NAUKA PROGRAMOWANIA W SZKOŁACH

Czas na upgrade?

CENTRUM CYFROWE

projekt: polska

MISTRZOWIE KODOWANIA

SPIS TREŚCI

KOMENTARZE:

Autorzy raportu:
Mirosław Filiciak, Kamil Sijko, Alek Tarkowski

Projekt graficzny:
Paulina Tyro-Niezgoda

Warszawa, czerwiec 2013

CENTRUM CYFROWE
projekt:polska

WSTĘP

Stwierdzenie, że komputery i inne urządzenia elektroniczne sterowane oprogramowaniem wdzierają się we wszystkie sfery naszego życia, brzmi jak truizm. Nowe technologie wywarły wpływ na to jak pracujemy, jak spędzamy czas wolny, jak uczymy się i kontaktujemy się z bliskimi, jak zdobywamy wiedzę i informacje. Do Sieci przenosi się coraz więcej usług, tak publicznych jak i komercyjnych. Jak pokazują choćby dane z badania „Diagnoza społeczna 2011", z telefonów komórkowych korzysta 85% Polaków, a komputery znajdują się w 66% gospodarstw domowych w naszym kraju. Równocześnie jednak choćby podstawowe umiejętności programowania tych urządzeń zastrzeżone są dla bardzo wąskiej grupy Polaków i nie zmienia się to w czasie - w roku 2007 umiejętność napisania programu deklarowało 8,7% Polaków, w roku 2011 - 8,8%.[1]

Ten rozdźwięk pomiędzy znaczeniem i powszechnością programowalnych urządzeń w życiu codziennym, a słabą znajomością umiejętności programowania uważamy za niepokojący. Technologie te bowiem stają się kluczowym czynnikiem kształtującym dziś społeczeństwa – a to dzięki programowaniu można mieć nad tymi technologiami kontrolę. To nie przypadek, że socjologowie opisujący współczesne społeczeństwa sięgają po odniesienia do technologii, jak choćby Manuel Castells piszący o „społeczeństwie sieci", czy Barry Wellman i Lee Reinie opisujący zmiany społeczne w kategoriach „społecznego systemu operacyjnego".[2] Po inne metafory sięgają pisarze i futuryści, jak William Gibson, który nasze zrośnięcie z elektronicznymi narzędziami charakteryzuje mówiąc o drugim, „zewnętrznym układzie nerwowym", którym są dla nas komputery i inne urządzenia podpięte do internetu[3]. Bo przecież nasze „ja" rozciąga się – bardziej niż kiedykolwiek wcześniej - poza ciało, czego przykładem może być choćby przechowujący kontakty i zdjęcia telefon komórkowy, stanowiący dla nas rodzaj zewnętrznej pamięci. Jednocześnie coraz częściej korzystamy z technologii cyfrowych w miejsce analogowych przedmiotów. Tu znów dobrym przykładem jest zastąpienie albumu z rodzinnymi zdjęciami w postaci odbitek katalogiem zdjęć w telefonie lub komputerze.

Przemiany społeczne – zarówno te na globalnym poziomie makro, jak i te zachodzące w bardziej osobistej i intymnej skali, oparte są dziś na technologiach wykorzystujących oprogramowanie. Równocześnie bez kompetencji programowania oraz powiązanej z nią zdolności zrozumienia, jak działają te technologie - nie możemy do końca zrozumieć przyczyn zachodzących zmian, ani nad nimi zapanować..

Mitem jest też przeświadczenie o technologicznej biegłości młodych ludzi[4]. Nie zmienia tego obecność lekcji informatyki w gimnazjach i szkołach średnich czy zajęć komputerowych w szkole podstawowej. Zapewne krokiem w dobrym kierunku są nowe podstawy programowe, na efekty wdrożenia których przyjdzie jeszcze poczekać. Zgodnie z nimi, edukacja informatyczna stanowi ważny element procesu nauczania - młodzi Polacy zajęcia z komputerami odbywają przez cały czas nauki w szkole podstawowej, już od I klasy. Na pierwszym etapie nauczania zajęcia mają być integrowane z innymi działaniami uczniów i jest to założenie, z którym trudno się nie zgodzić. Autorzy podstawy programowej kładą zresztą nacisk na dobre wykorzystanie zajęć w gimnazjum i liceum, a więc etapu, na którym pojawia się informatyka na poziome rozszerzonym. Równocześnie jednak trudno oczekiwać, że wszystkie problemy, o których piszemy w niniejszym opracowaniu, dzięki nowym zapisom znikną jak za dotknięciem czarodziejskiej różdżki. Dobrze pokazują to komentarze

nauczycieli, którzy znaleźli się wśród osób, które poprosiliśmy o wzbogacenie tego raportu własnymi opiniami. Daleko w nich do hurraoptymizmu. To fakt, że w preambule do podstawy programowej, wśród najważniejszych umiejętności zdobywanych w szkole podstawowej pojawia się „umiejętność posługiwania się nowoczesnymi technologiami informacyjno-komunikacyjnymi", a podobne sformułowania pojawiają się też w odniesieniu do nauczania na kolejnych etapach edukacji, gdzie zwiększa się rola informatyki. Widać jednak wyraźny rozdźwięk pomiędzy podstawą programową a kompetencjami nauczycieli – szkoła nie zawsze jest atrakcyjnym miejscem dla programistów. Konsekwencje tego spostrzeżenia widać zresztą w standardach przygotowania nauczycieli w zakresie technologii informacyjno-komunikacyjnych, gdzie – w odróżnieniu od podstaw programowych - informacje o programowaniu nie pojawiają się wcale. Rzeczywistość nauki programowania w większości polskich szkół nie wygląda więc ani tak ponuro, jak głoszą krzywdzące stereotypy, ani tak różowo, jak mogłoby wynikać z wyrywkowej lektury wytycznych Ministerstwa Edukacji.

Wobec powyższego, uważamy, że wciąż istotnym problemem pozostaje propagowanie znaczenia choćby podstawowych umiejętności informatycznych we współczesnym świecie. Przede wszystkim istotna jest jednak wymiana doświadczeń i współpraca pomiędzy różnymi środowiskami zainteresowanymi rozwijaniem znajomości programowania wśród Polaków. Dlatego piszemy m.in. o dobrych praktykach, często o charakterze oddolnym, wypracowanych modelach działania i zasobach, z których można skorzystać.

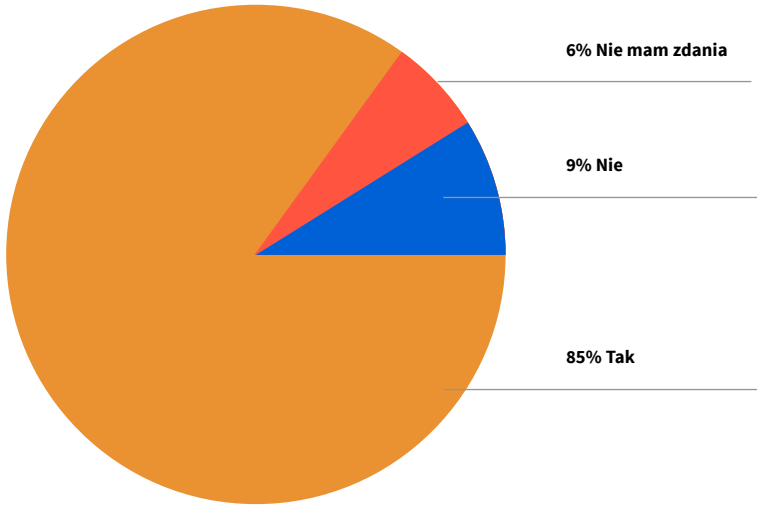
Polacy bywają słusznie dumni z sukcesów naszych informatyków w międzynarodowych konkursach. Nikt nie ma wątpliwości, że w naszym kraju nie brakuje zdolnych programistów. Sukcesy informatycznej elity i możliwie powszechna znajomość programowania to jednak dwie zupełnie różne sprawy. Odwołując się do sportowych porównań, które będą w tym opracowaniu powracać można stwierdzić, że nie chodzi nam o to, by garstka naszych rodaków zdobywała olimpijskie medale. Raczej o to, by znacząca część całej populacji była aktywna, np. biegając. Aby ludzie ćwiczyli, byli sprawniejsi, ale też lepiej rozumieli swój organizm. Organizm, który w szerokim rozumieniu nie kończy się dzisiaj na czubkach naszych palców – bo klawisze, w które tymi palcami stukamy, są przedłużeniem nas samych.

Niniejszy raport ma na celu przedstawienie idei powszechnej nauki programowania dla dzieci i młodzieży oraz kontekstu realizowania tego rodzaju programów. W pierwszej części prezentujemy wynik badania sondażowego badającego poglądy Polaków na znaczenie nauki programowania. W części drugiej prezentujemy teoretyczne uzasadnienia istotności umiejętności programowania w świecie przepełnionym technologiami cyfrowymi. Część trzecia dotyczy rynku pracy i zapotrzebowania na kadry posiadające odpowiednie kompetencje informatyczne. Część czwarta jest analizą nauczania informatyki w obecnym programie nauczania. Część piąta i ostatnia to prezentacja dobrych praktyk: studium przypadku projektu Coder Dojo, najciekawszych projektów polskich i zagranicznych, oraz stosowanych w nauczaniu języków programowania. Raport uzupełniają wypowiedzi ekspertów zajmujących się edukacją medialną oraz nauczaniem ■

OPINIE POLAKÓW

Wykres 1

Czy nauka programowania komputerów w szkołach przyniosłaby uczniom korzyści?



Okazuje się, że przekonanie o przydatności umiejętności programowania dla uczniów jest powszechne. To nie tylko opinia osób zajmujących się informatyką, potrzebami rynku pracy czy edukacją medialną. Pokazuje to przeprowadzone przez nas badanie, w którym respondenci odpowiadali na pytanie „Czy nauka programowania komputerów w szkołach przyniosłaby uczniom korzyści?" oraz wskazywali obszary tych korzyści.

Aż 85% pełnoletnich Polaków uważa, że nauka programowania przyniosłaby korzyści dla uczniów. Z taką opinią nie zgadza się zaledwie 9% ankietowanych. Pozostałe 6% nie posiada na ten temat wyrobionego zdania. O korzyściach nie trzeba przekonywać zwłaszcza bezpośrednio zainteresowanych - opiekunów osób uczących się, jak i ich samych.

W poparciu dla nauki programowania zgodni są kobiety i mężczyźni, za to czynnikiem różnicującym jest wiek - ale tylko w jednym, specyficznym przypadku. Wyraźnie niższe od uśrednionego wyniku poparcie widoczne jest wśród osób powyżej 60. roku życia (71% poparcia). Inną grupą, która na tle całej dorosłej populacji Polski rzadziej dostrzega korzyści z nauki programowania są osoby z wykształceniem podstawowym (78% udziela poparcia, podczas gdy w innych grupach jest to od 87% do 89%). Pewne rozbieżności w opiniach można też zaobserwować pomiędzy osobami mieszkającymi w ośrodkach różnej wielkości (w największych miastach, z powyżej 500 tys. mieszkańców, odpowiedzi pozytywnej udzieliło aż 93% ankietowanych) oraz w różnych regionach kraju (odpowiedzi pozytywne wahają się pomiędzy 75% w regionie wschodnim i 77% w regionie śląskim, a 90% w Małopolsce i 93% w regionie zachodnim).

Nieco częściej potrzebę nauki programowania dostrzegają osoby będące opiekunami dzieci - na „tak" było 89% tej grupy (w wypadku reszty populacji - 84%). Spore znaczenie ma też wykonywane zajęcie/zawód. Najniższe, choć wciąż bardzo wysokie poparcie dla idei uczenia programowania w szkołach zgłaszają emeryci (77%) i rolnicy (82%), najwyższe - osoby uczące się i studiujące (91%). Wyraźne różnice w częstotliwości odpowiedzi pozytywnych widać też pomiędzy internautami a osobami, które nie korzystają z internetu. Wśród tych pierwszych przydatność nauki programowania w szkołach dostrzega aż 90% ankietowanych - wśród niekorzystających już tylko 76%. Zdecydowanie na „nie" jest 8% internautów i 11% niekorzystających z Sieci. Rozbieżności widać też w kwestii częstotliwości udzielania odpowiedzi „nie wiem/nie mam zdania". Wybrało ją zaledwie 3% internautów i aż 13% osób niekorzystających z Sieci.

Zaobserwowane różnice wskazują wyraźnie, że potencjał nauki programowania częściej dostrzegają osoby młodsze, lepiej wykształcone i aktyw-

Wykres 2

Jakiego typu mogą być to korzyści?

ne na rynku pracy lub planujące taką aktywność w najbliższym czasie. Pewne znaczenie mają też doświadczenia z korzystania komputera i internetu, choć trzeba podkreślić, że w Polsce niekorzystanie z Sieci nakłada się na inne podziały - internauci to po prostu statystycznie młodsza, lepiej wykształcona, bardziej zamożna i mieszkająca w większych ośrodkach część populacji naszego kraju.

Widać, że nauka programowania w szkołach może spełnić istotną rolę nie tylko wpisując się w powszechne oczekiwania, ale też wyrównując różnice społeczne i docierając do osób z mniejszych ośrodków, których świadomość w kwestii znaczenia choćby podstawowych umiejętności informatycznych jest niższa niż w wypadku pozostałej części populacji.

W drugiej części badania osoby, które udzieliły pozytywnej odpowiedzi na pierwsze pytanie, poprosiliśmy o wskazanie typu korzyści, jakie według nich może przynieść uczniom nauka programowania. Najpopularniejszą odpowiedzią był „wzrost szans na rynku pracy" - tę opcję zaznaczyło aż 92% ankietowanych. Na korzyści związane ze sprawniejszym korzystaniem z komputera lub innych urządzeń zwróciło uwagę 90% badanej grupy, minimalnie mniej, bo 89%, wskazało na powiązanie nauki programowania z rozwojem umiejętności logicznego myślenia. Za ciekawe hobby programowanie uznało 85% dorosłych Polaków, a korzyści związane z powszechniejszym wsparciem dla otwartych projektów informatycznych rozwijanych w internecie wskazało 81% badanych. Najmniej liczna grupa - ale wciąż aż 3 na 4 ankietowany osoby - wskazała na efekty w postaci bardziej krytycznego podejścia do gotowego oprogramowania.

Podobnie jak w wypadku pytania pierwszego, w odpowiedziach zauważalne były różnice pomiędzy różnymi grupami. We wzrost szans na rynku

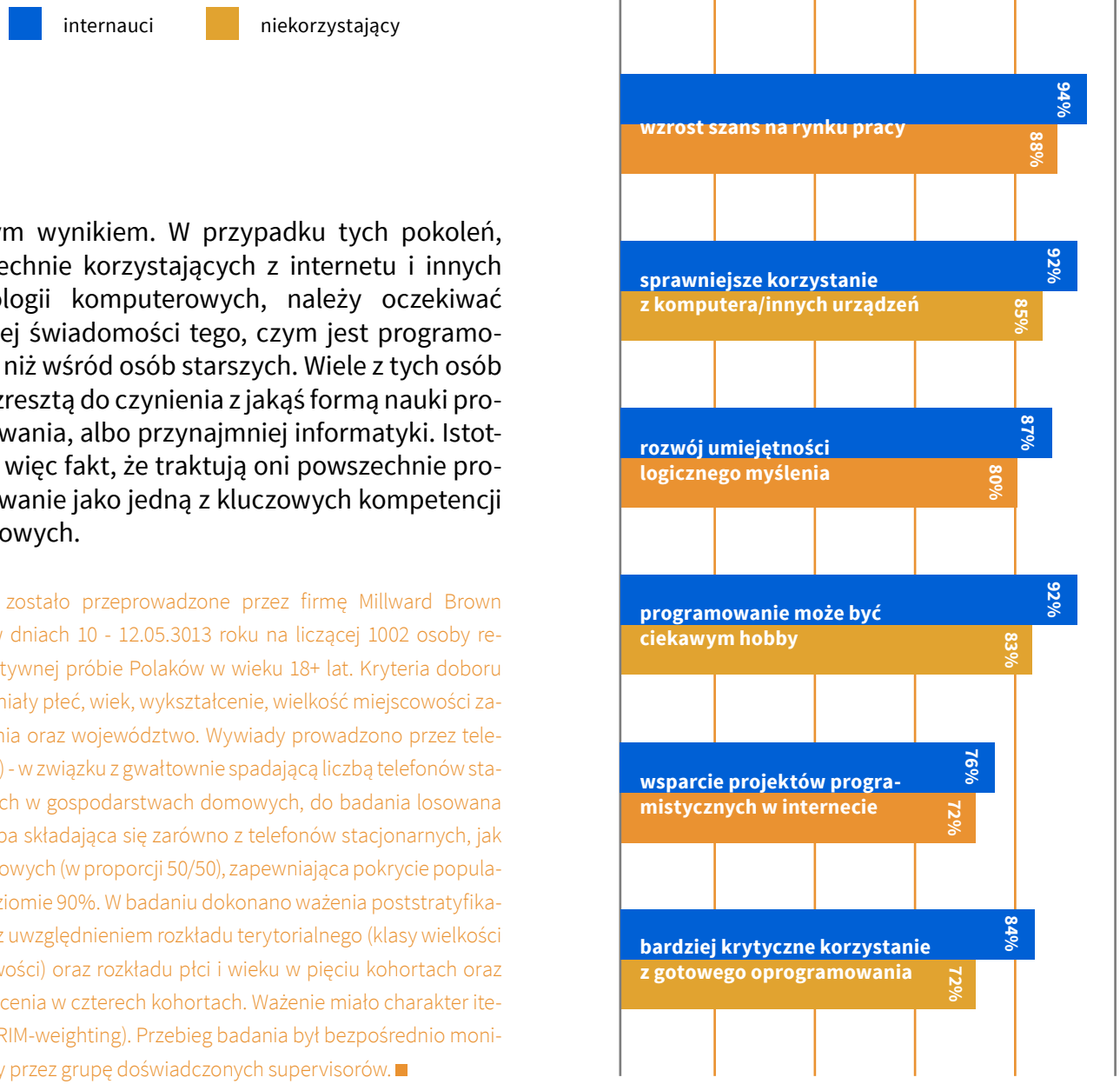
pracy najsilniej wierzą osoby młode, w wieku 18-24 lata, zapewne realnie tej pracy poszukujące lub planujące takie poszukiwania w nieodległej perspektywie - w tym segmencie pozytywnych wskazań było aż 98%. Na wzrost szans na rynku pracy wskazał też identyczny odsetek osób uczących się lub studiujących. Najmniej przekonaną o takim wpływie na dalsze losy uczniów są za to osoby z wykształceniem zawodowym - 87% wskazań (co zapewne jest logiczną konsekwencją dostępnych tej grupie na rynku pracy zawodów). Osoby z wykształceniem zawodowym w wyraźnie niższym niż średnia stopniu postrzegają też naukę programowania jako szansę na bardziej krytyczne korzystanie z komputera (66%). Z kolei osoby z wykształceniem wyższym znacznie częściej są przekonane o wspieraniu rozwoju logicznego myślenia (95%). Znaczące, że w każdej z wymienionych kategorii pozytywnych wskazań częściej udzielały osoby korzystające z internetu. Wśród nie-internautów 3% wybrało odpowiedź „nie wiem/trudno powiedzieć" - wśród internautów nie było takich osób.

Powszechność poparcia dla nauki programowania w szkołach w najmłodszych pokoleniach, wraz z przekonaniem o znaczeniu tej nauki – na przykład dla szans na rynku pracy – jest szczególnie istotnym wynikiem. W przypadku tych pokoleń, powszechnie korzystających z internetu i innych technologii komputerowych, należy oczekiwać większej świadomości tego, czym jest programowanie, niż wśród osób starszych. Wiele z tych osób miało zresztą do czynienia z jakąś formą nauki programowania, albo przynajmniej informatyki. Istotny jest więc fakt, że traktują oni powszechnie programowanie jako jedną z kluczowych kompetencji zawodowych.

Badanie zostało przeprowadzone przez firmę Millward Brown Polska w dniach 10 - 12.05.3013 roku na liczącej 1002 osoby reprezentatywnej próbie Polaków w wieku 18+ lat. Kryteria doboru uwzględniały płeć, wiek, wykształcenie, wielkość miejscowości zamieszkania oraz województwo. Wywiady prowadzono przez telefon (CATI) - w związku z gwałtownie spadającą liczbą telefonów stacjonarnych w gospodarstwach domowych, do badania losowana była próba składająca się zarówno z telefonów stacjonarnych, jak i komórkowych (w proporcji 50/50), zapewniająca pokrycie populacji na poziomie 90%. W badaniu dokonano ważenia poststratyfikacyjnego z uwzględnieniem rozkładu terytorialnego (klasy wielkości miejscowości) oraz rozkładu płci i wieku w pięciu kohortach oraz wykształcenia w czterech kohortach. Ważenie miało charakter iteracyjny (RIM-weighting). Przebieg badania był bezpośrednio monitorowany przez grupę doświadczonych supervisorów. ■

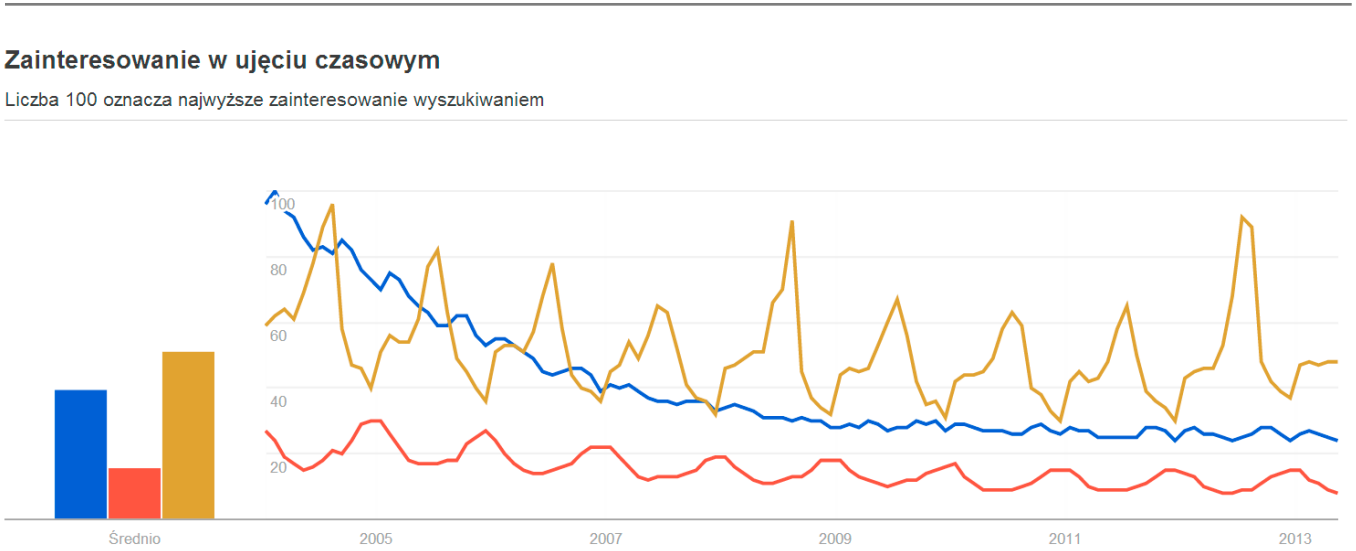
Wykres 3

Korzyści z nauki programowania w szkole - różnice pomiędzy internautami i osobami niekorzystającymi z internetu



DWIE KULTURY

Wykres 4



Pomimo przedstawionego obok wysokiego deklarowanego poparcia dla popularyzacji programowania, użytkownicy komputerów nie są szczególnie zainteresowani tą tematyką. Przyjrzyjmy się na przykład statystykom największej na świecie wyszukiwarki Google dla hasła „programming" (programowanie - kolor niebieski), zestawiając je z hasłami opisującymi inne formy aktywności: „knitting" (szydełkowanie - kolor czerwony) oraz „swimming" (pływanie - kolor żółty)[5]: [Wyk.4]

Wynika z niego, że po spadku zainteresowania w latach 2004-2008 mamy obecnie do czynienia ze stagnacją – kontrastowe hasła dotyczące szydełkowania i pływania wykazują w tym samym czasie stabilny udział w zapytaniach, choć oczywiście zróżnicowany sezonowo (zimą spada zainteresowanie pływaniem, a rośnie szydełkowaniem). Innym kontrintuicyjnym wynikiem jest ten dotyczący pochodzenia zapytań o programowanie – zaobserwować można odejście od intuicyjnych Ameryki, Europy i Azji na rzecz przede wszystkim Afryki, krajów takich jak Etiopia, Kenia, Tanzania, Botswana, Nigeria, oraz z kontynentu azjatyckiego – Indie (należy jednak zastrzec, że wykres przedstawia dane wyskalowane i znormalizowane – oznacza to, że bezwzględna liczba zapytań mogła w tym okresie pozostać taka sama, albo nawet wzrosnąć – to co obserwujemy na wykresie to spadek relatywnego zainteresowania programowaniem w stosunku do wszystkich innych zapytań).

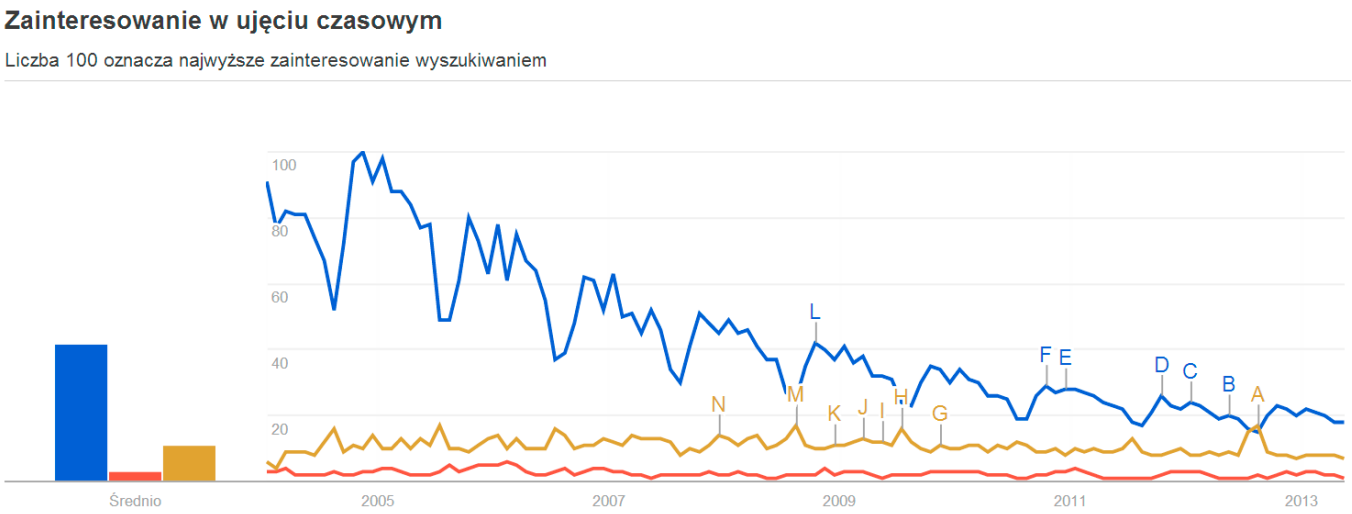
Z drugiej jednak strony trend ten nie musi dotyczyć Polski, bo przecież Polacy wyszukują po polsku „programowanie". Niestety tak nie jest[6]: [wyk.5]

Polska niemal idealnie wpisuje się w światową linię trendu, w Polsce również – podobnie jak na całym świecie – zaobserwować możemy fluktuacje regionalne zainteresowania w czasie: [Wyk.6]

W roku 2004 zaobserwować można wzmożone zainteresowanie wokół największych polskich miast: Warszawy, Poznania, Wrocławia, Krakowa oraz aglomeracji śląskiej. Obecnie relatywnie dużo więcej zapytań dotyczących programowania kierowanych jest z regionów peryferyjnych: województwa podlaskiego, podkarpackiego czy świętokrzyskiego.

Relatywny spadek zainteresowania programowaniem następuje w momencie, kiedy coraz większa liczba miejsc pracy wymaga takich kompetencji. Co istotne, mowa tutaj nie tylko, ani nawet nie przede wszystkim o miejscach pracy dla specjalistów informatyków – mamy bowiem do czynienia z rosnącą liczbą zawodów nieinformatycznych i nie inżynieryjnych w których kompetencje programistyczne będą coraz bardziej potrzebne (np. analitycy, badacze, menedżerowie, itd.).. Dlaczego tak się dzieje?

Wykres 5



Być może przyczyn należy szukać w kulturze. W opublikowanej już w roku 1964, a więc niemalpół wieku temu, książce *Dwie kultury*[7], C.P. Snow postulował niwelowanie podziału między humanistyką a naukami ścisłymi – podziału, który jest coraz mniej uzasadniony, jednak wciąż obecny. Wciąż myślimy w kategoriach opozycji pomiędzy tymi dwiema sferami, sprawy technologii pozostawiając „fachowcom" - choć tak naprawdę funkcjonujemy w „trzeciej kulturze", w której elementy nauk ścisłych (w tym informatyki) i humanistycznych splatają się ze sobą. Dobrze widać to w obszarze humanistycznej refleksji nad mediami, zapośredniczającymi dziś większą część naszych relacji z kulturą i innymi ludźmi. Wybitny badacz mediów, Lev Manovich, pisze o tym tak: „Żeby zrozumieć nowe media, musimy odwołać się do informatyki. To właśnie tam znajdziemy nowe terminy, kategorie i funkcje charakteryzujące media, które stały się programowalne. Od medioznawstwa zmierzamy w stronę czegoś, co można by nazwać programoznawstwem, czyli od teorii mediów do teorii oprogramowania". W swej ostatniej książce dodaje: „>>Społeczeństwo informacyjne", >>społeczeństwo wiedzy<<, >>społeczeństwo sieci<<, >>media społecznościowe<< - bez względu na to, na której z nowych cech współczesności koncentruje się dana teoria społeczna, każda z nich jest związana z działaniem oprogramowania. Czas więc, byśmy to na nim się skoncentrowali"[8]. Z kolei niemiecki badacz Friedrich A. Kittler stwierdził, że aby we współczesnym świecie móc powiedzieć cokolwiek na temat kultury, należy znać przynajmniej dwa języki programowania.[9] Zadaniem, jakie staje dziś przed szkołą, jest więc zasypanie podziałów między kulturą i technologią. Bowiem osoby zdolne funkcjonować ponad tym podziałem, i płynnie przechodzić między sferą kultury i sferą technologii, będą najlepiej przygotowane do działania we współczesnym świecie.

W tym ujęciu programowanie jawi się jako umiejętność bliska znajomości alfabetu. Służy zarówno – przez analogię do czytania – rozumieniu otaczającego nas świata, jak i daje możliwość wyrażania siebie – na wzór umiejętności pisania.. Jak napisał Douglas Rushkoff, „Technologie cyfrowe są programowalne. To sprawia, że są profilowane przez tych, którzy posiadają umiejętność programowania. W erze cyfrowej musimy nauczyć się jak tworzyć oprogramowanie, albo zaryzykować, że sami będziemy programowani. Programowanie nie jest trudne - i nie jest zbyt późno, by nauczyć się kodu zarządzającego przedmiotami, które używamy, lub przynajmniej zrozumieć, że za interfejsem kryje się kod. W innym razie znajdziemy się na łasce tych, którzy programują, ludzi, którzy im płacą, czy nawet samej technologii"[10]. To słowa z książki pod znaczącym tytułem: *Programuj, albo daj się programować*. Bo znajomość programowania staje się ważną częścią świadomego bycia w świecie.

Analogie z alfabetyzacją rozwija Piotr Celiński, piszący wręcz o ścieraniu się dwóch alfabetyzmów: „Gutenbergowski alfabetyzm wyrasta z ducha społeczeństwa tradycyjnego, zorganizowanego hierarchicznie i myślącego linearnie. Ukształtowały je najpierw pismo, a potem technologie analogowe służące jednokierunkowej, scentralizowanej komunikacji na skalę masową: druk, radio, kino i telewizja." Zakorzeniony w historii, dziś coraz wy-

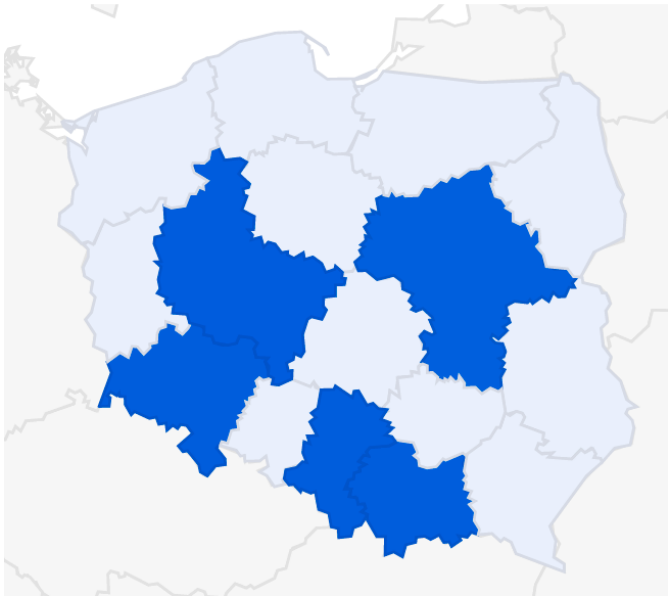
raźniej zastępowany przez nowy alfabetyzm cyfrowo sieciowy, oparty na nowych mediach i sieci. Te media składniają „nie tylko do >>czytania<< już >>napisanego<< świata, ale także do jego >>współredagowania<< i >>współpisania<<. Podważają istniejące hierarchie komunikacji masowej według modelu >>jeden do wielu<<, zamiast nich oferując możliwość komunikacji wszystkich ze wszystkimi na potencjalnie równych prawach (protokoły IP). Oferują alternatywne ekologie zasobów kultury i tworzą nieoczekiwane modele i mapy ich cyrkulacji (wiki, torrent). Wreszcie, pozwalają na niedostępną wcześniej kontrolę samych mediów przez ich użytkowników, dotąd pozostającą w rękach specjalistycznych instytucji i określonych środowisk (programowanie, personalizacja).”[11]

Reasumując: świadomość tego, jak działa kod komputerowy jest istotna, bo bez niej trudno o realne wykorzystanie potencjału nowych technologii. Trudno wówczas aktywnie wpływać na technologie i procesy, które oparte są na zaprogramowanych technologiach. Dobrze ilustrują to zresztą porażki programów edukacyjnych, w których obecność komputerów w szkole czy szerzej, w procesie nauczania, nie została wzmocniona zmianą logiki całego procesu edukacji. Komputery nie są przecież tylko kolejnym obszarem „do opanowania” – raczej elementem obecnym w innych obszarach ludzkiej aktywności. Stąd potrzeba otwierania się szkół na logikę, w której komputer byłby nie tylko sprzętem zamkniętym w osobnej sali i wykorzystywanym podczas lekcji informatyki, a integralną częścią uczenia się – tak jak jest integralną częścią funkcjonowania młodych osób poza szkołą.[22] Logikę, w której jest też miejsce na współpracę z projektami spoza szkoły, umożliwiającą profilowanie zajęć pod kątem indywidualnych zainteresowań i możliwości uczniów. To prowadzi nas w kierunku kolejnych wyzwań – bo przecież zadaniem szkoły jest przygotowanie uczniów do dobrego funkcjonowania po ukończeniu procesu edukacji. Także: na rynku pracy. ■

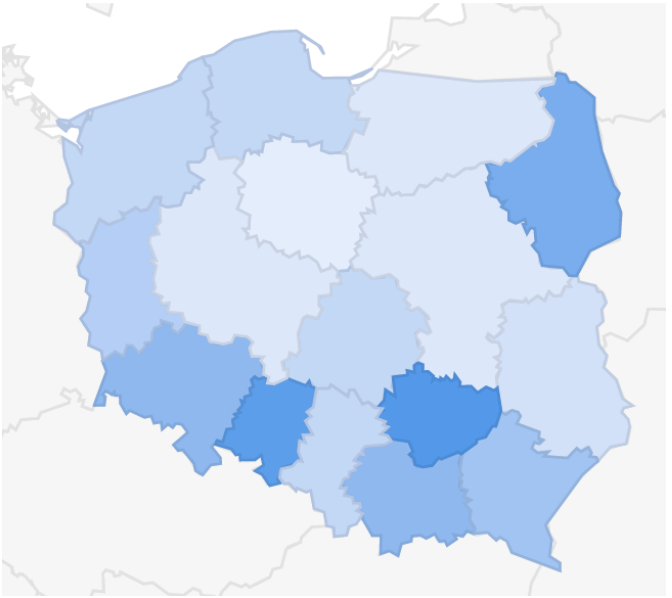
Wykres 6

Wyszukiwanie terminu „programowanie“ - zmiany w czasie z podziałem na regiony kraju (ciemniejsze barwy - większa ilość zapytań).

2004



2013



RYNEK PRACY

Polska gospodarka nigdy nie była gospodarką produkującą dużą liczbę innowacji – raczej gospodarką kopiującą wytwarzane gdzie indziej rozwiązania. Pomimo rozpowszechnionego przekonania o znaczeniu innowacji, ten stan rzeczy wciąż się nie zmienia. Jesteśmy raczej bezkrytycznymi naśladowcami, przejmującymi technologie i sposoby myślenia o nich z zagranicy, niż twórcami i innowatorami. Krytyczny – co nie znaczy, że bezwarunkowo nieufny – stosunek do technologii, oparty na zrozumieniu ich funkcjonowania, to coś, co mogłoby ten stan rzeczy stopniowo zmienić. Nie sposób jednak wywierać na tę sferę wpływ bez edukacji, bo innowacyjności nie da się „zaimportować".

Anna Giza-Poleszczuk i Renata Włoch w raporcie *Świt innowacyjnego społeczeństwa*, piszą: „(…) nie ulega wątpliwości, że procesy tworzenia i upowszechniania innowacji zależą od społecznego kapitału, to znaczy mobilności ludzi i idei, elastyczności tworzenia wokół nowych idei zespołów, mobilizacji zasobów, kultury współdziałania, gotowości eksperymentowania, zaufania itp. Wszystkie wymienione czynniki nie są jednak tłem czy podłożem, ale fundamentalnym wręcz czynnikiem samej innowacyjności. Podobnie przedmioty i procesy materialne (roboty, maszyny, rozwiązania informatyczne, do których zazwyczaj sprowadza się innowacje) nie są czymś osobnym, co zostało stworzone poza społeczeństwem (w laboratorium naukowym) i na społeczeństwo oddziałuje czy wręcz je determinuje. Przeciwnie, przedmioty i technologie, które materializują się w postaci innowacji, są wytworami procesu społecznego, również dlatego, że to, czy pojawią się jako nowość, zależy od negocjowania znaczenia i sensu odkrycia".[13] Giza-Poleszczuk i Włoch słusznie podkreślają, że procesy kulturowe czy społeczne nie zachodzą osobno od zmiany technologicznej, lecz są z nią bezpośrednio splecione. Jednak w praktycznym doświadczeniu większości osób – w szczególności tych nie posiadających umiejętności programowania i związanej z tym wrażliwości – technologie jednak są czymś obcym wobec społeczeństwa. Są urządzeniami, które działają – choć nie wiadomo za bardzo jak (co jest prawdą dotyczącą świadomości funkcjonowania niemal każdej złożonej technologii, z której korzystają współczesne społeczeństwa.

Oczywiście nie chodzi o tworzenie złudnych wizji, w których za sprawą wyższych kompetencji programistycznych swoich obywateli Polska miałaby się stać od razu technologiczną potęgą. Składa się na to wiele czynników poza skalą występowania tych kompetencji. Ciekawych inspiracji można szukać nie tylko w Krzemowej Dolinie, ale też w krajach takich jak Indie, gdzie karierę robią tzw. skromne innowacje (*jugaad innovation*) – niskonakładowe projekty, zakorzenione w tradycji „radzenia sobie", lepiej niż globalne rozwiązania spełniające potrzeby lokalnego rynku. Lokalne innowacje już dziś powstają w Polsce na poziomie oddolnym, warto jednak zwrócić uwagę, że ich twórcy (np. środowisko start-upów) to osoby w pewnym sensie uprzywilejowane. Warto byłoby zakres tego typu działań stymulować na szerszą skalę – a przecież niwelowanie podziałów społecznych jest zadaniem szkoły. Bez jej aktywnego wsparcia, rozwój nowych technologii przysłuży się tylko uprzywilejowanej kompetencyjnie i ekonomicznie mniejszości.

Wysokie kompetencje związane z obsługą komputerów to oczywiście nie tylko sprawa wyzwań innowacyjnej przyszłości – to także element oczekiwań dzisiejszych pracodawców. Z analizy Justyny

Jasiewicz wynika, że ogromna część dostępnych miejsc pracy zakłada pracę z komputerem. W 16% analizowanych ogłoszeń rekrutacyjnych polscy pracodawcy wymagają znajomości specjalistycznego oprogramowania lub nawet języków programowania.[14] Z kolei opracowania Polskiej Agencji Rozwoju Przedsiębiorczości pokazują, że specjaliści w dziedzinach informatyczno-komunikacyjnych ponadprzeciętnie często kontynuują swoją ścieżkę zawodową – co może sugerować, że nie mają kłopotów ze znalezieniem zatrudnienia i są zadowoleni z jego warunków. Oczywiście na rynku pracy doskonale radzą sobie informatycy - wśród zawodów postrzeganych przez Ministerstwo Spraw Wewnętrznych i Administracji jako strategiczne, wśród informatyków notowany jest najniższy odsetek bezrobotnych.[15] Równocześnie jednak – choć informatyków traktujemy jako specyficzną grupę, której poszerzenie nie jest głównym celem postulowanych w tym opracowaniu działań - odsetek studentów kierunków informatycznych w Polsce jest niższy od europejskiej średniej. Podobnie sytuacja wygląda, gdy przyjrzymy się liczbie osób w wieku 16-24 lata, które napisały kiedykolwiek program komputerowy, w tym rankingu zajmujemy dopiero 22. pozycję na 27 krajów Europy.[16] Z kolei w opracowaniu „Polska 2030. Wyzwania rozwojowe", przeczytać możemy: „Jednym z problemów polskiej edukacji wyższej jest niskie i malejące zainteresowanie naukami ścisłymi i studiami technicznymi, które są istotne z perspektywy gospodarki opartej na wiedzy. Na 1000 mieszkańców przypada u nas średnio 11,1 absolwenta kierunków przyrodniczych i technicznych. Dla porównania współczynnik ten dla krajów UE wynosi 12,9. Niepokojący jest przy tym wyraźny trend spadkowy: w 2007 r. zanotowano blisko 6-procentowy spadek zainteresowania wśród kandydatów na uczelnie studiami na kierunkach inżynieryjno-technicznych i informatycznych".[17] I choć – powtórzmy to raz jeszcze – nie utożsamiamy popularyzacji nauki programowania z edukowaniem informatyków, to oczywiste jest, że i w tym kontekście oddziaływać mógłby efekt skali: więcej osób programujących w ogóle to zapewne także większa liczba informatyków.

Odrębną kwestią są dysproporcje płciowe widoczne wśród absolwentów kierunków technicznych – choć liczba kobiet rośnie, wciąż stanowią one wyraźną mniejszość.[18] W tym kontekście zwiększenie nacisku na kształcenie umiejętności programowania na wczesnych etapach kształcenia mogłoby wyrównywać skutki tych dysproporcji. Należy niemniej podjąć starania, by nauka programowana nie została sprofilowana jako „zajęcie dla chłopców" i podkreślać znaczenie traktowania tych kompetencji jako powszechnych. ■

komentarz

Kamila Stępniowska *- COO Geek Girls Carrots – ogólnopolskiej społeczności promującej większą obecność kobiet w branży IT. Jest odpowiedzialna za nowe projekty, organizuje warszawskie spotkania GGC. Równocześnie pracuje nad doktoratem w Zakładzie Socjologii Kultury Instytutu Socjologii Uniwersytetu Warszawskiego. Współorganizatorka Offtopicarium.*

PROGRAMOWANIE DLA KAŻDEGO

Umiejętność programowania w niedalekiej przyszłości może być tak samo istotna jak umiejętność czytania, pisania, czy znajomość języka angielskiego. Od niej może zależeć otrzymanie ciekawej i dobrze płatnej pracy. Bardzo ważne jest, żeby nie zrażać młodzieży do nauki programowania, żeby umieć przekazać ile można stworzyć za pomocą pisanego kodu - nie tylko gry komputerowe, ale również aplikacje mobilne, czy programy sterujące robotami. W ramach nauczania szkolnego częściej powinny pojawiać się indywidualne i grupowe projekty pokazujące młodzieży owe możliwości.

Ważne jest również to, żeby podobnie jak w przypadku matematyki i innych przedmiotów stawiano takie same wymagania chłopakom i dziewczynom. Wydaje się, że w Polsce nadal pokutuje przekonanie, że nauki humanistyczne są „dla dziewczyn", a nauki ścisłe „dla chłopaków". Te stereotypy rzutują na wybór kierunków studiów, późniejsze możliwości zatrudnienia, ale też cały obraz branży IT. ■

***Michał Madej** - jako pracownik CD Projekt RED był głównym projektantem gry „The Witcher", pracował też przy najlepiej sprzedającej się polskiej grze w historii, „Dead Island" firmy Techland. Obecnie pracuje w szanghajskim oddziale jednego z największych światowych producentów gier, firmy Ubisoft, której roczne obroty przekraczają miliard dolarów. Znalazł się m.in. w zespole odpowiedzialnym za grę „Far Cry 3", nominowaną do tytułu najlepszej gry roku 2012.*

PERSPEKTYWA RYNKU GIER

Nikt nie wątpi, że nauka programowania przydać się może w przyszłości w naukach ścisłych i przyrodniczych. Co z humanistami? W najbliższej przyszłości dominować będą multimedialne formy komunikacji, do ich opanowania i zrozumienia znajomość języków programowania potrzebna jest w takim samym stopniu, jak znajomość alfabetu. Szczególnie tworzenie mediów wchodzących w interakcję z odbiorcą jest niemożliwe bez chociażby podstawowej wiedzy na temat programowania. Przykładem są gry komputerowe, dynamiczne rozwijająca się branża, która potrzebują nie tylko inżynierów, ale też artystów, pisarzy, a nawet krytyków.

Warto też dodać, że rynek gier, z którym jestem związany, to bardzo chłonna branża, w której w zasadzie nieustannie trwa nabór nowych pracowników. W wielu projektach mamy wakaty w zasadzie od ich rozpoczęcia do samego końca. Kłopot z pozyskaniem pracowników wynika także z tego, że poszukujemy ludzi, którzy łączą kompetencje humanistyczne i artystyczne - potrafią tworzyć i opowiadać historie, projektować interakcje, ale też np. rysować czy animować - z kompetencjami technicznymi, znajomością zasad funkcjonowania oprogramowania. Rozziew pomiędzy tymi sferami to z naszej perspektywy ogromny kłopot. ■

PROGRAMOWANIE W POLSKIM SYSTEMIE EDUKACJI

Systemy edukacyjne znacznie różnią się pomiędzy sobą. Najlepszym tego świadectwem jest fakt, że na poziomie Unii Europejskiej powołano specjalną instytucję o nazwie *Eurydice*[19] [20], której głównym zadaniem jest właśnie tłumaczenie zasad organizacji i funkcjonowania europejskich systemów edukacji. W tym celu *Eurydice* wydaje cyklicznie od roku 1995 publikacje z serii *Key Data* oraz inne raporty. Dla interesującej nas tu tematyki niezwykle ciekawy jest przetłumaczony w całości na język polski raport „Kluczowe dane o kształceniu i innowacjach z zastosowaniem technologii informacyjno-komunikacyjnych w szkołach w Europie. Wydanie 2011"[21]. Raport ten zwraca uwagę na podobieństwa w zakresie działań edukacyjnych pomiędzy państwami, wynikające z przynależności do Unii Europejskiej:

> W roku 2010 Komisja Europejska przyjęła nową Agendę Cyfrową dla Europy (Komisja Europejska, 2010b), która potwierdza i określa wyzwania na nadchodzące lata. Celem Agendy jest maksymalizacja społecznego i ekonomicznego potencjału technologii informacyjno-komunikacyjnych. Można to osiągnąć rozwijając wysoki poziom umiejętności związanych z technologiami informacyjno-komunikacyjnymi, w tym biegłości cyfrowej i medialnej.
>
> We wszystkich państwach europejskich przyjęto krajowe strategie mające zwiększać zastosowanie technologii informacyjno-komunikacyjnych w różnych dziedzinach, w tym strategie edukacyjne. Strategie te mają na celu wykształcenie u uczniów umiejętności koniecznych do stosowania technologii informacyjno-komunikacyjnych (zwłaszcza biegłości w tym zakresie), jak i kształcenie nauczycieli w tej dziedzinie.

Zgodnie z raportem, sytuacja ta dotyczyła Polski w sposób szczególny: była ona jednym z niewielu państw, które:

- w dziedzinie TIK (technologii informacyjnych i komunikacyjnych) w szkołach nie mają przyjętej strategii działania
- co za tym idzie nie finansują strategii TIK w edukacji oraz jej nie monitorują
- finansują TIK w edukacji głównie ze środków prywatnych

Od czasu publikacji tego raportu zaszły dwa ważne wydarzenia. Po pierwsze, przyjęto nową podstawę programową, obejmującą zajęcia komputerowe oraz nauczanie informatyki. Byliśmy też świadkami gorącej debaty społecznej dotyczącej edukacji medialnej, czy kompetencji cyfrowych – najpierw pod hasłem „komputer dla ucznia", a później szerszym „Cyfrowa Szkoła". Nowa podstawa programowa, w połączeniu z kompleksowym programem, jakim jest „Cyfrowa szkoła" – obejmującym nie tylko kwestie sprzętowe, ale też nauczanie kompetencji i dostarczanie odpowiednich zasobów – mają szansę wprowadzić realne zmiany dotyczące wykorzystania w szkołach TIK oraz nauczania informatyki. „Cyfrowa szkoła" jest jednak na razie projektem pilotażowym i nie jest pewne, do jakiego stopnia „Cyfrową szkołę" można uznać za realną, długoterminową strategię działania, a do jakiego za kolejny, doraźny program doposażania szkół w sprzęt.

Polska jest jednocześnie krajem, gdzie „sugeruje się" (s. 34) używanie TIK na każdym z monitorowanych 8 monitorowanych obszarów. Najciekawsze jest jednak to, co dokładnie kryje się pod hasłem „używanie TIK" w danym kraju [Wyk.7]

Z punktu widzenia tematyki tego raportu najistotniejszym punktem jest tutaj „rozwój umiejętności programowania". Jak widać pojawia się on w Polsce na poziomie szkół średnich, ale nie na poziomie szkół podstawowych. Do uwzględnienia programowania na poziomie szkoły podstawowej przyznaje się zaledwie 3 spośród krajów spośród uwzględnionych w analizie 31 państw, są to Grecja, Węgry oraz częściowo Wielka Brytania (dokładnie Anglia, Walia oraz Irlandia Północna, ale nie Szkocja).

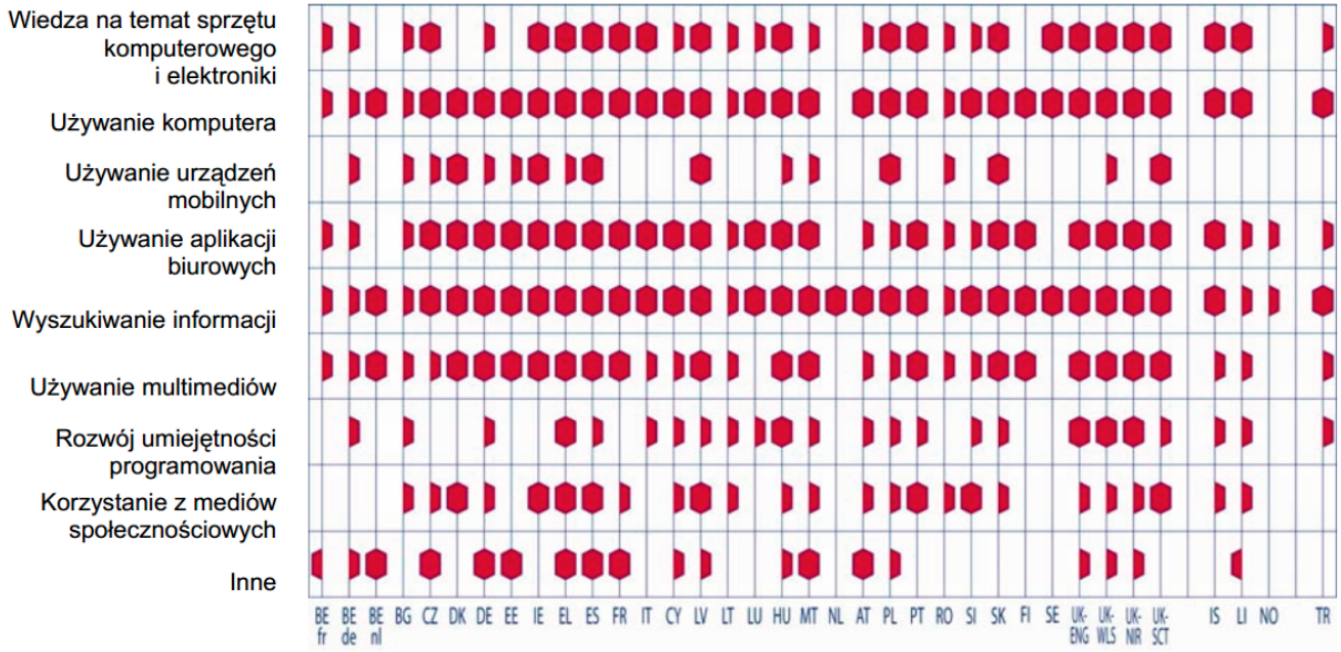
Przypadek Wielkiej Brytanii jest tutaj szczególnie interesujący, bowiem minister edukacji tego kraju otwierając w roku 2012 targi BETT (a więc już po publikacji raportu Eurydice) pozwolił sobie określić obowiązujący program TIK jako „bałagan" (ang. mess)[22] oraz postulował jego radykalne od-

Lewa
Szkoły podstawowe

Prawa
Szkoły średnie

Wykres 7

Obszary użycia TIK w programach szkolnych krajów europejskich.



Źródło: **Eurydice.**

nowienie tak, aby od września 2012 roku wprowadzić zmienioną podstawę programową, która pozwoliłaby przede wszystkim na decentralizację decyzji o tym czego i jak nauczać do poziomu szkół tak, aby „zamiast dzieci zanudzonych tym jak używać Worda i Excela móc obserować 11-latki, które w Scratchu produkują proste animacje 2D i 16-latki obeznane z zasadami logiki formalnej, która do tej pory była wykładana jako przedmiot uniwersytecki". Minister Gove dotrzymał tej obietnicy i obecnie w Wielkiej Brytanii nie ma obowiązkowej podstawy programowej dla przedmiotu TIK[23], a szkoły są uprawnione do napisania swojej własnej, wedle uznania. Obecnie z kolei trwają konsultacje społeczne dotyczące zmiany nazwy samego przedmiotu z TIK na *computing*. Zakończona zaledwie miesiąc temu (kwiecień 2013) pierwsza fala pokazuje jednak, że nie wszyscy interesariusze (głównie nauczyciele) są zgodni z Ministerstwem co do kierunku zmian[24]: co prawda większość spośród niemal 3000 odpowiedzi byłą pozytywna (39%), ale w tym samym czasie niemal tyle samo interesariuszy było przeciw (35%) lub nie miało zdania (26%).

W tym kontekście polska nowa podstawa programowa dla szkoły podstawowej (dla klas IV-VI, na przedmiocie „zajęcia komputerowe") wydaje się być dość nowoczesna, bowiem już obecnie przewiduje „rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera"[25], które dalej doprecyzowane jest następująco: „za pomocą ciągu poleceń tworzy proste motywy lub steruje obiektem na ekranie", co przywodzić może na myśl proste języki programowania, jak np. *Scratch,* czy *Logo*. ■

prof. Maciej M. Sysło *– matematyk i informatyk, autor przeszło 150 publikacji matematycznych, informatycznych i dydaktycznych. Redaktor naczelny pisma „Komputer w edukacji”, współtwórca krajowej olimpiady informatycznej. Jest współautorem podstawy programowej przedmiotów informatycznych.*

DIABEŁ TKWI W SZCZEGÓŁACH

Wielki celebryta wśród osób zajmujących się innowacjami w nauczaniu związanymi z nowymi technologiami, Marc Prensky, ostatnio zaczął głosić, że podstawową kompetencją XXI wielu będzie umiejętność programowania. To znakomicie wspiera nasze dążenia w rozwoju edukacji informatycznej, chociaż może rodzić wątpliwości, a nawet opór. W podstawie programowej znalazł się już przedmiot informatyka dla wszystkich uczniów w szkołach ponadgimnazjalnych, w którym pojawiają się elementy programowania. Ale to wcale nie znaczy, że chcemy kształcić samych informatyków. Stale pojawiają się też wątpliwości, na ile nauczyciele są przygotowani do prowadzenia takich zajęć.

Ale programowanie może dotyczyć narzędzi takich jak Logo czy Scratch, które służą do tworzenia środowisk uczenia się, gdzie uczący się może „programować” swoje kształcenie. Twórca Scratcha, Mitchel Resnick, swoją pracę doktorską pisał u Seymoura Paperta - człowieka, który pod koniec lat 70. XX wieku odwrócił relacje w myśleniu o technologii: uznał, że to dziecko ma programować komputer, panować nad nim, a nie odwrotnie. Scratch świetnie wpisuje się w filozofię Paperta.

Samo programowanie jest tłem. Dopiero w gimnazjum i liceum ogólnokształcącym pojawia się programowanie, które nie może obejść się bez algorytmiki. Co wcale nie znaczy, że przy odrobinie inwencji te proste narzędzia nie mogą służyć realizacji naprawdę ciekawych projektów. Zapisane w podstawie programowej „proste motywy” to na przykład grafika żółwia, Balti, czy Scratcha, a „sterowanie obiektem”, odnosić się może do elementów robotyki, sterowania urządzeniami podłączonymi do komputera. Oczywiście wciąż myślę, że to wstęp do działań bardziej zaawansowanych - ale do nich dopiero dojdziemy, i nie chodzi mi tylko o kolejne stopnie kształcenia, ale o przygotowanie kadry i ogólny stosunek społeczeństwa do roli programowania.

Niestety, pomimo spektakularnych osiągnięć naszych najlepszych młodych informatyków na arenach konkursów i olimpiad krajowych i międzynarodowych, średnie umiejętności uczniów z polskich szkół w zakresie programowania wypadają bardzo miernie - w skali europejskiej Polska zajmuje 20. miejsce (dane EuroStat, z 2012 roku). Przed nami więc ogrom pracy w zakresie powszechnego kształcenia umiejętności programowania.

Myślę, że ważne jest także szersze spojrzenie na programowanie, a raczej przygotowanie do niego np. w zakresie matematyki i logicznego myślenia. W tym zakresie są oferowane zajęcia typu „CS unplugged”, czyli informatyka bez komputerów, lub takie, jak w konkursie Bóbr. ■

Na poziomie gimnazjum sprawa jest już jednoznaczna, bowiem podstawa programowa dla przedmiotu „informatyka" mówi wprost, że jedną z trzonowych kompetencji, którą uczeń ma posiąść jest[26]:

Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Uczeń:

wyjaśnia pojęcie algorytmu, podaje odpowiednie przykłady algorytmów rozwiązywania różnych problemów;

formułuje ścisły opis prostej sytuacji problemowej, analizuje ją i przedstawia rozwiązanie w postaci algorytmicznej;

stosuje arkusz kalkulacyjny do rozwiązywania prostych problemów algorytmicznych;

opisuje sposób znajdowania wybranego elementu w zbiorze nieuporządkowanym i uporządkowanym, opisuje algorytm porządkowania zbioru elementów;

wykonuje wybrane algorytmy za pomocą komputera.

Dalej z kolei można przeczytać, że „Na III etapie edukacyjnym dopuszcza się wprowadzenie języka programowania, takiego jak *Logo* lub *Pascal*, które mają duże walory edukacyjne i mogą służyć kształceniu pojęć informatycznych.". Program ten i wymagania są rozwijane i rozszerzane w umiarkowanym stopniu dla szkół średnich na poziomie podstawowym i w znaczącym stopniu dla szkół średnich na poziomie zaawansowanym.

Raport *Eurydice* zwraca również uwagę na fakt, że Polska jest jednym z niewielu krajów, gdzie **zajęcia komputerowe / informatyka stanowią odrębny przedmiot** (w dodatku do zaleceń stosowania na wszelkich innych przedmiotach):

„Oprócz używania TIK jako narzędzia o charakterze ogólnym, TIK jest też odrębnym przedmiotem w szkołach podstawowych w ośmiu państwach/regionach: w Republice Czeskiej, na Łotwie, w Polsce, na Słowacji, w Zjednoczonym Królestwie (Anglia i Walia), na Islandii i w Turcji. Na tym poziomie TIK jest też częścią przedmiotów technicznych w Bułgarii, Francji, we Włoszech, na Cyprze, w Zjednoczonym Królestwie i na Islandii. Na poziomie szkół średnich TIK jest odrębnym przedmiotem i/lub jest częścią przedmiotów technicznych w niemal wszystkich systemach edukacji. Do wyjątków należą: Dania, Irlandia, Norwegia, Finlandia i Szwecja, gdzie TIK jest ogólnym narzędziem stosowanym w przypadku wszystkich przedmiotów".

Kolejnym dokumentem, który może rzucić więcej światła na sytuację pod względem wykorzystania TIK, w również nauki programowania w Polsce jest raport główny i raporty krajowe z badania *Survey of schools: ICT in Education*[27]. Projekt ten został przeprowadzony z inicjatywy Komisji Europejskiej, był nadzorowany przez *DG Connect* (*European Commission Directorate General for Communications Networks, Content and Technology*) jako narzędzie monitorujące cele stawiane sobie przez Unię Europejską w dokumentach *Digital Agenda for Europe* (DAE) oraz EU2020. Projekt zainicjowany został w roku 2011, a samo badanie odbywało się jesienią 2011 roku, badane były kraje EU27 + Chorwacja, Islandia, Norwegia oraz Turcja. Respondentami w badaniu byli dyrektorzy, nauczyciele oraz uczniowie. Pośród wielu wniosków płynących z badania najistotniejsze dla poruszanego w tym raporcie tematu są dane obrazujące ogólną sytuację Polski w dziedzinie TIK. Z raportu wynika, że charakterystyką Polski na tle zbadanych krajów jest **gorsza infrastruktura TIK.** Na przykład w klasie 2. gimnazjum na jeden komputer przypada 8 uczniów (10 na komputer z dostępem do internetu), podczas kiedy średnia dla badania to 5 uczniów – Polska w tym kryterium zajmuje 7 miejsce od końca (10 w kontekście komputera z dostępem do internetu), gorzej niż w Polsce jest jedynie w Chorwacji, Bułgarii, Włoszech, Rumunii, Grecji i Turcji. Nieco lepiej wypadają klasy czwarte szkoły podstawowej. Podobnie wyraźnie poniżej przeciętnej wygląda wskaźnik dostępu do tablic interaktywnych, projektorów, gorsza jest również przepustowość łącz internetowych. Ten niezbyt optymistyczny obraz rozjaśnia jedynie kilka pytań, w których Polska infrastruktura wypada jednak lepiej niż przeciętna dla całego sondażu, np. zdecydowana większość polskich gimnazjalistów (85%) uczy się w szkołach w których zdecydowana większość komputerów działa (ponad 90% komputerów) – jest to pewien zasób na którym można opierać projekt przyszłych programów interwencji.

Wobec gorszych wyników w kontekście infrastruktury TIK zaskakująco może brzmieć fakt, że **TIK używany jest w Polskich szkołach z częstotliwością zbliżoną do przeciętnej dla całego badania**, choć obraz ten jest zróżnicowany dla różnych wskaźników – generalnie nieco lepiej wypadają szkoły średnie, z kolei w interesujących nas tutaj szkołach podstawowych 44% uczniów przynależy do szkół gdzie TIK jest wykorzystywany na co najwięcej 5% lekcji, a w 24% jest to mniej niż skromny 1%. Bardzo intrygująco wyglądają przedstawiane przez autorów badania wyniki, które świadczą o tym, że uczniowie w polskich gimnazjach oraz szkołach ponadgimnazjalnych ponadprzeciętnie często **wykorzystują własny sprzęt** do celów naukowych na lekcjach, a w szczególności wyraźną różnicę obserwować można w wykorzystaniu uczniowskich **telefonów komórkowych** (pod tym względem zajmują 5 miejsce w Europie na 27 krajów).

Istotne dla projektowanych programów może być również to, że pomimo wspomnianej gorszej niż przeciętna infrastruktury polsc**y uczniowie oceniają swoje kompetencje cyfrowe wyżej niż przeciętna dla UE – w każdym aspekcie i na każdym poziomie edukacji** – kompetencjach w mediach społecznościowych, operacyjnych, odpowiedzialnego używania internetu oraz bezpiecznego używania internetu, podkreślić jednak należy, że **jest to samoocena, a nie zmierzony poziom kompetencji.**

Uzupełnieniem obrazu nauki programowania w polskich szkołach są inicjatywy konkursowe, z definicji przeznaczone dla lepszych uczniów. W Polsce ze względu na prestiż, zasięg i staż warto wymienić następujące konkursy dotyczące programowania:

1. Olimpiada informatyczna
2. Olimpiada informatyczna gimnazjalistów
3. Bóbr
4. Ogólnopolski konkurs informatyczny INTERSIEĆ

Omówione one zostaną kolejno:

OLIMPIADA INFORMATYCZNA

Olimpiada Informatyczna jest olimpiadą przedmiotową **adresowaną do uczniów szkół średnich**, przeprowadzaną corocznie pod egidą Ministerstwa Edukacji Narodowej[28]. Olimpiada funkcjonuje od roku szkolnego 1993/1994, a więc już 19 lat, choć korzenie Olimpiady sięgają nawet wcześniej – do końca lat osiemdziesiątych i I Międzynarodowej Olimpiady Informatyczna (IOI) w roku 1989, w Varnie, w Bułgarii. W ciągu 15 pierwszych lat funkcjonowania[29] olimpiada wyłoniła 457 finalistów, a zadania łącznie wypełniało 12 059 prac. Wyłonieni laureaci reprezentowali Polskę na ogólnoświatowej olimpiadzie informatycznej, zdobywając (w ciągu 15 pierwszych lat) na Międzynarodowej Olimpiadzie Informatycznej: 21 złotych, 22 srebrne, 17 brązowych medali, na Olimpiadzie Informatycznej Krajów Europy Środkowej: 16 złotych, 19 srebrnych, 17 brązowych oraz na I Bałtyckiej Olimpiadzie Informatycznej: 20 złotych, 22 srebrne, 12 brązowych.

Poza samą Olimpiadą działalność organizatorów obejmuje również prowadzenie portalu internetowego (m.in. archiwalne zadania, www.oi.edu.pl), własne wydawnictwa (tzw. niebieskie książeczki), coroczne obozy naukowo-treningowe im. Antoniego Kreczmara, warsztaty olimpijskie dla nauczycieli oraz portal edukacyjny www.main.edu.pl.

Zawody Olimpiady Informatycznej składają się z trzech etapów. W każdym etapie uczniowie rozwiązują od czterech do sześciu zadań algorytmiczno-programistycznych. Rozwiązaniem każdego zadania jest program komputerowy lub plik z wynikami. Etap I jest etapem domowym. Około 400 najlepszych uczniów z pierwszego etapu awansuje do etapu II, który jest rozgrywany w warunkach kontrolowanej samodzielności, w ośmiu różnych ośrodkach w kraju. Około 80 uczniów awansuje do finału Olimpiady.

OLIMPIADA INFORMATYCZNA GIMNAZJALISTÓW

Organizowana przez innych organizatorów (Stowarzyszenie Talent) Olimpiada informatyczna gimnazjalistów to ogólnopolska olimpiada przedmiotowa skierowana do uczniów szkół gimnazjalnych. Olimpiada organizowana jest w ramach projektu „Opracowanie i wdrożenie

kompleksowego systemu pracy z uczniem zdolnym" prowadzonego przez Ośrodek Rozwoju Edukacji. Pierwsza edycja miała miejsce w roku szkolnym 2006/2007, a więc obecnie organizowana jest jej 6. edycja. Jak piszą sami organizatorzy: „Jej głównym celem jest zainteresowanie uczniów informatyką poprzez rozwiązywanie ciekawych i inspirujących zadań i wyzwań informatycznych z zastosowaniem podejścia algorytmicznego i podejmowania decyzji z wykorzystaniem najnowszych technologii.". Również i ta olimpiada objęta jest patronatem MEN.

OGÓLNOPOLSKI KONKURS INFORMATYCZNY INTERSIEĆ

Kolejny z omawianych konkursów kierowany jest do **uczniów szkół ponadgimnazjalnych,** choć planowane jest rozszerzenie na „pozostałe grupy wiekowe". Rozegrano już VII edycji konkursu, a z edycji na edycję konkurs staje się coraz popularniejszy – w ostatniej edycji wzięło udział ponad 13000 uczestników.

BÓBR

Wyjątkowy w kontekście dwóch poprzednich konkursów jest konkurs Bóbr, bo **jest adresowany do uczniów we wszystkich typach szkół.** Bóbr to polska nazwa powołanego do życia w 2004 roku na Litwie międzynarodowego konkursu *Bebras* z zakresu informatyki oraz technologii informacyjnej i komunikacyjnej. Pod adresem znajduje się strona Konkursu *Bebras*[30]. Zasady konkursu są podobne do zasad bardzo popularnego w szkołach konkursu matematycznego Kangur. Organizatorem po stronie polskiej jest Wydział Matematyki i Informatyki Uniwersytetu Mikołaja Kopernika w Toruniu oraz firmy Vulcan, przy wsparciu Polskiego Towarzystwa Informatycznego, Stowarzyszenia Nauczycieli Technologii Informacyjnej oraz Uniwersyteckiego Centrum Nowoczesnych Technologii Nauczania. Głównym celem konkursu jest – tak jak w przypadku większości tego typu inicjatyw – rozwój i kształtowanie myślenia algorytmicznego oraz popularyzacja posługiwania się technologią informacyjną i komunikacyjną wśród wszystkich uczniów na wszystkich etapach edukacyjnych. ■

DOBRE PRAKTYKI

Najciekawsze działania dotyczące nauczania programowania dzieci i młodzieży są dzisiaj podejmowane nie w ramach standardowych zajęć lekcyjnych z informatyki czy technik komputerowych. Należy ich szukać w projektach często o charakterze pozaszkolnym, wykorzystujących model blended learningu albo nawet e-learningu. Część inicjatyw opiera się na tworzeniu środowisk edukacyjnych, także spotykających się w świecie rzeczywistym – inne zakładają przede wszystkim indywidualną naukę z pomocą serwisu WWW. Poniżej prezentujemy szczegółową prezentację jednego takiego projektu – CoderDojo, istotnego także ze względu na trwające, pilotażowe wdrożenie w Polsce. Dodatkowo przedstawiamy szereg najważniejszych inicjatyw tego rodzaju, realizowanych w Polsce i na świecie. Prezentujemy też najważniejsze języki programowania stosowane w tego rodzaju projektach.

CODERDOJO – studium przypadku nauczania dzieci i młodzieży programowania

Inicjatywa o nazwie CoderDojo została powołana do życia w Irlandii zaledwie dwa lata temu (w roku 2011), jednak zarówno samo japońskie Dojo, jak i jego styczne z programowaniem (np. CodingDojo) mają dłuższą tradycję.

Samo **dojo** to słowo pochodzące z języka japońskiego (jap. 道場 dōjō) i oznaczające miejsce, budynek, świątynię nauki[31]. Dojo zwykle przeznaczone były do nauki sztuk walki, te przeznaczone do medytacji nazywano sōdō (jap. 僧堂). W japońskich dojo, podobnie jak w późniejszych codingdojo / coderdojo obowiązywały **kodeksy** (jap. dōjō kun), które regulowały zachowanie osób w nim przebywających. Wiele ćwiczeń miało charakter **kata** (jap. 型), czyli sformalizowanych układów ataku i obrony, stosowanych szczególnie tam, gdzie wykonanie ćwiczenia w sparingu zagrażałoby życiu sparingpartnerów (np. w sztuce walki kendo, gdzie walczono na miecze). Inne ćwiczenia z kolei wykonywane są w zwarciu dwóch zawodników i noszą nazwę **kumite** (jap. 組手). Nad szkoleniem adeptów sztuk walki czuwa **Sensei** (jap. 先生)

IDEA DOJO I PROGRAMOWANIE

Trudno jednoznacznie stwierdzić kiedy, kto i gdzie rozciągnął i zaadaptował ideę dojo do nauki programowania. Niektóre źródła[32] wskazują na Dave'a Thomasa[33], który w 2007 roku opublikował na swoim blogu artykuł w którym podkreślał, że zawód programisty to jeden z niewielu takich, w których nie kładzie się dużego nacisku na ćwiczenia – po kursie teoretycznym (np. odpowiednich studiach) programiści od razu „rzucani są na głęboką wodę" wyzwań zawodowych – inaczej niż np. sportowcy czy artyści muzycy, dla których żmudne ćwiczenia są chlebem codziennym. W związku z tym – argumentuje Thomas – programiści popełniają błędy w pracy, bo dopiero tam uczą się praktyki. Żeby tą niewłaściwą sytuację naprawić Dave Thomas sugerował powołanie do życia instytucji Coding Kata, a więc zestawu wyzwań dla programistów, którzy w tej sposób zyskiwaliby poligon do ćwiczeń, podnoszenia swoich kompetencji w bezpiecznym, ćwiczeniowym środowisku. Thomas na swoim blogu prezentuje 20 takich kata. Idea przeniesienia japońskiej terminologii sztuk walki na grunt kodowania okazała się nośna i w przeciągu kilku lat powstały pomysły łączące ideę dojo z programowaniem, szczególnie programowaniem ekstremalnym, a więc takim typem programowania, w którym przy komputerze zasiada dwóch programistów (co zostało przeniesione do terminologii dojo jako kumite). Tak zrodziła się idea CodingDojo[34], a więc spotkań, na których grupa programistów spotyka się, aby trenować swoje

umiejętności podczas rozwiązywania wcześniej przygotowanych problemów programistycznych (coding kata). W centrum dojo siedzi dwóch programistów, którzy zajmują się problemem, z kolei skupieni wokół nich widzowie przyjmują jedynie czasem czynną rolę podpowiadając możliwe kierunki rozwoju kodu lub alternatywy. Kodeks dojo stanowi, że każdy widz z czasem powinien zasiąść do sparingu i przyjąć na siebie rolę aktywną. Nad całością czuwa z kolei mentor (Sensei), który naprowadza uczestników na rozwiązanie w momentach kryzysowych.

CodingDojo są zatem spotkaniami na których – co kluczowe – programiści rozwijają swoje kompetencje. Coding Dojo nie było pomyślane jako droga wejścia do bycia programistą, a jako droga samodoskonalenia dla już-programistów.

IRLANDZKIE CODERDOJO

Jest to organizacja non-profit powołana do życia przez Jamesa Wheltona oraz Billa Liao. Pierwszy z nich w momencie zakładania organizacji miał zaledwie 19 lat, jednak wydarzenia, które uczyniły go sławnym i doprowadziły do spotkania z Billem Liao miały miejsce jeszcze wcześniej – wszystko zaczęło się w grudniu 2010 roku, kiedy James Whelton wygrał iPoda nano szóstej generacji. W ciągu kilku zaledwie dni udało mu się – jako pierwszej osobie na świecie – obejść zabezpieczenia tego urządzenia tak, aby było w stanie wykonywać normalnie niedostępne funkcje – odtwarzać filmy, uruchamiać gry, czy choćby kasować aplikacje – swoimi osiągnięciami podzielił się przy pomocy strony internetowej http://nanohack.me co przyniosło mu niezwykłą popularność w Internecie (np. „Wired”[35]) i w ojczystej Irlandii. Na fali popularności James założył w swojej szkole klub komputerowy, w którym uczył innych przede wszystkim projektowania stron www. W roku 2011 doszło do spotkania Wheltona z Liao, który zauważył w tej inicjatywie duży potencjał. Założona przez nich wspólnie w czerwcu inicjatywa CoderDojo miała przedłużyć żywot założonego przez Wheltona klubu komputerowego i spopularyzować ideę na szerszą skalę. Tak istotnie się stało – w momencie pisania tego artykułu funkcjonują już 184 CoderDojo w 25 krajach, w tym w Polsce, choć większość z nich znajduje się w Irlandii (42%) USA (17%) oraz Wielkiej Brytanii (11%) – w pozostałych krajach obserwujemy po kilka lub wręcz po jednym CoderDojo (jak np. w Polsce)[36].

Organizacja CoderDojo założona przez Wheltona i Liao nie wiąże jednak poszczególnych Dojo ściśle. Należy ją widzieć raczej jako federację autonomicznych bytów złączonych jedną nazwą, celem i do pewnego stopnia metodologią pracy (choć trudno to zweryfikować empirycznie – nie ma jeszcze żadnych badań na ten temat). Sama strona coderdojo.com oferuje jedynie krótkie poradniki na temat tego jak założyć Dojo – przy czym ich skrótowa forma jest wyrazem nie tyle braku czasu czy kompetencji autorów, a raczej ich folizofii niewielu zasad i dużej autonomii jaką powinni otrzymywać uczestnicy Dojo. W rezultacie najkrórszy przepis na Dojo[37] można przetłumaczyć tak:

1. Zdobądź dostęp do bezpłatnej sali, z której będziesz mógł korzystać raz w tygodniu
2. Ustal datę rozpoczęcia działalności Dojo
3. Znajdź dwóch programistów-mentorów, którzy zainspirują dzieci
4. Bądź „cool”

Ostatnia z zasad „be cool” jest przez autorów podkreślana na każdym kroku. CoderDojo to miejsce, w którym dzieci mają się przede wszystkim dobrze czuć. Oczywiście na stronie znajdują się również nieco bardziej pogłębione zasoby, z których możemy się dowiedzieć nieco więcej na temat wizji Liao i Wheltona:

1. Coder Dojo to miejsce:

a. **dla dzieci i młodzieży** w wieku 7-17 lat
b. **darmowe -** nikt nie pobierał i nie powinien pobierać w przyszłości opłaty za udział w zajęciach
c. **inkluzywne** – mile widziani są chłopcy, dziewczynki, dzieci z autyzmem, zespołem aspergera, na wózkach itd.
d. **dobrowolne** – dzieci przychodzą, bo chcą w czymś takim uczestniczyć, bo komputery, programowanie i mentorzy są fajni
e. **wolne** (ang. free, but not free-ride) – udział jest bezpłatny, a uczestnicy mają dużo autonomii, ale jednocześnie udział w Dojo to przywilej, a nie prawo dzieci

2. Mentorzy w CoderDojo

a. **nie powinni dotykać klawiatury** – dzieci mają programować samodzielnie, inaczej niczego się nie nauczą. Zalecana jest metoda „na babcię”,

a więc dopingującego, ale biernego obserwatora, którą promuje m.in. prof. Sugata Mitra[38]

b. **nie muszą koniecznie programować** – po pewnym czasie dzieci same zaczynają uczyć się wzajemnie i rolę mentora można zastąpić rolą bohatera Dojo (ang. champion), który odpowiada głównie za logistykę spotkania, nie merytorykę, zewnętrzni mentorzy są potrzebni przez pierwszy okres
c. **mogą nimi być (powinni!)** starsi uczestnicy Dojo – dzieci powinny przyjmować na siebie rolę mentora najszybciej jak to możliwe

3. **CoderDojo** polega na swoim otoczeniu

a. **duża w nim rola rodziców**, którzy nie powinni traktować Dojo jako darmowej niani, a raczej wspierać je w roli bohatera Dojo
b. **dobrze jeśli Dojo ma wsparcie środowiska**, np. lokalnych firm z branży IT (które mogą wspomagać Dojo drobnym sprzętem, prawdziwymi wyzwaniami programistycznymi), restauracji (darmowe jedzenie), czy samorządów lokalnych (mogą zapewniać pomieszczenia, czempionów itd.)
c. **sprzęt uczestników –** zwykle uczestnicy CoderDojo proszeni są o przyniesienie własnych komputerów

Strona udzielająca wsparcia zakładającym nowe Dojo nieustannie się rozrasta o nowe materiały (w ostatnich tygodniach np. poradnik wydany w formie audiobooka, reklamowy klip wideo itp.), podejmowane są też inicjatywy wymiany doświadczeń pomiędzy Dojo (np. konferencja *#DojoCon 2013*).

Warto podkreślić, że w zamyśle autorów CoderDojo to instytucja, która nie wymaga wkładu finansowego (laptopy uczesników, darmowa sala, darmowa praca bohaterów i mentorów) i nie przynosi żadnych zysków – we wzorcowo funkcjonującym CoderDojo nie ma żadnych przepływów finansowych. W to miejsce CoderDojo używa energii społecznej – zaangażowania mentorów i bohaterów oraz szerszej społeczności lokalnej. Kapitałem CoderDojo jest tzw. kapitał społeczny lokalnej społeczności, a więc m.in. zaufanie i chęć do współpracy. Ze względu na taką charakterystykę inicjatywa raz wdrożona ma duże szanse na samopodtrzymywanie się.

Drugą ważną i godną podkreślenia charakterystyką CoderDojo jest jego samodefiniowanie się w opozycji do szkoły, szczególnie złej szkoły. CoderDojo jest – w przeciwieństwie do szkoły – dobrowolne, dzieci przychodzą do niego, bo lubią je i chcą się uczyć. CoderDojo unika dominującej w szkole metody wykładu, a w to miejsce kładzie nacisk na stosunkowo rzadko używaną w środowisku szkolnym metodę uczenia się rówieśniczego i pracy metodą warsztatową. W CoderDojo sprzęt uczniowski (komputery, tablety, telefony) jest tym, co pozwala Dojo funkcjonować, podczas kiedy w większości szkół korzysta się ze sprzętu szkolnego, a uczniowski jest czasem nawet zabroniony na terenie szkoły. Przykłady takie można oczywiście mnożyć, ale składają się one na obraz CoderDojo jako antytezy szkoły publicznej.

Inicjatywa CoderDojo jest niezwykle otwarta i chętnie korzysta z osiągnięć innych programów i inicjatyw o podobnym celu (część z nich jest omówiona dalej). Najszerzej w CoderDojo promowanym językiem programowania jest *Scratch* wymyślony przez naukowców z *Massachusetts Institute of Technology (MIT)*. Poza nim jako cenne projekty wymienia się np.:

- *Robocode* – środowisko w którym zaprogramować należy robota bitewnego, żeby pokonać roboty innych graczy (również przez nich zaprogramowane)
- *Codeacademy* – portal e-learningowy z interaktywnymi kursami programowania (m.in. JavaScript, Python, Ruby) oraz tworzenia stron internetowych (m.in. HTML, CSS, PHP)
- *KhanAcademy* – portal e-learningowy zawierający tysiące filmów z rozmaitych dziedzin (historii, geografii, matematyki itd.) w tym programowania – dział oprogramowania zawiera elementy interaktywne, które pozwalają testować na bieżąco zdobywaną wiedzę.
- W związku z federacyjną strukturą Dojo wymienione wyżej zasady są wdrażanie w różnych miejscach z różną gorliwością, a poszczególne inicjatywy różnią się między sobą dość znacznie.

CODERDOJO ZAMBRÓW

Szczególnie ciekawą z punktu widzenia tego raportu inicjatywą typu CoderDojo jest klub, który powstał w Zambrowie, w województwie podlaskim w marcu 2013 roku. Jest to pierwsze i jak do tej pory jedyne w Polsce CoderDojo. Zastanawiać

może jego peryferyjna lokalizacja – w innych krajach w których inicjatywa ta dopiero się rozwija i Dojo jest jedno lub kilka, obecne są one w największych miastach, bardzo często stolicach np. Kapsztadzie (RPA), Lublanie (Słowenia), Sankt Petersburgu (Rosja), Lizbonie (Portugalia), Atenach (Grecja) i tak dalej. Inicjatorem i jednocześnie mentorem zambrowskiego CoderDojo jest Kamil Sijko – pracownik warszawskiego Instytutu Badań Edukacyjnych, który pomysł na Dojo przywiózł z angielskich targów nowych technologii w edukacji BETT 2013 (luty 2013), na których Whelton i Liao mieli wystąpienie oraz prowadzili pokazowe CoderDojo. Drugim współzałożycielem i mentorem został Łukasz Dziedziul, programista w jednej z podlaskich firm produkujących chemikalia. Od początku istnienia do momentu pisania tych słów odbyło się osiem spotkań, na które przychodzi dość stabilna grupa 20-30 młodych ludzi w wieku od 8 do 21 lat.

Do tej pory uczestnicy koncentrowali się przede wszystkim na programowaniu w języku *Scratch 1.4*, programowaniem z wykorzystaniem aplikacji *MIT AppInventor* (programowanie w systemie Android), oraz projektowaniu w 3D z wykorzystaniem oprogramowania *Sketchup 8*. Poza tym na zajęciach w mniejszym stopniu wykorzystywano również programy graficzne (np. *GIMP*) i eksperymentowano z innymi środowiskami programistycznymi (np. *C Sharp*, *Unity 3D*). Rezultaty pracy uczestników oraz najświeższe informacje można śledzić na oficjalnej stronie internetowej www.coder-dojo.pl, kanale na portalu Facebook[39] oraz Twitter[40].

Zambrowskie Dojo jest w większości zgodne z obrazem idealnego Dojo opisanym wyżej, ale napotkało już pewne bariery w rozwoju, które każą modyfikować Irlandzkie propozycje. Różnice można podsumować następująco:

- problemy związane z niskim z niskim poziomem kapitału społecznego w konkretnej społeczności
- Idea CoderDojo opiera się na dużym wsparciu społeczności lokalnej – o ile nie było problemem zorganizowanie darmowej sali na spotkania (odbywają się w Centrum Kultury, w sali konferencyjnej, sala jest udostępniana całkowicie za darmo), to dużym wyzwaniem okazało się pozyskanie dodatkowych mentorów i bohaterów Dojo. Podejmowane próby zrekrutowania do Dojo nauczycieli w charakterze mentorów, czy rodziców w charakterze bohaterów kończyły się jak do tej pory niepowodzeniami – nauczyciele mniej lub bardziej wprost wskazują na niski poziom kompetencji cyfrowych, brak czasu, czy czasem – zupełnie szczerze – brak woli do udziału w akcjach społecznych. Rodzice z kolei unikają kontaktu z Dojo – pomimo zaproszeń nie pojawiają się na samych zajęciach, pomimo tego, że część z nich dowozi dzieci na spotkania samochodem
- problemy sprzętowe uczestników Dojo

W oryginalnym pomyśle uczestnicy Dojo proszeni są o przyniesienie swojego własnego komputera. Pierwsze spotkanie poprzedzone było kampanią informacyjną (m.in. na popularnych lokalnych portalach informacyjnych[41]) w rezultacie której pojawiło się dużo pytań o to, czy własny sprzęt jest konieczny do uczestnictwa w programie. Założyciele Dojo wyżej cenili zasadę inkluzywności (każdy może przyjść), niż zasadę „własnego sprzętu", w związku z czym złagodzili ten punkt. W rezultacie obecnie ok. połowa uczestników przychodzi z własnym laptopem, a pozostali próbują dołączyć się do innych osób, które akurat przyniosły swój komputer. Powodami dla których niektórzy nie przynoszą sprzętu jest często brak komputera mobilnego w domu lub brak możliwości wyniesienia go z domu (np. rodzice obawiający się o sprzęt, jedyny komputer w domu itd.). Bardzo trudno oszacować jak wielu potencjalnych uczestników nie przychodzi na Dojo właśnie ze względu na brak swojego komputera. Istnieje również ryzyko „wykruszania się" uczestników, którzy nie mają własnego sprzętu, bo – pomimo starań – nie są oni w stanie uzyskać pełnego doświadczenia CoderDojo.

W rezultacie założyciele planują złamać jedną z podstawowych zasad CoderDojo – zasadę braku przepływów finasowych – i założyć fundację CoderDojo Polska, której zadaniem byłoby kompensowanie braków kapitałowych (kapitału społecznego i kapitału finansowego) poprzez zbieranie środków od sponsorów. Środki te miałyby być przeznaczone na podstawowe (laptopy) oraz nieco bardziej specjalistyczne wyposażenie CoderDojo (np. programowalne klocki do budowania robotów, tablety graficzne do rysowania itd.), które umożliwiłoby realizowanie zasady inkluzywności. Drugim celem dla którego powoływana jest funda-

cja jest stworzenie podręcznika o roboczej nazwie CoderDojo Toolkit, który mógłby służyć pomocą innym osobom w Polsce, które chciałyby otworzyć CoderDojo w swojej okolicy.

INNE PROJEKTY

KHAN ACADEMY

Khan Academy to organizacja non-profit, założona w roku 2006 przez Salmana Khana. Misją organizacji jest zapewnienie światowej jakości edukacji dla każdego chętnego. Głównym środkiem do osiągnięcia tego celu jest biblioteka ponad 4100 lekcji sporządzonych w formie klipów wideo, dostępnych do oglądania m. in. w serwisie YouTube.

Khan Academy to jednak nie tylko klipy „korepetycyjne", to cała filozofia i metodologia uczenia się, którą naszkicował jej założyciel podczas wystąpienia na konferencji TED w roku 2011[42]. **Screencasty** są podstawą funkcjonowania systemu, na której wyrosły kolejne elementy. Screencast (videocast, lub po prostu klip wideo) to multimedialna, elektroniczna lekcja udzielana słuchaczowi z wykorzystaniem animacji, obrazu, dźwięku oraz narracji, ważnym elementem definiującym screencast jest nauka ucznia bez udziału nauczyciela – to znaczy nauczyciel po jednorazowym nagraniu screenastu nie uczestniczy już dalej w jego przyswajaniu. Według Khana podstawową zaletą udzielania lekcji przez screencasty jest właśnie ich indywidualny odbiór: pierwszym pozytywnym skutkiem takiej sytuacji jest **brak kontroli społecznej**, dzięki czemu np. w przypadku niezrozumienia fragmentu lekcji można ją powtórzyć nawet i sto razy, bez najmniejszych reperkusji ze strony nauczyciela, czy kolegów (powtórzone stukrotnie pytanie byłoby irytujące); działa to również doskonale w kontekście starszych uczniów (czy wręcz starych) którzy mogą odwołać się do takiej lekcji bez wstydu związanego z tym, że zapomniał coś oczywistego, co pamiętać „powinien". Po drugie powtarzanie skutkuje również **indywidualizacją tempa nauczania** – jednemu uczniowi wystarczy zaledwie jednokrotne przesłuchanie lekcji (a i to z delikatnym przyspieszeniem lektora), a inny potrzebuje kilkunastu powtórzeń – w związku z tym, że uczniowie słuchają lekcji indywidualnie ich potrzeby nie kolidują ze sobą, nie pozostają więc w konflikcie interesów.

Drugim elementem jest **drzewo wiedzy**. Khan i współpracownicy przygotowali drzewo jako mapę prowadzącą poprzez poszczególne lekcje. Uczeń ma dużą dowolność w wybraniu punktu startowego oraz kolejności poznawania poszczególnych obszarów. Co prawda dostęp do każdego filmu ma każdy, w każdym momencie, jednak obecne są sugestie na temat tego co można obejrzeć w dalszej kolejności, żeby nauka była łatwiejsza. Przypomina to proces układania ścieżki edukacyjnej przez studentów, jednak z tą różnicą, że kurs trwa 30 godzin akademickich, a lekcja w Khan Academy około dziesięciu minut. W związku z tym decyzje są zdecydowanie częstsze, a poczucie kontroli większe. Przyczynia się to do zintensyfikowania indywidualizacji nauczania o element indywidualizacji zakresu tematycznego nauki. Drzewo umożliwia „szybszym" uczniom **dalszą naukę, bez czekania** na uczniów „wolniejszych" – wiedzą oni co mogą robić w dalszej kolejności. Drzewo zwalnia również nauczyciela z obowiązku wytyczania zadań, przez co **uwalnia jego czas** na lekcji.

Absolutnie kluczowym elementem filozofii Khana jest coś co nazywa on **zwrotem klasy o 180 stopni**. Rewolucyjna według Khana zmiana polega w tym wypadku na odwróceniu kolejności zapoznawania się z materiałem oraz ćwiczeniami. Według niego w tradycyjnej szkole na lekcji nauczyciel zajmuje się głównie wykładaniem wiedzy. Ćwiczenia są również istotne, jednak wykonywane są przez uczniów głównie w ramach prac domowych, na własną rękę, z ćwiczeń jest się później rozliczanym. W przypadku Khana rolę przekaźnika wiedzy pełnią screencasty, które ogląda się w domu, w swoim tempie. Do szkoły uczniowie przychodzą przygotowani i cały czas z nauczycielem spędzają na doskonaleniu swojego rozumienia materiału poprzez ćwiczenia. Podstawowym zyskiem z takiego zabiegu jest to, że cały czas nauczyciela poświęcony jest na pracę indywidualną, a nie z grupą. Nauczyciel może dzięki temu udzielać **spersonalizowanego wsparcia** zamiast ogólnych wytycznych (w ramach wykładu audytoryjnego). Z perspektywy ucznia zmienia się punkt w którym dostaje on najlepsze jakościowo wsparcie – zamiast na początku (kiedy jest łatwiej) dostaje je na etapie ćwiczeń, kiedy to praktyka obnaża punkty których nie zrozumiał. Czas pracy nauczyciela jest zatem **efektywniej** wykorzystywany.

Pomocą w pracy podczas „odwróconej lekcji" jest **panel nauczyciela**, który służy informacją o tym gdzie są poszczególni uczniowie i z jakimi aktualnie

borykają się problemami. Celem panelu jest dostarczenie informacji pozwalających lepiej zorganizować lekcję – skierować wsparcie tam, gdzie jest ono najpotrzebniejsze, z użyciem zasobów, które są dostępne. Jeżeli ktoś nie radzi sobie z dzieleniem ułamków i utknął na etapie ćwiczeń, to nauczyciel może próbować udzielić mu pomocy. Albo poprosić o to kolegę z drugiego końca sali, który kompetencję tę opanował doskonale (co również widać z panelu).

Działanie to łączy się z **filozofią oczekiwania perfekcji**. Khan Academy chce oceniać swoich uczniów zerojedynkowo – albo ktoś opanował stawiane mu wymagania i potrafi to udowodnić rozwiązując poprawnie kilka zadań z rzędu, albo nie i wówczas jest zachęcany do dalszych ćwiczeń, aż do momentu, kiedy uda mu się udowodnić swoją kompetencję. Według Khana (i nie tylko – dyskusja o tym aspekcie oceniania trwa od dłuższego czasu) podejście takie jest o wiele lepsze, bowiem nie tworzy sytuacji nawarstwiających się zaległości – jeżeli ktoś nie do końca jeszcze opanował dodawanie i mnożenie, to prawdopodobnie trudniej będzie mu opanować odejmowanie i dzielenie itd. Khan w swoich wystąpieniach wskazywał na przykłady szkół pracujących jego metodą, w których dało się obserwować dzieci dłużej niż pozostałe pracujące nad opanowaniem partii materiału, ale w końcu dochodzące do poziomu swoich kolegów.

Ostatni element to popularna ostatnio **grywalizacja edukacji** poprzez stosowanie odznaczeń (*badges, achievements*). Odznaczenia te są przyznawane – w odróżnieniu od ocen szkolnych – głównie za wysiłek wkładany w naukę, np. czas poświęcony na studiowanie, liczbę prób, przejście po dłuższym impasie testu itd., a nie za efekty pracy. Korzyścią z tego działania jest **podnoszenie motywacji uczniów** do nauki – nie tylko tych, którym już idzie dobrze, ale w tym samym stopniu tych, którym idzie słabiej – premiujemy wysiłek, zatem staramy się o sytuację, w której każdy osiągnie tyle, na ile go stać.

Dział dotyczący nauk o komputerze i programowania jest w tym kontekście traktowany szczególnie, posiada oddzielną pod-stronę z unikalną funkcjonalnością[43]. Nacisk w jego wypadku położony jest na większą interaktywność – obecne są screencasty (głos lektora i tablica na której pojawia się odpowiedni kod), ale co ciekawe – można je w dowolnym momencie zatrzymać i zmodyfikować wyświetlany obecnie kod, a w odpowiednim oknie pojawią się wprowadzone przez nas zmiany, bez potrzeby kompilacji, czy nawet polecenia wykonania kodu / odświeżenia. Zintegrowano zatem moduł ćwiczeniowy z modułem screencastów, z kolei nieobecna jest tutaj strona serwisu umożliwiająca monitorowanie uczniów przez ich nauczyciela, bo nie ma jeszcze zupełnie modułu oceny pracy ucznia. Część serwisu poświęcona programowaniu jest obecnie **zdecydowanie bardziej nastawiona na własną aktywność**, niż wykorzystanie w typowej klasie. Uczniowie są zachęcani do interakcji z lekcją oraz „bawienia się kodem", remiksowania go (w terminologii programistycznej *forking*) zupełnie jak na profesjonalnych serwisach programistycznych np. GitHub. W chwili pisania tych słów wyniki remiksowania są imponujące – pod każdą lekcją istnieje już bardzo bogata galeria prac uczniów wykonanych na podstawie lekcji nauczycieli Akademii. Duży nacisk położony jest również na możliwość dyskutowania społeczności nad projektami nauczycieli i samych uczniów.

KhanAcademy zyskuje na świecie wielu zwolenników, m.in. samego Billa Gatesa, który przeznaczył na jego rozwój pokaźne sumy pieniędzy, a w Polsce Lecha Mankiewicza oraz współpracowników, dzięki którym już ponad 800 filmików zostało w całości przetłumaczonych na polski – część w formie napisów, a część w formie polskiej ścieżki dźwiękowej.

CODE ACADEMY

Założona przez dwóch dwudziestolatków w 2011 roku firma Codeacademy (codeacademy.com) jest – w przeciwieństwie do większości prezentowanych tu inicjatyw – przedsiębiorstwem komercyjnym, choć dopiero w fazie „Startup", a więc inwestycji, a nie przynoszenia zysków. Na swojej stronie Codeacademy przedstawia się jako firma z misją wynalezienia po raz drugi edukacji od podstaw. Podstawami tymi i inspiracją dla ich wizji są portale takie jak *Facebook* czy firmy jak *Zynga* (producenci m.in. gry *Farmville*). Dostępna pod adresem http://www.code.org/learn/codecademy platforma e-learningowa *Codeacademy* skoncentrowana jest – w przeciwieństwie do KhanAcademy – wyłącznie na programowaniu (m.in. *JavaScript, Python, Ruby*) oraz tworzenu stron internetowych (m.in. HTML, CSS, PHP). Podobnie jak KhanAcademy, również i tutaj dysponujemy konsolą w której możemy testować swój kod bezpośrednio na stronie internetowej – nie ma żadnego dodatkowego oprogramowania. W miejsce narratora z KhanAcademy tutaj pojawia się tylko małe

okienko z tekstem od nauczyciela, mniejszy jest również udział w lekcjach społeczności. Z drugiej strony sam projekt strony wygląda dużo poważniej i zdaje się jest kierowany do starszego, albo bardziej zaawansowanego użytkownika (KhanAcademy wydaje się być bardziej dostępna). Efektem pracy kursanta jest tekst, nie ma możliwości pracowania z grafiką – to kolejna róznica w stosunku do KhanAcademy, gdzie efektem jest przede wszystkim wektorowa grafika, animacje. Siłą Codeacademy jest z kolei liczba pokrywanych języków i – jak się zdaje – większa kompletność prezentowanego materiału.

CODE.ORG

Code.org to organizacja non-profit, której głównym celem jest promowanie programowania wśród młodzieży. Założona została roku przez przedsiębiorcę i inwestora z branży TIK – Hadiego Patrovi, który pracował przy tworzeniu usług iLike oraz Tellme oraz był inwestorem tak znanych startupów jak Facebook czy Dropbox. Organizacja ta zasłynęła ostatnio inspirującym klipem filmowym w którym udział wzięły takie gwiazdy sceny TIK jak Bill Gates czy Mark Zuckerberg zatytułowany *What most schools don't teach*[44]. Film zyskał dużą uwagę zarówno zagranicznych, jak i polskich odbiorców, pisano o nim nie tylko w serwisach internetowych (np. *Huffington Post*, a w Polsce wykop.pl, antyweb.pl), ale również w mediach bardziej tradycyjnych (*CNN International*, w Polsce gazeta.pl). Obecnie opublikowany w lutym 2013 roku klip ma już ponad 10 milionów odsłon.

BALTIE

Baltie to jednocześnie konkurs, język programowania oraz ekosystem wsparcia dla szkół (w tym m.in. podręczniki). Samo oprogramowanie jest zbliżone swoją koncepcją do Logo czy Scratcha – oferuje zatem naukę algorytmiki w wizualnej, uproszczonej formie oraz szybką informację zwrotną. Podstawową różnicą np. w stosunku do Scratcha jest to, że Baltie to program płatny (np. rocznie 500-1000zł w zależności od wielkości szkoły za licencję całościową – na wszystkie szkolne komputery, dla wszystkich nauczycieli oraz wszystkich uczniów). Program znajduje się na liście zalecanego oprogramowania do nauczania na poziomie szkoły podstawowej i gimnazjum (nr zalecenia: 1685/2004). Oprogramowanie posiada dodatkowy panel internetowy do śledzenia postępów uczniów, który może z jednej strony służyć nauczycielowi, a z drugiej strony służy do rozgrywania konkursów programistycznych dla uczniów. Autorem tej aplikacji oraz właścicielem firmy SGP Systems jest programista z Czech, Bohumir Soukup, co być może tłumaczy dużą popularność tego języka w Polsce. Ze statystyk strony internetowej producenta można dowiedzieć się, że obecnie w projekcie „Twórcza informatyka z Baltie" bierze w Polsce udział 2265 szkół, 3005 nauczycieli oraz – co najważniejsze - 22494 uczniów.

PRZEDSIĘWZIĘCIA KOMERCYJNE

Oferta dla polskich dzieci zainteresowanych programowaniem nie jest niestety przesadnie bogata. Jeśli mają one mało szczęścia, to w ich szkole nie ma nauczyciela, który mógłby pracować z nimi na poziomie wyższym niż proste animacje w Logo, nie ma w ich okolicy również CoderDojo (które w naszym kraju dopiero raczkuje). Olimpiady i konkursy przedmiotowe z programowania są z kolei najczęściej zbyt trudne dla początkujących. Dzieci mogą oczywiście próbować korzystać z zasobów anglojęzycznych (np. wymienione wcześniej Codeacademy, czy KhanAcademy), jednak żeby móc to zrobić muszą operować językiem angielskim w co najmniej średnim stopniu (co nie jest bardzo prawdopodobne w przypadku przeciętnego dziecka w wieku 7-13 lat). Część zasobów jest tłumaczona na język polski (patrz wyżej), ale rzadko dotyczy to materiałów z programowania, które zwykle mają charakter nieco bardziej skomplikowanych stron internetowych, a nie prostych *screencastów* (filmików).

Taki stan rzeczy prowadzi do sytuacji odpowiedniej do rozwoju komercyjnych inicjatyw uzupełniających ofertę szkoły oraz organizacji pozarządowych (NGO). Przykładami są mlodziprogramisci.pl oraz e-matplaneta.pl. Oba serwisy są wizytówkami szkół zorganizowanych na kształt szkół językowych – oferują zajęcia z programowania w trójmieście (mlodziprogramisci.pl) oraz Warszawie (matplaneta), w przypadku firmy trójmiejskiej dostępne są również zajęcia e-learningowe (wówczas bez ograniczeń terytorialnych). Zajęcia są oczywiście płatne – za semestralny kurs w matplanecie płaci się obecnie 980zł, a w mlodziprogramisci.pl 845zł. Obie szkoły proponują rozbudowany program w skład którego wchodzi programowanie nie tylko w prostych językach (np. Scratch), ale również podstawy języka C++, Java czy php. ■

dr Grzegorz D. Stunża *- adiunkt w Pracowni Edukacji Medialnej w Zakładzie Filozofii Wychowania i Studiów Kulturowych Instytutu Pedagogiki UG. Asystent koordynatora i badacz w projektach „Dzieci sieci - kompetencje komunikacyjne najmłodszych" i „Dzieci sieci 2.0 - kompetencje komunikacyjne młodych". Członek zespołu projektu „Cyfrowa przyszłość". Prowadzi edukatormedialny.pl, współtworzy Medialab Gdańsk budowany przy Instytucie Kultury Miejskiej.*

DZIECI SIECI

Komputery, które wkroczyły w nasze życia już dobrych kilkadziesiąt lat temu, radykalnie zmieniły i zmieniają nasze nawyki. Zmieniamy podejście do uczenia się, uczestnictwa w kulturze, kontaktowania się z innymi ludźmi. Komunikacja zapośredniczona medialnie lub wspomagana multi- i hipermediami nie zawsze musi być lepsza niż ta odbywająca się twarzą w twarz. Zachłyśnięcie technologiami, charakterystyczne np. dla lat dziewięćdziesiątych i objawiające się m.in. nastawieniem na wyposażanie szkolnych pracowni komputerowych, niekoniecznie mija, ale jest coraz częściej podszyte krytycznym spojrzeniem na zachodzące zjawiska. Mamy już za sobą, poza kilkoma dogasającymi ogniskami, demonizowanie mediów, do czego często przyczyniali się ci, którzy niewiele wspólnego mieli z korzystaniem z nowych tekstów kultury. I tak jak trudno powiedzieć dzisiaj jednoznacznie, że komputer to zło, które wyłącznie pożera czas, trudno mówić, że twarde, inżynieryjne umiejętności potrzebne przy budowaniu i zarządzaniu medialną infrastrukturą, powinni mieć tylko specjaliści, kończący określone studia.

„Wylewanie się" umiejętności medialnych i informacyjnych poza krąg specjalistów postępuje właściwie od początku rozwoju domowych komputerów. Wprowadzenie wizualnego interfejsu i symboliczne rozbijanie młotem przez kobietę ekranu Wielkiego Brata w kultowej reklamie Apple z 1984 można potraktować jako granicę pomiędzy starym światem powszechnej, jednostronnej najczęściej komunikacji, nad którą władzę dzierżyli biali mężczyźni o techniczno-inżynieryjnym wykształceniu, a światem hipermedialnego zapośredniczenia, uruchamiania indywidualnych ścieżek lektury. To właściwie granice między modernistycznym podejściem do informacji, a informacyjnym anarchizmem ponowoczesności. Można się oczywiście spierać, czy żyjemy już w czasie po-ponowoczesnym i czy korzystamy z nowych, a może nowych nowych mediów i czy podana interpretacja nie jest przesadzona. Opisy są oczywiście ważne, ważniejsza wydaje się jednak dokonywana regularnie diagnoza sytuacji i ocena, na ile potencjał najnowszych technologii i odmiennej od „starej" strategii komunikowania, jest wykorzystywany przez młodych ludzi.

W badaniu „Dzieci sieci - kompetencje komunikacyjne najmłodszych", eksplorując grunt pod

większe działanie naukowe, obserwowaliśmy uczniów trzech ostatnich klas szkół podstawowych, którzy niekoniecznie funkcjonowali jako cyfrowi tubylcy. Ich świadomość tego, jak funkcjonują media, była ograniczona (badanie miało charakter jakościowy, mówię zatem tylko o badanych), co można oczywiście tłumaczyć wiekiem (tu jednak mam wielkie wątpliwości, ucząc studentów pierwszego roku uniwersytetu i to kierunków pedagogicznych). Wydaje się jednak, że wchodząc w nastoletni wiek powinni nie tylko znać potencjał manipulacyjny mediów i go rozpoznawać, ale poważniej podchodzić do kwestii bezpieczeństwa. A ich zachowania w tym zakresie np. w serwisach społecznościowych, wydawało się być błądzeniem w ciemnościach. Zabawa na chybił-trafił z bezpieczeństwem, prezentowaniem wizerunku, udostępnianiem prywatnych informacji i danych osobowych może rodzić konkretne konsekwencje w ich dorosłym życiu, jeśli na kolejnym etapie edukacji formalnej lub w ramach ratunkowych działań organizacji pozarządowych, instytucji kultury czy innych organów nie uzupełnią braków.

Niedawno prowadziłem wspólnie z Anną Miler warsztat z uczniami ostatnich klas szkół podstawowych na temat bezpieczeństwa. Większość dzieci nie wiedziała, jak usunąć historię korzystania z przeglądarki, jak wyczyścić hasła, formularze, co to są ciasteczka. Poinformowanie, że warto chronić hasłem profil na komputerze było dla nich jak odkrywanie nowego lądu. Świetnie radzili sobie z zabawami w Paincie, nie mieli jednak prawie wcale pojęcia, jak chronić wizerunek, dane i bronić się przed cyberprzemocą. Wystarczyło kilka godzin warsztatu, by przez godzinę opowiadały, kierując słowa do kamery, dla swoich rówieśników, jak bezpiecznie korzystać z mediów. Negatywne przykłady niewiedzy i braku medialnych i informacyjnych umiejętności zarysowałem bardzo skrótowo. Na pewno dałoby się pokazać jednostki czy nawet grupy dzieci i młodzieży, którzy radzą sobie świetnie. Dyskusja na temat pochodzenia i kapitału kulturowego i również związanych z tym potrzeb byłaby w tym miejscu zapewne ciekawa. Ograniczę się jednak do podsumowania, że skoro wielu młodych ludzi ma problemy z kwestiami, jakie uznajemy za podstawowe - odsyłam m.in. do „Katalogu kompetencji medialnych i informacyjnych” Fundacji Nowoczesna Polska, który współtworzyłem - musimy jak najszybciej zacząć działać. Inaczej możemy zapomnieć o wyjściu poza podstawy edukacji np. w zakresie programowania z wykorzystaniem narzędzi, które leżą pod ręką i są bardzo interesujące dla nastolatków i młodszych dzieci. Cyfrowa rewolucja trwa i coraz częściej wchodzi w świat niezapośredniczony medialnie. Wystarczy spojrzeć na rozwój drukarek 3D, sieci fablabów i medialabów, które choć mogą pełnić rolę nieustających awangardowych instytucji, mogłyby, przy odpowiednim podejściu, działaniach i finansowaniu, stać się inicjatorem i katalizatorem zmian i upowszechnienia nowości, z korzyścią dla nas wszystkich. Tak jak trzydzieści lat temu wizualne interfejsy, a półtora dekady wcześniej prezentacja sieciowych możliwości przez Douglasa Engelbarta. ■

JĘZYKI PROGRAMOWANIA

Jedną z kluczowych kwestii, związanych z pewną filozofią podejścia do nauki programowania, jest wybór odpowiedniego języka programowania. Poniżej przedstawiamy wybrane – w naszej opinii wyróżniające się, nie tylko ze względu na swoje właściwości, ale też wsparcie społeczności i dostępność materiałów pomocniczych w języku polskim – propozycje, wraz z komentarzem praktyka.

SCRATCH

Język programownia *Scratch* powstał z inicjatywy *Massachusetts Institute of Technology (MIT),* a dokładniej wydziału *Lifelong Kindergarten Group MIT Media Lab*. Jest to język programowania zaprojektowany z myślą o dzieciach w wieku od 8 do 16 lat, ale korzystają z niego osoby z o wiele szerszego przedziału, „miliony użytkowników" (jak piszą na swojej stronie autorzy). Mocną strona *Scratcha* jest jego lokalizacja do 40 różnych języków, w tym pełna polonizacja. *Scratch* jest obecnie szeroko wykorzystywany w szkołach, również polskich. Platforma ta ma kilka cech szczególnych, które składają się na jej dużą popularność w edukacji:

Pierwsza grupa zalet wiąże się z kwestiami **logistycznymi**. Środowisko *Scratch* do niedawna wymagało instalacji na komputerze na którym miało odbywać się programowanie instalacji specjalnego programu (ostatnio w wersji 1.4). Obecna, niedawno zaprezentowana finalnie wersja 2.0 działa już zupełnie w modelu oprogramowania „w chmurze", to znaczy uruchamianego całkowicie wewnątrz przeglądarki internetowej – przynajmniej Chrome 7, Firefox 4, lub Internet Explorer 7 – wszystkie z zainstalowanym dodatkiem Adobe Flash Player w wersji przynajmniej 10.2. Taki model oznacza wiele korzyści dla mobilnego użytkownika – a takim przecież jest uczeń – wszystkie jego projekty przechowywane są na serwerach MIT, przez co uczeń może powrócić do swojego projektu z dowolnego miejsca na ziemi, aby podłączonego do internetu. Zmienne i środowisko pozwalające na uruchomienie aplikacji ucznia również przechowywane jest na centralnym serwerze, w związku z czym uruchomić aplikację (o ile tak zdecyduje uczeń) można również z dowolnego komputera podłączonego do internetu. Niskie wymagania sprzętowe (podłączenie do internetu + przeglądarka internetowa) sprawiają, że korzystanie ze *Scratcha* możliwe jest na zdecydowanej większości komputerów szkolnych, w tym na coraz popularniejszych w edukacji komputerach typu *Chromebook*, których system operacyjny pozwala na uruchomienie wyłącznie tego typu aplikacji („w chmurze"). *Scratch* jest zatem dostępny dla nauczycieli i uczniów – jest darmowy, nie trzeba go instalować, tłumaczyć i można go uruchomić wszędzie tam, gdzie są komputery i opcjonalny dostep do internetu[45] (zatem według danych GUS – ponad 90% szkół na poziomie podstawowym[46]).

Druga grupa zalet wiąże się z tym jak **szybko można uzyskać efekty** przy pomocy *Scratcha*. Zwykle – jest to doświadczenie, które było udziałem również autorów tego raportu – pracując z dziećmi w wieku od 8 lat wzwyż możliwe jest uzyskanie prostych programów natychmiast, już po pierwszych 10-15 minutach. Mówiąc językiem psychologii – program ten udziela bardzo szybko wzmocnień, co sprawia, że większa liczba uczniów ma ochotę i motywację do dalszej pracy – ciągle coś im się udaje. Dzieje się tak z pewnością częściowo ze względu na wizualizację fragmentów kodu, które mają postać klocków o różnych kształtach tak, że bardzo łatwo wizualnie stwierdzić z jakimi elementami mogą się łączyć, a z jakimi nie. Bardzo ważne – zważywszy na mło-

dy wiek grupy docelowej – jest to, że efekty są bardzo konkretne i wymierne – nie jest to napis „gratulajce, bardzo dobrze!", a wizualny efekt na ekranie – kotek (który jest postacią przewodnią *Scratcha*), który chodzi w rytm muzyki, obraca się zgodnie z instrukcjami, miauczy, reaguje na bodźce z zewnątrz (naciśnięcia klawiszy, myszkę komputerową, a nawet dźwięki otoczenia!) – starsza wersja 1.4 pozwalała w tym względzie na jeszcze więcej, np. programowanie rzeczywistych przedmiotów, np. klocków *LEGO We DO®* lub *Mindstorms NXT®*. Kolejne wersje *Scratcha* (tym razem 2.0) wnoszą do niego coraz więcej atrakcyjnych wizualnie, bardzo efektownych funkcji, jak np. wykorzystanie kamerki internetowej laptopa do sterowania napisanym przez siebie programem. Można szybko być efektywnym i efektownym.

Kolejną kluczową właściwością *Scratcha* jest **bardzo dobre jakościowo i szerokie wsparcie** zarówno ze strony samych autorów, jak i społeczności. Dla każdego kto chciałby uczyć programowania w *Scratchu* dostępna jest strona *ScratchEd*[47], na której znaleźć można bardzo wiele gotowych zasobów do wykrorzystania na zajęciach: hostorie wdrożeń (obecnie 208), gotowe zasoby edukacyjne (takie jak propozycje ćwiczeń, przykładowe programy, konspekty zajęć, quizy, podręczniki i poradniki itd.) do darmowego wykorzystania skategoryzowane ze względu na poziom edukacji (od przedszkola aż po uniwersytet i doszkalanie zawodowe). Jeśli to nie wystarczy, to na wszystkich chętnych czeka bardzo aktywne forum. Za pośrednictwem strony ScratchEd można również zdobyć informację o nadchodzących wydarzeniach, takich jak konferencje, seminaria i warsztaty (AFK, czyli *away from keyboard* oraz *online*). W kontekście polskim warto wspomnieć o ciekawej inicjatywie stworzenia bezpłatnego, polskiego podręcznika do *Scratcha*. Autorem jest matematyk z Bielska-Białej, Piotr Szlagor, a sam podręcznik jest do pobrania bezpłatnie (jako *ebook*)[48]. Na osobne zdanie zasługuje w tym kontekście również polityka MIT, która wymusza na każdym dodanym publicznie projekcie licencję *Creative Commons Share Alike* (CC BY-SA 3.0), dzięki czemu obecnie na stronach portalu można przeglądać mrowie (tysiące? setki tysięcy? niestety właściciele strony milczą na ten temat) aplikacji i gier stworzonych przez innych programistów *Scratch*. Bez problemu znajdziemy w tym zbiorze remiksy klasycznych gier, takich jak *Pong*, *Mario* czy *Snake*, ale również nowszych, jak np. bardzo popularne *Angry Birds*. Reprodukcje te prezentują bardzo różny poziom, ale kluczowe jest to, że jednym kliknięciem możemy „wywrócić je" na drugą stronę, czyli stronę kodu *Scratch* i sprawdzić jak działają, albo skopiować cały kod i rozpocząć własną pracę na tym fundamencie.

APPINVENTOR

Scratch – pomimo bardzo wielu zalet – posiada również pewne ograniczenia. Poważnym jest używanie technologii *Adobe Flash* jako podstawy zarówno do tworzenia programów (*Scratch* 2.0), jak i ich odtwarzania (2.0 i starsze wersje). Niektórzy producenci systemów operacyjnych (najbardziej znany przykład to oczywiście firma *Apple* i system *iOS* wykorzystywany w telefonach *iPhone*, odtwarzaczach *iPod* i tabletach *iPad*) nie umożliwiają działania *Flasha.* W związku z tym prace uczniów są niedostępne na tych – coraz bardziej popularnych – platformach.

AppInventor – kolejna aplikacja od pracowników *Lifelong Kindergarten Group MIT Media Lab* to odpowiednik *Scratcha*, który produkuje autentyczne aplikacje dla systemu *Android*. Sama zasada działania jest podobna jak w przypadku *Scratcha* – aplikacje buduje się poprzez przeciąganie bloków „legopodobnych" klocków kodu na ekran, choć oczywiście same klocki są nieco inne, dopasowane do platformy na którą się tworzy (np. są wśród nich takie, które korzystają z unikalnych czujników telefonów, których nie ma zwykle w komputerach – np. akcelerometru). Tak jak i w *Scratchu*, tak również i tutaj możemy od razu obserwować efekty swojej pracy na ekranie własnego telefonu lub emulatora telefonu. Na koniec możemy stworzyć kompletny plik wykonywalny (*.app*), który można wykonywać na telefonach z Androidem. Poza tymi różnicami projekty *Scratch* i *AppInventor* są podobne – kładą nacisk na podobne zasady, które wokół jednego i drugiego programu stworzyły już dużą społeczność użytkowników. ■

Piotr Szlagor *- nauczyciel matematyki i przedmiotów informatycznych w Zespole Szkół Technicznych i Handlowych im. Franciszka Kępki oraz w Dwujęzycznych Szkołach im. Władysława Kopalińskiego w Bielsku-Białej. Pasjonat nowych technologii i ich szkolnych zastosowań, autor kilku publikacji dotyczących nowoczesnego nauczania. Szczególne zainteresowania wiąże z urządzeniami mobilnymi i nauką programowania w szkole.*

SCRATCH W SZKOLE PODSTAWOWEJ

Podstawa Programowa mówi o tym, iż każdy absolwent liceum czy technikum powinien wykazywać się umiejętnością rozwiązywania problemów z użyciem komputera oraz stosowaniem algorytmicznego podejścia do zadań. Cele te powinny być realizowane już od początku nauki w pierwszej klasie szkoły podstawowej i to nie tylko na lekcjach technologii informacyjnej, informatyki, czy zajęciach komputerowych.

Obecnie w szkołach nauka programowania jest traktowana po macoszemu. W szkole podstawowej nauczyciele często uznają, że uczniowie nie poradzą sobie z tak trudnym tematem. Na kolejnych etapach edukacyjnych, o ile pojawiają się elementy algorytmiki i programowania (sam jestem osobą, która przez cały cykl edukacji szkolnej nie zrealizowała w szkole nawet jednej godziny z tego tematu), to ich realizacja odbywa się przy użyciu anachronicznych narzędzi (takich jak Logo, ELI, Pascal). Nie są one ani atrakcyjne, ani przydatne, co skutecznie zniechęca uczniów do programowania. Postrzegają oni je jako trudną i zbędną umiejętność. Często jedynym powodem użycia nadmienionych środowisk jest fakt, iż są one znane przez nauczycieli prowadzących zajęcia lekcyjne.

Jak więc znaleźć środowisko, które byłoby dla uczniów atrakcyjne i przydatne, a z drugiej strony pozwalało na skuteczną naukę programowania? Musiałoby ono być przyjazne w obsłudze, darmowe, wieloplatformowe, a przy tym nowoczesne. Ideałem byłoby, gdyby zostało zaprojektowane z myślą o edukacji. Okazuje się, że takie środowisko już istnieje.

Scratch, bo o nim mowa, został stworzony przez specjalistów z MIT z misją, by skutecznie wspomagać naukę programowania. Aplikacje są dosłownie składane z dostępnych bloków funkcjonalnych - nie jest więc wymagana od uczniów znajomość komend, lecz samo rozumienie sensu treści na nich zapisanych. Bloczki są pogrupowane tematycznie, by ułatwić nawigację po programie. Dodatkowo Scratch pozwala na tworzenie własnych bloczków, gdy jest użytkownikowi to potrzebne. Od wersji 2.0, opublikowanej w maju 2013 roku, Scratch jest wyłącznie aplikacją internetową. Wszystkie programy tworzymy w oknie przeglądarki. Pozwala to na dostęp do swoich projektów z każdego miejsca na Ziemi. Dodatkowo swoje programy możemy opublikować na stronie Scratcha lub na własnym blogu tak łatwo, jak robi się to z filmami z serwisu YouTube.

Jako osoba aktywnie wykorzystująca Scratcha w nauczaniu informatyki i matematyki, mogę mówić o nim wyłącznie w samych superlatywach. Nie ma obecnie lepszego narzędzia uczącego myślenia algorytmicznego z takim potencjałem dydaktycznym. Scratcha wprowadzam już w pierwszej klasie szkoły podstawowej (na Zajęciach Komputerowych). Obsługa programu nie sprawia pierwszoklasistom żadnego programu, a mają przecież 5-7 lat. Ich pierwsze projekty to najczęściej proste animacje, w których dwie postacie (przygotowane przez nich w wbudowanym edytorze graficznym) prowadzą ze sobą krótki dialog. Następnie wprowadzane są elementy ruchu postaci, dźwięki itd. Dzieci chętnie wykonują stawiane przed nimi zadania, często wprowadzając do swoich aplikacji własne pomysły - to kolejna, ogromna zaleta Scratcha - nie ogranicza on w żaden sposób kreatywności uczniów.

Uczniowie, chodzący do klas starszych (gimnazjum i liceum) dostają zadania o większym poziomie trudności. Na tym poziomie Scratch doskonale sprawdza się jako narzędzie uczące algorytmiki. Można w bardzo prosty sposób pokazać, jak ważna jest dobrze przemyślana struktura programu. Jako przykład mogę podać aplikację, w której dwie postacie wymieniają dzielniki podanej przez użytkownika liczby. Każda z postaci ma zaimplementowany inny algorytm dla tej czynności. Uczniowie, widząc różnice w czasach wykonania algorytmów, mogą samodzielnie wyciągnąć odpowiednie wnioski dotyczące efektywnego programowania.

Scratcha wykorzystuję również na swoich lekcjach matematyki. Realizując tematy staram się co jakiś czas, w myśl zasady „jeśli chcesz coś zrozumieć - zaprogramuj to”, zlecać uczniom jako zadanie dodatkowe, zaprogramowanie aplikacji wykonującej określone zadania matematyczne np. stworzenie programu obliczającego pole i obwód czworokąta (dla uczniów IV klasy SP). Zadania tego typu są chętnie podejmowane przez moich uczniów. Mnie z kolei pozwalają na lepsze ukształtowanie pojęć matematycznych u podopiecznych.

Uważam, że każdy uczeń powinien mieć styczność z programowaniem już od wczesnych lat swojej nauki i to nie tylko na zajęciach informatyki (ale też matematyki, fizyki, biologii, czy języka polskiego). Idealnym narzędziem do tego jest Scratch, który możliwościami bije na głowę każde inne środowisko używane w szkołach. ■

Agnieszka Bilska *- anglistka w ZSO nr 10 w Gliwicach. Promuje szkołę w modelu PSK - „Przynieś Sobie Kompa” z otwartym dla uczniów dostępem do WiFi, edukację z wykorzystaniem gier i gamifikacji. Prywatnie mama 8-letniego Piotrusia.*

PROGRAMOWANIE OD KOŁYSKI

W ciągu ostatnich kilkunastu lat w każdej sferze naszego życia przybyło nowoczesnych zdobyczy technologicznych. Komputery stanowią kluczowy komponent sprzętów gospodarstwa domowego, samochodów, telefonów. Są obecne na każdym kroku, w urzędach, bankach, warsztatach, szkołach. Wiele zawodów wykonywanych przez dzisiejszych seniorów odchodzi do lamusa, a ich miejsce zajmują nowe, dostosowane do skomputeryzowanej rzeczywistości. Naprawa samochodu wymaga wysokiej klasy oprogramowania i kosztownego sprzętu, nie jest już możliwa dla każdego majsterkowicza w przydomowym garażu. Dodajmy do tego internet. Dzięki smartfonom możemy być w sieci połączeni ze światem i ze sobą nawzajem przez 24 godziny na dobę. Powszechny dostęp do internetu dla „cyfrowych tubylców” jest czymś zupełnie naturalnym, tylko my go jeszcze postrzegamy, jako dodatkową warstwę nałożoną na rzeczywistość.

Na naszych oczach dokonuje się przełom cywilizacyjny, o konsekwencjach porównywalnych jedynie do wynalezienia druku przez Gutenberga. W średniowieczu umiejętność czytania i pisania była zastrzeżona dla nielicznych - książąt, biskupów. Budziła szacunek, gwarantowała społeczny awans, lecz także dawała władzę, tym, którzy ją posiedli. Dzisiaj taką grupę „litterati” stanowią programiści. Zdobyli tajemną wiedzę i umiejętności, które przeciętnemu człowiekowi wydają się niedostępne. Potrafią z niczego tworzyć skomplikowane systemy, posiadają władzę nad maszynami, hackują rzeczywistość.

W internecie jak grzyby po deszczu pojawiają się anglojęzyczne publikacje, aplikacje, instruktaże, gry, a nawet całe szkoły programowania. Mają trafiać do jak największej rzeszy odbiorców. Świadczy o tym fakt, że ogromna ilość tych narzędzi jest skierowana do najmłodszych użytkowników komputerów.

Dzieci od urodzenia aktywnie korzystają z techologii. Od najmłodszych lat z tabletem lub smartfonem w rękach uczą się nimi manipulować. Nie są tylko odbiorcami. Całkiem naturalnie próbują zmieniać, dostosowywać do swoich potrzeb urządzenia, w tym - programy. Największą popularność zdobywają gry i aplikacje, w których młody użytkownik ma możliwość modyfikacji, jednocześnie staje się twórcą, czuje się ich właścicielem. Gry takie jak „Minecraft”, wymagają opanowania bardzo złożonych algorytmów działań. Stawiają gracza przed złożonymi problemami, wymagają od niego wnikliwej analizy sytuacji, stawiania hipotez, testowania opracowanych przez siebie rozwiązań, wprowadzania poprawek i ponownych testów. Tylko w ten sposób, na drodze iteracji, można robić postępy w grze. Nie trzeba znać programistycznej składni, aby uczyć się myśleć jak programista.

Przybywa aplikacji, które uczą programowania graficznego. Już kilkuletnie dzieci z wielką frajdą przestawiają kolorowe kocki, tak, żeby w końcu program lub zabawkowy robot zachował się tak jak sobie tego życzą. Z wiekiem, gdy ich potrzeby staną się większe, przyjdzie czas na poznanie i opanowanie “poważnych” języków programistycznych. W konsekwencji coraz częściej słyszy się o nastoletnich programistach wynalazcach, racjonalizatorach, którzy dla zabawy, z własnej potrzeby, czy dla lansu tworzą aplikcje warte miliony dolarów. Dziś młody wiek przestaje być barierą, staje się atutem, bo gwarantuje innowacyjność, niekonwencjonalność pomysłów.

Jeśli takie rzeczy dzieją się już dziś, gdy moda na programowanie dopiero się rodzi, co będzie za dziesięć lat? A co by było, gdyby ten trend wdrożyć jak najwcześniej w edukacji formalnej, tak jak to się dzieje w Estonii?

Nasuwa się oczywiste pytanie czemu przy galopującym postępie technologicznym, w publicznych szkołach na zajęciach komputerowych dzieci w podstawówce uczą się o komputerach w teorii z papierowych książek? Dlaczego na jednej lekcji informatyki tygodniowo gimnazjaliści poznają jedyne obsługę jednego słusznego edytora tekstu?

Żywię głęboką nadzieję, że pewnego dnia, w niezbyt odległej przyszłości, na planie lekcji mojego synka, obok polskiego i matematyki, znajdzie się programowanie. ■

PODSUMOWANIE: PORA NA UPGRADE?

Generowanie impulsów dla wzrostu zainteresowania programowaniem i realną popularyzację umiejętności nie jest zadaniem łatwym. Kluczowe wydaje się znalezienie odpowiedniego balansu pomiędzy masowym nauczaniem podstaw a rozbudzaniem pasji u osób, które z programowaniem mogą związać swoją karierę zawodową. Podobnych trudności nastręcza pytanie o to, jaką rolę w podnoszeniu umiejętności programistycznych Polaków powinna odgrywać szkoła – zwłaszcza, że jak pokazaliśmy, sposobów podnoszenia swoich kompetencji poza systemem szkolnym nie brakuje.

Na pytanie o proporcje pomiędzy „podstawami" a zaawansowanymi umiejętnościami programistycznymi nie ma jednej prostej odpowiedzi. Jest ona wypadkową odgórnych założeń (w wypadku szkoły – podstaw programowych) i lokalnych warunków, w szczególności nauczycieli, którzy mają te założenia realizować. Wydaje się, że wsparcie dla nich jest kwestią niezwykle ważną, co prowadzi nas w kierunku kwestii drugiej. Jesteśmy przekonani, że szkoła nie może być jedynym miejscem edukacji informatycznej, ale równocześnie powinna być podstawowym miejscem jej prowadzenia. Bez tego nie sposób mówić o masowym propagowaniu umiejętności programowania. Fałszywe są tezy o tym, że zainteresowani uczniowie sami sobie poradzą, korzystając z zasobów dostępnych w Sieci. To nie do końca tak: nie wszyscy po nie sięgną, ale też sami młodzi ludzie przyznają, że internet w nauce najczęściej bywa zaledwie namiastką kontaktu z nauczycielem i że taki bezpośredni kontakt nie daje się łatwo zastąpić niczym innym.[49] Dlatego przedstawione tu inicjatywy postrzegamy jako komplementarne wobec nauczania w szkole. Zapewne mogą one dostarczyć też wielu inspiracji, nie tylko reformatorom polskiej szkoły, ale też po prostu samym nauczycielom. Wierzymy też, że dokonująca się już stopniowo, ale w naszym odczuciu wciąż nie do końca satysfakcjonująca zmiana statusu edukacji informatycznej w polskiej szkole, może stanowić ważny element dyskusji o edukacji w ogóle. O tym, jak powinna się zmieniać i w jaki sposób powinna reagować na wyzwania współczesności.

Podsumowując niniejszy raport, prezentujemy charakterystykę inicjatyw, które odnoszą sukcesy. Oto ona:

1. Korzystają z tego co już jest - z istniejącej infrastruktury, czasem jedynie zrekonfigurowanej. Zarówno najgłośniejsza inicjatywa spośród wymienionych – KhanAcademy, jak i pozostałe nie silą się na dostarczanie zasobów dzieciom. Po części dlatego, że te zasoby dzieci (i szkoły) już posiadają, a po części dlatego, że do przeprowadzenia takiej inwestycji trzeba by niesamowitych nakładów finansowych przy być może nieproporcjonalnym efekcie. Zatem walka z istniejącą infrastrukturą oznacza albo olbrzymie obciążenie finansowe, albo mocne zawężenie grupy odbiorców. W to miejsce wszystkie inicjatywy kreatywnie wykorzystują istniejący już sprzęt i istniejących w systemie aktorów: np. KhanAcademy „jedynie" odwraca etap samodzielnych ćwiczeń i wykładu nauczycielskiego na ćwiczenia z nauczycielem i samodzielne (wspomagane akademią) wykłady. Scratch wykorzystuje najprostszy system dostarczenia oprogramowania do ucznia i nauczyciela – przeglądarkę internetową – nie wymaga nawet tak małych zmian infrastruktury jak instalacja (darmowego przecież!) oprogramowania.

2. Dostarczają szybkiego, niemal natychmiastowego feedbacku, często bardzo konkretnego (rysunkowego, wizualnego). Wszystkie ze wspomnianych platform e-learningowych i programistycznych łączy to, że dostarczają bardzo szybkich, niemal natychmiastowych efektów. Pierwsza lekcja na Codeacademy polega tylko na wpisaiu w cudzysłowach swojego imienia, które następnie komputer wyświetli – pierwszy sukces po 10 sekundach. Przesunięcie i zmuszenie do „tańczenia" kota ze Scratcha to zaledwie kilkanaście minut (a już można chwalić się rodzicom i kolegom!). Jest to niezwykle ważne, bo radykalnie zmniejsza chęć potencjalnej rezygnacji z kursu. Dobrze zaprojektowana inicjatywa dozuje uczesnikom poczucie sukcesu zupełnie jak dobra gra komputerowa – poziom pierwszy jest niezwykle łatwy, ale potem jest trudniej, jednak o tyle tylko trudniej, żeby graczowi się udało i miał z tego satysfakcję. Najlepsze inicjatywy – podobnie jak najlepsze gry – można reklamować hasłem „easy to pick up and play, hard to master".

Dodatkowo należy podkreślić konkretny charakter feedbacku – nie ogólna informacja zwrotna (jesteś super), a ruszający się kot, migające światełka, zaprogramowany robot. To kluczowe jeśli weźmie się pod uwagę fakt, że dzieci do pewnego wieku myślą bardzo konkretnie i taka informacja zwrotna łatwiej do nich trafia i jest przyswajana jako sukces!

3. Bazują na motywacji wewnętrznej i wzmagają ją przy użyciu technik takich jak gamifikacja. Kolejna styczna wszystkich programów - wynikająca poniekąd z poprzedniej – to bazowanie na motywacji wewnętrznej uczestników. Programy te są poniekąd w luksusowej sytuacji, bo uczestnicy nie mają obowiązku szkolnego wobec nich, przychodzą dobrowolnie, a sami autorzy pomimo tego, że pewnie chcieliby pomóc jak największej liczbie osób uważać będą za sukces jeśli pomogą komukolwiek, w związku z tym mogą sobie pozwolić (jak CoderDojo w oficjalnym podręczniku) na hasła w rodzaju „Don't stress – Some young people will come back and some won't. Some will drop in occasionally and some will complain. Don't worry its not for everyone just do the best you can and stay cool.". Sprawia to, że uczestnicy są otoczeni przez podobnych sobie zapaleńców, ma to ogromny wpływ na kształt i zachowanie społeczności (która ma dużą motywację m.in. do samopomocy) i sprzyja osiąganiu – znowu motywujących – sukcesów. Można powiedzieć, że organizacje te poprzez odrzucenie idei pomocy wszystkim „jak leci" ustawiły swoje programy w „błędnym kole wysokiej motywacji i sukcesów".

Jednak to nie wszystko. Najlepsze programy – jak np. KhanAcademy – dodatkowo domotywowują uczestników poprzez mechanizmy tzw. „gamifikacji" – oferują odznaki i punkty za najdrobniejsze sukcesy, a ważne też jak te sukcesy definiują – czasem jako zaprezentowanie wysokich kompetencji (np. zaliczenie testu), ale dużo częściej sukcesem jest w ogóle praca, wysiłek.

4. Włączanie ducha, kultury i doświadczeń ruchu *open-source* i *creative commons.* Większość z prezentowanych tu programów i inicjatyw to przedsięwzięcia not-for-profit. Ich autorzy pozyskują środki na życie z innych dziedzin swojej działalności, a programy mają inny cel, czysto społeczny. W związku z tym prawdopodobnie tak dobrze i wydawałoby się bezwysiłkowo integrują osiągnięcia, ducha i najlepsze doświadczenia ruchów open-source i creative commons. Prawie w każdym przypadku sam program – czy to kod źródłowy jak w wypadku *Scratcha*, czy to kod serwisu internetowego jak w przypadku *KhanAcademy*, czy w końcu know-how jak w wypadku *CoderDojo* jest otwarty dla wszystkich i dostępny do modyfikacji – tak jak kod źródłowy wolnego oprogramowania. Żaden z tych programów nie ściga się z innymi, żaden nie boi się skopiowania, a wprost przeciwnie – zachęca do niego. Kluczowe jest również zachęcanie do tego samego użytkowników – każdy publikujący aplikację w Scratchu musi pogodzić się z tym, że jego praca będzie mogła być „podejrzana" od środka, skopiowana i rozwinięta bez zgody i wiedzy autora. To automatycznie sprawia, że każdy uczący się ma dostęp do olbrzymiej bazy przykładów, sprawia to, że społeczność żyje. ■

przypisy

1. Dominik Batorski, *Korzystanie z technologii informacyjno-komunikacyjnych*, w: *Diagnoza społeczna 2011. Warunki i jakość życia Polaków*, red. J. Czapiński, T. Panek, Warszawa 2012.
2. Por. Manuel Castells, *Społeczeństwo sieci*, Warszawa 2007; Lee Rainie, Barry Wellman, *Networked: The Social Operating System*, Cambridge, MA 2012.
3. William Gibson w filmie dokumentalnym *No Maps For These Territories* (2000, reż. Mark Neal).
4. Por. raporty *Dzieci Sieci: kompetencje komunikacyjne najmłodszych*, red. P. Siuda i inni < www.dzieci-sieci.pl/raport_IKM_dzieci_sieci.pdf>; M. Filiciak, M. Danielewicz, M. Halawa, P .Mazurek, A. Nowotny, *Młodzi i media* <www.wyborcza.pl/mlodziimedia>.
5. http://www.google.pl/trends/explore#q=Programming%2C%20Knitting%2C%20Swimming&cmpt=q
6. http://www.google.pl/trends/explore#q=programowanie%2C%20szyde%C5%82kowanie%2C%20p%C5%82ywanie&cmpt=q
7. Charles P. Snow, *Dwie kultury*, Warszawa 1999.
8. Lev Manovich, *Język nowych mediów*, przeł. P. Cypryański, Warszawa 2006, s. 117; Lev Manovich, *Software Takes Command,* <http://www.manovich.net/DOCS/Manoich.Cultural_Software.2011.pdf>.
9. Matthew Griffin, Susanne Herrmann, *Technologies of Writing: Interview with Friedrich A. Kittler*, „New Literary History" 27.4 (1996), s. 740.
10. Douglas Rushkoff, *Program or Be Programmed*, New York 2010, s. 128.
11. Piotr Celiński, *Medialab: silnik nowego alfabetyzmu*, w: *Medialab. Instrukcja obsługi*, red. M. Filiciak, A. Jałosińska, A. Tarkowski, Chrzelice 2011, s. 58-9 < http://labkit.pl/wp-content/uploads/2011/12/Medialab.-Instrukcja-obs%C5%82ugi.pdf>.
12. Informacje nt. krytycznej oceny programów sprzętowych: Andrew Zucker, Daniel Light, *Laptop programs for students*, „Science", 2 January 2009: Vol. 323 no. 5910, s. *82-85*.
13. Anna Giza-Poleszczuk, Renata Włoch, *Innowacje a społeczeństwo*, w: *Świt innowacyjnego społeczeństwa. Trendy na najbliższe lata*, Kraków 2013, s. 65 < http://www.parp.gov.pl/index/more/31715>.
14. Justyna Jasiewicz, *Przygotowanie do pracy w środowisku informacyjnym*, Warszawa 2012, s. 18, <http://cyfrowagospodarka.pl/wp-content/uploads/2012/07/09_SrodowiskoPracy.pdf>.
15. *Młodość czy doświadczenie? Kapitał ludzki w Polsce*, Kraków 2012; s. 48, 134 <http://www.parp.gov.pl/files/74/81/626/16433.pdf>.
16. Za: Eurostat, *Computer Skills in the EU27 in figures*, <http://europa.eu/rapid/pressReleasesAction.do?reference=STAT/12/47&format=HTML&aged=0&language=en&guiLanguage=en>.
17. *Polska 2030. Wyzwania rozwojowe*, s. 219, <http://zds.kprm.gov.pl/sites/default/files/pliki/pl_2030_wyzwania_rozwojowe.pdf>.
18. Por. Artur Grabek, *Już co trzeci inżynier to kobieta*, „Rzeczpospolita", <http://www.ekonomia24.pl/artykul/495345,834292-Co-trzeci-inzynier-w-Polsce-to-kobieta.html>.
19. http://eacea.ec.europa.eu/education/eurydice/about_eurydice_en.php
20. http://www.eurydice.org.pl/informacja_o_programie
21. http://www.eurydice.org.pl/sites/eurydice.org.pl/files/KD_ict_PL.pdf
22. http://www.guardian.co.uk/education/2012/jan/11/digital-literacy-michael-gove-speech
23. http://www.education.gov.uk/schools/teachingandlearning/curriculum/primary/b00199028/ict
24. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/193838/CONSULTATION_REPORT_CHANGING_ICT_TO_COMPUTING_IN_THE_NATIONAL_CURRICULUM.pdf
25. http://www.men.gov.pl/images/stories/pdf/Reforma/6d.pdf s. 100 wg. numeracji w PDF
26. http://www.men.gov.pl/images/stories/pdf/Reforma/6e.pdf s. 105 wg. numeracji w PDF
27. https://ec.europa.eu/digital-agenda/node/51275
28. http://www.oi.edu.pl/l/35/
29. http://www.oi.edu.pl/static/attachment/20110331/Folder.pdf
30. http://www.bebras.org/
31. Bill Liao na targach BETT 2013
32. http://www.dtsato.com/blog/wp-content/uploads/2008/06/sato-codingdojo.pdf
33. http://codekata.pragprog.com/2007/01/code_kata_backg.html#more
34. Por. np. http://www.youtube.com/watch?v=gav9fLVkZQc
35. http://www.wired.com/gadgetlab/2010/12/ipod-nano-hack-could-enable-video-games/
36. http://zen.coderdojo.com/dojo
37. http://coderdojo.com/help-us/i-want-to-start-a-coderdojo-right-now/
38. http://www.ted.com/talks/sugata_mitra_shows_how_kids_teach_themselves.html
39. https://www.facebook.com/CoderDojoPL
40. https://twitter.com/CoderDojoPL
41. http://zambrow.org.pl/aktualnosci/strona.php?strona=artykuly_pokaz&id=12261
42. http://youtu.be/gM95HHI4gLk
43. http://www.khanacademy.org/cs
44. http://youtu.be/nKIu9yen5nc
45. Autorzy *Scratcha* zapowiadają, że jeszcze w tym roku pojawi się w wersji 2.0 możliwość pracy *offline*
46. http://www.stat.gov.pl/cps/rde/xbcr/gus/e_oswiata_i_wychowanie_2011-2012.pdf, s. 110.
47. http://scratched.media.mit.edu/
48. https://edustore.eu/publikacje-edukacyjne/63-programowanie-wizualne-dla-kazdego-przewodnik-po-scratch.html
49. Por. *Młodzi i media*, dz. cyt., s. 116.