanej przez użytkownika. Pierwszy z duszków zadaje pytanie i po odpowiednim czasie wyświetla gratulacje lub zachęca do dalszej nauki.

Rysunek 3. Wyświetlenie odpowiedzi przez duszka.

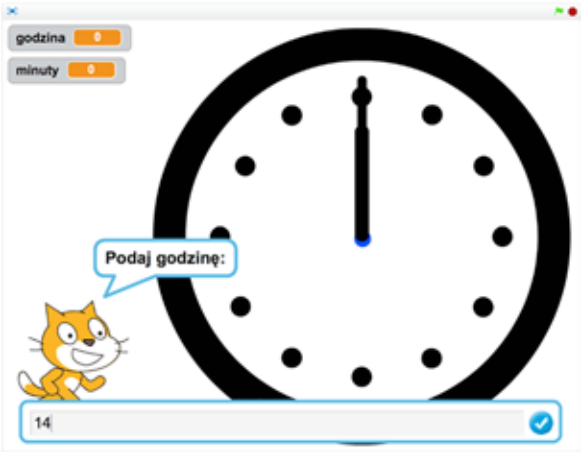
Najtrudniejszym elementem dla uczniów może okazać się wyświetlenie pytania. By duszek wyświetlił całe zdanie typu „Ile to jest 4 * 5?" musimy użyć aż 6 bloczków. Podobne problemy wystąpią, gdy będziemy budować złożone warunki lub działania matematyczne do wykonania przez duszka.

Rysunek 4. Przykładowy skrypt duszka zadającego pytanie.

Interesującym zadaniem może być taka modyfikacja projektu, by duszek wielokrotnie pytał się o wynik mnożenia bez konieczności ponownego uruchamiania projektu za pomocą zielonej flagi. Możemy także dodać zliczanie poprawnych odpowiedzi i w zależności od uzyskanych wyników wyświetlać podsumowanie znajomości tabliczki mnożenia przez ucznia.

Ustawiamy zegar

W projekcie **Zegar** podajemy dwie liczby określające czas pokazywany przez zegar analogowy. Tło sceny zawiera tarczę zegara z zaznaczoną podziałką godzinową. Dwa duszki reprezentują wskazówki: godzinową i minutową. Na początku obie wskazówki skierowane są do góry, ponieważ zegar nie został jeszcze ustawiony.

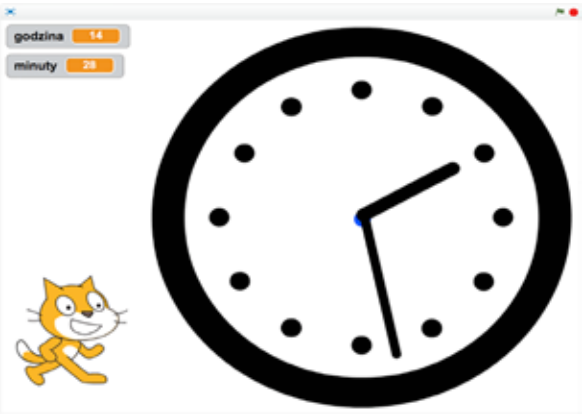


Rysunek 5. Podawanie godziny ustawianej na zegarze

Duszek-kot prosi o podanie kolejno dwóch liczb – godziny i minut. Podane liczby zostają zapamiętane w pomocniczych zmiennych *godzina* i *minuty*, wyświetlanych w lewym górnym rogu sceny. Zostaje wysłany komunikat do duszków z prośbą o ustawienie zegara.

Rysunek 6. Skrypt ustawiający wskazówkę godzinową.

Możemy przedyskutować z uczniami, w jaki sposób duszki-wskazówki powinny zachować się po otrzymaniu komunikatu. W najprostszej wersji projektu każda ze wskazówek reaguje na wartość odpowiedniej zmiennej, ustawiając kierunek lub obracając się o właściwy kąt.



Rysunek 7. Zegar wyświetla podany czas

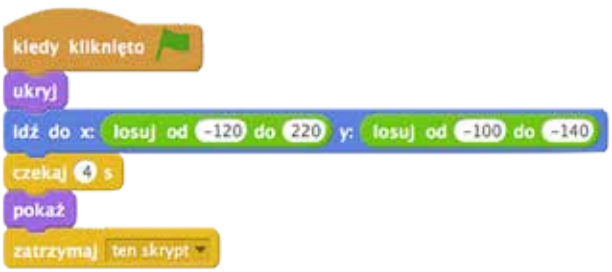
Projekt **Zegar** można rozszerzyć, poprawiając położenie wskazówki godzinowej. Powinna się ona nie tylko przesuwać skokowo co godzinę, ale także podczas danej godziny o pewien kąt, w zależności od tego, ile upłynęło minut. Inną modyfikacją może być odczytywanie przez ucznia czasu z tarczy zegara po wylosowaniu położenia wskazówek. Możemy także przygotować projekt, w którym wskazówki pokazują bieżący czas, korzystając z czujników *aktualna godzina* 

i *aktualna minuta* 

.

Rozpoznajemy figury

Bardzo uniwersalnym zadaniem jest podział duszków na dwie grupy lub więcej. Możemy na przykład rozpoznawać figury geometryczne, dzielić liczby na dodatnie i ujemne lub parzyste i nieparzyste. Ten sam projekt po niewielkiej modyfikacji można wykorzystać na lekcji języka polskiego (np. pisanie wyrazów z **rz** i **ż**), przyrody (podział na owoce i warzywa, drzewa liściaste i iglaste itp.) czy dowolnego innego przedmiotu. Do rozpoznania, czy przedmioty zostały prawidłowo podzielone na grupy, można użyć tła sceny.

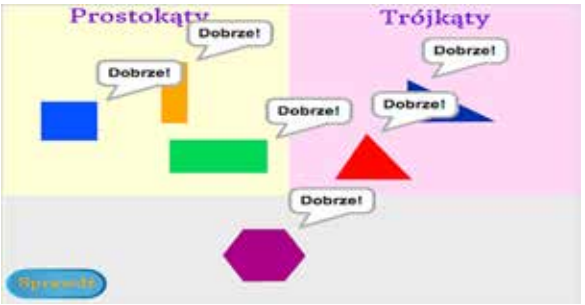


Rysunek 8. Losowanie położenia figury.

Po uruchomieniu projektu za pomocą zielonej flagi, duszki-figury rozsypują się w sposób losowy w dolnej części sceny. Analizujemy z uczniami skrypt zmieniający położenie duszków, zastanawiając się wspólnie nad zakresem losowanych współrzędnych. Uczniowie mogą samodzielnie przygotować tło sceny, wtedy odczytują za pomocą kursora myszki, jakie współrzędne mogą mieć duszki, by mieścić się w całości na szarym tle. Zauważamy, że jeśli prawidłowo dobraliśmy zakres losowania, to wszystkie duszki-figury mogą korzystać z tego samego skryptu.

Rysunek 9. Sprawdzanie, czy trójkąt został prawidłowo umieszczony na scenie.

Naszym zadaniem jest przesunięcie myszką figur do właściwej części sceny. Należy pamiętać, by włączyć opcję *można przeciągać w odtwarzaczu* dla każdego duszka-figury. Po naciśnięciu przycisku **Sprawdź**, zostaje nadany komunikat *sprawdź figury*. Każdy duszek-figura reaguje na komunikat, sprawdzając, jakiego koloru tła dotyka. Następnie wypowiada odpowiednio komentarz „Dobrze" lub „Źle", w zależności od tego, w jakiej części sceny się znalazł.



Rysunek 10. Wybieramy trójkąty i prostokąty.

Łatwo sprawdzić, że proponowane rozwiązaie nie jest idealne. Jeśli figurę umieścimy w taki sposób, że dotyka kilku kolorów (nie leży całkowicie wewnątrz właściwej części sceny), to również uzyskamy komentarz „Dobrze". Warto spróbować z bardziej zaawansowanymi uczniami zmienić warunki sprawdzane przez duszki, by rozwiązać ten problem. Projekt można urozmaicić, dodając dodatkowe kostiumy dla duszków. Po kliknięciu w zieloną flagę będzie losowany kostium dla każdego duszka. Uczeń za każdym razem otrzyma nowy zestaw figur różniących się kształtem lub kolorem od poprzedniego.

Wielokrotności liczb

Przygotowując specjalne kostiumy dla duszków, możemy tworzyć ciekawe projekty i przyspieszać jednocześnie proces ich budowania. Jeden ze scenariuszy omawianych na zajęciach w ramach projektu **Warszawa programuje!** dotyczy układania wyrazu z poruszających się po ekranie literek. Aby można było automatycznie tworzyć różne wyrazy, niezbędne jest przygotowanie duszka-litery, którego kostiumy to kolejne litery alfabetu. W standardowej bibliotece kostiumów Scratcha dostępne są jedynie pojedyncze litery, nie ma także znaków diakrytycznych potrzebnych do tworzenia słów w języku polskim. Dlatego kostiumy dla duszka musimy wcześniej samodzielnie przygotować lub pobrać z internetu.

Do projektu **Wielokrotności liczb** warto najpierw przygotować duszka, którego kostiumami będą kolejne liczby od jeden do stu. Podobne duszki możemy później wykorzystać w innych projektach związanych z liczeniem.

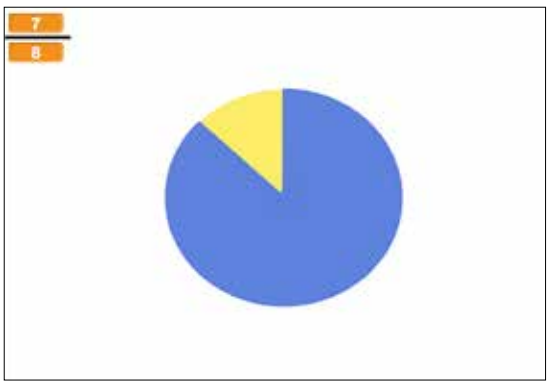
Rysunek 11. Wielokrotności liczby 3.

Dla wzorcowego duszka-liczby definiujemy zmienną prywatną *numer klona,* a następnie klonujemy go i ustawiamy na właściwej pozycji. Uczniowie mogą samodzielnie ustawić sto duszków, zaczynając od pojedynczego wiersza z dziesięcioma liczbami, a potem dodając drugą pętlę i tworząc kolejne klony. Wspólnie tworzymy skrypt określający zachowanie duszków – każdy z nich powinien mieć inną wartość zmiennej *numer klona* i inny kostium. Jeśli dany klon dotyka wskaźnika myszki, traktowany jest jako wybrany przez użytkownika. Wszystkie pozostałe duszki, o numerze będącym wielokrotnością wybranego, zmieniają swój rozmiar lub dodają efekt *kolor* albo *jasność*, aby się wyróżnić. Reszta klonów stosuje standardowe ustawienia.

Tablicę liczb wyświetlanych w projekcie można wykorzystać także na inne sposoby. Możemy na przykład wrócić do tabliczki mnożenia. Użytkownik podaje dwie liczby, a klony odpowiadające wynikowi mnożenia zostają wyróżnione. Starsi uczniowie mogą spróbować przygotować wizualizację wyszukiwania liczby pierwszej metodą sita Eratostenesa.

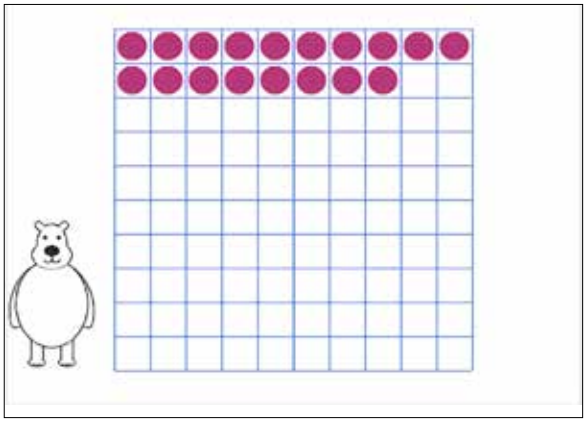
Ułamki

W Scratchu możemy przygotowywać skrypty pomagające uczniom rozumieć ułamki zwykłe i dziesiętne. Do wizualizacji ułamków zwykłych można wykorzystać bloczki z grupy *Pisak*. Po otrzymaniu informacji o liczniku i mianowniku ułamka duszek rysuje odcinki w dwóch kolorach. Niebieski kolor to nasz ułamek, żółty – dopełnienie do jedności. Możemy ilustrować ułamek za pomocą wycinka koła lub jako dwukolorowy prostokąt.



Rysunek 12. Ułamek 7/8.

Rysowania można także użyć do przedstawiania ułamków dziesiętnych. Przygotowujemy tło sceny zawierające siatkę kwadratów o wymiarach 10 na 10. Po podaniu wartości ułamka, duszek skacze do środka kolejnych kwadratów i w odpowiedniej części z nich rysuje dużą kropkę.



Rysunek 13. Ułamek 0.18

Oba przedstawione projekty można prosto rozbudować. Po dodaniu wyświetlania drugiego ułamka zwykłego możemy porównywać ułamki i wskazywać na przykład, który z nich jest większy. Siatki kwadratów z drugiego projektu można użyć do ilustrowania dodawania lub odejmowania ułamków dziesiętnych.

Cyfry i systemy liczbowe

Korzystając z operacji dzielenia możemy przygotować projekt, w którym duszek „czyta" kolejne cyfry podanej przez nas liczby. Potrzebna będzie zmienna *liczba* i bloczki *zapytaj* oraz *odpowiedź* do komunikacji z użytkownikiem. Wykorzystamy operację *mod* – reszta z dzielenia oraz część całkowitą z dzielenia. Niestety w środowisku Scratch nie ma operacji dzielenia całkowitego. Zamiast tego można skorzystać z funkcji *podłoga*, która dla danej liczby obcina jej część ułamkową, zostawiając największą liczbę całkowitą mniejszą lub równą podanej wartości. Warto zwrócić uwagę uczniów na kolejność wyświetlanych cyfr.

W przedstawionym na rysunku 14 skrypcie algorytm można także wykorzystać do przeliczania z systemu dziesiętnego na inny system liczbowy. Wystarczy zastąpić dzielenie przez dziesięć dzieleniem przez podstawę tego systemu. Zamiast kazać

Rysunek 14. Wyświetlenie kolejnych cyfr liczby.

duszkowi wyświetlać kolejne cyfry otrzymanej liczby, można je zapamiętać w liście albo wyświetlić na scenie jako kolejne duszki. Każdy z duszków będzie miał tyle kostiumów, ile jest różnych cyfr w systemie liczbowym, na który przeliczamy. Dla systemu binarnego będą to zero i jedynka, dla systemu trójkowego zero, jedynka i dwójka itd.

Rysunek 15. Przeliczenie na system dwójkowy.

Prostszym projektem dla uczniów może być przeliczanie z systemu dwójkowego na dziesiętny. Wystarczy, jeśli przygotują oni tyle duszków, ile maksymalnie cyfr może liczyć wyświetlana wartość binarna. Każdy duszek ma dwa kostiumy – zero i jedynkę. Po kliknięciu w duszka następuje przełączenie kostiumu. Z duszkami związane są zmienne odpowiadające potędze liczby 2. Jeśli duszek pierwszy z prawej ma kostium z liczbą zero – to właściwa zmienna ma wartość zero, jeśli kostium z jedynką – to zmienna ma wartość jeden. Dla kolejnego duszka zeru odpowiada zero, a jedynce dwójka. U duszków następnych są to zero i czwórka, zero i ósemka itd. Po każdej zmianie

kostiumu dowolnego z duszków liczona jest suma wszystkich zmiennych i przypisywana pomocniczej zmiennej *dziesiętna*.

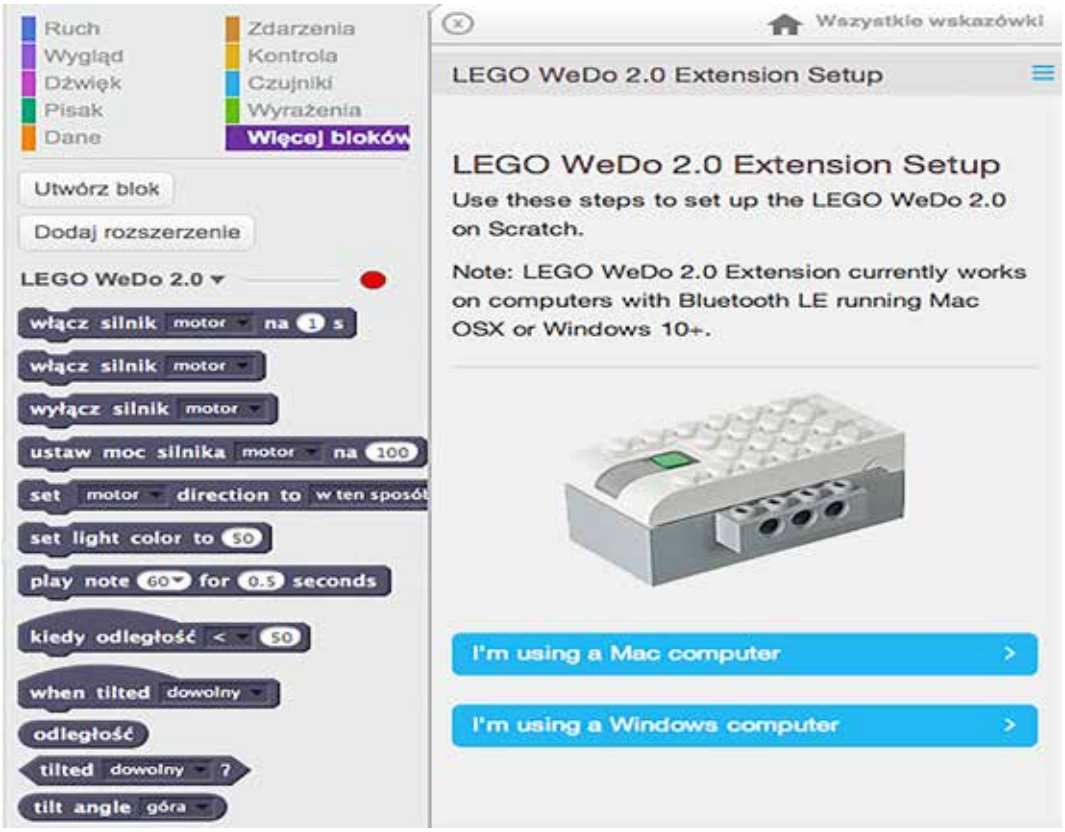
Uczniowie mogą samodzielnie przygotować analogiczny projekt do przeliczania z systemu trójkowego na dziesiętny.

Zbuduj z klocków LEGO i baw się ze Scratchem

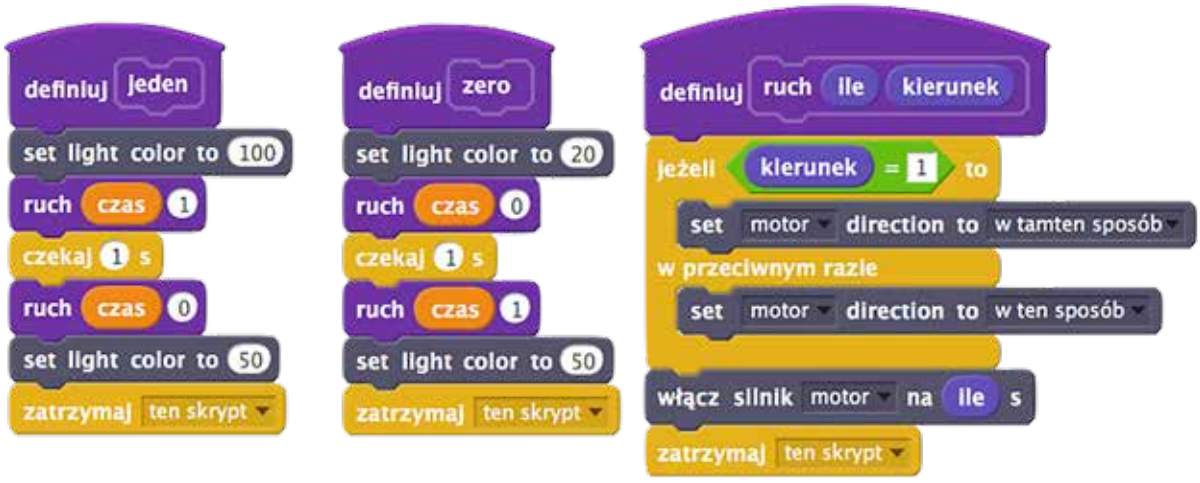
Każdy z nas w dzieciństwie bawił się klockami. Zapewne najbardziej atrakcyjne dla dzieci są klocki LEGO. Dzisiaj możemy połączyć zabawę z programowaniem, budując z klocków Lego WeDo różne konstrukcje, a następnie sterując nimi za pomocą oryginalnego oprogramowania lub specjalnego rozszerzenia dodawanego do Scratcha. Bazowy zestaw klocków WeDo 2.0 zawiera ponad 150 elementów, w tym silnik, dwa czujniki oraz jednostkę sterującą. Oprogramowanie można bezpłatnie zainstalować na tabletach, laptopach lub komputerach stacjonarnych. Jedynym wymaganiem jest wyposażenie sprzętu komputerowego w standard Bluetooth Low Energy. Jeśli nie mamy wbudowanego takiego interfejsu, możliwe jest podłączenie go zewnętrznie przez wejście USB.

Do programu sterującego robotami WeDo dołączonych jest ponad 20 gotowych scenariuszy składających się z wprowadzenia, opisu budowania robota krok po kroku oraz krótkiego programu sterującego zbudowaną konstrukcją. Programy budujemy z bloczków, podobnie jak w Scratchu Juniorze. W internecie można także znaleźć przykłady oryginalnych konstrukcji wymyślonych i zbudowanych przez uczniów.

W Scratchu istnieje możliwość dodania rozszerzeń pozwalających łączyć się z urządzeniami zewnętrznymi, takimi jak PicoBoard, LEGO WeDo 1.0 i LEGO WeDo 2.0. Wybieramy grupę *Więcej bloków* i naciskamy przycisk **Dodaj rozszerzenie**. Jeśli chcemy kierować robotem WeDo, musimy jeszcze połączyć się z jego jednostką sterującą. Należy zainstalować na komputerze program Scratch Device Manager, uruchomić go i włączyć robota. Po wykryciu i sparowaniu robota z komputerem możemy sterować jego zachowaniem za pomocą skryptów tworzonych w Scratchu.



Rysunek 16. Bloczki sterujące robotem WeDo i instrukcja podłączenia go do Scratcha.



Rysunek 17. Przykładowe skrypty sterujące robotem Milo.

Ponieważ zestaw WeDo zawiera tylko jeden silnik, roboty mają ograniczoną możliwość ruchu – mogą jeździć tylko do przodu i do tyłu, nie potrafią jednak skręcać. Istnieje możliwość dokupienia i podłączenia dodatkowego silniczka, co pozwoli zwiększyć możliwości sterowania robotem. Jednak już nawet standardowa konstrukcja zalecana do zbudowania jako pierwsza, tzw. robot Milo, może być wykorzystana do tworzenia ciekawych projektów. Wystarczy przygotować dwie plansze, po których będzie poruszał się Milo. Na jednej drukujemy lub piszemy cyfrę zero, na drugiej zaś jedynkę. Otwieramy w Scratchu projekt przeliczający liczby z systemu dziesiętnego na dwójkowy i zapamiętujący wynik w postaci listy. Dodajemy do niego skrypty sterujące robotem: po odczytaniu z listy cyfry zero Milo jedzie na planszę z cyfrą zero, a następnie wraca do wyjściowej pozycji. Dla cyfry jeden Milo jedzie w przeciwną stronę. Dodatkowo możemy zmieniać kolor światełka na jednostce sterującej.

Podsumowanie

Scratch jest ciekawym środowiskiem i warto z niego korzystać na lekcji. Uczniowie łatwo uczą się budować z bloczków skrypty, wymyślają różne historyjki, projektują scenę i kostiumy dla duszków. Należy zwrócić uwagę na to, jakie projekty będziemy tworzyć. Już samo sterowanie duszkami wprowadza nas w świat matematyki – ustalanie współrzędnych duszka, wyliczanie kątów obrotu, budowanie warunków logicznych. W swoich poszukiwaniach warto posunąć się jednak dalej i przygotować bardziej rozbudowane projekty matematyczne. Jeśli możemy dodatkowo połączyć się ze światem realnym, chociażby poprzez sterowanie robotami, nauka programowania stanie się ulubionym zajęciem naszych uczniów, a matematyka przestanie być strasznym i stresującym przedmiotem.

Agnieszka BOROWIECKA jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Od programowania wizualnego do tekstowego[1]

Maciej BOROWIECKI, Krzysztof CHECHŁACZ

Wstęp

Nowa podstawa programowa przedmiotu informatyka kładzie duży nacisk na rozwiązywanie problemów z pomocą komputera oraz programowanie. Na wczesnych etapach edukacyjnych zwykle rozpoczynamy od programowania wizualnego. Podczas tworzenia projektu unikamy w ten sposób problemów związanych z zawiłościami składni języka programowania. Gdy próbujemy połączyć elementy, które do siebie nie pasują, mechanizm bloczków nie pozwala nam na to. Ponadto szybko uzyskujemy efekt w postaci działającego projektu i w ten sposób zachęcamy uczniów do kolejnych aktywności.

W pewnym momencie należy jednak przejść na programowanie tekstowe. Języki tekstowe mają zwykle większe możliwości i choćby dlatego warto zainteresować się programowaniem w środowisku tekstowym. Trudno też będzie podtrzymać zainteresowanie uczniów przez 12 lat, pozostając wyłącznie przy programowaniu wizualnym. Komentarz do nowej podstawy programowej wskazuje jako dobry moment na rozpoczęcie nauki w środowisku tekstowym VII-VIII klasę zreformowanej szkoły podstawowej. Uczniowie w tym wieku zaczynają myśleć także abstrakcyjnie.

Pisząc program w środowisku tekstowym, musimy znać składnię języka i sprawnie posługiwać się klawiaturą. Są to podstawowe problemy, na jakie napotykają osoby rozpoczynające naukę programowania tekstowego. Pojawiają się błędy składniowe. Wielu uczniów zniechęca się i wyraża chęć powrotu do programowania wizualnego. W dalszej części artykułu przedstawiamy propozycje przejścia od programowania wizualnego do tekstowego w sposób łagodny i akceptowalny dla uczniów. Co więcej, w ten sposób pokazujemy, że najważniejszy jest sam algorytm rozwiązania zadania, a sposób zapisu jest mniej istotny.

Wykorzystamy narzędzia, które umożliwiają napisanie programu w pewnym języku, a następnie automatyczne przełożenie kodu na inny język.

Algorytm Euklidesa

W nowej podstawie programowej znajdujemy wiele odwołań do różnych algorytmów. W szczególności uczeń na poziomie klas VII-VIII stosuje przy rozwiązywaniu problemów podstawowe algorytmy na liczbach całkowitych: bada podzielność liczb, wyodrębnia cyfry danej liczby, przedstawia działanie algorytmu Euklidesa w obu wersjach iteracyjnych (z odejmowaniem i z resztą z dzielenia). Zajmijmy się przykładowo algorytmem Euklidesa. Umożliwia on wyliczenie największego wspólnego dzielnika dwóch liczb naturalnych. W najprostszej wersji polega on na tym, że póki liczby są różne, od większej z nich odejmujemy mniejszą. Gdy są równe, oznacza to, że otrzymaliśmy wynik.

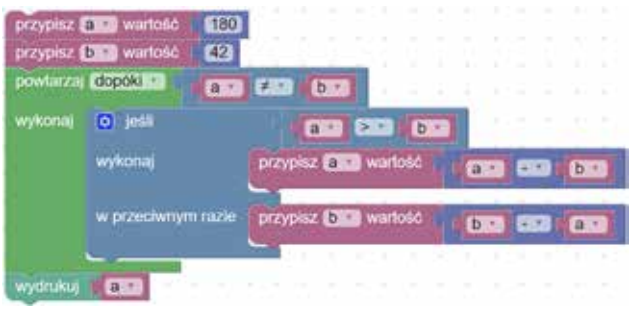
Rozwiązanie zapiszmy w języku programowania. Na razie nie zastanawiamy się nad kwestią

[1] Artykuł został opracowany na podstawie warsztatów przygotowanych przez autorów na konferencję Informatyka w Edukacji 2017 w Toruniu oraz artykułu „Od programowania wizualnego do tekstowego" [1].

eleganckiego przekazania danych do programu, wrócimy do tego później. Przykładowe rozwiązanie dla liczb 180 i 42 w najbardziej popularnym w Polsce środowisku programowania wizualnego Scratch może wyglądać następująco:

Rysunek 1. Algorytm Euklidesa w Scratchu.

Podobnie będzie ono wyglądać w środowisku *Blockly* dostępnym na stronie `https://blockly-demo.appspot.com/static/demos/code/index.html`. Środowisko to jest jednym z demonstracyjnych projektów inicjatywy *Google Blockly*, w którym powstała m.in. szeroko znana *Godzina Kodowania*. Podobnie jak w Scratchu możemy używać języka polskiego, mamy elementy – bloczki, które możemy łączyć, tworząc własny projekt, a następnie taki projekt zapisać i uruchomić.



Rysunek 2. Algorytm Euklidesa w Google Blockly.

W obu przypadkach otrzymujemy wynik 6. Jest on poprawny – w istocie dzielnikami 180 są 1, 2, 3, 4, 5, 6, 9, 10, 12, 15, 18, 20, 30, 36, 45, 60, 90 oraz 180, dzielnikami 42 są 1, 2, 3, 6, 7, 14, 21 oraz 42 – liczba 6 jest największą, która występuje w obu przypadkach.

W *Blockly* mamy możliwość obejrzenia, jak wyglądałoby rozwiązanie zapisane w kilku innych językach. Wystarczy wybrać jedną z zakładek.

Rysunek 3. Wybór języka docelowego.

Kilka ostatnich zakładek daje możliwość obejrzenia kodu w dość rzadko używanych językach lub kod jest mało czytelny. Zajmiemy się językami z dwóch pierwszych zakładek: *Python* i *JavaScript*.

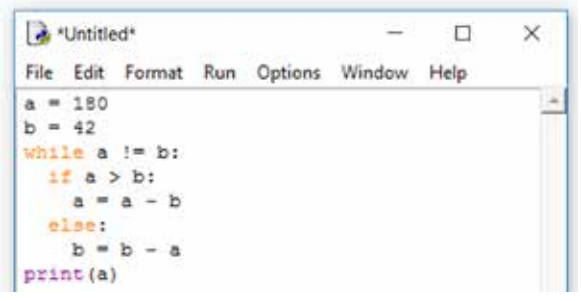
Python

Kod w języku *Python* jest krótki i czytelny, nie pozostawia wątpliwości co do sposobu działania; poniżej z pominiętymi dwoma pierwszymi wierszami.

```
a = 180
b = 42
while a != b:
  if a > b:
    a = a - b
  else:
    b = b - a
print(a)
```

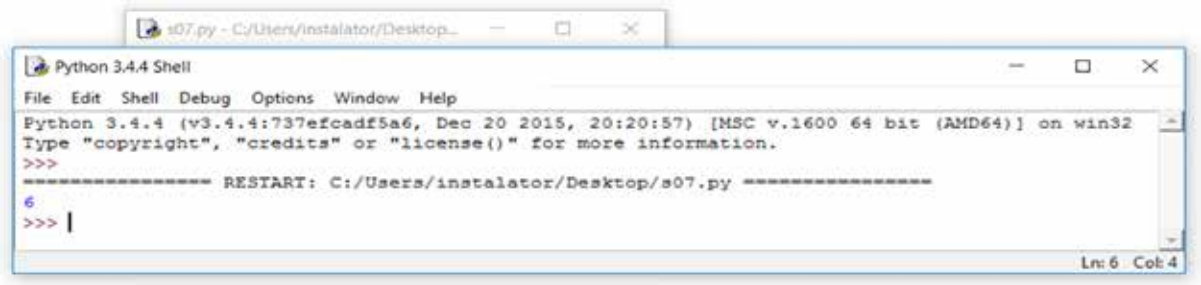
Rysunek 4. Kod w Pythonie wygenerowany automatycznie.

Tak uzyskany kod możemy skopiować do środowiska języka *Python*, najlepiej do nowego (pustego) okna.

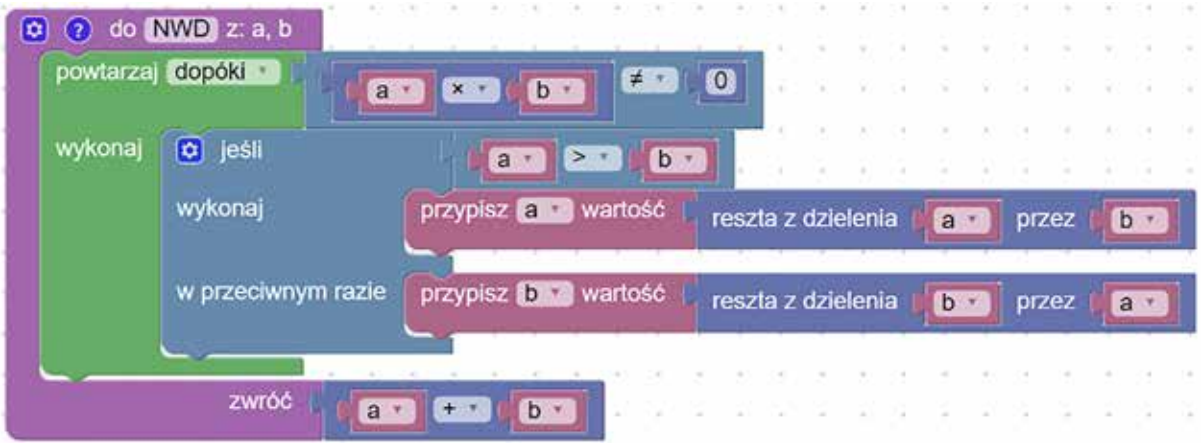


Rysunek 5. Kod źródłowy w Pythonie.

Od razu widać, jak elegancko środowisko *Pythona* koloruje składnię – dzięki temu wiemy, że słowa kluczowe wpisane zostały poprawnie. Po uruchomieniu programu wynik zostanie wypisany w głównym oknie.



Rysunek 6. Po uruchomieniu programu w Pythonie.



Rysunek 7. Kod funkcji w Google Blockly.

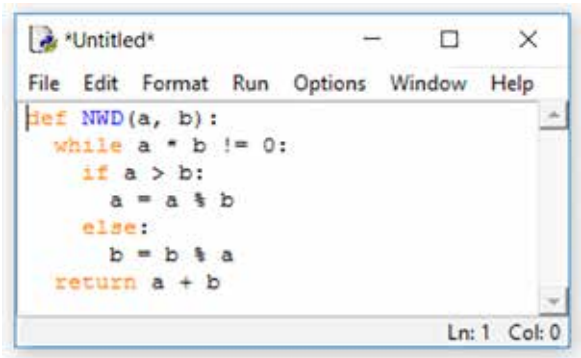
Tak więc, bez znajomości *Pythona*, udało się nam „napisać" i uruchomić program w tym języku. W sposób oczywisty, bez większej trudności, uczniowie poznali składnię najważniejszych instrukcji – przypisania wartości, warunkowej, pętli, a także zobaczyli sposób tworzenia wyrażeń i warunków logicznych. Warto w tym miejscu przeprowadzić dyskusję na temat znaczenia wcięć stosowanych w kodzie programu – *Python* wymusza ich stosowanie. Jest to sytuacja korzystna i nie rodzi dyskusji, bo to nie nauczyciel poleca stosowanie wcięć, ale jest to formalny wymóg składni języka. Uczniowie uczą się na podstawie przykładu, dzięki czemu ich wiedza będzie trwalsza i użyteczniejsza.

W dalszej części zajęć z uczniami można rozbudowywać program. Na przykład korzystając z *Google Blockly*, budujemy funkcję znajdującą największy wspólny dzielnik, a sam algorytm Euklidesa modyfikujemy tak, by zamiast odejmowania używał reszty z dzielenia (rysunek 7). Budowanie funkcji jest możliwe w *Google Blockly*, natomiast w *Scratchu* nie ma możliwości zwrotu wartości funkcji.

Kod otrzymany w *Pythonie* będzie zawierał nowe informacje – o tym, jak wygląda struktura funkcji i jak zapisywać resztę z dzielenia.

Wywołanie funkcji wymaga użycia nazewnika funkcji i podania parametrów wywołania. Efekt jest identyczny, jak poprzednio.

Warto dać uczniom do wykonania inne zadanie polegające na zapisie pewnego algorytmu w wybranym przez siebie języku, np. algorytmu wyznaczenia dzielników zadanej liczby naturalnej bądź wyodrębnienia cyfr danej liczby.

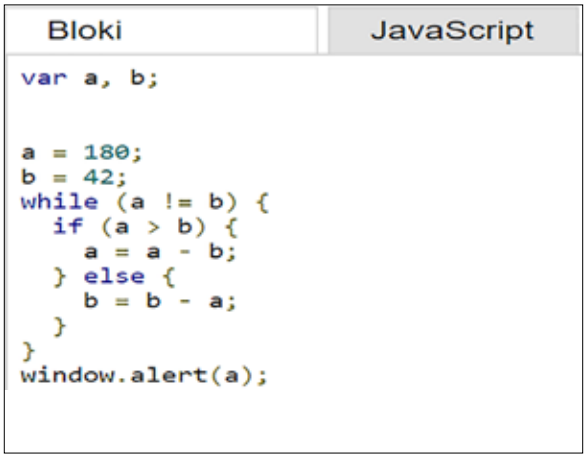


Rysunek 8. Kod funkcji automatycznie wygenerowany i przeniesiony do Pythona.

Rysunek 9. Wywołanie funkcji.

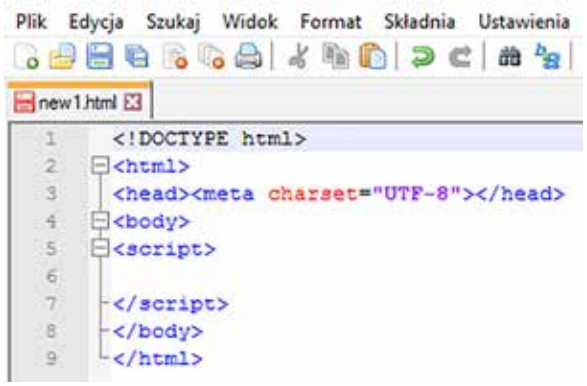
JavaScript

Spróbujmy teraz skorzystać z kodu *JavaScript*, który został wygenerowany z użyciem *Google Blockly* dla algorytmu Euklidesa:



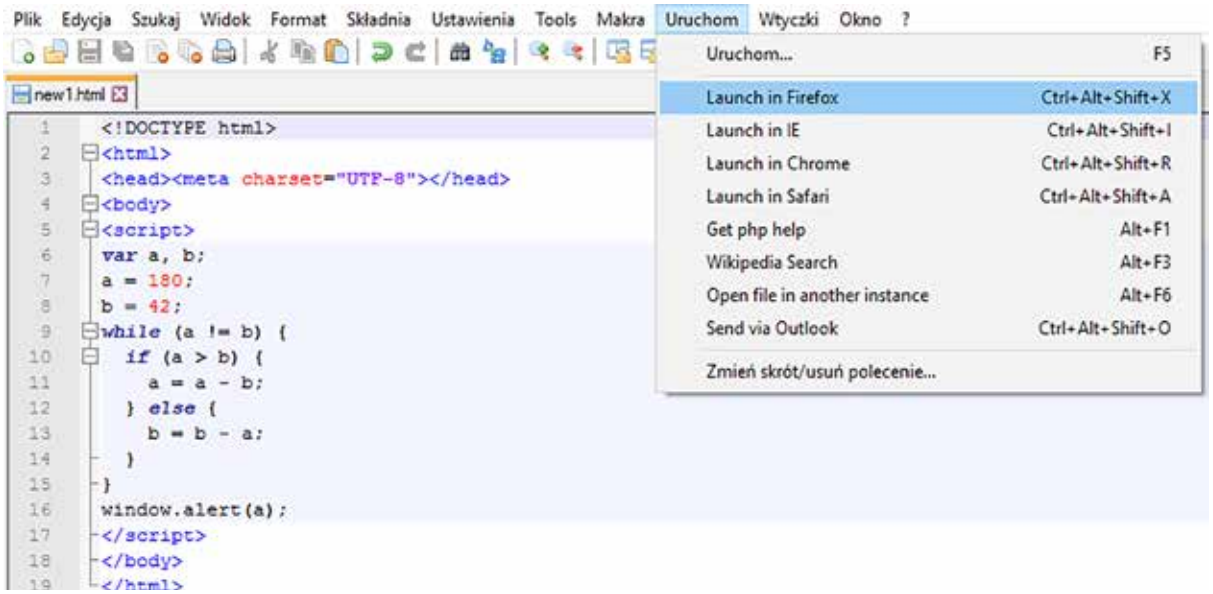
Rysunek 10. Kod w JavaScript wygenerowany automatycznie.

Najprościej będzie kod *JavaScript* zanurzyć w dokumencie *html*, a następnie tak utworzony



Rysunek 11. Szablon opisu strony internetowej.

dokument przekazać przeglądarce internetowej do zinterpretowania. Dla utworzenia dokumentu html możemy użyć programu *Notatnik++* z możliwością bezpośredniego przejścia – uruchomienia kodu *html* w przeglądarce internetowej. Można też użyć systemowego notatnika, ale wówczas niektóre czynności będą bardziej złożone.

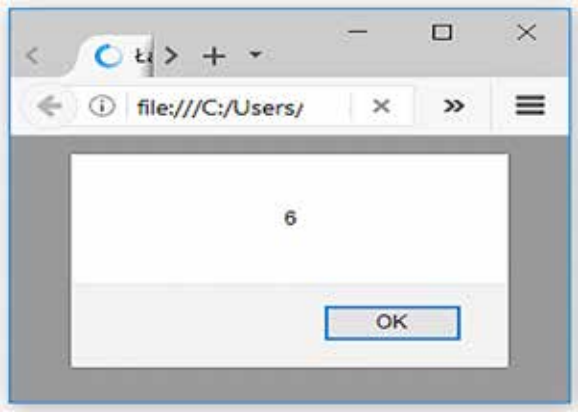


Rysunek 12. Opis strony internetowej po umieszczeniu w nim kodu

W pierwszym kroku tworzymy szablon opisu strony internetowej. Utworzony skrypt będzie działać także wówczas, gdy ograniczymy się do piątego i siódmego wiersza, ale nie polecamy takiego rozwiązania, ponieważ tworzy to złe nawyki dotyczące budowy plików opisujących strony internetowe.

W miejscu pustego szóstego wiersza umieszczamy kod z *Google Blockly*. Dokument zapisujemy (to ważne, bo program edycyjny komunikuje się z przeglądarką, używając wersji pliku *html* zapisanej na dysku) i przekazujemy przeglądarce do zinterpretowania, wybierając *Uruchom* i *Launch in* (tu-nazwa- przeglądarki).

Przeglądarka zinterpretuje kod i wypisze wynik.



Rysunek 13. Wynik działania JavaScript.

Podsumowanie

Przedstawiliśmy rozwiązanie dotyczące automatycznej zamiany kodu zapisanego w środowisku programowania wizualnego na kod w języku programowania tekstowego. Pokazuje ono, że dość łatwo można zachęcić uczniów, nawet tych, którzy przez długi czas programowali wizualnie, do podjęcia trudu tworzenia własnych programów z użyciem narzędzi programowania tekstowego. Koncentrujemy się na rozwiązywanym problemie, algorytmie. Składnia języka pozostaje w tle, wprowadzamy tylko te instrukcje, które są potrzebne.

Bibliografia

1. Borowiecki M., Chechłacz K. *Od programowania wizualnego do tekstowego*, Informatyka w Edukacji, Toruń 2017.
2. Borowiecki M. *Python na lekcjach informatyki w szkole ponadgimnazjalnej*, Informatyka w Edukacji, Toruń 2013.
3. Borowiecki M. *Python w szkole od podstawówki do liceum*, EduFakty – Uczę Nowocześnie nr 23, styczeń-luty 2013.
4. Podstawa programowa kształcenia ogólnego dla szkoły podstawowej, http://www.dziennikustaw.gov.pl/DU/2017/356, dostęp 3.09.2017.
5. Polewczyński A. *Nauka kodowania z Google Blockly*, Informatyka w Edukacji, Toruń 2014.
6. Zając K. *Python jako alternatywa dla Pascala*, Informatyka w Edukacji, Toruń 2012.
7. https://scratch.mit.edu, dostęp 3.09.2017.
8. https://blockly-demo.appspot.com/static/demos/code/index.html?lang=pl, dostęp 3.09.2017.
9. https://www.python.org, dostęp 3.09.2017.

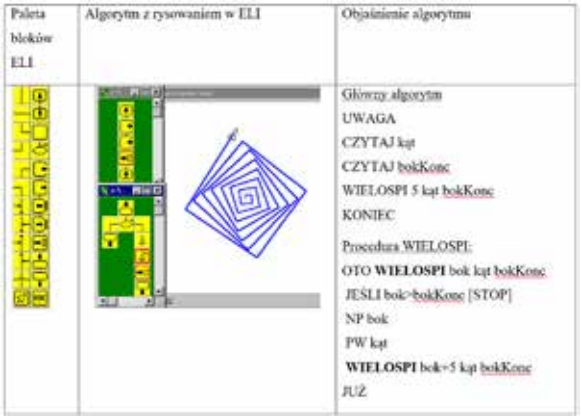
Maciej BOROWIECKI jest wicedyrektorem ds. edukacyjnych, **Krzysztof CHECHŁACZ** jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

W poszukiwaniu środowiska do nauki programowania – poza Scratchem[1]

Witold KRANAS

Programowanie wizualne

W 2017 roku obchodziliśmy dziesiąte urodziny Scratcha – środowiska, które bardzo szybko rozpowszechnia się w polskich szkołach, zastępując język Logo. Scratch jest sztandarowym przykładem wykorzystania programowania wizualnego. Pomysł ułatwienia nauki programowania poprzez wizualizację poleceń języka i składanie programu z gotowych bloków nie jest nowy. Już w latach 90. poprzedniego wieku był wykorzystywany w polskich szkołach program ELI Laboratorium Informatyki, w którym można było układać i uruchamiać schematy blokowe algorytmów.



Rysunek 1. Paleta klocków i program ułożony w ELI.

W tych czasach dominującym środowiskiem programistycznym w edukacji był język Logo, a na wyższym poziomie również Pascal. Dokonujące się dziś przejście na środowiska wizualne spowodowało, że trzeba szukać sposobów przeniesienia nabytych w tych środowiskach umiejętności programistycznych do środowisk programowania tekstowego wymaganych na egzaminie maturalnym (C/C++, Java, Python).

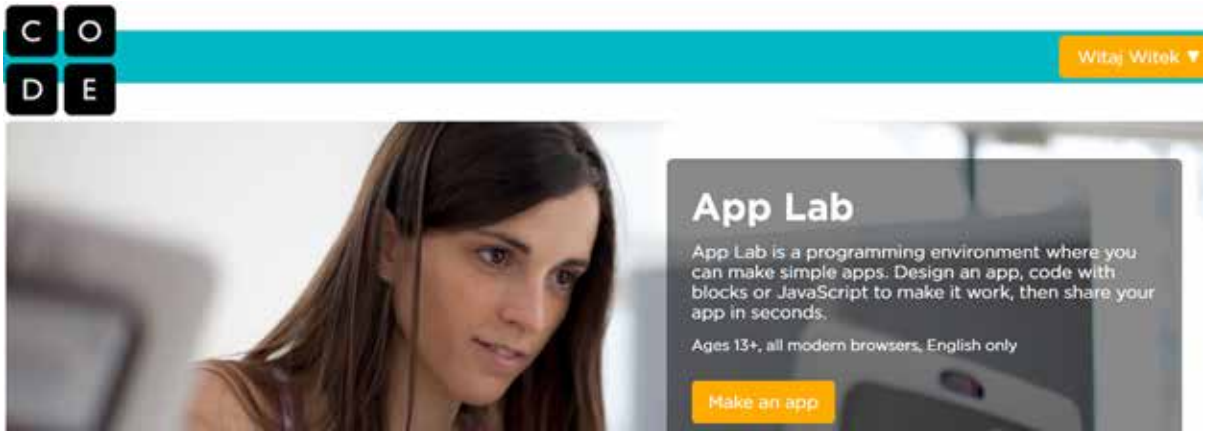
Zaczęły już powstawać środowiska łączące budowanie programu z gotowych bloków i programowanie tekstowe. Przedstawię jedno z nich, *App Lab*, pozwalające na zbudowanie za pomocą zestawu bloków aplikacji smartfonowej. Ułożone bloki można jednym kliknięciem zamienić na kod programu w JavaScript (i odwrotnie).

App Lab – aplikacja code.org

Od ponad roku `code.org`, instytucja, która zorganizowała akcję *Godzina Kodowania*, rozwija własne środowisko programowania wizualnego o nazwie *App Lab*, adresowane do uczniów 13+.

Można w nim zarówno programować wizualnie, korzystając z zestawu gotowych bloków, jak i tekstowo w języku JavaScript oraz jednym kliknięciem przechodzić od bloków do kodu, i odwrotnie. Środowisko jest nastawione na tworzenie prostych aplikacji na smartfony i tablety. Działa ono wyłącznie w wersji angielskiej. Na głównej stronie jest obfitość materiałów wprowadzających do korzystania z *App Lab*, utrzymanych w stylu charakterystycznym dla *Godziny Kodowania*. Jest wśród

[1] Artykuł został opracowany na podstawie warsztatów przygotowanych przez autora na konferencję Informatyka w Edukacji 2017 w Toruniu oraz artykułu „Między blokami a kodem programu – w poszukiwaniu środowiska do uczenia programowania" [1].



Rysunek 2. Główna strona środowiska App Lab (code.org).

nich kurs wprowadzający do programowania (CSP Unit 3 – Programming), szereg krótkich filmów demonstrujących możliwości środowiska (RESOURCES), próbki gotowych projektów (CHALLENGES) oraz rosnący zasób filmów edukacyjnych (VIDEO LIBRARY).

Przejście do budowania projektu wymaga zalogowania się na konto w `code.org`, to samo, z którego korzystamy w trakcie pracy z *Godziną Kodowania*.

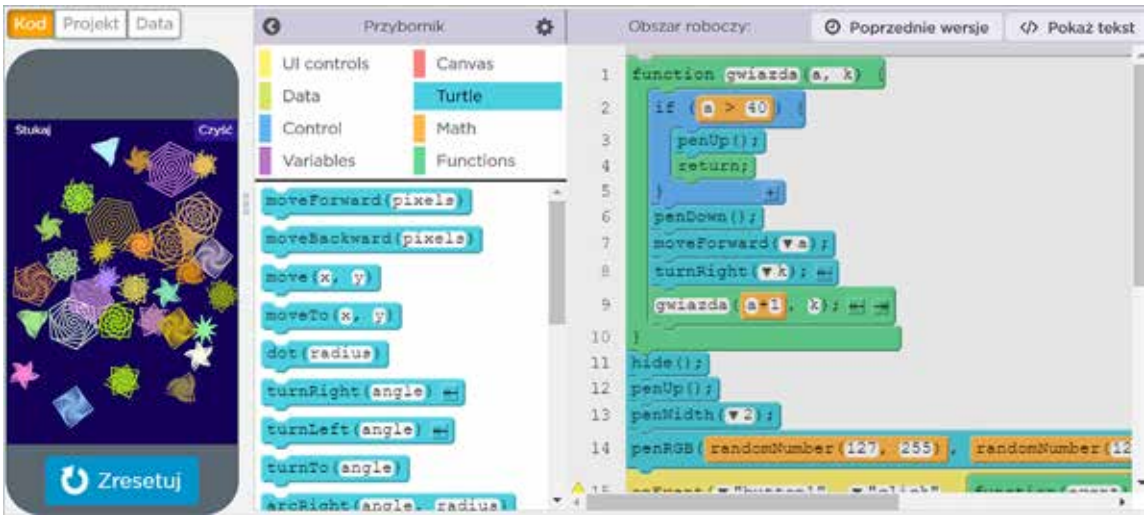
Prosty projekt – rysowanie gwiazdek

W projekcie wykorzystamy głównie zestaw bloków Turtle, pozwalający na korzystanie z grafiki żółwia. Będziemy rysować spiralne gwiazdki w miejscu kliknięcia na ekranie.

Procedura rysująca gwiazdki w wersji JavaScript wygląda następująco:

```
function gwiazda(a, k) {
  if (a > 40) {
    penUp();
    return;
  }
  penDown();
  moveForward(a);
  turnRight(k);
  gwiazda(a+1, k);
}
```

Korzysta ona z grafiki żółwia i rekurencji. Ta możliwość jest jedną z zalet środowiska, gdyż pozwala na wykorzystanie materiałów opracowanych w języku Logo. Pozostała część projektu to obsługa zdarzeń: kliknięcia na ekranie oraz kliknięcia przycisku „Czyść”.



Rysunek 3. Projekt „Gwiazdy” w środowisku App Lab – widok bloków.

```
hide();
penUp();
penWidth(2);
penRGB(randomNumber(127, 255),
       randomNumber(127, 255),
       randomNumber(0, 255));
onEvent("button1", "click",
  function(event) {
    moveTo(160, 240);
    penColor("#140058");
    dot(300);
  }
);
onEvent("screen1", "click",
  function(event) {
    moveTo(event.x, event.y);
    gwiazda(1, randomNumber(60, 179));
    penRGB(randomNumber(127,255),
           randomNumber (127,255),
           randomNumber(0,255));
  }
);
```

Ważną częścią projektu jest budowanie jego elementów, takich jak przyciski, wprowadzanie danych, etykiety, obrazki, pola tekstowe, itp. Odbywa się to w zakładce Project. Nasz projekt zawiera przycisk „Czyść".

Budowane obiekty mogą generować zdarzenia definiowane w zakładce EVENTS.

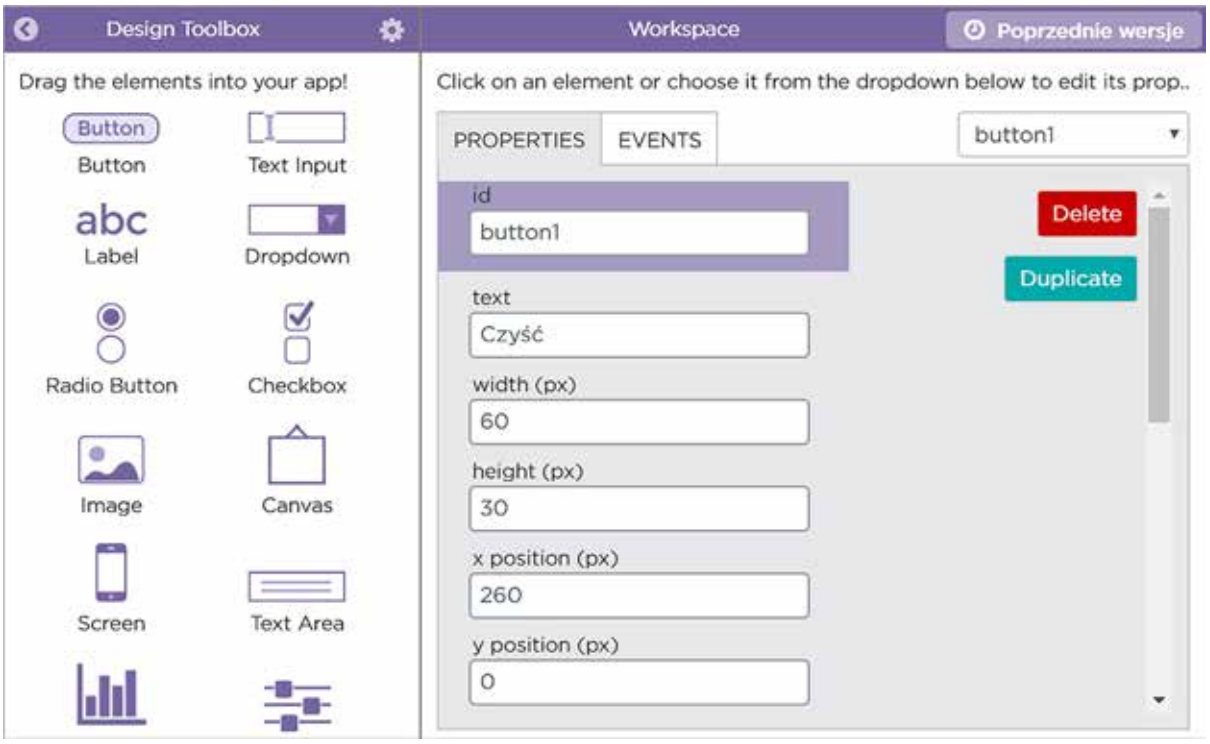
Jedną z istotnych cech środowiska, pomagającą w nauczaniu algorytmiki, jest możliwość śledzenia programu. Działa ona zarówno w wersji blokowej, jak i tekstowej. Przed uruchomieniem programu trzeba jednak przestawić szybkość jego wykonywania z „zająca" na „żółwia".

Aby uruchomić projekt na smartfonie, wystarczy przesłać do niego link, odebrać go na smartfonie i stuknąć. W pełni funkcjonalna aplikacja zostanie otwarta w przeglądarce.

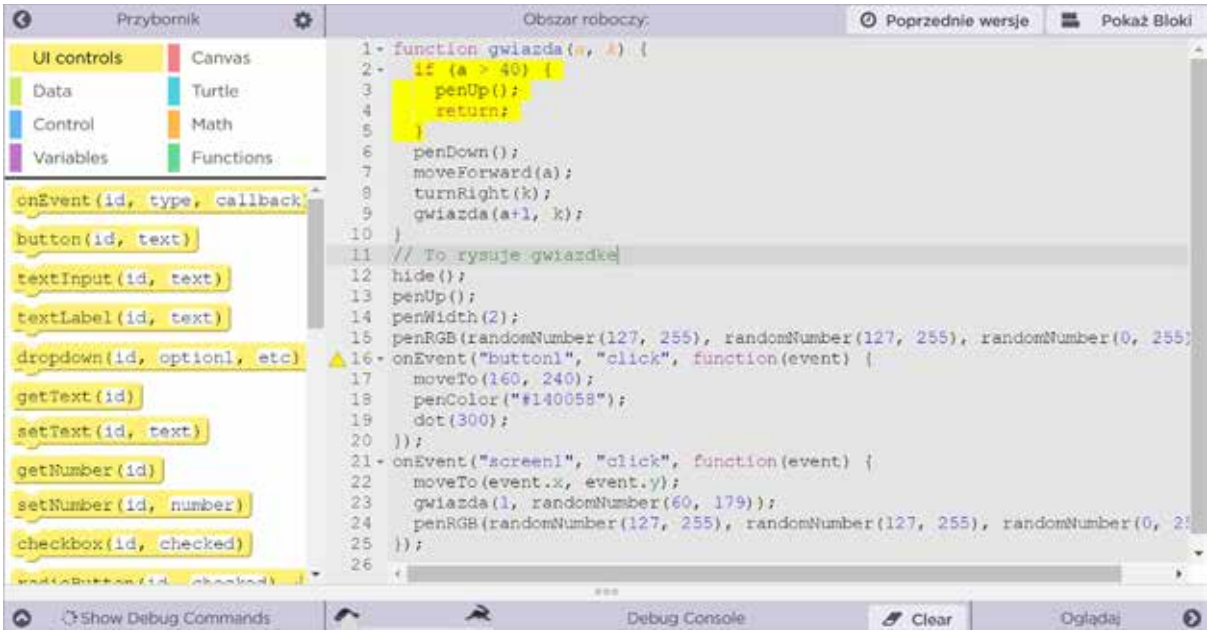
Nauczanie programowania

Nowa podstawa programowa kładzie nacisk na programowanie i algorytmikę. W klasach VII i VIII wskazane jest wprowadzanie tekstowego środowiska programowania. O ile podstawę dla klas IV-VI można w pełni zrealizować, wykorzystując środowisko wizualne, takie jak Scratch, to w dwóch ostatnich klasach szkoły podstawowej powinniśmy mieć aplikację umożliwiającą programowanie tekstowe.

App Lab pozwala zarówno na budowanie projektu z bloków, jak i wpisywanie kodu programu.



Rysunek 4. Zakładka Project w środowisku App Lab.



Rysunek 5. Projekt w App Lab, widok tekstu, śledzenie.

Nie bez znaczenia jest też możliwość korzystania w środowisku *App Lab* z bloków grafiki żółwia. Część nauczycieli przez wiele lat wykorzystywała w nauczaniu język Logo i w *App Lab* będą oni mogli wykorzystać przynajmniej część swoich materiałów. Wreszcie możliwość uruchomienia stworzonej aplikacji na smartfonie może dodatkowo motywować uczniów do pracy w tym środowisku.

Bibliografia

1. Kranas W. *Między blokami a kodem programu – w poszukiwaniu środowiska do uczenia programowania*, Informatyka w Edukacji, Toruń 2017.

2. Dokumentacja środowiska *App Lab*: https://docs.code.org/applab, dostęp sierpień 2017.

3. Projekt „Gwiazdy" w środowisku *App Lab*: https://studio.code.org/projects/applab/h4ca4yTVkbPDCHn2x2Ephw, dostęp maj 2017.

4. Nowa podstawa programowa z informatyki dla szkoły podstawowej: https://men.gov.pl/wp-content/uploads/2016/11/podstawa-programowa-z-informatyki-szkola-podstawowa.pdf, dostęp sierpień 2017.

5. Strona główna środowiska App Lab: https://code.org/educate/applab, dostęp sierpień 2017.

Witold KRANAS jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Czy można zaprzyjaźnić się z Pythonem?

Wanda JOCHEMCZYK, dr Katarzyna OLĘDZKA

Mimo podejmowania wielu wysiłków człowiekowi nie udaje się oswoić wszystkich zwierząt. Istnieje przekonanie, że do zwierząt, których nie sposób oswoić, należą węże. Gad ten jedynie przyzwyczaja się do obecności człowieka w swoim otoczeniu. Możemy nauczyć się dbać o węża i polubić jego obecność, ale on ze swojej natury zawsze pozostanie dzikim stworzeniem, a więc w dużym stopniu nieprzewidywalnym w swoim zachowaniu. Należy pamiętać, dla własnego dobra i dobra innych ludzi, że spotkanie z wężem może być niebezpieczne, szczególnie, gdy jest on jadowity. Nie o tym jednak będziemy rozważać. Opiszemy naszą przygodę z Pythonem – językiem programowania, którego nazwa w gruncie rzeczy nie pochodzi od zwierzęcia – pełzającego gada. Nie odwołuje się także od potwora znanego z mitologii greckiej. W latach 70. BBC emitowało serial komediowy zatytułowany „Monty Python's Flying Circus" – Latający Cyrk Monty Pythona. To właśnie do tego serialu odnosi się nazwa języka, za którego twórcę uznawany jest Guido van Rossum, holenderski programista.

Rysunek 1. Logo Pythona.

Dlaczego Python?

Istnieje wiele różnych języków programowania. Dlaczego zatem promujemy naukę właśnie języka Python? Zapytaliśmy o walory edukacyjne tego języka naszych nauczycieli – uczestników szkolenia e-learningowego dotyczącego programowania w Pythonie.

- *Jego jasna i prosta składnia pozwala skupić się na faktycznym problemie i rozwiązaniu zadania. Uczy on dobrych nawyków, m.in. organizacji kodu.* (p. Marta)

- *Czy Python? Początkowe trudności z wcięciami mijają, a po pewnym czasie są postrzegane jako ułatwienie. Właściwe też jest początkowe wprowadzanie grafiki, a następnie algorytmiki.* (p. Andrzej)

- *Python jest bardzo często wykorzystywany do tworzenia prototypów rozwiązań, bo osiąga się szybkie efekty w krótkim czasie. Jest to język o bardzo szerokim spektrum zastosowań. Pewnie ciężko dziś znaleźć dziedzinę, w której nie dałoby się go zastosować.* (p. Grzegorz)

- *Python jest językiem wymagającym bardzo precyzyjnego zapisu, ważne jest każde wcięcie, dwukropek. Przez to kod jest bardzo klarowny, jasny. Uczeń od razu uczy się struktury pętli i widzi, co w tej pętli chce zawrzeć.* (p. Katarzyna)

I jeszcze jedna wypowiedź dotycząca nauki programowania:

- *Rozwiązywanie problemów to nic innego, jak tworzenie algorytmów. Już niemowlęta tworzą algorytmy polegające na odpowiednio długim darciu dzioba czy wylewaniu łez, żeby uzyskać wzięcie na ręce, jedzenie czy suchą pieluchę na... Wówczas jeszcze nie mają świadomości, że właśnie wymyślają algorytm działania, żeby uzyskać zaplanowany efekt, czyli rozwiązać napotkany problem.* (p. Krzysztof)

Warto podkreślić, że w Pythonie mogą programować zarówno początkujący, nawet uczniowie szkoły podstawowej, jak i osoby zawodowo zajmujące się programowaniem. Ten aspekt może zachęcić uczniów do poważnego zajęcia się programowaniem. W Pythonie można prosto zapisać wiele algorytmów, a przy wykorzystaniu dostępnych bibliotek przygotować oprogramowanie dostosowane na naszych potrzeb. Ponadto sprzyja on zwięzłemu zapisowi i wymusza poprawne formatowanie kodu poprzez ścisłe zasady stosowania wcięć. Kolejną zaletą edukacyjną jest możliwość stosowania strukturalnego, obiektowego lub funkcyjnego stylu programowania. Jako projekt Open Source jest darmowy, a jego interpretery są dostępne na wiele systemów operacyjnych, m.in. Windows, GNU/Linux, MacOS. Jest to bardzo ważne w edukacji, ponieważ nie promujemy komercyjnych produktów, ale wprowadzamy uczniów w świat wolnego oprogramowania. Wkrótce także będzie można zdawać egzamin maturalny z informatyki w tym języku.

Python już w szkole podstawowej

Najmłodszych uczniów wprowadzamy w świat programowania, korzystając z języka wizualnego. Uczeń, tworząc program, nie tyle wpisuje polecenia z klawiatury, co przeciąga bloczki. Nie musi pamiętać składni poleceń, ale powinien je rozumieć, by efektywnie stosować. W zasadzie nie popełnia też błędów składniowych. Dlatego dzieci chętnie zaczynają naukę programowania od Scratcha lub międzynarodowego projektu *Godzina Kodowania*. Jednak, gdy chcemy napisać trochę dłuższy program, okazuje się, że bloczki stają się pewną przeszkodą. Kod jest długi i mało czytelny, stąd w sposób naturalny rodzi się potrzeba przejścia na tekstowy język programowania. Warto jednak pamiętać, że przejście to powinno być łagodne, a uczniom należy stawiać takie zadania, które zachęcą ich do pracy, a nie zniechęcą do nauki programowania.

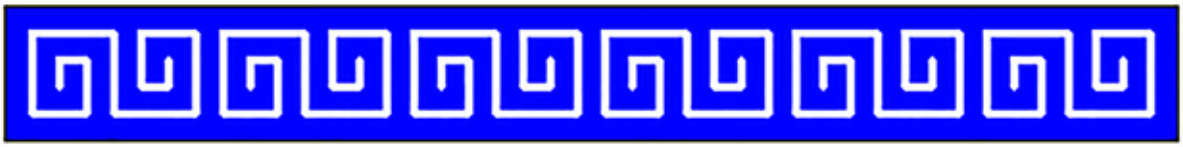
Jak rozpocząć i dlaczego? Proponujemy zacząć naukę programowania od sterowania obiektem na ekranie. Po pierwsze, stosujemy proste, a przez to zrozumiałe komendy, zbliżone do języka naturalnego – idź naprzód o daną liczbę kroków, obróć się o podany kąt, podnieś pisak itp. Po drugie, postać żółwia to graficzny symbol, który pozwala dziecku wyobrazić sobie wykonawcę algorytmu zapisanego w języku programowania. Po trzecie, równie ważna, jak dwa poprzednie aspekty, jest semantyka operacyjna. Uczeń widzi efekty działania swoich programów. Interpretacja kodu powoduje na ekranie skutek, który może ocenić pod względem zgodności ze wzorcem. Uczeń obserwuje, jak żółw rysuje, i ocenia, czy wykonuje zadanie prawidłowo. Może też znaleźć ewentualny błąd. Wszystkie te elementy znajdziemy zarówno w Scratchu – gdy rysujemy, wykorzystując pisak – jak i w projekcie *Godzina Kodowania* – sterując różnymi obiektami, w Logo, a także w Pythonie – korzystając z modułu Turtle. Rozszerzamy w ten sposób ideę Seymoura Paperta – programowania grafiki z wykorzystaniem żółwia – i zachęcamy uczniów zarówno do nauki, jak i do zabawy.

Sterowanie żółwiem

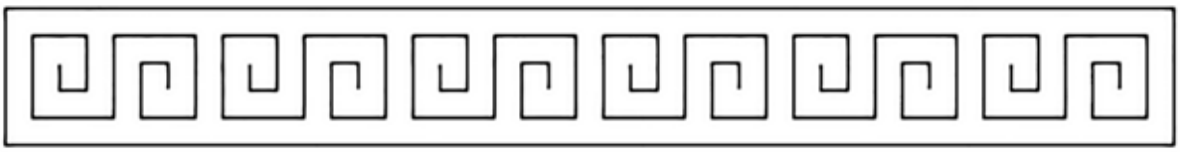
Do rysowania wykorzystujemy polecenia:

- `fd(a)` – przesuwa żółwia o podaną liczbę kroków, w zależności od stanu pisaka żółw rysuje kreskę (pisak opuszczony) lub nie rysuje (pisak podniesiony),
- `rt(kąt)` – obraca żółwia w prawo o podany kąt,
- `lt(kąt)` – obraca żółwia w lewo o podany kąt,
- `pu() / pd()` – podnosi / opuszcza pisak.

Podczas tworzenia rysunków z wykorzystaniem grafiki żółwia jedną z podstawowych umiejętności jest odnajdywanie elementów powtarzających się



Rysunek 2. Motyw grecki.



Rysunek 3. Motyw grecki – rysunek konturowy.

i odpowiednie stosowanie instrukcji iteracji. O ile przy prostych przykładach uczniowie radzą sobie dobrze z zadaniem, przy bardziej skomplikowanych pojawiają się trudności. Warto zauważyć, że zagadnienie iteracji obejmuje zadania, w których występuje proste powtarzanie, powtarzanie ze zmiennym elementem oraz zagnieżdżone pętle.

Przyjrzyjmy się następującemu zadaniu. Mamy napisać funkcję, po wywołaniu której zostanie narysowany motyw grecki (rysunek 2).

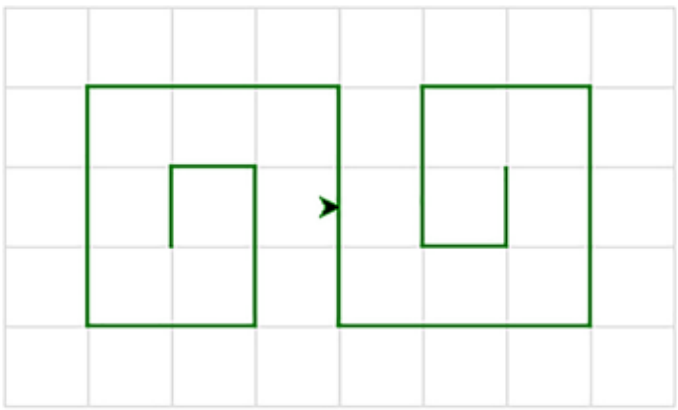
Od czego zacząć rozwiązywanie takiego zadania? Spróbujmy przedstawić rysunek motywu w sposób bardziej przejrzysty, aby widać było rysowane elementy.

Następnie możemy narysować powtarzające się elementy na pomocniczej kratce, aby dokładnie zbadać proporcje jednego elementu.

Miejsce, w którym widzimy żółwia, jest optymalnym miejscem rozpoczęcia rysowania. Żółw powinien rozpocząć i zakończyć rysowanie w tym samym miejscu. Warto stosować taką zasadę, gdyż ułatwia to łączenie elementów.

```
from turtle import *
def zawijas(a):
   for k in range(2):
      lt(90)
      fd(1.5*a);lt(90)
   for i in range(2):
      fd(3*a);lt(90)
   for i in range(2):
      fd(2*a);lt(90)
   for i in range(2):
      fd(a);lt(90)
   pu();fd(2*a);rt(90);
   bk(0.5*a);rt(90);pd()
```

Najtrudniejszy fragment rozwiązania jest poza nami. Zostało narysowanie prostokąta i powtórzenie 6 razy zdefiniowanego elementu. Motyw grecki jest rysowany białym pisakiem na niebieskim tle, musimy zatem wybrać odpowiednią kolejność rysowania, aby efekt był taki, jak na wzorcowym rysunku. Jako pierwszy narysujemy prostokąt zamalowany kolorem niebieskim, następnie po zmianie koloru pisaka na biały narysujemy 6 powtarzających się elementów.



Rysunek 4. Element powtarzający się, narysowany na pomocniczej kratce.

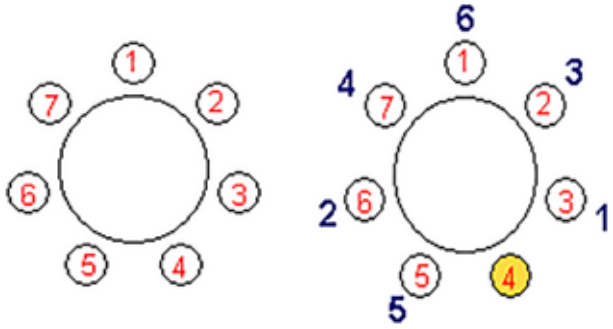
```
def motyw():
   #ustalenie wielkości
   a=10
   fillcolor("blue")
   #rysowanie prostokąta
   begin_fill()
   for i in range(2):
      fd(43*a);lt(90);fd(5*a);lt(90)
   end_fill()
   #przejście w odpowiednie miejsce
   pu();fd(4*a);lt(90)
   fd(2.5*a);rt(90);pd()
   pensize(3)
   pencolor("white")
   #rysowanie 6 elementów
   for i in range(6):
      zawijas(a)
      pu();fd(7*a);pd()
```

Rysowanie złożonego motywu trwa długo. Żółw wykonuje powoli każdy ruch. Ułatwia to naukę, ale przeszkadza w testowaniu, zajmuje za dużo czasu. Warto przyśpieszyć rysowanie, uruchamiając tworzenie motywu w pamięci i na końcu wyświetlając go na ekranie za pomocą poleceń:

```
tracer(0);motyw();update()
```

Przykład całkiem poważnego problemu algorytmicznego

Przyjrzyjmy się zadaniu „Okrągły stół". Zadanie nawiązuje do zagadnienia zwanego problemem Józefa Flawiusza, a pochodzi z 3. etapu Konkursu Informatycznego dla gimnazjalistów LOGIA13 (rok szkolny 2012/2013). Konkurs jest rokrocznie organizowany dla uczniów szkół województwa mazowieckiego (`http://logia.oeiizk.waw.pl`).



Rysunek 5. Osoby wstają kolejno z krzeseł o numerach: 3, 6, 2, 7, 5, 1

Przy okrągłym stole siedzi **n** uczestników spotkania, na krzesłach ponumerowanych od **1** do **n**. Kolejno co **k**-ta osoba wstaje i opuszcza spotkanie. Zadaniem Antka jest wskazanie osoby, która pozostanie przy stole jako ostatnia. Pomóż Antkowi i napisz funkcję **ostatni(n, k)**. Wynikiem funkcji jest numer krzesła zajętego przez uczestnika spotkania, który pozostanie przy stole. Parametry **n** i **k** mogą przyjmować wartości z zakresu od **1** do **100**. Przykład: wynikiem **ostatni(7, 3)** jest **4**.

Nasuwającym się rozwiązaniem jest stworzenie listy wszystkich uczestników i kolejno wykreślanie osoby opuszczającej miejsce. Powstaje pytanie, jak usuwać osobę, by móc efektywnie wskazać kolejną do wykreślenia. Jednym z rozwiązań, choć nie jedynym, jest budowanie za każdym razem nowej listy uczestników tak, by pierwszą część stanowiła lista uczestników bezpośrednio po usuniętym, a drugą część lista osób do tej, którą usuwamy. Dokładniej, tak długo jak lista zawiera więcej niż jednego uczestnika, wykonuj następujące kroki:

- oblicz numer osoby, którą trzeba usunąć (dana liczba `k` może być większa niż długość listy, wiec stosujemy operację modulo),
- zbuduj listę `pocz` od pierwszego elementu, do osoby usuwanej (bez tej osoby),
- zbuduj listę `kon` osób występujących bezpośrednio za usuwaną,
- połącz obie listy `kon` i `pocz`.

Wynikiem jest pierwszy i jedyny element listy.

```
def ostatni(n, k):
   #tworzenie listy od 1 do n
   lista = []
   for i in range(1, n+1):
      lista.append(i)
   #kolejno wstaje jeden uczestnik,
   #aż zostanie 1
   while len(lista) > 1:
      #gdy index większy od długości
      #listy
      nr = (k-1) % len(lista)
      #tworzymy nową listę bez numeru
      #usuniętego
      #i z początkiem na końcu
      pocz = lista[:nr]
      kon = lista[nr+1:]
      lista = kon + pocz
   return lista[0]
#przykłady wywołań
print(ostatni(7,3))
print(ostatni(6,2))
```

Na komputerze i w chmurze

W języku Python możemy pracować lokalnie na komputerze, po wcześniejszej jego instalacji. Warto dodać, że w różnych dystrybucjach Linuksa oraz MacOS-a Python jest już zainstalowany z systemem. Możemy też korzystać z serwisów, np. `http://ideone.com` lub `http://pythontutor.com`. Otwieramy przeglądarkę internetową, a następnie piszemy i uruchamiamy nasz program. W tym drugim serwisie możemy napisać skrypt, a następnie go uruchomić i wykonać krok po kroku. Dodatkowo mamy narzędzie do wizualizacji – możemy obserwować, jakie są bieżące wartości zmiennych i jak się zmieniają w czasie. Znajdziemy tam kilkanaście przykładów zaimplementowanych podstawowych algorytmów.

Warto również dodać, że mamy dostępne różne biblioteki i dodatki do tego języka. Przykładem przydatnej biblioteki dla uczniów jest *Pygame.* Jest ona przeznaczona do tworzenia gier oraz aplikacji multimedialnych. Można tworzyć w nim grafikę statystyczną i dynamiczną, odtwarzać i edytować dźwięki czy korzystać z materiałów filmowych. W sieci opublikowano wiele programów stworzonych z wykorzystaniem biblioteki *Pygame*, można je pobrać i z nich skorzystać oraz „podejrzeć", jak zostały zrobione, by się wiele nauczyć. Z kolei rysować wykresy można z wykorzystaniem biblioteki *SageMath* (*Software for Algebra and Geometry Experimentation*). Jest to dość rozbudowany pakiet, w którym obok wykresów w 2D i 3D możemy zajmować się zagadnieniami związanymi z różnego typu obliczeniami, rozwiązywaniem równań czy zagadnieniami z dziedziny kombinatoryki.

Nasze spotkanie z Pythonem

Zbliżając się do końca, pragniemy podzielić się naszym doświadczeniem uczenia się i propagowania języka Python. Zainteresowałyśmy się nim w czasie udziału w kursie e-learningowym *An Introduction to Interactive Programming in Python* prowadzonym przez profesorów z Department of Computer Science Rice University w Houston (USA) na portalu `www.coursera.org`. Szkolenie dotyczyło tworzenia aplikacji interaktywnych w tym języku. Postanowiłyśmy poszerzyć zdobytą wiedzę i bliżej zainteresować się Pythonem. Potem zaproponowałyśmy szkolenia dla nauczycieli i konkurs dla uczniów. Obecnie prowadzimy całą serię szkoleń, głównie w systemie online. Na stronie `python.oeiizk.edu.pl` są dostępne materiały pomocnicze wprowadzające do programowania w tym języku. Chociaż Python nie jest jedynym językiem, w którym można stawiać pierwsze kroki w programowaniu, jest dość specyficzny i wart poznania. Zachęcamy naszych Czytelników, o ile jeszcze tego nie uczynili, aby rozpoczęli swoją przygodę z Pythonem.

Bibliografia

1. Borowiecki M. *Python na lekcjach informatyki w szkole ponadgimnazjalnej*, IwE 2013.
2. Jochemczyk W., Olędzka K. *Python dla wszystkich*, IwE 2016.
3. Jochemczyk W., Olędzka K. *Pythonowe wyzwania dla początkujących*, IwE 2017.
4. https://code.org
5. http://python.oeiizk.edu.pl
6. http://python.org

Wanda JOCHEMCZYK i **dr Katarzyna OLĘDZKA** są nauczycielami konsultantami w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

25 lat Polskiej Olimpiady Informatycznej. Następny krok[1]

Prof. dr hab. Krzysztof DIKS

Olimpiada Informatyczna (www.oi.edu.pl) jest jedną z 17 olimpiad przedmiotowych organizowanych pod egidą Ministerstwa Edukacji Narodowej, których laureaci i finaliści są zwolnieni z egzaminu maturalnego z danego przedmiotu. Pierwsza edycja Olimpiady Informatycznej miała miejsce w roku szkolnym 1993/1994. W roku szkolnym 2017/2018 Olimpiada Informatyczna po raz dwudziesty piąty zaoferuje uczniom zainteresowanym informatyką możliwość sprawdzenia swojej wiedzy i umiejętności w zakresie układania algorytmów i programowania. Wyniki laureatów Olimpiady Informatycznej w rywalizacji z rówieśnikami z całego świata sytuują naszą Olimpiadę wśród najlepszych krajowych olimpiad przedmiotowych. Pod względem liczby wszystkich medali zdobytych w Międzynarodowej Olimpiadzie Informatycznej (`http://www.ioinformatics.org`) Polska ze 105 medalami zajmuje drugie miejsce na świecie po Chinach (115 medali). Na 105 polskich medali składa się 38 medali złotych, 38 medali srebrnych i 29 medali brązowych, co w klasyfikacji medalowej daje nam czwartą pozycję na świecie po Chinach (77 medali złotych, 26 medali srebrnych, 12 medali brązowych), Rosji (odpowiednio 56, 36, 12) i Stanach Zjednoczonych (46, 34, 15). W tym artykule pokazuję, że ten niewątpliwy sukces młodych polskich informatyków nie jest przypadkowy oraz postaram się wskazać kierunki działań, które umożliwią wykorzystanie doświadczeń Olimpiady Informatycznej w powszechnej edukacji uczniów uzdolnionych informatyczne tak, żeby można było w pełni zrealizować hasło symbolizujące cel wszelkich przedsięwzięć związanych z tą Olimpiadą: *Informatyka specjalnością młodych Polaków*.

Olimpiada Informatyczna została powołana przez Instytut Informatyki Uniwersytetu Wrocławskiego 10 grudnia 1993 roku na podstawie zarządzenia nr 28 Ministra Edukacji Narodowej z dnia 14 września 1992 roku w sprawie organizacji konkursów i olimpiad przedmiotowych. W skład pierwszego Komitetu Głównego Olimpiady Informatycznej weszli wybitni naukowcy, edukatorzy i popularyzatorzy informatyki oraz przedstawiciele Ministerstwa Edukacji Narodowej odpowiedzialni za edukację informatyczną:

- prof. dr hab. Jacek Błażewicz
  Politechnika Poznańska
- prof. dr hab. Jan Madey
  Uniwersytet Warszawski
- prof. dr hab. Andrzej W. Mostowski
  Uniwersytet Gdański
- prof. dr hab. Wojciech Rytter
  Uniwersytet Warszawski
- prof. dr hab. Maciej M. Sysło
  Uniwersytet Wrocławski
- dr hab. inż. Stanisław Waligórski
  Uniwersytet Warszawski
- dr Piotr Chrząstowski-Wachtel
  Uniwersytet Warszawski
- dr Andrzej Walat OEIiZK

[1] W artykule wykorzystano fragmenty artykułu „Od szkolnych konkursów programistycznych do sukcesów w zawodzie informatyka" autorstwa Krzysztof Diksa i Jana Madeya oraz opracowanie „CMI – Centrum Mistrzostwa Informatycznego. Całościowa koncepcja wzmocnienia polskiej edukacji informatycznej w kierunku zwiększonego kształcenia wybitnych specjalistów" autorstwa Krzysztofa Diksa, Andrzeja Kisielewicza i Krzysztofa Lorysia.

- dr Bolesław Wojdyło UMK
- mgr Jerzy Dałek MEN
- mgr Jerzy Święcicki MEN
- Tadeusz Kuran OEIiZK
- mgr Krystyna Kominek
  II LO im. Stefana Batorego Warszawie

Komitet Główny wybrał następujących członków prezydium:

- przewodniczący
  dr hab. inż. Stanisław Waligórski
- zastępca przewodniczącego
  prof. dr hab. Maciej Sysło
- sekretarz naukowy dr Andrzej Walat
- kierownik organizacyjny Tadeusz Kuran
- sekretarz mgr Krystyna Kominek

Siedzibą Olimpiady Informatycznej niezmiennie od samego początku jest Ośrodek Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Twórcy Olimpiady Informatycznej wykonali tytaniczną pracę – opracowali standardy organizacji i przeprowadzania Olimpiady, które w swoich podstawach obowiązują do dziś. W przyjętym regulaminie określono następujące cele Olimpiady Informatycznej:

1. Stworzenie motywacji dla zainteresowania nauczycieli i uczniów nowymi metodami informatyki.
2. Rozszerzenie współdziałania nauczycieli akademickich z nauczycielami szkół w kształceniu młodzieży uzdolnionej.
3. Stymulowanie aktywności poznawczej młodzieży informatycznie uzdolnionej.
4. Kształtowanie umiejętności samodzielnego zdobywania i rozszerzania wiedzy informatycznej.
5. Stwarzanie młodzieży możliwości szlachetnego współzawodnictwa w rozwijaniu swoich uzdolnień, a nauczycielom – warunków twórczej pracy z młodzieżą.
6. Wyłanianie reprezentacji Rzeczypospolitej Polskiej na Międzynarodową Olimpiadę Informatyczną.

Jak już wspomniałem, Olimpiada Informatyczna jest konkursem przedmiotowym dla uczniów szkół średnich, ale mogą w niej startować także młodsi uczniowie. Zwycięzcy (pierwsza czwórka) krajowych konkursów programistycznych reprezentują swoje kraje w zawodach Międzynarodowej Olimpiady Informatycznej (IOI). Polska Olimpiada Informatyczna składa się z trzech etapów. Pierwszy etap jest etapem szkolnym rozgrywanym na przełomie października i listopada i gromadzi około tysiąca uczestników. Uczestnicy mają do rozwiązania zazwyczaj pięć zadań i pracują nad nimi w domu. Wyniki swojej pracy przesyłają przez Internet organizatorom do oceny. Do drugiego etapu awansuje około 360 najlepszych zawodników z etapu pierwszego. Drugi etap jest organizowany w kilku ośrodkach regionalnych współpracujących ściśle z najlepszymi uczelniami informatycznymi w kraju i trwa trzy dni. Pierwszy dzień jest poświęcony na zapoznanie się z warunkami rozgrywania zawodów. W każdym z następnych dwóch dni uczestnicy mają do samodzielnego rozwiązania 2-3 zadania w trakcie pięciogodzinnej kontrolowanej sesji. Rozwiązania z całej Polski są zbierane centralnie i wszystkie są oceniane w takim samym środowisku i na tych samych testach. Około 80 najlepszych uczestników drugiego etapu awansuje do finału Olimpiady. Finał jest rozgrywany w jednym miejscu i trwa pięć dni. Trzy dni są przeznaczone na same zawody, a dwa dni na rekreację i turystykę. Sposób rozgrywania finału jest podobny do tego z etapu drugiego. Czterej najlepsi zawodnicy z finału reprezentują Polskę na międzynarodowych zawodach informatycznych, w tym Międzynarodowej Olimpiadzie Informatycznej. Międzynarodowa Olimpiada Informatyczna jest rozgrywana latem każdego roku i gromadzi najlepszych na świecie młodych informatyków – uczniów szkół średnich. Pierwsza Międzynarodowa Olimpiada Informatyczna miała miejsce w roku 1989. W latach 2006 i 2007 Polacy (odpowiednio) Filip Wolski i Tomek Kulczyński zostali absolutnymi zwycięzcami Międzynarodowej Olimpiady Informatycznej. Do polskich multimedalistów Międzynarodowej Olimpiady Informatycznej należą: Filip Wolski (4 medale złote), Andrzej Gąsienica-Samek (3 medale złote, 1 medal srebrny), Marcin Andrychowicz i Jarosław Kwiecień (po 3 medale złote). Cała czwórka znajduje się w pierwszej dziesiątce multimedalistów Międzynarodowej Olimpiady Informatycznej.

Sukces Olimpiady Informatycznej wynika przede wszystkim z niezwykle wysokiego poziomu organizacyjnego i merytorycznego samego konkursu, który pełni rolę nie do przecenienia w wyławianiu i kształceniu uczniów wyjątkowo uzdolnionych informatycznie. Uczestnictwo i sukcesy w Olimpiadzie wymagają od uczniów wiedzy i umiejętności wybiegających daleko poza wymagania szkolne. Minimalnych wymogów (http://ioi2017.org/files/ioi-syllabus-2017.pdf) stawianych uczestnikom Międzynarodowej Olimpiady Informatycznej nie spełniają często studenci wielu uczelni informatycznych w kraju. Olimpiada dotyka jądra informatyki – algorytmiki i programowania – a wiedza i umiejętności zdobyte podczas Olimpiady nie są ulotne i dają niezbędne podwaliny do dalszego dziedzinowego rozwoju. Jeszcze ważniejsze jest, że konkurs kształci w młodych ludziach umiejętności, które są niezbędne w ich późniejszej aktywności zawodowej: pracowitość, systematyczność, samodyscyplina, dociekliwość, samodoskonalenie, praca w zespole, uczciwość, ambicja, chęć konkurowania, dążenie do sukcesu. Startowanie w Olimpiadzie jest wyzwaniem intelektualnym dla młodego człowieka, a sukces nobilituje. Z drugiej strony organizatorzy Olimpiady dbają o to, żeby uczestnicy mieli okazję poznać się i nawiązać bliskie kontakty, które później mogą zaowocować w ich życiu zawodowym.

Na sukces Olimpiady ma także wpływ ścisła systemowa współpraca Ministerstwa Edukacji Narodowej, najlepszych uczelni w kraju (Uniwersytet Warszawski, Uniwersytet Jagielloński, Uniwersytet Wrocławski, Uniwersytet im. Mikołaja Kopernika w Toruniu, Politechnika Białostocka, Politechnika Gdańska, Politechnika Poznańska, Akademia Górniczo-Hutnicza w Krakowie, Politechnika Śląska), nauczycieli i firm IT w wyławianiu uczniów utalentowanych informatycznie i rozwijaniu ich talentów. Cele te są realizowane poprzez umożliwienie uczniom szlachetnej rywalizacji w rozwiązywaniu zadań informatycznych. Zadania są przygotowywane zarówno przez naukowców-informatyków o światowej renomie, jak i byłych uczestników Olimpiady, osiągających sukcesy w konkursach studenckich. Co więcej, byli olimpijczycy aktywnie uczestniczą w pracach Olimpiady, przygotowując wzorcowe rozwiązania zadań olimpijskich i są autorami wyrafinowanego oprogramowania olimpijskiego, służącego do automatyzacji prac przy Olimpiadzie, w szczególności automatycznego sprawdzania rozwiązań zawodników.

Olimpiada prowadzi intensywną działalność edukacyjną. Co roku wydawane są materiały poolimpijskie, zawierające szczegółową dyskusję na temat rozwiązań zadań olimpijskich oraz wzorcowe programy. Byli olimpijczycy prowadzą portal edukacyjny dla początkujących adeptów programowania i algorytmiki (szkopul.edu.pl), dzięki któremu nawet uczniowie z małych miasteczek i wiosek mają szansę poznawać tajniki „prawdziwej" informatyki. Finaliści Olimpiady mają co roku okazję uczestniczyć w wakacyjnych obozach wypoczynkowo-naukowych, na których wysłuchują wykładów przygotowywanych przez pracowników naukowych i starszych kolegów. Mogą też doskonalić swoje umiejętności algorytmiczno-programistyczne, biorąc udział w praktycznych warsztatach programistycznych.

Ważną rolę w rozwoju najlepszych odgrywa też Krajowy Fundusz na Rzecz Dzieci – organizacja, która statutowo zajmuje się młodzieżą uzdolnioną (nie tylko informatycznie). Na warsztatach i obozach Funduszu uczniowie mają szanse poznawać dziedziny informatyki, z którymi niekoniecznie muszą zetknąć się, startując w konkursach.

Współpracownicy Olimpiady są często autorami lub tłumaczami najważniejszych podręczników informatycznych, które umożliwiają młodym ludziom naukę w języku ojczystym.

Pierwszą osobą, która może dostrzec talent ucznia i skierować go na właściwe drogi rozwoju, jest jego nauczyciel. Dlatego w ramach Olimpiady organizowane są warsztaty olimpijskie dla nauczycieli, podczas których mogą oni zapoznać się praktycznie ze specyfiką konkursów informatycznych.

Tak szeroka działalność nie byłaby możliwa bez wsparcia finansowego. Działalność Olimpiady Informatycznej jest finansowana zarówno ze środków publicznych (dotacja Ministerstwa Edukacji Narodowej), jak i środków prywatnych. Ograniczenia finansowe są jednak największą bolączką w upowszechnianiu i edukacji informatycznej mającej na celu przyciągnięcie większej rzeszy młodzieży.

Niemniej ważne dla sukcesu Olimpiady jest to, że najlepsi olimpijczycy, zwycięzcy konkursów programistycznych, tworzą elitę i są wzorcem dla następnych pokoleń. Bycie elitą nobilituje. Sprzyja temu też polityka czołowych uczelni w kraju, które przyjmują finalistów Olimpiady na studia bez postępowania kwalifikacyjnego.

Podsumowując, kluczami do sukcesu Olimpiady Informatycznej są:

- pasjonaci informatyki zaangażowani w jej promowanie i organizację (naukowcy, nauczyciele, studenci, uczniowie),
- wspierające instytucje (uczelnie wyższe, Ministerstwo Edukacji Narodowej, szkoły, Ośrodek Edukacji Informatycznej i Zastosowań Komputerów w Warszawie, firmy informatyczne),
- bezpieczeństwo finansowe, choć ciągle niedostateczne (MEN, firmy prywatne),
- wysoka jakość (zarówno przygotowywanych zadań, jak i organizacyjna).

Podczas zawodów olimpijskich uczniowie dostają pewną liczbę zadań do rozwiązania, z których każde składa się z krótkiej historyjki przedstawiającej sytuację problemową. Rozwiązaniem zadania jest zazwyczaj algorytm zapisany w postaci programu w wybranym przez zawodnika (algorytmicznym) języku programowania. Najczęściej używanym obecnie językiem programowania jest C++. W ostatnich latach obserwuje się wyraźny spadek zainteresowania językiem programowania Pascal, nad czym bardzo boleję, ponieważ Pascal jest doskonałym dydaktycznym językiem programowania – zwartym, ale na tyle bogatym, żeby rozwinąć umiejętności programowania od najprostszych do najbardziej wyrafinowanych.

Poprawnie kompilujące się programy są następnie uruchamiane na nieznanych zawodnikom testach przygotowanych przez organizatorów. Testy są tak dobrane, żeby wykrywały programy niepoprawne i różnicowały rozwiązania o różnej złożoności obliczeniowej, przy czym głównie chodzi o złożoność czasową, a złożoność pamięciowa jest wymuszana przez podane explicite ograniczenia na wielkość wykorzystywanej przez program pamięci. W Olimpiadzie liczba punktów otrzymana za zadanie zależy od jakości zaproponowanego algorytmu i jego implementacji.

Przedstawię teraz przykład zadania olimpijskiego, żeby Czytelnik mógł sam posmakować, na czym polega start w Olimpiadzie Informatycznej. Zadanie opisane poniżej, autorstwa Piotra Chrząstowskiego-Wachtela, pochodzi z zawodów I stopnia I Olimpiady Informatycznej.

Zadanie trójkąty

autor: Piotr Chrząstowski-Wachtel

Dany jest skończony, co najmniej trzyelementowy zbiór A odcinków o długościach wymiernych. Chcemy zbadać, czy z każdych trzech odcinków z tego zbioru można zbudować trójkąt.

Zestaw danych wejściowych jest co najmniej trzyelementowym ciągiem długości wszystkich odcinków ze zbioru A ułożonych w jakiejś kolejności. Każda długość odcinka (liczba wymierna) jest zapisana w postaci *licznik/mianownik*, gdzie *licznik* oraz *mianownik* są dodatnimi liczbami całkowitymi nie większymi niż 9999. Kolejne długości w tym ciągu są oddzielone odstępem lub pojedynczym znakiem końca wiersza.

Należy otrzymać odpowiedź:

» TAK, jeśli z każdej trójki odcinków można zbudować trójkąt.
» NIE, jeśli nie z każdej trójki odcinków można zbudować trójkąt.
» NONSENS, jeśli zestaw danych jest niepoprawny, to znaczy nie spełnia podanych wyżej warunków.

Przykłady

Dla zestawu danych wejściowych:
13/10 1/2 6/5 11/6 9/7 3/5 9/7 13/10 9/5 8/5
odpowiedź brzmi NIE, bo na przykład nie da się zbudować trójkąta z odcinków o długościach: 6/5 3/5 9/5.

Dla zestawu danych wejściowych:
1/2 3/5 2/3 4/7 1/1 4/6
odpowiedź brzmi TAK.

Dla zestawu danych wejściowych:
1/2 3/5 2/3 4/7 1 4/6
odpowiedź brzmi NONSENS, bo 1 nie jest parą liczb przedzielonych znakiem /.

Zadanie

Ułóż program, który kolejno dla każdego zestawu danych z pliku TKT.IN generuje właściwą odpowiedź: TAK, NIE albo NONSENS i zapisuje ją w pliku TKT.OUT.

Już to pierwsze zadanie z zawodów I stopnia I Olimpiady Informatycznej doskonale charakteryzuje zadania, z którym mierzą się uczniowie w zawodach olimpijskich:

- każde zadanie wymaga dokładnej analizy popartej wiedzą matematyczną,
- nie każdy zaproponowany algorytm jest akceptowalny, liczy się wydajność zaproponowanego rozwiązania,
- potrzebna jest wiedza na temat reprezentacji danych w komputerze i problemów wynikających ze skończoności reprezentacji,
- na koniec, ale nie mniej ważne – do sukcesu niezbędna jest sprawność w pisaniu programów na komputerze.

Zachęcam Czytelników do zmierzenia się z tym zadaniem. Podpowiem tylko, że do otrzymania odpowiedzi wystarczy jednokrotne przejrzenie danych wejściowych.

Sukcesy młodych polskich algorytmików i programistów w konfrontacji z najlepszymi z całego świata są niewątpliwe i bezprzykładne. Laureaci konkursów i olimpiad w dorosłym życiu odnoszą światowe sukcesy zawodowe jako znakomici naukowcy, pracownicy firm technologicznych zmieniających oblicze świata lub właściciele własnych firm. Można śmiało powiedzieć, że najlepsi młodzi polscy informatycy są kontynuatorami słynnej przedwojennej Polskiej Szkoły Logiczno-Matematycznej i ówczesnych kryptologów. Stanowią oni dzisiaj nasze naturalne zasoby intelektualne, na dodatek unikalne, bo trudne do skopiowania przez inne społeczeństwa. Ten szczególny kapitał powinien być utrzymywany i rozwijany, ponieważ stwarza jedyną w swoim rodzaju możliwość zbudowania globalnej i strategicznej przewagi konkurencyjnej. Wymaga to jednak podjęcia odpowiednich systemowych działań. Na prośbę Ministerstwa Cyfryzacji profesorowie Krzysztof Diks, Krzysztof Loryś i Andrzej Kisielewicz przygotowali całościową koncepcję wzmocnienia polskiej edukacji informatycznej adresowanej do uczniów uzdolnionych informatycznie i ich rodziców, nauczycieli, studentów, szkół i uczelni. Istotą proponowanej koncepcji jest budowa w szkołach, w ścisłej współpracy z uczelniami wyższymi, sieci warsztatów (kół zainteresowań) o profilu informatycznym, powiązanych z zawodami i konkursami z zakresu algorytmiki i programowania. Głównym zadaniem byłoby wyławianie w całym kraju uczniów mających wyjątkowe predyspozycje do informatyki, a w szczególności do rozwiązywania trudnych problemów algorytmicznych, zintensyfikowane kształcenie takich uczniów, zachęcanie do samokształcenia, zapewnianie różnorodnych możliwości rozwoju. Proponowane szczegółowe rozwiązania oparte byłyby na dotychczasowych osiągnięciach nauczycieli i kadry akademickiej odnoszących szczególne sukcesy w dziedzinie informatyki, na szerzeniu i powielaniu dobrych sprawdzonych praktyk, zapewnianiu wsparcia finansowego i logistycznego dla dotychczasowych oraz nowych inicjatyw, na inspirowaniu powstawania nowych węzłów takiej sieci. Szczególnie ważną sprawą jest system kształcenia i wynagradzania nauczycieli angażujących się w takie przedsięwzięcia. Niewielka grupa najlepszych nauczycieli w tej dziedzinie korzysta obecnie ze stypendiów prywatnej Fundacji im. Łukasiewicza (`http://lukasiewicz.org.pl`), ale system wynagradzania nauczycieli wymaga zdecydowanego rozszerzenia i ustabilizowania. Powodzenie projektu wymaga powołania instytucji nadzorującej funkcjonowanie i rozwój całego systemu, z odpowiednim stałym budżetem. Szczegóły propozycji można znaleźć pod adresem `mc.bip.gov.pl/fobjects/download/196905/codi-pdf`. Jestem przekonany, że realizacja zaproponowanej koncepcji pozwoli w niedługim czasie na zbudowanie dużego środowiska polskich informatyków, którzy nie tylko pomogą w pełni zinformatyzować nasz kraj, ale także przyczynią się do powstania polskiego markowego produktu informatycznego.

Prof. dr hab. Krzysztof DIKS Instytut Informatyki UW, Komitet Główny Olimpiady Informatycznej.

Programowanie (nie tylko) dla artystów

Agnieszka BOROWIECKA

We współczesnym cyfrowym świecie korzystanie z technologii informacyjno-komunikacyjnych stało się koniecznością. Dotyczy to zarówno życia codziennego (programowane telewizory, lodówki, pralki czy nawet odkurzacze), jak i zawodowego. Z komputerów muszą obecnie korzystać także artyści: graficy, projektanci, muzycy. Jednak czy mają je traktować jako zwykłe narzędzie pracy, takie jak pędzel lub ołówek, czy może podejść do tego nieco inaczej?

Cytując Pavla V. Orlova: *„Зачем художнику нужно программировать? Почему нельзя просто взять и нарисовать от руки? А что, фотошоп отменили? Неужели у художников есть математическая логика?" Мне эти вопросы кажутся замечательными и напоминают художника в момент появления фотографии, когда художники спорили с фотографами примерно теми же словами, разве что слова фотошоп еще не было*. W wolnym tłumaczeniu można to ująć: *„Po co artyście programowanie? Dlaczego po prostu nie może wziąć i narysować czegoś odręcznie? A co, Photoshopa skasowali? Czy artyści znają logikę matematyczną?". Pytania te wydają mi się cudowne i przypominają moment pojawienia się fotografii, gdy artyści spierali się z fotografami w mniej więcej tych samych słowach, z wyjątkiem tego, że Photoshopa jeszcze nie było.*

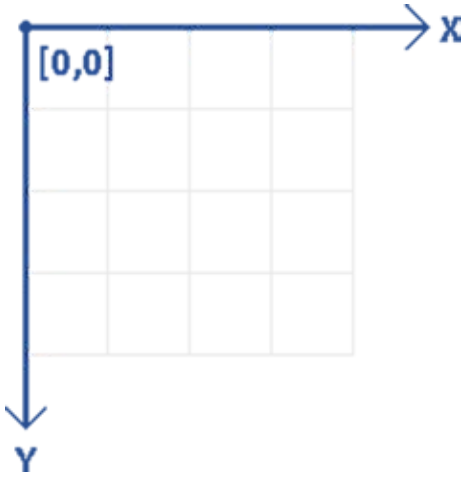
Programowanie i artyści to wydawałoby się dwa odmienne światy. Jednak warto zwrócić uwagę, że wszystko zależy od tego, jakie środowisko pracy im zaproponujemy i w jaki sposób będą z niego korzystać.

W 2001 roku Casey Reas i Benjamin Fry zaczęli pracę nad językiem i środowiskiem Processing dla grupy Aesthetics and Computation w Massachusetts Institute of Technology Media Lab. Korzenie tego środowiska tkwią w języku DBN (*Design by Numbers*) opracowanym przez profesora Johna Maeda z MIT. Czym Processing wyróżnia się wśród innych języków programowania? Cóż, właściwie wszystkim. Celem jego powstania miały być wizualizacja danych, sztuka mediów, sztuka elektroniczna i nauczanie. Powstał jako narzędzie do uczenia artystów programowania. Tak naprawdę nie tworzymy w nim aplikacji, ale serię szkiców (sketch) – które trafiają do szkicownika (sketchbook), co jest naturalnym stylem pracy dla artystów. Processing jest oparty na licencji open source, wzbogacają go liczne biblioteki tworzone przez niezależnych twórców i rozszerzające znacznie jego możliwości. Możemy pracować, korzystając z różnych systemów operacyjnych (MacOS, Windows, Linux). Tworząc programy, w łatwy sposób uzyskujemy konkretne efekty wizualne, podobnie jak pracując z grafiką żółwia w Logo. Możemy także rozwiązywać skomplikowane problemy, nie tylko dla artystów.

Jeśli w swojej klasie mamy uczniów, którzy pięknie rysują, projektują gry lub tworzą komiksy, spróbujmy wprowadzić ich w świat programowania, korzystając właśnie z Processingu.

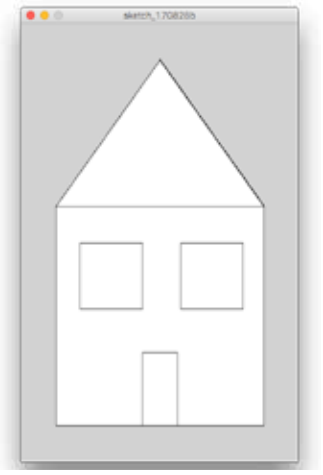
Logika dla artystów, a może zaczniemy od „zwykłej" matematyki?

Pisząc programy w Processingu, czy może raczej tworząc szkice, nie unikniemy odrobiny matematyki. Jest to w zasadzie nieuniknione dla każdego, kto chce tworzyć rysunki nie za pomocą myszki, ale wydając odpowiednie polecenia dla komputera. Processing, uruchamiając program, wyświetla małe okienko, w którym pojawią się zdefiniowane przez nas obiekty: punkty, linie, elipsy, trójkąty, czworokąty, napisy. Aby cokolwiek pojawiło się na ekranie, musimy określić położenie i wielkość rysowanego obiektu. Warto najpierw zorientować się, ile mamy miejsca na rysunek i jak określamy współrzędne ekranowe.



Domyślnie dostępny do rysowania obszar ma wymiary 100 na 100 pikseli. Punkt o współrzędnych (0, 0) znajduje się w lewym górnym narożniku okna, współrzędne x-owe rosną w prawo, zaś współrzędne y-owe w dół. Rozmiar okna można zmienić za pomocą polecenia `size()`. Tworząc dowolny rysunek, musimy odpowiednio wyliczać współrzędne wierzchołków rysowanych figur oraz dbać o kolejność rysowania obiektów (figury kreślone z wypełnionym wnętrzem mogą się zasłaniać). Warto wykorzystać kartkę w kratkę do zaprojektowania wyglądu okna aplikacji, a następnie zaprogramować rysunek na komputerze. Tworząc szkice w Processingu, definiujemy zawsze dwie standardowe funkcje – `setup()` do ustalania początkowych wartości parametrów oraz `draw()`, odpowiedzialną za zmianę zawartości okna aplikacji. Poniżej prezentujemy przykład programu rysującego dom z trójkątnym dachem, dwoma oknami i drzwiami.

```
void setup()
{
   size(400, 600);
}
void draw()
{
   // dom
   rect(50, 250, 300, 300);
   // dach
   triangle(50, 250, 350, 250,
            200, 50);
   // drzwi
   rect(175, 450, 50, 100);
   // okna
   rect(85, 300, 90, 90);
   rect(230, 300, 90, 90);
}
```



Olbrzymią zaletą Processingu jest łatwość tworzenia interakcji z użytkownikiem. Dostępne są specjalne zmienne i funkcje pozwalające komunikować się z myszką i klawiaturą. Projekt, który chcielibyśmy opisać, to prosty edytor graficzny. Przesuwając kursorem myszki po ekranie, będziemy tworzyć rysunek, do przygotowania ciekawszych efektów wizualnych wykorzystamy symetrię i funkcje matematyczne. Uczniowie, tworząc taki program, muszą zastanowić się, jak wyliczać współrzędne odpowiednich punktów w oknie aplikacji, by uzyskać symetryczne odbicie kreślonych linii względem środka okna. Jeśli będą chcieli uzależnić sposób rysowania (np. kolor linii, kształt pisaka) od pewnych dodatkowych warunków, pojawi się konieczność skorzystania z logiki matematycznej i instrukcji warunkowej. Na przykład budując w aplikacji przycisk pozwalający określić kolor pędzla, musimy zbadać położenie kursora myszki

względem tego przycisku – czyli zapisać złożony warunek, korzystając z koniunkcji i alternatywy. Im bardziej złożone projekty będziemy przygotowywać, tym bardziej znajomość logiki matematycznej będzie nam potrzebna.

Projekt Malowanka[1]

Najprostsza wersja projektu wymaga od nas napisania zaledwie kilku instrukcji. Rysunki będziemy tworzyć za pomocą pędzla w kształcie koła, korzystając z polecenia `ellipse()`. Należy podać cztery liczby określające współrzędne środka elipsy oraz długości jej małej i wielkiej osi. Jeśli osie mają taką samą długość, na ekranie zostanie narysowane koło. Możemy skorzystać ze zmiennych systemowych `mouseX` i `mouseY`, by środek rysowanej elipsy pokrywał się ze współrzędnymi kursora myszki.

Za pomocą poleceń `fill()` i `stroke()` ustalamy kolor wypełnienia i obwódki elipsy, taki sam podczas całego działania aplikacji. Możemy także zmienić domyślny rozmiar i kolor tła okna aplikacji – polecenie `background()`.

```
void setup()
{
   size(500, 500);
   fill(255);
   stroke(255);
   background(0);
}
void draw()
{
   ellipse(mouseX, mouseY, 30, 30);
}
```

Funkcja `draw()` jest wywoływana automatycznie 60 razy na sekundę, jeśli w tym czasie zmieni się położenie kursora myszki, to na ekranie zostaną narysowane kolejne elipsy.

Pierwsza wersja aplikacji jest gotowa. Jednak łatwo można zauważyć jej niedoskonałość – nowe elipsy są rysowane przy każdej zmianie położenia kursora myszki, niezależnie od tego, czy w danym miejscu chcieliśmy coś narysować, czy nie. Użytkownik ma ograniczoną kontrolę nad powstającym rysunkiem. Lepszym rozwiązaniem byłoby rysowanie nie przy każdym ruchu kursora myszki, lecz jedynie wtedy, gdy sobie tego życzymy – czyli po wciśnięciu przycisku myszki.

Należy wprowadzić dwie zmiany w programie. Dodajemy nową funkcję `mouseDragged()`. Jest ona wywoływana automatycznie, gdy przytrzymamy wciśnięty dowolny przycisk myszki i zaczniemy przesuwać jej kursor po ekranie. Instrukcję rysującą elipsę przenosimy z funkcji `draw()` do funkcji `mouseDragged()`. Poprawiony program wygląda następująco:

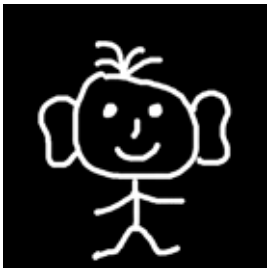
```
void setup()
{
   size(500, 500);
   fill(255);
   stroke(255);
   background(0);
}

void draw()
{

}

void mouseDragged()
{
   ellipse(mouseX, mouseY, 30, 30);
}
```

Tworzenie czarno-białych rysunków może się szybko znudzić. Uzyskane efekty byłyby ciekawsze, gdyby można było zmieniać kolor, grubość i kształt pędzla. Bardziej energiczny użytkownik zauważy również, że przy szybszym przemieszczaniu kursora myszy pojawiają się przerwy w kreślonych liniach – widać wyraźnie, w jaki sposób one powstają. Zamiast tego możemy rysować łamaną,

[1] Opisany scenariusz zajęć pochodzi z projektu Warszawa programuje!, realizowanego od 2016 roku przez OEIiZK oraz WCIES.

wykorzystując polecenie `line()`, a nie stawiać dużą liczbę kropek poleceniem `ellipse()`. Kreśląc linie, będziemy łączyć punkt wskazywany w danej chwili przez kursor myszy z poprzednią pozycją kursora (zmienne systemowe `pmouseX` i `pmouseY`). Aby kreślone linie były wyraźniejsze, a punkty łączenia łamanej mniej „kanciaste", warto zmienić grubość rysowanych krawędzi, korzystając z polecenia `strokeWeight()`. Możemy także usunąć definicję koloru wypełnienia, gdyż nie będzie już nam potrzebna.

```
void setup()
{
  size(500, 500);
  stroke(255);
  strokeWeight(10);
  background(0);
}
void draw()
{
}
void mouseDragged()
{
   line(pmouseX, pmouseY, mouseX, mouseY);
}
```



Wprowadzone zmiany znacznie poprawiły jakość tworzonych rysunków. W następnym kroku dodamy do aplikacji możliwość zmiany koloru rysowanych linii. Użytkownik może określać kolor rysowania, wciskając odpowiedni klawisz na klawiaturze lub klikając myszką w wybrany obszar okna projektu. To drugie rozwiązanie ograniczyłoby obszar rysowania przez konieczność zdefiniowania na ekranie palety barw, czyli paska narzędziowego dla użytkownika aplikacji. W tym celu należałoby narysować kolorowe prostokąty np. w górnej części okna aplikacji. Po kliknięciu w prostokąt zmieniany byłby odpowiednio kolor pędzla. W podobny sposób można dodać możliwość wyboru grubości kreślonych linii. Wykorzystanie palety barw polecamy dla aplikacji, którą chcielibyśmy uruchamiać na urządzeniach mobilnych – rozwiąże to problem z dostępem do klawiatury ekranowej.

Ustalanie koloru rysowania za pomocą klawiatury lub myszki wymaga dodania do projektu funkcji `keyPressed()` lub `mousePressed()` oraz skorzystania z instrukcji warunkowej. Zamiast tego proponujemy wprowadzenie automatycznego określania koloru linii. Definiując kolor podajemy zwykle trzy składowe RGB – czerwoną, zieloną i niebieską. Każda składowa przyjmuje domyślnie wartości z zakresu od 0 do 255. W projekcie uzależnimy wartość składowej od współrzędnych kursora myszy. Na przykład składowa czerwona może zależeć od współrzędnej x, składowa zielona od współrzędnej y, zaś składowa niebieska od odległości między kursorem myszy a środkiem okna programu. Do obliczeń wykorzystamy funkcje `map()` i `dist()`. Pierwsza z nich pozwala przeliczać wartości z jednego zakresu na drugi, druga oblicza odległość między dwoma punktami.

Współrzędna x-owa kursora myszy może przyjmować wartości od zera (lewa krawędź okna aplikacji) do szerokości okna zdefiniowanej w funkcji `setup()` (zmienna systemowa `width`). Musimy znaleźć liczbę, jaka odpowiadałaby położeniu pierwszej współrzędnej kursora myszki, po przeliczeniu na zakres od 0 do 255, czyli wartości odpowiadające składowej RGB. Definiujemy pomocniczą zmienną `red`:

```
float red = map(mouseX, 0, width, 0, 255);
```

W analogiczny sposób wyliczamy składową zieloną, tym razem odwołując się do zmiennej systemowej `height`, określającej wysokość okna aplikacji:

```
float green = map(mouseY, 0, height, 0, 255);
```

Ostatnią składową wyliczamy jako odległość kursora myszki od środka ekranu:

```
float blue = dist(mouseX, mouseY,
                  width / 2, height / 2);
```

Zmienna `blue` powinna przyjmować wartości z zakresu od 0 do 255, tymczasem istnieją punkty w oknie aplikacji, których odległość od środka okna jest większa od 255. Możemy ponownie skorzystać z funkcji `map()` lub użyć funkcji `constrain()` do przycięcia wartości zmiennej do właściwego zakresu.

```
blue = constrain(blue, 0, 255);
```

Na podstawie wyliczonych wartości definiujemy kolor kreślonych linii:

```
stroke(red, green, blue);
```

Możemy przedyskutować z uczniami, jaki efekt spowodują wprowadzane przez nas zmiany. Warto definiować najpierw jedną składową, pozostałym przypisując wartość zero. Pozwoli to zauważyć, jak zmiana położenia kursora myszki wpływa na zmianę koloru kreślonych linii, zarówno na ich barwę, jak i jaskrawość. Warto również zastanowić się, jak zmieni się wygląd rysunku, gdy zamienimy rodzaj wyliczeń przypisanych poszczególnym składowym.

Ostatnia wersja programu pozwala uzyskać naprawdę interesujące efekty. Warto dodać do projektu dwie nowe funkcjonalności – możliwość zapisywania rysunku oraz czyszczenia ekranu bez konieczności ponownego uruchomienia aplikacji. Znaki wprowadzone za pomocą klawiatury są przechowywane w zmiennej `key`. Przyjmijmy, że naciśnięcie klawisza z literą *c* spowoduje wyczyszczenie ekranu, klawisz *s* pozwoli zapisać gotowy rysunek pod podaną nazwą.

```
void keyPressed()
{
   if (key == 'c') background(0);
   if (key == 's') saveFrame("rysunek.png");
}
```

Wróćmy na chwilę do matematyki. Chcielibyśmy dodać do tworzonych rysunków odbicie lustrzane, czyli symetrię względem pionowej osi przechodzącej przez środek okna. Jakie zmiany należałoby wprowadzić w programie? Warto zacząć od dyskusji z uczniami, ilustrując zachowanie współrzędnych punktów w kartezjańskim układzie współrzędnych. Następnie próbujemy przekształcić otrzymane wzory tak, by pasowały do rozmiarów okna i położenia układu współrzędnych w aplikacji tworzonej w Processingu.

Po wprowadzeniu ostatnich poprawek funkcja `mouseDragged()` przyjmie następującą postać:

```
void mouseDragged()
{
  float red = map(mouseX, 0, width, 0, 255);
  float green = map(mouseY, 0, height, 0, 255);
  float blue = dist(mouseX, mouseY,
                    width / 2, height / 2);
  blue = constrain(blue, 0, 255);
  stroke(red, green, blue);
  line(pmouseX, pmouseY, mouseX, mouseY);
  line(width - pmouseX, pmouseY,
       width - mouseX, mouseY);
}
```

Opisany projekt pozwala na stopniowe wprowadzanie wielu pojęć, łączy tworzenie interesującej grafiki z nauką pisania programów, rozwija także umiejętności matematyczne uczniów. Tworząc program, zapisujemy w języku formalnym – przy wykorzystaniu metod z matematyki i informatyki – to, co rozumiemy intuicyjnie. Aplikacja **Malowanka** może stanowić bazę do tworzenia wielu interesujących modyfikacji: kreślenia symetrycznych modeli – projektów naczyń, dodania efektu „zanikania" rysunku, dodania kolejnych symetrii oraz możliwości włączania i wyłączania wybranych funkcjonalności za pomocą klawiatury itd.

Trochę więcej matematyki

Bawiąc się rysowaniem, możemy testować działanie różnych poleceń i wprowadzać skomplikowane pojęcia. Przypuśćmy, że będziemy rysować proste kwiatki złożone z żółtego środka i kilku niebieskich płatków. Do rysowania płatków wykorzystamy iterację – pętlę `for`. Wyliczenie położenia kolejnych płatków może się wydawać zbyt skomplikowane dla naszych uczniów, wymaga bowiem znajomości funkcji trygonometrycznych, które w nowej podstawie programowej są wprowadzane dopiero w liceum. Jednak Processing przychodzi nam z pomocą. Wystarczy wyliczyć położenie jednego płatka, a pozostałe zostaną narysowane dzięki funkcji `rotate()`. Załóżmy, że środek kwiatka rysowany jest w środku układu współrzędnych. Wykorzystamy do tego poniższe dwie instrukcje:

```
fill(250, 250, 0);

ellipse(0, 0, 20, 20);
```

Przyjęliśmy, że średnica koła będącego środkiem kwiatka wynosi 20. Zastanówmy się, jak wyliczyć współrzędne środka dowolnego płatka. Najprościej jest narysować płatek, przesuwając się wzdłuż jednej z osi układu współrzędnych o sumę promienia środka kwiatka i promienia płatka, np.

```
ellipse(25, 0, 30, 30);
```

Pozostałe płatki są tak samo odległe od środka układu współrzędnych, jednak zamiast wyliczać ich współrzędne wykonamy „sztuczkę" – obrócimy cały układ współrzędnych o odpowiedni kąt, a następnie narysujemy identyczną elipsę. Funkcja `rotate()` obraca układ współrzędnych o podany kąt wyrażony w radianach. Możemy jednak podać wartość kąta w stopniach, a następnie przeliczyć go za pomocą funkcji `radians()`. Do rysowania pięciu płatków moglibyśmy użyć następującej pętli `for`:

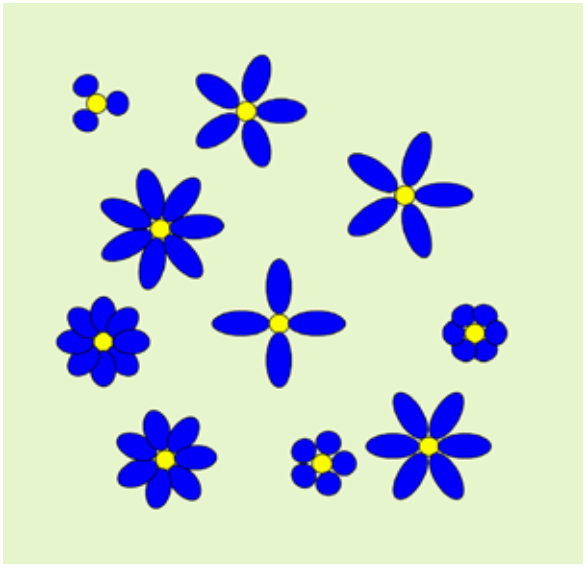
```
fill(0,0,250);
for (int i=0; i<5; i++)
{
   rotate(radians(360 / 5));
   ellipse(25, 0, 30, 30);
}
```

Kwiatek zostanie narysowany, jednak nie będziemy pewni, czy uzyskany efekt jest prawidłowy – w oknie aplikacji widoczny jest tylko jego fragment. Pamiętajmy, że środek układu współrzędnych, będący jednocześnie środkiem kwiatka, leży w górnym lewym rogu okna aplikacji. W Processingu jest dostępna także funkcja `translate()`, pozwalająca przesunąć środek układu współrzędnych w inne miejsce, np. do punktu wskazywanego przez kursor myszy. Do naszego programu dodamy jeszcze dwie funkcje: `pushMatrix()`, zapamiętującą bieżące położenie układu współrzędnych oraz `popMatrix()`, przenoszącą układ współrzędnych do zapamiętanego położenia. Dzięki temu można na przykład dodać do projektu wypisywanie licznika narysowanych kwiatków bez konieczności zastanawiania się, w jakim punkcie ekranu powinien być rysowany.

```
void setup() {
   size(600, 600);
   background(#E7F5CD);
}

void draw() {
}

void mousePressed() {
   pushMatrix();
   translate(mouseX,mouseY);
   fill(250, 250, 0);
   ellipse(0, 0, 20, 20);
   fill(0, 0, 250);
   for (int i=0; i<5; i++) {
      rotate(radians(360 / 5));
      ellipse(25, 0, 30, 30);
   }
   popMatrix();
}
```

Program możemy wzbogacić o możliwość rysowania kwiatków o zmiennej liczbie płatków. Do ustalania liczby płatków wykorzystamy funkcję `random()`, losującą liczbę rzeczywistą z zakresu od zera do podanej wartości. Wynik musimy przekonwertować na wartość całkowitą i zapamiętać w pomocniczej zmiennej. Uzyskaną wartość użyjemy do określenia liczby powtórzeń pętli oraz wyliczenia kąta, o jaki obracamy układ współrzędnych.

Każdy rysowany kwiatek może mieć od trzech do ośmiu płatków.



```
void mousePressed() {
   pushMatrix();
   translate(mouseX, mouseY);
   fill(250, 250, 0);
   ellipse(0, 0, 20, 20);
   fill(0, 0, 250);
   int n = 3 + int(random(6));
   for (int i=0; i<n; i++) {
      rotate(radians(360 / n));
      ellipse(25, 0, 30, 30);
   }
   popMatrix();
}
```

Uczniowie mogą eksperymentować w celu uzyskania innego kształtu płatków, dodać losowy kolor kwiatków itp. Możemy także zastanowić się, jak utworzyć bukiet kwiatków, dorysowując zielone łodyżki wychodzące z jednego punktu.

Programowanie w szkole

Programowanie nie jest łatwym zagadnieniem dla uczniów. Powinniśmy się zastanowić jak ich zachęcić do nauki. Ucząc programowania, warto dobrać właściwe przykłady, aby wyjaśnić uczniom sposób działania poszczególnych instrukcji i rozwijać prawidłowe intuicje. Podczas pracy nad projektem **Kwiaty** możemy zwrócić uwagę uczniów na łatwość automatyzowania pewnych czynności w celu uzyskania regularnych wzorów. Podobny efekt byłby bardzo trudny do uzyskania podczas odręcznego tworzenia grafiki. W obu projektach warto zauważyć, że praca koncepcyjna, planowanie wyglądu i działania aplikacji jest bardzo ważna i poświęcamy jej więcej uwagi niż pracy „manualnej" – zaprogramowaniu aplikacji. Przejście od planu do realizacji jest płynne, polega bardziej na sformalizowaniu wiedzy posiadanej intuicyjnie niż na zapamiętywaniu gotowych formułek i instrukcji.

Równie istotnym zagadnieniem jest właściwy dobór języka programowania i środowiska, w którym będziemy pracować. Zachęcamy do korzystania ze środowiska Processing, dostępnego na różne systemy operacyjne i umożliwiającego m.in. tworzenie aplikacji mobilnych dla systemu Android. Język, jakim się posługujemy, to uproszczona wersja Javy. W opisanych projektach w sposób naturalny wprowadzamy pojęcie zmiennej, instrukcję warunkową, pętlę oraz losowość. Jednak możliwości środowiska są dużo większe, pozwalają definiować proste gry logiczne i planszowe, symulacje zjawisk czy przygotowywać wygodne narzędzia do podstawowej obróbki zdjęć. Warto także dodać, że Processing jest wykorzystywany na kursach dotyczących programowania, oferowanych przez bardzo popularny portal Akademia Khana.

Agnieszka BOROWIECKA jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Nauka programowania na iPadzie – od zera do bohatera.

Programowanie na tablecie lub z robotem – nie tylko na informatyce

Janusz S. WIERZBICKI

Programowanie najczęściej kojarzy się ze ślęczeniem nad klawiaturą komputera i mozolnym pisaniem kodu. Niewiele osób wie, że do nauki podstaw programowania doskonale nadaje się iPad. Istnieje wiele aplikacji ułatwiających to zadanie – niezależnie od wieku i stopnia zaawansowania uczniów!

Zgodnie z nową podstawą nauczania informatyki, **elementem powszechnego kształcenia staje się umiejętność programowania**. Programowanie jest tu rozumiane znacznie szerzej niż tylko samo napisanie programu w języku programowania. To cały proces, informatyczne podejście do rozwiązywania problemu: od specyfikacji problemu (określenie danych i wyników), przez znalezienie i opracowanie rozwiązania, do jego zaprogramowania, przetestowania poprawności i ewentualnej korekty przy użyciu odpowiednio dobranej aplikacji lub języka programowania. Tak rozumiane programowanie jest częścią zajęć informatycznych od najmłodszych lat, wpływa na sposób nauczania innych przedmiotów, służy właściwemu rozumieniu pojęć informatycznych i metod informatyki. Wspomaga kształcenie takich umiejętności, jak: logiczne myślenie, precyzyjne prezentowanie myśli i pomysłów; sprzyja dobrej organizacji pracy, buduje kompetencje potrzebne do pracy zespołowej i efektywnej realizacji projektów. W warunkach szybko zmieniającej się technologii te umiejętności są ponadczasowe, trwalsze niż jakiekolwiek środowisko programowania czy aplikacje. Umiejętności nabyte podczas programowania są przydatne na zajęciach z innych przedmiotów, jak i później w różnych zawodach, niekoniecznie informatycznych. Umożliwiają przejście z pozycji cyfrowego konsumenta na pozycję cyfrowego twórcy oraz przyjęcie roli osoby władającej technologią, a nie tylko korzystającej z niej*.

*na podstawie:

- http://dziennikustaw.gov.pl/du/2017/356/1
- skrót adresu: http://patrz.link/NPP

Technologia otacza nas na każdym kroku. Niezależnie od tego, kim jesteśmy i czym się zajmujemy – ma wpływ na nasze życie. Zatem zrozumienie podstaw działania urządzeń i uruchomionych na nich aplikacji ułatwia nasze funkcjonowanie. Nauczenie się programowania w podstawowym zakresie to jak umiejętność komunikowania się w kolejnym obcym języku. Przydatna tak samo dla mechanika samochodowego, architekta, inżyniera, księgowego czy grafika. Dlatego programowanie jest dzisiaj postrzegane jako jedna z kluczowych, bardzo przydatnych w XXI wieku umiejętności i powinna być kształcona od najmłodszych lat.

Warto zauważyć, że koncepcja uczenia się programowania od najmłodszych lat nie jest nowa. Już w książce z 1980 roku Seymour Paper, twórca dydaktycznego języka Logo, pisał: *Każde dziecko uczy się mówić. Dlaczego zatem dzieci nie miałyby się uczyć rozmawiać z komputerem.* Chodzi o to, by to nie komputery „sterowały" nami. To my powinniśmy umieć sterować nimi.

Należy podkreślić, że nauka programowania związana jest bezpośrednio z kształceniem kompetencji, które dziś uważa się za bardzo ważne dla funkcjonowania w XXI wieku. Należą do nich między innymi:

- rozwiązywanie problemów i podejmowanie decyzji,
- twórcze i krytyczne myślenie,
- komunikacja, współpraca, negocjacje,
- wyszukiwanie, selekcja, porządkowanie i ocenianie informacji,
- wykorzystywanie wiedzy w nowych sytuacjach,
- integrowanie technologii z kształceniem i własnym rozwojem.

Powstaje więc pytanie, jak uczyć programowania przez zabawę od najwcześniejszych lat? Czy można w tym celu wykorzystać umiejętności już posiadane przez naszych uczniów? Dzieci i młodzież posługują się w sposób naturalny urządzeniami mobilnymi z dotykowymi ekranami. Głównie w celach rozrywkowych. Warto więc przyjrzeć się możliwości ich wykorzystania do nauki programowania. Dla iPadów powstało wiele doskonałych aplikacji – środowisk edukacyjnych przeznaczonych do tego celu.

Scratch Jr. dla najmłodszych

Scratch Jr. to przeznaczona dla najmłodszych uczniów wersja bardzo popularnego w ostatnich latach środowiska Scratch wykorzystywanego do nauki programowania. Została zaprojektowana specjalnie dla ekranów dotykowych, dlatego doskonale sprawdza się na tabletach. Tutaj także program układamy z bloczków (klocków) przypominających puzzle. W odróżnieniu od „starszego brata" w Scratch Jr. uczniowie **nie muszą nawet umieć czytać**, ponieważ bloczki opisane są symbolami obrazkowymi. Za pomocą stworzonych programów uczniowie sterują postaciami na wirtualnej scenie. W ten sposób mogą układać animowane historyjki, planować interakcje między bohaterami swojej opowieści, a całość wzbogacić efektami dźwiękowymi oraz nagraną przez siebie narracją. Bawiąc się, poznają podstawowe instrukcje i konstrukcje programistyczne (np. pętle, instrukcje warunkowe), które są potrzebne do osiągnięcia zaplanowanego przez nich efektu. Tworzone programy mogą być wykorzystywane na lekcjach różnych przedmiotów. Możliwe jest między innymi budowanie interaktywnych scenek dialogowych, stanowiących ilustrację historii z bajek czy wierszy. Dziecko może rozwijać swoją kreatywność, tworząc do nich własną narrację oraz poszerzać kompetencje związane z wypowiadaniem się i komunikacją. W aplikację wbudowany jest także edytor graficzny, dzięki któremu uczniowie mogą zaprojektować własną scenerię lub bohaterów tworzonych historyjek, gier i innych programów.

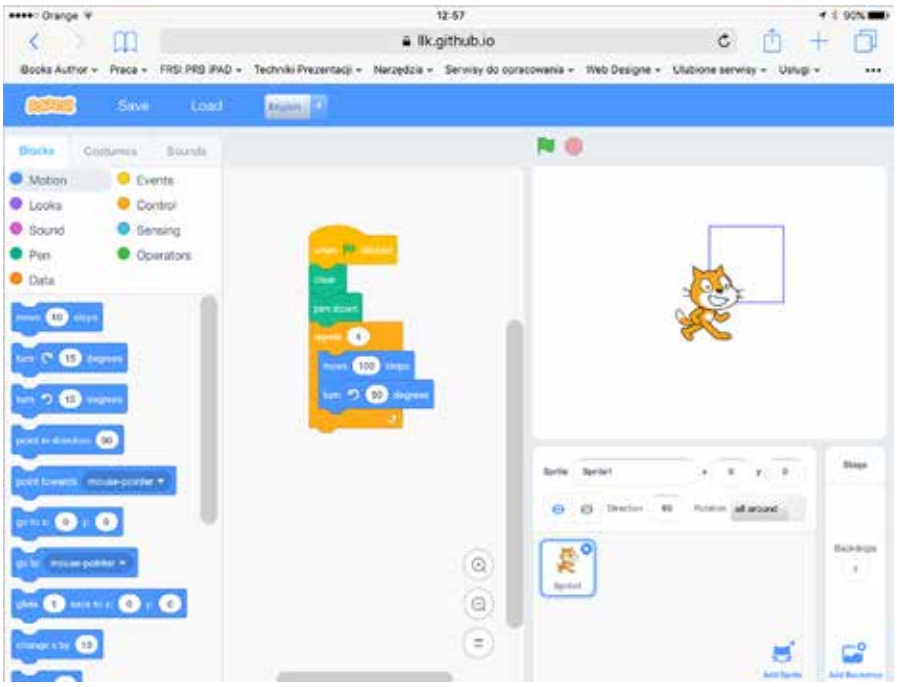
W rozpoczęciu pracy z aplikacją pomagają dostępne w niej samouczki wideo – demonstrujące krok po kroku, w jaki sposób stworzyć i oprogramować własną interaktywną historyjkę. Możemy także skorzystać z minipodręcznika-przewodnika.

Link do pobrania aplikacji:
`https://itunes.apple.com/pl/app/scratchjr/id895485086?l=pl&mt=8`
Skrót: `http://patrz.link/ScratchJr`

Rysunek 1. Środowisko Scratch Jr na iPadzie pozwala programować nawet uczniom, którzy jeszcze nie potrafią czytać.

Dotychczas korzystanie na iPadzie ze środowiska Scratch przeznaczonego dla starszych uczniów było utrudnione, a wręcz niemożliwe. Dobrą wiadomość stanowi informacja, że już w 2018 roku ma być dostępna nowa wersja Scratch 3.0 opracowana w nowej technologii, z której będziemy mogli skorzystać na wszystkich urządzeniach! Dzięki temu, po wprowadzeniu najmłodszych w świat programowania przy użyciu Scratch Jr. będzie możliwe przejście na oferującą znacznie większe możliwości wersję Scratch 3.0 – nadal korzystając z tabletów! Wszystkie projekty opracowane we wcześniejszej wersji Scratch 2.0 mają poprawnie działać w nowym wydaniu środowiska.

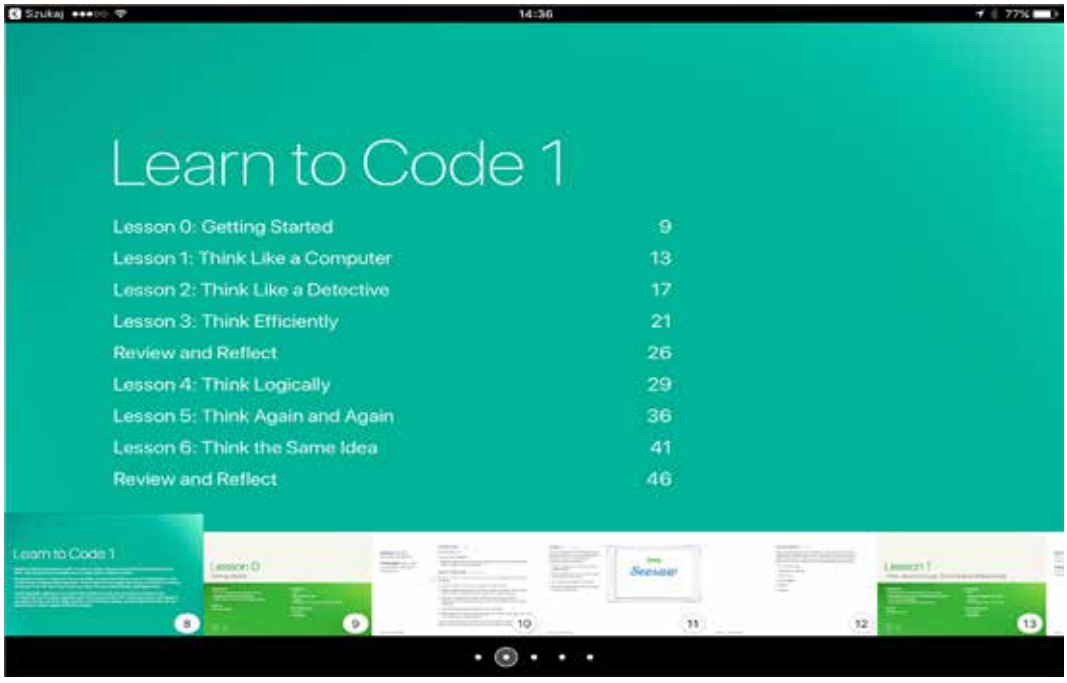
Wersję próbną środowiska Scratch 3.0 można przetestować pod adresem: `http://llk.github.oi/scratch-gui` (skrót: `http://patrz.link/scratch3`)



Rysunek 2. Działająca wersja próbna Scratch 3.0 w przeglądarce na iPadzie.

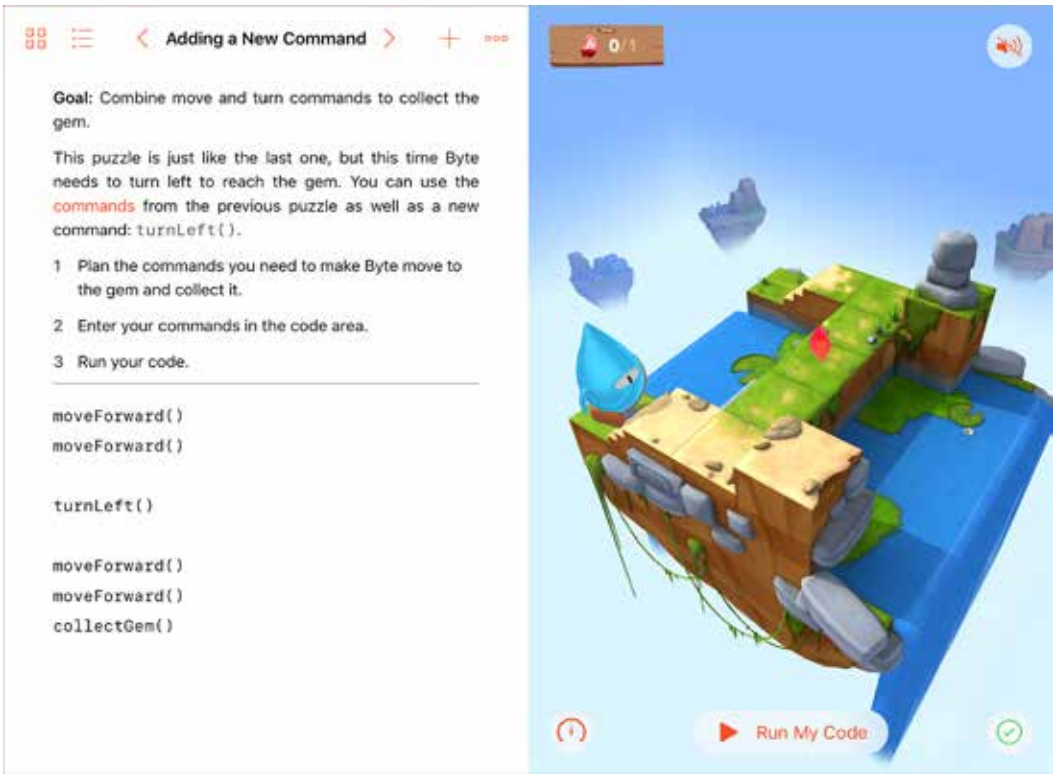
Programowanie tekstowe na tablecie? To możliwe ze Swift Playgrounds!

Przygotowane przez Apple środowisko Swift Playgrounds do nauki programowania można wykorzystać na niemal każdym etapie edukacji, niezależnie od stopnia zaawansowania uczniów. Warto zaznaczyć na wstępie, że oprócz aplikacji powstało dla nauczycieli wiele materiałów przydatnych do prowadzenia lekcji (podręcznik, prezentacje, kursy online). Dostępna jest również książka dla uczniów umożliwiająca samodzielną pracę i naukę wykraczającą poza zagadnienia omawiane na lekcjach. Zarówno aplikacja, jak i wszystkie materiały dydaktyczne dostępne są w wersji elektronicznej **bezpłatnie!**



Rysunek 3. Elektroniczne podręczniki dla ucznia i nauczyciela ułatwiają naukę i prowadzenie zajęć z wykorzystaniem Switf Playgrounds.

Rysunek 4. Przygotowane zostały trzy moduły umożliwiające poprowadzenie lekcji od podstaw po zaawansowane zagadnienia.



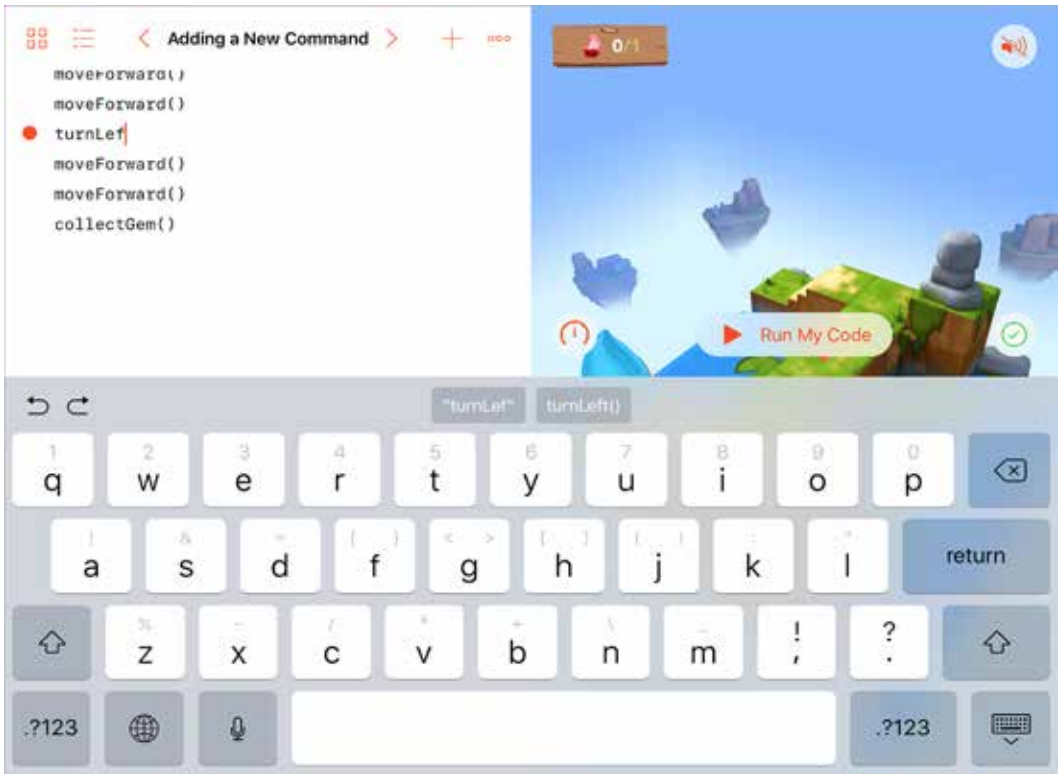
Rysunek 5. Jedna z początkowych lekcji – należy ułożyć program, który doprowadzi bohatera do miejsca docelowego i pozwoli zebrać (zjeść?) znajdujący się tam skarb.

W aplikacji można pobrać gotowe scenariusze lekcji, dzięki którym uczymy się programowania od podstaw, ale możemy dojść do zaawansowanego poziomu wtajemniczenia. Na początku – sterując postacią w trójwymiarowym świecie i rozwiązując kolejne zadania i łamigłówki. Warto podkreślić, że nie ograniczamy się tylko do tworzenia własnych rozwiązań. Czasem musimy nanieść poprawki do już istniejących programów, co bywa znacznie trudniejszym wyzwaniem. Stopniowo wprowadzane są nowe polecenia i instrukcje, jak również konieczność konstruowania własnych funkcji.

Do budowy programów wykorzystujemy komendy języka Swift – używanego przez profesjonalistów do pisania aplikacji na urządzenia z system iOS oraz MacOS. Dzięki temu w dalszym toku nauki możemy tworzyć coraz bardziej wyrafinowane aplikacje korzystając także z wbudowanych w iPada czujników i urządzeń. Interfejs środowiska jest przystosowany do ekranu dotykowego. Program możemy budować, przeciągając polecenia, podobnie jak ma to miejsce np. w Scratchu. Jednocześnie możemy je wpisać, korzystając z odpowiednio zmodyfikowanej, ułatwiającej to zadanie klawiatury ekranowej.

W aplikacji znajdziemy więcej gotowych materiałów demonstrujących możliwości środowiska oraz gotowe szablony pozwalające sterować różnego rodzaju akcesoriami z poziomu tworzonego programu – np. robotami Dash & Dot, Sphero czy klockami Lego Mindstorms EV3. Playgrounds pozwala także na tworzenie własnych programów przez uczniów lub szablonów przez nauczyciela. Pomogą w tym gotowe przykłady, np. do lekcji matematyki. W ten sposób można bowiem uczyć różnych przedmiotów – ucząc jednocześnie programowania lub je wykorzystując.

Przygotowanymi szablonami, lekcjami i programami można się dzielić, przesyłając je między iPadami. Wysłać je można bezprzewodowo, mailem lub w wiadomości. Odbiorca nie tylko będzie mógł obejrzeć i uruchomić projekt, ale także wprowadzić w nim zmiany. Co więcej – wbudowane w aplikację narzędzie pozwala na przygotowanie filmów szkoleniowych ułatwiających tworzenie materiałów dla uczniów lub odpowiadanie na ich pytania!



Rysunek 6. Wbudowana klawiatura ekranowa ułatwia wprowadzanie komend, jeśli nie chcemy ich przeciągać.

Korzystając z Playgrounds, można uczyć programowania w przyjemny, zabawny i wciągający sposób. Jednocześnie – dzięki dodatkowym materiałom – uczniowie, których zainteresuje programowanie, mogą samodzielnie rozwijać swoje umiejętności. Dzięki poznawaniu od początku języka używanego do programowania komercyjnych aplikacji, gdy osiągną odpowiedni poziom, mogą bez problemu przesiąść się do profesjonalnego środowiska programistycznego Xcode, w którym powstają aplikacje na komputery Mac oraz dla urządzeń z systemem iOS z wykorzystaniem języka Swift.

Programowanie robotów na tablecie

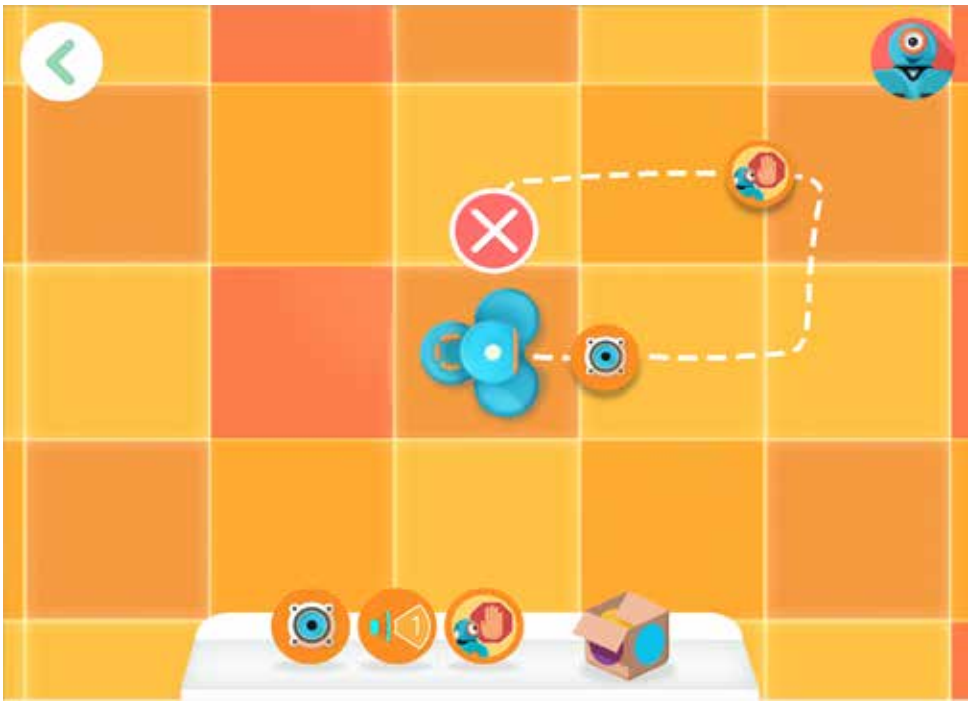
iPad współpracuje z wieloma akcesoriami wspomagającymi naukę programowania. Jedną z ciekawszych propozycji stanowią roboty Dash & Dot. Przygotowanych zostało także kilka aplikacji z nimi współpracujących – od prostego zdalnego sterowania, przez specjalne środowiska pozwalające zaprogramować robota także młodszym uczniom nieumiejącym czytać i przez definiowanie ścieżki ruchu lub określanie stanów i akcji do wykonania, po pisanie programów w języku Blockly – podobnym do Scratcha języku bloczkowym.

Nawet starsi uczniowie mogą rozpocząć od nauki zdalnego sterowania robotem i poznać w ten sposób właściwości motoryczne urządzenia. Na tym etapie roboty mogą być wykorzystywane do nauki dowolnych zagadnień z zakresu nauczania zintegrowanego. Na przykład uczniowie mogą wskazywać z użyciem robota prawidłowe odpowiedzi umieszczone na kartkach rozłożonych na podłodze. Jeśli dysponujemy kilkoma robotami – możliwe jest rozegranie za ich pomocą meczu lub przeprowadzenie konkursu, kto w najkrótszy czasie przeprowadzi robota przez wyklejony na podłodze labirynt. Pozwala to zaobserwować uczniom wiele ciekawych właściwości – np. łatwiej i precyzyjniej steruje się robotem przy mniejszej prędkości. Precyzja sterowania zależy także od rodzaju podłoża (dywan, linoleum, lakierowana deska lub kafelki).

W dalszym toku nauki robot może być programowany przez uczniów na różne sposoby. Od bardzo prostego, szczególnie dla najmłodszych, rysowania drogi, którą ma pokonać urządzenie, poprzez zdefiniowanie kolejnych akcji tworząc

Rysunek 7. Aplikacja GO!, będąca zdalnym sterowaniem, pozwala poznać własności motoryczne robota, a także sprawdzić, jak działają poszczególne funkcje: diody, głośnik czy sterowanie głową.

specjalny diagram z dostępnych elementów, po użycie zmodyfikowanej wersji programowania w Blockly, gdzie komendy mają postać bloczków przypominających te znane ze środowiska Scratch. Można także skorzystać ze Switf Playgrounds, w którym zamieszczono odpowiednie szablony z instrukcjami umożliwiającymi nie tylko sterowanie robotem, ale wykorzystanie jego czujników – podobnie jak ma to miejsce w natywnych aplikacjach!



Rysunek 8. Aplikacja Path pozwala zaprogramować robota przez naszkicowanie drogi, którą ma pokonać i umieszczenie na niej dodatkowych poleceń (np. zatrzymaj się na jakiś czas, zmień kolor drogi czy wydaj dźwięk).

Rysunek 9. Aplikacja Wonder pozwala zaprogramować robota przez określenie stanów oraz warunków przejścia między nimi (np. po uruchomieniu programu, czekaj na klaśnięcie dłońmi, a następnie zapal na czerwono diody w uszach, następnie odczekaj 2 sekundy i przejedź 20 centymetrów do przodu).

Rysunek 10. Robota można programować także w aplikacji Blockly, bardzo podobnej do popularnego Scratcha – program tworzymy za pomocą bloczków.

Robot wyposażony jest w czujniki wykrywające przeszkodę przed nim lub za nim. Dysponuje także mikrofonem i głośnikiem oraz kilkoma przyciskami. Dzięki nim może odtwarzać nagrane wcześniej dźwięki lub reagować na dźwięki wydawana przez otoczenia – np. klaśnięcie. Ma także kolorowe diody z przodu korpusu oraz w „uszach" – możemy zaprogramować ich kolor, częstotliwość i sposób jego zmiany. Białe diody umieszczone z przodu „głowy" pozwalają zmieniać wyraz „twarzy". Dzięki możliwości ich programowania możemy wyrażać emocje – uśmiech na twarzy, mruganie okiem. Możliwe jest także obracanie i kiwanie głową. Dołączenie do głowy odpowiednich akcesoriów pozwala m.in. zamienić robota w wyrzutnię piłeczek lub odegrać muzykę na cymbałkach. Wszystko przez zaprogramowanie ruchów głowy. Dostępne są również interfejsy umożliwiające zbudowanie własnych elementów z klocków zgodnych z Lego i ich zamontowanie na robocie. Dzięki czujnikom i możliwości zastosowania wielu akcesoriów możemy wykorzystać robota nie tylko na lekcjach informatyki. Odpowiednio napisany program pozwoli robotom np.: odegrać przedstawienie teatralne na języku polskim, scenkę dialogową po angielsku, zagrać w kosza lub wziąć udział w imaginowanej akcji ratunkowej (musząc ominąć przeszkody, przejść przez labirynt i dotrzeć do „potrzebującego pomocy" pluszaka). Pomysły na programy ogranicza tylko wyobraźnia uczniów i nauczycieli.

Sterowanie robotem przy użyciu programu pozwala przenieść zagadnienia ze świata wirtualnego do rzeczywistego. Uczniowie szybko mogą się przekonać, że znacznie łatwiej jest poprowadzić postać (duszka) po obwodzie kwadratu na ekranie monitora (np. w środowisku Scratch) niż strącić robotem kręgle rozstawione na wierzchołkach kwadratu wyznaczonego na podłodze. By napisać odpowiedni program, będą musieli zmierzyć bok kwadratu, a potem dobrać odpowiednio parametry prędkości, odległości w programie robota odpowiednio do powierzchni, po której będzie się poruszał. Przekonają się, że rzeczywista odległość czy zaplanowany obrót mogą być nieco inne w rzeczywistości, w zależności od panujących warunków lub napotkanych przeszkód, w stosunku do określonej w programie, a sam program będzie wymagał naniesienia poprawek.

Przed bardziej zaawansowanymi uczniami mamy możliwość stawiania coraz ciekawszych wyzwań. Np. napisanie prostego programu symulującego poruszanie się robota w sposób podobny do automatycznych odkurzaczy – a tym samym omijanie napotkanych przeszkód i „odkurzenie" całej powierzchni. Pomysłowość uczniów wobec napotykanych podczas eksperymentowania problemów niejednokrotnie może nas zaskoczyć, a ich przygotować do nieszablonowego, bardziej kreatywnego podejścia do rozwiązywania zadań. Innym pomysłem (dla starszych uczniów) może być napisanie programu pozwalającego robotowi pokonać dowolny, zbudowany przez innych uczniów (np. z pustych kartonów) labirynt. Dla mniej zaawansowanych młodszych uczniów zadanie może stanowić napisanie programu na podstawie mapy labiryntu wykonanej w odpowiedniej skali i wypróbowanie go w rzeczywistym, zbudowanym na jej podstawie labiryncie. Połączymy w ten sposób w jednym projekcie konieczność wykorzystania umiejętności z zakresu różnych przedmiotów szkolnych (matematyka, geografia/przyroda).

Do poprowadzenia ciekawych lekcji wystarczy nawet jeden robot, choć przydałoby się kilka (np. jeden na czterech uczniów). Program można pisać bowiem bez dostępu do urządzenia, a następnie po podłączeniu go – wypróbować. Uczy to także skrupulatności, przewidywania różnych sytuacji, a nie tylko eksperymentowania na zasadzie „może się uda" – które, choć także ważne, nie zastąpi myślenia.

Inne możliwości wykorzystania iPadów do nauki programowania

Warto podkreślić, że istnieje wiele aplikacji, dzięki którym możemy wykorzystać tablet na wyższych etapach edukacji do uczenia bardziej zaawansowanych algorytmów i tradycyjnych języków programowania, takich jak C/C++, Python, Java. W sklepie App Store można spędzić wiele czasu, poszukując odpowiednich rozwiązań i zawsze dopasować je do określonych celów i potrzeb.

Janusz Stanisław WIERZBICKI jest specjalistą ds. merytorycznych w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.

Programowanie na specjalne zamówienie

Zyta CZECHOWSKA, Jolanta MAJKOWSKA

Uczenie podstaw programowania we współczesnym świecie jest ważne, bez względu na to, co dzieci chcą robić w przyszłości. Programowanie bowiem uczy kreatywności, logicznego myślenia, rozwiązywania problemów, korygowania błędnych decyzji i wytrwałości.

Czy programowanie możliwe jest w szkole specjalnej?

Oczywiście, że tak! Uczniowie z niepełnosprawnością intelektualną mogą, potrafią i powinni również programować. Należy jednak tak zorganizować zajęcia, aby dziecko mogło każdorazowo odnieść sukces. Należy przede wszystkim dobrać takie narzędzia i metody nauczania, aby poprzez zabawę dziecko rozwijało swoją kreatywność i nabywało nowe umiejętności.

Należy wziąć pod uwagę, że dzieci niepełnosprawne intelektualnie mają deficyty w zakresie logicznego myślenia, orientacji przestrzennej i nie potrafią abstrahować, co utrudnia z pewnością naukę programowania.

W tym roku nasza szkoła – Zespół Szkół Specjalnych w Kowanówku – brała udział w pilotażowym programie nauki programowania. Na naszych zajęciach edukacyjnych, a także dodatkowych, prowadzonych w ramach programu, zachęcałyśmy uczniów do twórczej aktywności, pokazując im, że nie trzeba być tylko odbiorcą cyfrowej rzeczywistości, ale można ją również współtworzyć.

Na podstawie naszych obserwacji uważamy, że umiejętność programowania przynosi wiele korzyści uczniom niepełnosprawnym intelektualnie. Wyposaża ich w ważne sprawności życiowe, takie jak planowanie, przewidywanie tego, co się wydarzy, wdraża do samooceny, poza tym kształci kompetencje społeczne, uczy zasad dobrej współpracy, efektywnego rozwiązywania zadań w grupie. Rozwija logiczne myślenie, orientację przestrzenną i wdraża do wielozmysłowego podejścia do problemu i zadania.

W ostatnim czasie w naszej szkole pojawiły się małe sympatyczne urządzenia – robociki Ozoboty. Wielu uczniów podczas zajęć edukacyjnych, rewalidacyjnych, dodatkowych, świetlicowych, a także na zajęciach komputerowych miało okazję poznać ich działanie. Wszyscy uczniowie, bez wyjątku, byli zachwyceni możliwością pracy z tak atrakcyjnymi urządzeniami.

Dla nas jednak była to kolejna okazja do tego, aby zaobserwować, z jakimi dysfunkcjami i deficytami borykają się nasi uczniowie. Mamy pełen obraz obserwacji, ponieważ w zajęciach brali udział uczniowie na wszystkich etapach edukacyjnych i z różnymi niepełnosprawnościami intelektualnymi. Wiemy już, jak wykorzystać roboty do terapii zaburzeń oraz wyrównywania różnych deficytów. Jak można zatem wykorzystać roboty w naszej pracy – pedagogów specjalnych? Jakie umiejętności uczniów pomogą nam rozwinąć, jakie zaburzenia zminimalizować?

- **budowanie motywacji do pracy**

Uczniowie ze specjalnymi potrzebami edukacyjnymi borykają się z obniżoną motywacją. Szczególnie widoczne jest to u dzieci z autyzmem. Warto i trzeba zatem podejmować takie działania, które nie tylko wzmocnią, ale także wykształcą właściwą motywację do pracy. Ozoboty są atrakcyjne, bo nie są dostępne dla każdego, a mimo że są małe, potrafią wiele, poruszają się, przy okazji zmieniając kolory migających światełek, co dostarcza dodatkowych bodźców percepcyjnych.

- **wykorzystanie jako wzmocnienie w pracy z dziećmi z autyzmem – nagrody**

W pracy z dziećmi ze specjalnymi potrzebami edukacyjnymi, szczególnie z dziećmi z autyzmem bazujemy na pozytywnych wzmocnieniach. Na początku są to wzmocnienia pierwotne – biologiczne, czyli jedzenie i picie, a następnie wtórne, takie jak żetony, znaczki, naklejki. Socjalne, czyli przytulenia, pochwały, i stymulujące, czyli np. zabawki. Do tych ostatnich możemy zaliczyć właśnie roboty. Dzieci nie mają do nich stałego dostępu, są one bardzo atrakcyjne, niezwykle angażujące, zatem możemy je wykorzystać do pracy terapeutycznej. Robot nie musi być celem samym w sobie, ale sposobem dojścia do niego.

- **przełamywanie barier emocjonalnych, komunikacyjnych, społecznych**

U dzieci z autyzmem bardzo często widoczny jest brak intencji komunikacyjnej, zainteresowania ludźmi i ignorowanie ich obecności. Bardzo rzadko dzieci te inicjują kontakt z innymi osobami, a jeśli już tak się stanie, kontakt wzrokowy jest bardzo ograniczony. Nie potrafią niestety uczestniczyć w interakcji na przemian ze swoim partnerem. Charakterystyczny dla dzieci z autyzmem jest także brak zainteresowania rówieśnikami, zabawkami, przedmiotami znajdującymi się w zasięgu. Problemem jest również to, że na ogół nie podejmują żadnej wspólnej aktywności, nie rozumieją prostych zasad gier i zabaw. Bawią się same, zaabsorbowane przedmiotami, którymi bawią się na co dzień. Bardzo często jest to zabawa niewnosząca nic do rozwoju, a przeradzająca się często w stereotypowe zachowania. Ozoboty mogą naszym zdaniem pomóc w przełamywaniu tych dysfunkcji i uzupełnianiu wspomnianych powyżej deficytów, ponieważ nie ingerują w świat dziecka ze spektrum autyzmu, są atrakcyjne, efekty pracy są widoczne natychmiast i – co ważne – są atrakcyjne dla dziecka. Często dzieci, widząc, jak zostały sprawcami ruchów robotów, podejmują kolejne próby ich „ożywienia" i wprawiania w ruch, otwierając się tym samym na współpracę z innymi i nawiązywanie właściwych relacji. Pokazują efekty swojej pracy i chętnie obserwują innych.

- **korygowanie zaburzeń związanych z orientacją przestrzenną i właściwym rozmieszczeniem tekstu czy obrazu w zeszycie, na płaszczyźnie kartki**

Dużym problemem dzieci ze specjalnymi potrzebami są zaburzenia orientacji przestrzennej, nieumiejętność określania kierunków. Bardzo często tych zasad po prostu nie rozumieją, nie zapamiętują i nie wdrażają ich w życie. Widoczne jest to szczególnie w zapisie tekstu czy umieszczaniu rysunków na przestrzeni kartki w zeszycie. Nasi uczniowie, mimo że mają do dyspozycji całą wolną przestrzeń, najczęściej swój tekst czy obraz umieszczają w rogu kartki, kumulując wszystko w jednym miejscu. Programowanie trasy robotów wymusza na nich konieczność zmiany tych dysfunkcji, ponieważ już w trakcie pracy przekonują się, że źle rozrysowana trasa (zbyt blisko brzegu kartki) sprawi, że Ozobot nie pojedzie lub źle odczyta informacje. Mając to na uwadze, maksymalnie skupiają się na tym, aby wyrysować jego trasę na środku kartki, by droga była właściwej szerokości i precyzyjnie zamalowana.

- **niwelowanie dysfunkcji manualnych motoryki małej**

Aby właściwe wyrysować trasę robota, trzeba wcielić kilka ważnych zasad. Musi być ona wyrysowana precyzyjnie, nie może być zbyt cienka i zbyt gruba. Zarówno kontury drogi, jak i kody muszą być dokładnie zamalowane. Trzeba użyć właściwych kolorów. To powoduje, że dzieci maksymalnie skupiają się na zadaniu, starając się precyzyjnie narysować trasę, przy okazji używając właściwych kodów, kolorów i usprawniając motorykę małą.

Rysunek 1. Uczniowie klasy 1 gimnazjum przygotowali plakat „Wędrówki Ozobotów po kontynentach", na którym zaplanowali trasę robota i zaprogramowali jego aktywność. Poruszający się po elipsie Ozobot wykonywał wskazane czynności, np. obracał się wokół własnej osi, wjeżdżał do kontynentu i zawracał.

- **ćwiczenia logicznego i abstrakcyjnego myślenia**

Dzieci ze specjalnymi potrzebami edukacyjnymi mają ograniczone możliwości poznawcze, trudność stanowią dla nich treści o charakterze abstrakcyjnym, wymagające logicznego myślenia, wyciągania wniosków, abstrahowania i samodzielnych rozwiązań. Praca z Ozobotami nad planowaniem i programowaniem ich trasy i aktywności z pewnością sprawi, że dzieci będą te deficyty chętniej wyrównywać. Co bardzo ważne, efekt ich pracy będzie możliwy do zaobserwowania i natychmiastowego skorygowania.

- **ćwiczenia koncentracji uwagi**

Praca nad materiałem, który jest atrakcyjny, inspirujący dla ucznia zawsze sprawia, że wykonywana jest dłużej, chętniej i maksymalnie go angażuje. Tak też jest w przypadku ćwiczeń z Ozobotami. Uczniowie maksymalnie skupiali uwagę na zadaniu, kreślili trasy robota i starali się dokładnie je narysować. Gdy zadzwonił dzwonek były prośby, by przedłużyć pracę z robotami.

- **współpraca w grupie**

Z pewnością pracę z robotami można wykorzystać do rozwijania umiejętności współpracy w grupie. Programowanie tras robotów nie musi odbywać się indywidualnie. Możemy określić kryteria sukcesu dla trasy i aktywności Ozobota, a następnie wskazać konkretne grupy, które temu zadaniu muszą sprostać. Z pewnością będziemy świadkami niesamowitych prac i pomysłów wzrostu poczucia własnej wartości u uczniów, szczególnie tych słabszych, którzy nie wierzą we własne siły i możliwości.

- **rozwijanie sprawności językowych**

Uczniowie ze specjalnymi potrzebami edukacyjnymi mają duże problemy z budowaniem dłuższych wypowiedzi powiązanych w logiczną całość. Dużą trudność sprawia im opisywanie sytuacji życia codziennego. Należy więc zaplanować takie działania, które wzbogacą słownictwo dziecka i będą doskonalić umiejętność prowadzenia rozmowy, opowiadania. Dzięki Ozobotom uczniowie dużo łatwiej tworzą spójne plany wydarzeń, wokół których projektują trasy dla robotów. Następnie w towarzystwie robota opowiadają wymyślone historie.

- **ćwiczenia w określaniu związków przyczynowo-skutkowych, chronologia wydarzeń**

Na lekcjach historii duże trudności dzieciom z zespołem Aspergera sprawia zapamiętanie faktów historycznych i określanie ich związków przyczynowo-skutkowych. Należy więc tak zaplanować zajęcia, aby dziecko jak najczęściej powtarzało i utrwalało wiedzę. Jak to zrobić, by nie znudzić i nie zniechęcić ucznia? Warto tworzyć osie czasu z tras dla robotów, wplatając w nie ułożone w porządku chronologicznym wydarzenia historyczne i graficzne kody. Wspomnieć tutaj należy o doskonałej aplikacji tabletowej TIMELINE do tworzenia takich osi, która może być uzupełnieniem naszych działań, np. podczas zajęć rewalidacyjnych.

- **ćwiczenia z mapą**

Na lekcjach przyrody i geografii największe trudności dzieciom ze specjalnymi potrzebami edukacyjnymi sprawiają ćwiczenia z wykorzystaniem map. Dlatego należy ćwiczyć nie tylko odczytywanie z nich konkretnych informacji, ale również określanie kierunków głównych i pośrednich. Przewodnikiem dziecka po zadaniach z mapą mogą być roboty. Ich trasy wklejone lub narysowane na mapach konturowych skutecznie utrwalą nauczane treści i nauczą określania kierunków. Z mapą dzieci pracują również na lekcji historii. Umieszczają wydarzenia historyczne na mapie, a wytyczone trasy Ozobotów znacznie ułatwiają im to zadanie. Przykładem może być lekcja, podczas której dzieci uczyły się o wielkich odkryciach geograficznych. W jaki sposób? Wystarczyło przygotować karty pracy, na których oprócz kontynentów znalazły się trasy wypraw: Krzysztofa Kolumba, Vasco da Gamy i Ferdynanda Magellana. Po dodaniu odpowiedniego kodu graficznego Ozoboty zamieniły się w portugalskich i hiszpańskich żeglarzy odkrywających nowe lądy.

Poznajemy kontynenty

Zbliżający się Międzynarodowy Dzień Dziecka skłonił nas do stworzenia kreatywnych niestandardowych ćwiczeń dla naszych uczniów. Wykorzystałyśmy w tym celu Ozoboty. Na specjalnie przygotowanej karcie pracy uczniowie programowali trasę robotów, których zadaniem było dotarcie do wszystkich kontynentów. Przy okazji uczniowie uczyli się spełniać warunki. Jakie? Takie, które pozwalały na zmianę tempa poruszania się robota, zmianę koloru i kierunku jazdy. Ozobot miał za zadanie np. jadąc pod

Rysunek 2. Na specjalnie przygotowanej karcie pracy uczniowie programowali trasę robotów, których zadaniem było dotarcie do wszystkich kontynentów. Przy okazji uczniowie uczyli się spełniać określone warunki (np. zbliżając się do Afryki, Ozobot musiał zatrzymać się na 3 sekundy).

Afryką, zapalić czerwone światło i zwolnić, zbliżając się do Ameryki Południowej, zatrzymać się na 3 sec. itd. Przy okazji uczyliśmy się nazywać i rozpoznawać kontynenty. Podjęliśmy również temat tolerancji dla odmienności ludzi ze względu na ich kolor skóry czy kraj pochodzenia.

Dodatkowo wykonaliśmy dużą planszę z kontynentami, by na większym formacie zaprogramować aktywności Ozobotów. Taka forma zajęć z pewnością pozwoli na szybsze zapamiętanie nazw geograficznych, zachęci do pracy, a przede wszystkim zintegruje uczniów.

- **roboty na lekcjach języka polskiego**

Robotyka na lekcji języka polskiego ułatwia dzieciom z niepełnosprawnością intelektualną zapamiętanie wielu wiadomości. Takim przykładem może być lekcja, podczas której roboty pomogły naszym uczniom napisać list.

Rysunek 3. Programowanie robotów na lekcji języka polskiego. Każdy uczeń otrzymał kartę pracy i zredagował krótki list do swojej mamy, pamiętając o stosowaniu zwrotów grzecznościowych. Zakodował kartę pracy i wspólnie z robotem przeczytał napisany list.

Na początku lekcji dzieci obejrzały filmową instrukcję pisania listu, opowiedzianą przez uczniów pani Katarzyny Krywult ze Szkoły Podstawowej w Bestwinie, znalezioną na blogu „Instrukcje obsługi języka polskiego". Następnie wkleiły do zeszytu ilustrację listu, do której w aplikacji *Aurasma* został dołączony wyświetlony na lekcji plik wideo. W domu każdy uczeń po zeskanowaniu ilustracji będzie mógł jeszcze raz obejrzeć filmową instrukcję.

Następnie na dużej kartce kartonu z wyrysowaną kopertą umieścili elementy listu, dbając o ich odpowiedni układ graficzny. Później dzieci dodały kody dla Ozobota, który jeżdżąc po kopercie, zatrzymywał się przy kolejnych informacjach i dawał dzieciom czas na ich powtórzenie.

- **ćwiczenia w programowaniu**

Uczniowie szkół specjalnych, realizując podstawę programową z zajęć komputerowych czy informatyki, powinni być wdrażani do programowania. Aby zwizualizować te treści, warto korzystać z programów, które wykorzystują bloczki. Takim programem jest Ozoblockly, dostępny w języku polskim, co nie jest bez znaczenia dla uczniów ze specjalnymi potrzebami edukacyjnymi. Aplikacji tej nie trzeba instalować, jest dostępna z poziomu przeglądarki.

Ozoblockly oferuje różne poziomy trudności. Dzięki temu już nawet dzieci, które jeszcze nie umieją czytać, mogą rozpocząć swoją przygodę z programowaniem.

Poziom 1 składa się z trzech kategorii: *ruch*, *efekty świetlne* oraz *czekaj*. Nasi uczniowie świetnie potrafią stworzyć kilka prostych ruchów czy komend, które przybliżają do programowania. Dzieci mają do dyspozycji 39 różnych komend.

Poziom 2 to możliwość zastosowań poleceń tekstowych oraz grafik. Uczniowie mają do dyspozycji więcej możliwości, zwiększa się spektrum trudności, ale polecenia nadal pozostają jasne i czytelne. Pojawia się również kategoria *pętle*, pozwalająca powtarzać stworzony kod.

Poziom 3 to nowe kategorie: *nawigacja po linii* i *logika*. Aplikacja jest nie tylko dobrze zintegrowana z Ozobotem, ale przede wszystkim atrakcyjna graficznie dla dzieci, które w związku z tym nie mają większych problemów z korzystaniem z niej.

Dash i Dot jako pomocnicy w edukacji i terapii dzieci ze SPE

Te sympatyczne, atrakcyjne dla dzieci roboty nie tylko uczą programowania, ale również uatrakcyjniają lekcje i pomagają w nabywaniu nowych umiejętności, utrwalaniu wiedzy i z pewnością wspomagają terapię dzieci ze spektrum autyzmu, gdyż niwelują zachowania niepożądane, uczą okazywania emocji, zwiększają motywację do pracy i bardzo często stają się dla dzieci wzmocnieniem, czyli nagrodą.

Dash dzięki licznym sensorom reaguje na głos, odnajduje przedmioty, porusza się i wydaje dźwięki, a wszystko to z poziomu kilku aplikacji do wyboru. Robotem kieruje się za pomocą intuicyjnych, graficznych darmowych aplikacji, które dziecko z łatwością obsłuży z tabletu lub smartfona.

Robot ma charakter edukacyjny, ponieważ poprzez zabawę z nim dzieci wchodzą w świat programowania (tworzenie zdarzeń, algorytmów, budowanie sekwencji i pętli, i wiele innych). Przy tym zupełnie nieświadomie uzupełniają deficyty naszych uczniów i rozwijają kluczowe w naszych czasach umiejętności logicznego myślenia skupionego na rozwiązywaniu problemów, kreatywnego podejścia, precyzyjnego prezentowania swoich myśli, współpracy i podstawy języka angielskiego (nauka poprzez zabawę).

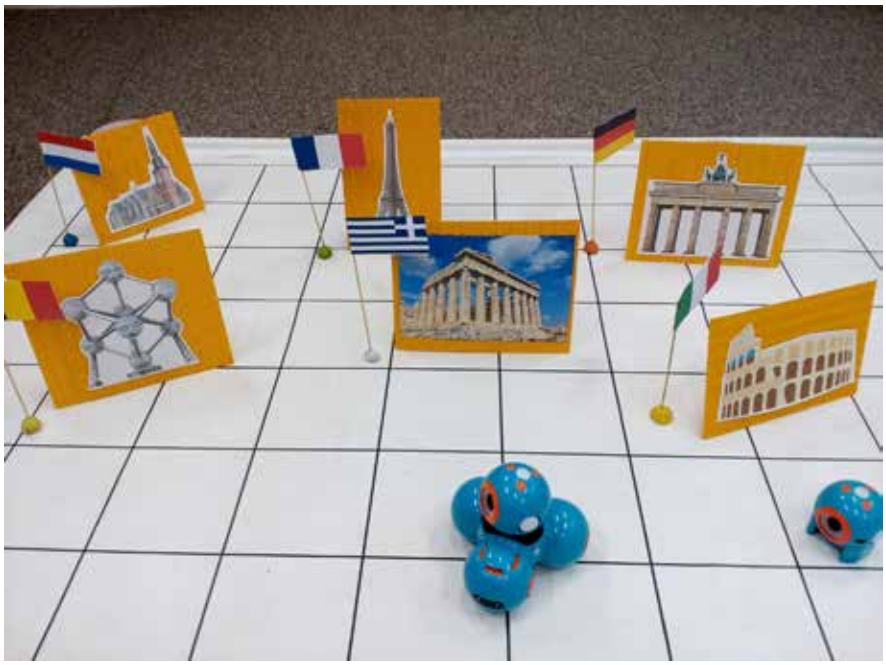
Jak można wykorzystać te sympatyczne niebieskie roboty na naszych lekcjach czy zajęciach rewalidacyjnych? Na wiele sposobów, choćby z wykorzystaniem Maty Mistrzów Kodowania, która naszym zdaniem jest produktem nieocenionym. Dash może np. pomagać w ćwiczeniach porządkowania liczb, przeliczania, ćwiczeniu orientacji przestrzennej, przemieszczając się po szachownicy.

Uczniowie mogą także sterować robotem w przód, w tył, w bok, w lewo, w prawo, ćwicząc przy okazji kierunki i usprawniając koordynację wzrokowo-ruchową. Jest to możliwe np. dzięki kompatybilnej z Dashem omawianej już aplikacji GO! Zabawa jest nie tylko doskonała, ale przy okazji pomaga w opanowaniu i ćwiczeniu stosunków przestrzennych, wartości liczb. Dzięki możliwości nagrywania głosu możemy ćwiczyć także wymowę trudnych wyrazów i rozumienie mowy.

To od kreatywności nauczycieli tak naprawdę zależy, jakie miejsce i rolę w nauczaniu będą spełniały roboty. W ostatnim czasie, podczas Święta Europy, Dash pomagał uczniom zapamiętać najważniejsze państwa Unii Europejskiej, rozróżniać ich flagi oraz nazywać charakterystyczne zabytki.

Wśród aplikacji, przy pomocy których można programować roboty Dash i Dot, warto zwrócić uwagę na Blockly. Wprowadza ona uczniów w świat algorytmów, ucząc logicznego myślenia,

Rysunek 4. Zabawy z Dashem i Dotem na specjalnej macie. Najmłodsi uczniowie naszej szkoły uczyli się wprawiać w ruch roboty, a następnie poprzez aplikację GO! ćwiczyli kierunki i usprawniali koordynację wzrokowo-ruchową, planując konkretną trasę robotów.



Rysunek 5. Podczas Święta Europy wykorzystaliśmy Dasha do utrwalenia nazw stolic państw Unii Europejskiej oraz rozpoznawania najważniejszych zabytków. To wszystko z wykorzystaniem Maty Mistrzów Kodowania.

Rysunek 6. Sterowanie robotem z użyciem aplikacji GO! Zadaniem ucznia było pokierowanie robotem w taki sposób, aby dotarł do konkretnych budowli w określonej kolejności. Po dotarciu do celu Dash wypowiadał nazwę tego zabytku. Konkretne nazwy zostały wcześniej zapisane i zaprogramowane przez uczniów.

przewidywania tego, co się wydarzy, korygowania błędów, cierpliwości i wytrwałości w dążeniu do rozwiązywania zadań programistycznych. Po ułożeniu kodu wystarczy połączyć robota z aplikacją za pomocą Bluetooth Smart i wczytać zaprogramowane aktywności.

Uczniowie z niepełnosprawnością intelektualną w stopniu umiarkowanym i znacznym najchętniej korzystają z aplikacji GO! Ta aplikacja zaznajamia dzieci z podstawowymi funkcjami robota: zmianą koloru świateł, ruchem kół, pozwala nagrać na dostępnych ścieżkach dźwiękowych określone informacje. Przesuwając palcem po konsoli, wprawiają robota w ruch i dobierają odpowiednią prędkość.

Wśród akcesoriów zaprojektowanych dla Dasha znajdują się kolorowe cymbałki. Dzieci, korzystając z kolejnej aplikacji – Xylo – mogą skomponować krótką melodię, czyli uczą się programowania poprzez muzykę.

Na rynku sukcesywnie pojawiają się nowe propozycje robotów, które w podobny sposób pomagają w nauce logicznego myślenia, wdrażają do nauki programowania, a przede wszystkim uatrakcyjniają lekcje i aktywizują uczniów. Należy do nich

na pewno robot Photon, który zapewnia dzieciom nie tylko zabawę i zdrową rywalizację, ale przede wszystkim naukę przez interakcję z otoczeniem. Producenci zapewniają, że to nie zabawka, a pomoc dydaktyczna rozwijająca się razem z dzieckiem. Zdajemy sobie sprawę, że pedagog specjalny musi nieustannie poszukiwać inspiracji do pracy, motywując tym samym uczniów do nauki.

Programowanie offline

Programowanie to nie tylko praca z tabletem lub komputerem, ale również aktywności offline, które nasi uczniowie bardzo lubią.

Aby dostosować formę pracy i stopień trudności do naszych uczniów, przygotowujemy autorskie karty pracy, które są nie tylko ćwiczeniami wprowadzającymi do programowania, ale wdrażają także do nauki logicznego myślenia, ćwiczeń orientacji przestrzennej oraz percepcji wzrokowo-ruchowej.

Jednym z wielu takich ćwiczeń, dostępnym na naszym blogu edukacyjnym www.specjalni.pl, jest wdrażająca do nauki programowania zabawa – *Zakodowany alfabet*. To ćwiczenia polegające na szyfrowaniu wyrazów, zdań i informacji za pomocą symboli. Każda litera ma odpowiedni symbol, za pomocą którego kodujemy wyrazy, zadania itp.

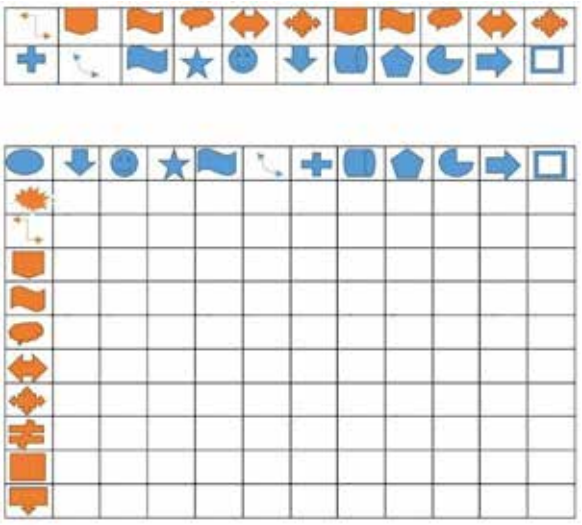
Jak można wykorzystać zakodowany alfabet?

- Utwórz wyrazy za pomocą symboli i poproś uczniów o ich odkodowanie.
- Podaj uczniom wyrazy, które muszą zakodować za pomocą symboli.
- Zakoduj wyrazy i zdania naprzemiennie, używając liter i symboli. Zadaniem ucznia będzie ich odczytanie.
- Wykorzystaj zasady gry w wisielca, ale z wykorzystaniem symboli.
- Zakoduj symbole na kratownicy z układem współrzędnych. Następnie poproś o odczytanie tych symboli i dopasowanie odpowiadających im liter na czystej kratownicy. Można spróbować zakodować hasło.
- Dzieciom, którym trudność sprawi narysowanie symboli, można je wydrukować i poprosić o ich wycięcie i naklejenie. Możemy też symbole wydrukować i zalaminować.

Przykładowe karty pracy dostępne są po zeskanowaniu kodu QR.

Rysunek 7. Kod QR prowadzący do przykładowych kart pracy.

Innym przykładem zadania jest diagram, który należy rozszyfrować za pomocą obrazkowego kodu (legendy). Wykorzystałyśmy ten kod na lekcji matematyki z uczniem z zespołem Aspergera, podczas której omawiałyśmy rodzaje kątów. Możemy zaproponować uczniowi, aby zakodował dalszy szyfr, który stworzy podstawę trójkąta, i otrzymamy figurę geometryczną. Stworzyłyśmy także specjalną kartę do kodowania, która pomogła uczniom odkodowywać i kodować rysunki.



Rysunek 8. Przykładowa plansza przedstawiająca ćwiczenie, które polega na zakodowaniu lub odkodowaniu konkretnej informacji za pomocą wcześniej ustalonych symboli.

Mucha inspiruje, bawi, ale przede wszystkim uczy. Niesamowitą, niezwykle angażującą uczniów zabawę do nauki programowania wykorzystaliśmy na lekcji matematyki. Pomogła nam w nauce programowania, ćwiczeniach logicznego myślenia, utrwalania kierunków i stosunków przestrzennych.

Uczniowie I klasy gimnazjum sami wykonali plansze do gry, do której wymyślaliśmy wiele różnych zastosowań. Nikt na takiej lekcji z pewnością nudzić się nie będzie. Natomiast podczas zajęć rewalidacyjnych i zajęć z funkcjonowania osobistego i społecznego uczniowie utrwalali symbole narodowe i programowali najkrótszą i najdłuższą drogę muchy do domu.

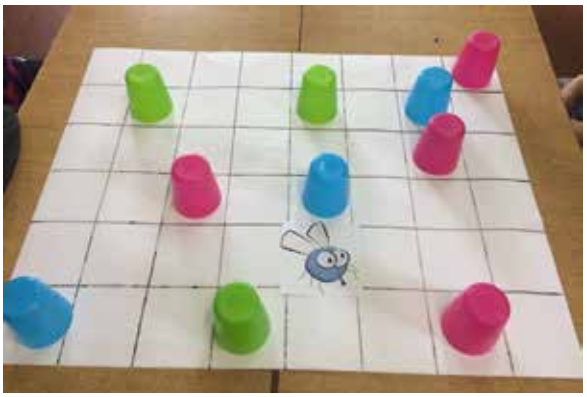
Zabawę zaplanowałyśmy i dostosowałyśmy do naszych zajęć inspirując się grą w muchę autorstwa Anny Grzegory oraz modyfikacjami Mistrzów Kodowania.

Jakie modyfikacje muchy zostały zastosowane?

- Dzieci utrwalały z muchą symbole. Na początku miały problem z kierunkami, więc mucha chodziła z nimi po macie, później już ćwiczyły, wodząc wzrokiem. W momencie, gdy mucha weszła na pole z danym symbolem, dzieci musiały wypowiedzieć jego nazwę.
- Po opanowaniu zasad gry z symbolami można wprowadzić aktywności ruchowe. Można się umówić, że gdy mucha wejdzie na rozłożony na macie symbol lub np. kolorową kartkę, dzieci muszą wykonać jakiś ruch, np. wstać, klasnąć itd.
- Dla starszych uczniów przygotowałyśmy karty pracy. Dodatkowym zadaniem dla uczniów było ułożenie kodu, dzięki któremu wskazały najkrótszą i najdłuższą drogę muchy do domku. Super zabawa! Karty pracy można zmodyfikować.

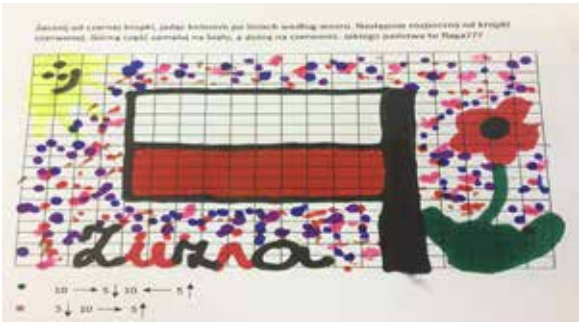
W związku ze zbliżającym się Świętem Unii Europejskiej i Świętami Majowymi przypomnieliśmy uczniom informacje dotyczące naszego państwa i państw członkowskich UE.

Tym razem proponujemy nietypowe ćwiczenia. Utrwalanie nazw państw Unii oraz rozpoznawanie



Rysunek 9. Plansza przedstawiająca przykładowe ćwiczenia przygotowujące do kodowania. Zadaniem uczniów było określenie kodu, którym posłuży się mucha, by dojść do wskazanego miejsca na planszy, omijając przeszkody, którymi są na przykład kolorowe kubki.

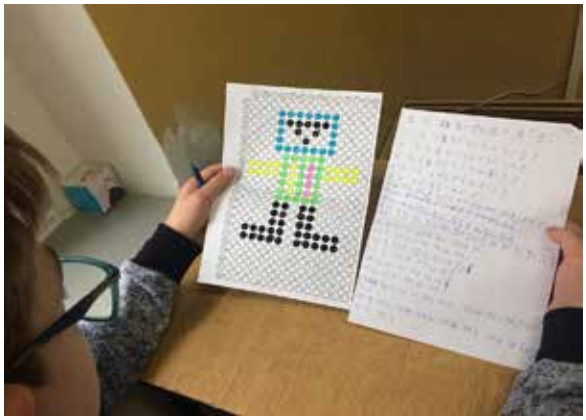
flag poprzez programowanie. W tym celu stworzyłyśmy karty pracy, które dotyczą omawianych zagadnień. Programowanie było tylko jednym z zadań, ponieważ uczniowie wyzwolili swoją aktywność twórczą i przy okazji ozdobili swoje karty pracy dodatkowymi elementami, a także chętnie poszerzali słownictwo dotyczące tych elementów.



Rysunek 10. Karta pracy przygotowana na zajęcia rewalidacyjne, podczas których zadaniem uczniów było odkodowanie informacji-kodu. Efektem końcowym okazała się flaga.

Rysunek 11. Uczennice podczas zajęć rewalidacyjnych programują najkrótszą i najdłuższą trasę dotarcia do flagi Unii Europejskiej, wpisując jej kod. Przy okazji utrwalały nazwy państw członkowskich Unii.

Inną, budzącą wśród uczniów pozytywne emocje formą programowania offline jest aktywność *zakodowane obrazki*, którą zaproponowałyśmy uczniom klasy 1 gimnazjum na lekcji matematyki. Na specjalnie przygotowanej karcie do kodowania uczniowie mieli za zadanie odkodować szyfr. Okazało się, że był pod nim umieszczony domek. Kolejne zadanie polegało na samodzielnym zakodowaniu własnego obrazka i zapisaniem do niego kodu. Uczniowie bardzo aktywnie wykonywali swoje niełatwe zadanie. A oto efekt ich pracy.



Rysunek 12. Podczas lekcji matematyki uczniowie na specjalnie przygotowanej karcie do kodowania tworzyli rysunki, do których następnie zapisywali konkretny kod. Ćwiczeniem odwrotnym było odszyfrowywanie wcześniej przygotowanej przez nas informacji, dzięki której powstał konkretny obraz.

Dla najmłodszych uczniów wartościową pomocą jest wspomniana już Mata Mistrzów Kodowania. Dwustronna plansza wykonana z lekkiego tworzywa, łatwego w utrzymaniu czystości. Podzielona na mniejsze płaszczyzny, na których dzieci mogą układać kolorowe, atrakcyjne wizualnie i sensorycznie klocki podzielone na 3 kategorie: rośliny, zwierzęta i środki transportu. Mata wraz z klockami służy nie tylko do wstępnych aktywności związanych z programowaniem, ale także do rozwoju kompetencji matematycznych, orientacji w przestrzeni, w schemacie własnego ciała, do tworzenia zbiorów, porównywania ilości, odczytywania i określania położenia przedmiotów na osiach, określania kierunków. Matę możemy także wykorzystać do rozwijania słownictwa poprzez układanie historyjek obrazkowych, dzielenia wyrazów na sylaby, wyodrębniania głosek. Dzieci mogą z pomocą maty uczyć się kolorów, określać wielkości czy stosunki przestrzenne. Z pewnością mata ułatwi nam usprawnianie zaburzonych funkcji u dzieci, a także wspomoże ich rozwój i doskonalenie nabytych już umiejętności.

Podczas zajęć z matą na zajęciach rewalidacyjnych z uczniami klas 1-2 doskonaliłyśmy umiejętność przeliczania, utrwalałyśmy poczucie wartości liczby. Ćwiczyłyśmy także stosunki przestrzenne, rozwijałyśmy słownictwo i kategoryzowanie. Wdrażałyśmy dzieci do porównywania i porządkowania liczb. Aktywność dzieci przerosła nasze oczekiwania. Taka forma nauki wywarła na nich duże wrażenie i z pewnością nie mogą się już doczekać kolejnych zajęć.

Rysunek 13. Konkurs Matematyczny z wykorzystaniem Maty Mistrzów Kodowania. Zadanie polegało na odczytaniu przez zespół zapisanego kodu i zamieszczeniu na macie konkretnych kół, które utworzyły widoczny obraz.

Matę Mistrzów Kodowania z powodzeniem wykorzystać można także ze starszymi uczniami, co uczyniliśmy podczas Szkolnego Konkursu Matematycznego.

Mamy nadzieję, że przedstawione przykłady utwierdziły Czytelników w przekonaniu, że programowanie to przestrzeń edukacyjna, którą warto odpowiednio zagospodarować także w pracy z dziećmi ze specjalnymi potrzebami edukacyjnymi.

Zyta CZECHOWSKA i **Jolanta MAJKOWSKA** – nauczycielki ZSS w Kowanówku, autorki blogów edukacyjnych
www.specjalni.pl,
www.tikzklasa.blogspot.com,
trenerki prowadzące szkolenia i warsztaty

Programowanie w Akademii Khana[1]

Witold KRANAS

Podstawowe informacje o Akademii Khana

Akademia Khana to portal internetowy, szkoła wybitnego nauczyciela Sala Khana, dostępna bezpłatnie dla każdego i wszędzie tam, gdzie dociera Internet. Zasoby Akademii Khana zawierają tysiące krótkich filmów – lekcji oraz ćwiczenia. Salman Khan z wyczuciem i w prosty sposób przedstawia problemy, ilustrując je zapisami na tablicy, a czasem również grafiką. Mówi zazwyczaj wyraźnie i niezbyt szybko, co pozwala śledzić tok wykładu. Portal jest nieustannie doskonalony – teraz może on wyglądać inaczej niż w momencie, gdy był tworzony ten opis. W Polsce głównym promotorem tłumaczenia Akademii Khana jest prof. Lech Mankiewicz z Centrum Fizyki Teoretycznej PAN. Polskie zasoby Akademii Khana znajdują się pod adresem `pl.khanacademy.org`.

Informatyka

Informatyka jest szybko rozwijającym się działem Akademii Khana. Składa się on z dwóch części: *Informatyka* i *Programowanie* (dodatkowo została wydzielona *Godzina Kodowania*). Część *Informatyka* zawiera obecnie 4 rozdziały/kursy:

- **Algorytmy** – sporo materiałów do czytania, wymaga znajomości środowiska JavaScript, w którym realizowane są przykłady algorytmów. Zaczyna się od wyszukiwania binarnego, a kończy na szybkim sortowaniu, reprezentacji grafów i przeszukiwaniu wszerz (BSF).

- **Podróż do kryptografii** – zaczyna się od historii kryptografii i szyfrów, by dojść do szyfrowania RSA, arytmetyki modularnej, testowania pierwszości i algorytmów probabilistycznych.

  Oba te kursy to raczej poziom liceum w zakresie rozszerzonym. Kolejne dwa wydają się nieco łatwiejsze i sporą część ich materiału można zrealizować na poziomie ostatnich klas szkoły podstawowej (gimnazjum).

- **Podróż do teorii informacji** – składa się z dwóch działów, historycznego – od historii alfabetu do kodu Morse'a i współczesnego, zawierającego zagadnienia prędkości modulacji, pomiaru informacji, łańcuchów Markowa, entropii i korekcji błędów. Jest tu świetny film „Pomiar informacji", na którym można oprzeć interesującą lekcję informatyki.

- **Internet: wprowadzenie** – krótki kurs z filmami wyjaśniającymi podstawy działania Internetu, terminologię i problemy związane z powszechnym korzystaniem z Internetu.

[1] Artykuł został opracowany na podstawie warsztatów przygotowanych przez Macieja Borowieckiego i Witolda Kranasa na konferencję Informatyka w Edukacji 2016 w Toruniu.

Rysunek 2. Kadr z filmu „Pomiar informacji".

Programowanie

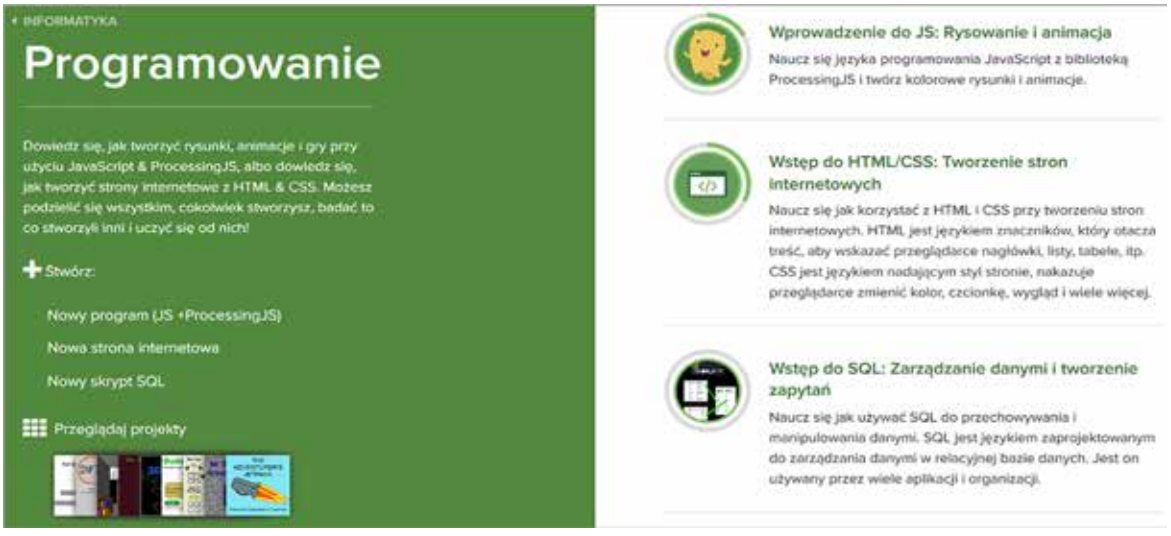
Ta część rozwija się bardzo szybko. Równie szybko postępuje opracowywanie polskiej wersji materiałów. Na potrzeby nauki programowania zostało utworzone specjalne oprogramowanie umożliwiające interaktywną pracę z kodem. Obecnie dostępne są kursy programowania: JavaScript i Processing, HTML/CSS i SQL. Najbardziej rozbudowane jest programowanie w JavaScript składające się z trzech kursów:

- Wprowadzenie do JS: Rysowanie i animacja.
- Zaawansowany JS: Gry i Wizualizacje.
- Zaawansowany JS: Symulacja natury.

Dwa dodatkowe kursy są rozwinięciem JS i HTML/CSS:

- HTML/JS: Tworzenie interaktywnych stron internetowych.
- HTML/JS: Tworzenie interaktywnych stron internetowych z jQuery.

Od ponad roku rozwijany jest jeszcze jeden kurs związany z programowaniem: *Pixar w pigułce* (`pl.khanacademy.org/partner-content/pixar`). Tworzy go Akademia Khana we współpracy z wytwórnią filmów animowanych *Pixar*. Jest to kurs animacji komputerowej, z tematami takimi jak: animowanie, chmary obiektów, efekty specjalne, renderowanie, wirtualne kamery. Każdy



Rysunek 3. Podstawowe kursy działu Programowanie w Akademii Khana.

z jego działów ma wprowadzenie – opis efektów, a następnie przedstawienie zagadnień matematycznych i informatycznych wykorzystywanych przy tworzeniu animacji.

Wprowadzenie do JavaScript

Ten kurs można próbować zrealizować na poziomie ostatnich klas szkoły podstawowej lub pierwszej klasy liceum. Znacznym ułatwieniem dla nauczyciela jest dokładne rozplanowanie kolejnych tematów. Przygotowanie do lekcji jest dość proste – należy przejrzeć kolejne tematy. Z kolei uczniowie mogą pracować we własnym tempie, a jeśli opuszczą lekcję, mają możliwość samodzielnej pracy w domu.

Kurs zaczyna się od podstaw: wyjaśnienia, na czym polega programowanie, objaśnienia środowiska, wprowadzenia do rysowania i kolorowania. Kolejne działy to: *Zmienne, Podstawy animacji, Programy interaktywne, Tekst i ciągi znaków, Funkcje, Logika i instrukcja if, Zapętlanie, Tablice, Obiekty*, a na końcu: *Stawanie się coraz lepszym programistą*. Kurs jest więc dość obszerny.

Klasa i postępy uczniów

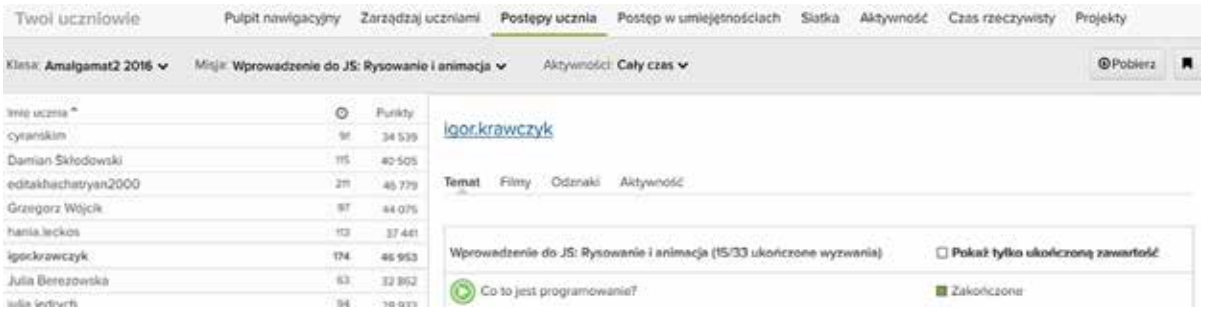
Nauczyciel może utworzyć klasę, wyznaczyć jej temat-misję i oglądać postępy uczniów. Uczniowie logują się do Akademii Khana (przez Gmaila lub Facebooka) i w okienku *Trenerzy/Dodaj trenera* wpisują kod klasy podany przez nauczyciela. Nauczyciel ma dobry przegląd pracy uczniów: globalny – liczbę uzyskanych punktów – i szczegółowy – ukazujący każdy temat przerabiany przez ucznia. Może indywidualizować uczenie przez wyznaczanie zadań poszczególnym uczniom.

Środowisko programistyczne

Jak zostało wspomniane wcześniej, na potrzeby kursów programistycznych w Akademii Khana zostało opracowane środowisko programistyczne umożliwiające interaktywną pracę z kodem (JavaScript) z wykorzystaniem biblioteki Processing. Charakteryzuje się ono prostym i przyjaznym interfejsem, cały czas dostępnym „pod ręką" systemem pomocy. Środowisko można wykorzystać niezależnie od kursu, proponując uczniom własne zadania do wykonania. Właściwą metodą wydaje się propozycja dodatkowych zadań w trakcie kursu. Przykłady problemów (do rozwiązania w połowie kursu) znajdują się poniżej.

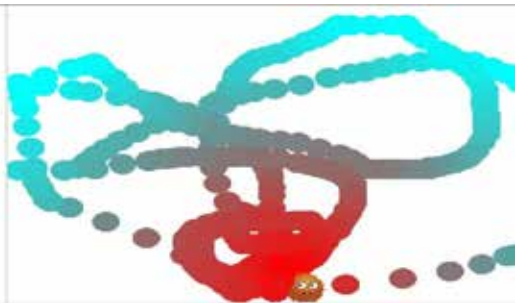
Kolorowe kółka

Pierwszy problem to symulacja zabawy w „Ciepło – zimno". Najpierw losujemy `(random)` położenie `(xl, yl)` niespodzianki na płaszczyźnie. Dalej trzeba umieć narysować kółko `(ellipse)` bez obwódki `(noStroke)` w miejscu, gdzie znajduje się kursor myszy `(mouseX, mouseY)` i wtedy, gdy naciskamy klawisz myszy `(if (mouseIsPressed))`. Wszystko to trzeba wykonywać w sposób ciągły w pętli (`draw`). Kolor kółka `(fill)` powinien być uzależniony od odległości `(dist)` kursora myszy od niespodzianki. Im bliżej niespodzianki, tym więcej koloru czerwonego `(R)`, a mniej zielonego `(G)` i niebieskiego `(B)`. Jeśli jesteśmy dostatecznie blisko, niespodzianka `(image)` powinna się pokazać. Rysunek 5 przedstawia listing i wynik działania programu.

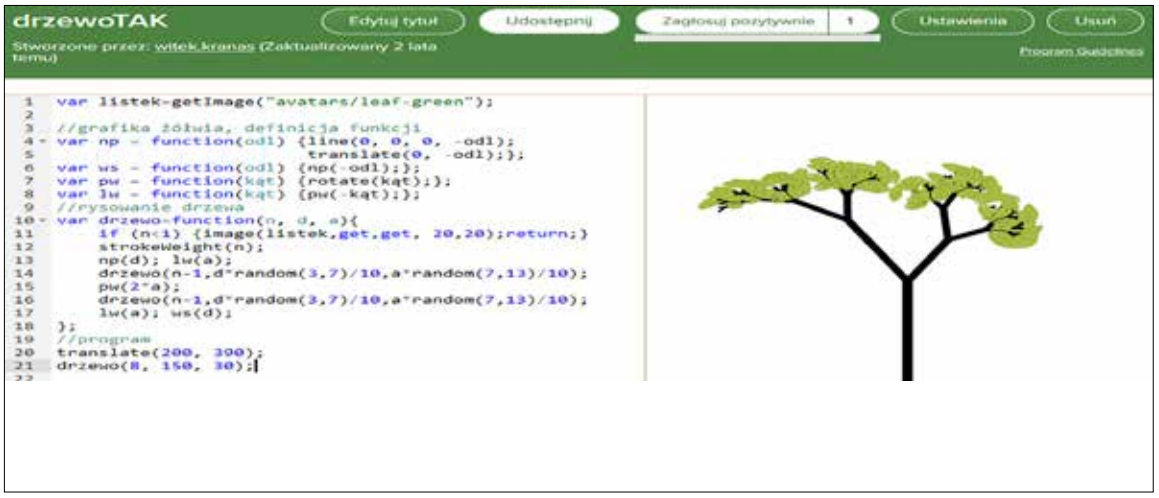


Rysunek 4. Klasa i postępy ucznia.

```
noStroke();

var xl=random(1,399);
var yl=random(1,399);

var draw = function() {
    var r=dist(xl,yl, mouseX,mouseY);
    var R=255-r;
    var G=r;
    var B=r;
    if (mouseIsPressed){
        fill(R, G, B);
        ellipse(mouseX,mouseY, 20,20);
        if(r<5){image(getImage("avatars/mr
-pink-orange"), xl, yl, 30,30);}
    }
};
```



Rysunek 5. Program „Ciepło – zimno".



Rysunek 6. Program rysujący drzewo binarne.

Rysowanie fraktali

W omawianym środowisku łatwo można przejść do grafiki żółwia i tworzyć np. fraktale. Wystarczy zdefiniować funkcje: naprzód (`np`), wstecz (`ws`), prawo (`pw`) i lewo (`lw`). Teraz będą działać proste algorytmy rysowania fraktali. Jednym z takich fraktali jest drzewo binarne stopnia n. Konstrukcję drzewa można podzielić na elementy podstawowe i drzewa niższego stopnia. Element podstawowy to pień (odcinek) o pewnej długości (d). Wyrastają z niego dwa poddrzewa (czyli drzewa stopnia $n-1$) o długości mniejszej niż pień. Jedno z nich jest odchylone od pnia w lewo o pewien kąt (a), drugie zaś w prawo o ten kąt. Algorytm jest więc rekurencyjny, zaczyna się od warunku stopu i zawiera dwa wywołania rekurencyjne. Najtrudniejszy moment to powrót do podstawy pnia na końcu. Program i narysowane drzewo przedstawia rysunek 6. Mając „bibliotekę" grafiki żółwia, bez problemu zrealizujemy w tym środowisku kolejne fraktale: trójkąt Sierpińskiego, krzywą i płatek Kocha.

Bibliografia

1. Program rysujący drzewo binarne: https://pl.khanacademy.org/computer-programming/drzewotak/4806732935069696, dostęp sierpień 2017.
2. Program zabawy w „Ciepło – zimno": https://pl.khanacademy.org/computer-programming/ciepo-zimno-3/5597997738164224, dostęp wrzesień 2017.
3. Strona Akademii Khana poświęcona programowaniu (w polskiej wersji): https://pl.khanacademy.org/computing/computer-programming, dostęp sierpień 2017.
4. Strona kursu Pixar w pigułce: https://pl. khanacademy.org/partner-content/pixar, dostęp wrzesień 2017.

Witold KRANAS jest nauczycielem konsultantem w Ośrodku Edukacji Informatycznej i Zastosowań Komputerów w Warszawie.