

Trudne Tematy w Najprostszy Sposób

Damian Kurpiewski, Krzysztof Skowronek



# Wstęp

Informatyka się rozwija, cały czas. Co chwilę powstają nowe technologie. Także w kwestii dydaktyki informatyki od pewnego czasu dostrzeżono potrzebę położenia większego nacisku na naukę programowania. I tak nowa podstawa programowa mówi, że dzieci powinny być uczone programowania już od najmłodszych lat. W klasach pierwszych mamy do czynienia z programowaniem wizualnym, a już od szóstej klasy mają być wprowadzane podstawy programowania tekstowego. Niewątpliwie jest to dobra zmiana, ale czy przyniesie spodziewane efekty, o tym się dopiero dowiemy. Nie od dzisiaj wiadomo, że wielu nauczycieli nie wie, jak skutecznie uczyć programowania. Nic w tym dziwnego, wielu przecież nawet nie umie programować. Żeby nauczyć się dobrze programować, należy skończyć odpowiednie studia, albo przynajmniej jakieś kursy. Jeżeli jednak ktoś umie dobrze programować, to jaka jest szansa, że pójdzie uczyć do szkoły, skoro może dostać posadę programisty za znacznie lepsze pieniądze. Nie ma co się oszukiwać, ciężko znaleźć dobrego programistę, który chciałby pójść uczyć do szkoły. Wystarczy spojrzeć na poziom zarobków programistów, albo koszty szkoleń z programowania. W takim razie potrzebny jest sposób, aby dobrze uczyć programowania, samemu nie umiejąc wcale tak dobrze programować. Ale czy rzeczywiście znajomość programowania jest potrzebna nauczycielowi? Należy tutaj zadać sobie pytanie, czego chcemy nauczyć naszych uczniów. Czy chcemy ich uczyć danego języka programowania, konkretnej technologii, dobrych praktyk programistycznych, czy też wszystkiego tego, czego mogą praktycznie samodzielnie nauczyć się z kursów internetowych, czy stacjonarnych szkoleń? A może lepiej skupić się na zrozumieniu koncepcji programistycznych, tworzeniu algorytmów i myśleniu komputacyjnym. Taka wiedza wydaje się być znacznie bardziej praktyczna i ułatwi im późniejszą naukę konkretnego języka. Niestety wiele materiałów przygotowywanych dla szkół i nauczycieli wydaje się skupiać właśnie na nauce konkretnego narzędzia. Ponadto rozwiązania proponowane szkołom, często przez róże firmy, bardzo często opierają się na ograniczeniu uczniów do tego jednego, konkretnego narzędzia, a lekcje polegają na ciągłej pracy z komputerem. Należy pamiętać, że informatyka to nie tylko komputer. To przede wszystkim różnego rodzaju koncepcje i algorytmy. Wielu tematów z dziedziny informatyki, w szczególności tych związanych z myśleniem komputacyjnym, możemy przeprowadzić bez pomocy komputera. Wystarczy spojrzeć na projekt Computer Science Unplugged [].

# Wprowadzenie do algorytmów

Algorytmy obecne są wszędzie. Wstając rano i przygotowując się do szkoły czy pracy, podążamy za pewnym algorytmem. Doświadczony programista powinien umieć te algorytmy dostrzegać – zauważać pewne powtarzające się wzorce, umieć je nazwać i wykorzystać. Instrukcje warunkowe, pętle, algorytmy liniowe – to wszystko są bardzo podstawowe pojęcia, bez których nie da się dobrze przyswoić wiedzy programistycznej.

# Zmienne

Bez zmiennych ciężko jest napisać jakikolwiek zaawansowany algorytm. W programowaniu i nauce pojawiają się one dość intuicyjnie i naturalnie, niemniej jednak zdarza się, że sprawiają duże problemy koncepcyjne. Być może ma to związek ze sposobem przedstawiania i wykorzystywania zmiennych na lekcji matematyki. Tam pod nazwą zmiennej zazwyczaj kryje się niewiadoma, której wartość musimy policzyć. W informatyce jednak zmienne praktycznie zawsze mają przypisaną konkretną wartość, która, jak wskazuje nazwa, może ulegać zmianom w trakcie wykonywania programu.

# Wprowadzenie do funkcji

Funkcje wykorzystujemy wszędzie. Są one obecne w każdym bardziej zaawansowanym programie, ale możemy znaleźć ich odwzorowania także w życiu codziennym. Weźmy na przykład taką maszynę do kawy – realizuje ona pewną funkcje, która jako dane przyjmuje ziarna kawy i wodę, a jako wynik zwraca świeżo zaparzoną kawę. Takie przykłady możemy odnaleźć w wielu miejscach, wystarczy odpowiednio dostosować swój sposób postrzegania świata.

Niezwykle ważne w zrozumieniu funkcji jest przyswojenie sobie takiej informacji: nie musimy wiedzieć, jak zrealizowana (zaimplementowana) jest funkcja, aby móc z niej skorzystać. Zauważmy, że robimy to na co dzień, choćby korzystając ze spłuczki w toalecie. Znamy ogólną zasadę działania, wiemy co musimy nacisnąć, ale to wszystko. Nie potrzebujemy bogatszej wiedzy, aby spuścić wodę. Podobnie jest w programowaniu. Funkcji używamy nawet w najprostszych programach, na przykład, gdy chcemy wypisać komunikat na ekran. Używamy odpowiedniego polecenia, które realizuje pewną funkcję, o której zwykle nie wiemy nic, poza tym, co przyjmuje i jaki jest jej wynik. Podajemy tekst, a on pojawia się na ekranie czy w konsoli. To wszystko, co musimy wiedzieć. Oczywiście zgłębianie wiedzy i ciekawość świata są bardzo ważne w nauce, jednak równie ważna jest umiejętność myślenia abstrakcyjnego i korzystania z narzędzi, które są nam dane.

# Wprowadzenie do rekurencji

Rekurencja jest niewątpliwie jednym z najtrudniejszych tematów, jeżeli chodzi o naukę programowania. Koncepcja funkcji rekurencyjnej jest o tyle trudna do przyswojenia, ponieważ wydaje się, że jest ona sprzeczna z naszym sposobem rozumowania. Człowiek, zazwyczaj, myśli iteracyjnie, krok po kroku.

Spis treści

[Wstęp 1](#_Toc520902991)

[Wprowadzenie do algorytmów 2](#_Toc520902992)

[Zmienne 3](#_Toc520902993)

[Wprowadzenie do funkcji 4](#_Toc520902994)

[Wprowadzenie do rekurencji 5](#_Toc520902995)