PROGRAMOWANIE
JAK UCZYĆ, Z POMOCĄ CZEGO I DLACZEGO

Damian Kurpiewski
Instytut Podstaw Informatyki, Polska Akademia Nauk
d.kurpiewski@ipipan.waw.pl

Abstract. Teaching computer science in school is a difficult task, both for teachers and for students. Especially considering programming. More and more tools are created to help children understand hard art of programming, so why not use some of them in school? I selected some of the tools and programming languages to talk about their pros and cons and usefulness.

1. Wstęp

Programowanie jest jednym z najtrudniejszych tematów poruszanych na lekcjach informatyki i nie bez powodu. Rzeczywiście, daje się zauważyć pewna zależność, jakoby programowanie wymagało pewnego specyficznego sposobu myślenia. Nieraz słyszymy o informatykach, że porozumieją się innym językiem. Niektórzy nawet twierdzą, że programistę można rozpoznać na pierwszy rzut oka. Jest to oczywiście pewien stereotyp, sam jednak programuję już od dłuższego czasu, skończyłem studia informatyki i muszę przyznać, że jednak kryje się w tym ziarnko prawdy. Zrozumienie programowania nie jest proste. Nie znaczy to jednak, że nauka programowania musi być trudna, żmudna i nudna. Mamy XXI wiek, technologia rozwija się w zadziwiającym tempie. Minęły już czasy, kiedy jedna niewłaściwa instrukcja mogła sprawić, że trzeba było zrestartować komputer, a wynik działania programu widoczny był jako komunikat na ekranie monitora. Obecne narzędzia pozwalają nawet na wskazanie linijki, w której popełniliśmy błąd i wyświetlenia sugestii, jak go naprawić. Mają znacznie większe możliwości niż tylko wyświetlenie komunikatu. Dlatego warto z nich korzystać. W tej pracy przybliżam czytelnikowi możliwości, jakie ma nauczyciel, który zdecyduje się nauczać programowania.

2. Programowanie blokowe

Gdy uczymy programowania najmłodszych, bardziej zależy nam na tym, aby poznali jego ideę, aniżeli nauczyli się jakiegoś konkretnego języka. Oczywiście

poznanie takiego języka programowania jak np. C++, czy Python, da uczniom znacznie większe możliwości w przyszłości, zarówno pod względem pisanych programów jak i nawet znalezienia potencjalnej pracy. Nie jest to jednak najlepszy sposób, w jaki można zacząć swoją przygodę z programowaniem. Jak wspomniałem wcześniej, sztuka ta jest trudna i tego się będę trzymał, dlatego zaczynamy od rzeczy jak najprostszych. W tym celu warto sięgnąć po jedno z dostępnych środowisk programowania blokowego. Nazywam je tutaj blokowym z racji tego, że program tworzymy zazwyczaj z różnokształtnych, kolorowych bloków, klocków, czy też puzzli. Możemy więc tutaj mówić o gatunku programowania wizualnego, gdzie suchy kod zastąpiony zostaje gotowymi elementami. Takie środowisko jest zazwyczaj proste w użyciu, posiada intuicyjny interfejs i jest znacznie prostsze do opanowania w porównaniu z konkretnym językiem programowania. Jego niezaprzeczalną zaletą jest fakt, że wszystkie bloki, z jakich uczeń może korzystać, są dla niego bezpośrednio widoczne. Może je przeglądać by znaleźć komendę, której akurat potrzebuje. W związku z tym uczniowie nie muszą zapamiętywać różnych poleceń, ciągów instrukcji, a jedynie je rozumieć. Unikamy w ten sposób bardzo częstych i nieraz czasochłonnych błędów składniowych, które bywają irytujące i zniechęcające dla uczniów. Uczniowie nie muszą uczyć się na pamięć, a my jako nauczyciele, możemy skupić się na tym co najistotniejsze, czyli nauczenia ich rozumienia istoty programu i konkretnych instrukcji, takich jak np. pętle.

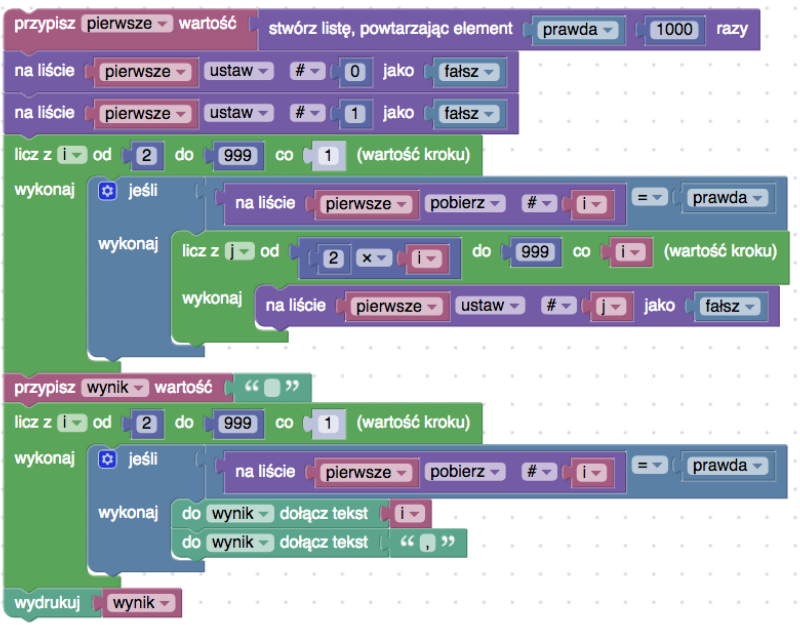
2.1. Scratch

Jednym z najpopularniejszych środowisk graficznych do nauki programowania dla dzieci jest właśnie Scratch. W tym narzędziu tworzymy program, który steruje tzw. duszkami. Duszki są to kolorowe postaci, rysunki przedstawiające zwierzęta, ludzi, rośliny czy też przedmioty. Układamy dla nich instrukcje, które opisują ich działania i sposób interakcji z innymi duszkami czy elementami. W ten sposób w Scratchu możemy tworzyć gry, animacje czy też interaktywne historie. Jest to zastosowanie dość ograniczone, ale za to idealnie nadające się dla młodszych dzieci. W zaledwie kilkanaście minut można stworzyć własną grę i pokazać ją znajomym. Samo środowisko jest bardzo intuicyjne i proste w obsłudze, a ponadto posiada polską wersję językową.

Nbędę się rozpisywał o Scratchu, gdyż jest to narzędzie dobrze znane i stosowane w wielu szkołach podstawowych. Spotkałem się także z próbami stosowania go w gimnazjum, chociaż osobiście nie uważam tego za najlepszy pomysł. Niewątpliwą zaletą Scratcha jest jego atrakcyjność dla młodszych dzieci, niestety stanowi to też jego dość duże ograniczenie. Jest praktycznie niemożliwym zastosowanie go do innych celów niż proste gry czy animacje. Prezentacja konkretnych algorytmów w tym środowisku jest bardzo trudna i mało intuicyjna. Poza tym jego charakter sprawia, że dla starszych uczniów narzędzie to wydaje się być po prostu dziecinne.

2.2. Blockly

Blockly jest graficznym środowiskiem programistycznym, które powstało kilka lat temu i od tego czasu prężnie się rozwija. Program tworzony jest tutaj z poszczególnych klocków, których pełną listę mamy dostępną w odpowiednim menu, podobnie jak w Scratchu. To, co przede wszystkim odróżnia Blockly od Scratcha, jego możliwości. Podczas gdy Scratch pozwala na tworzenie prostych gier, animacji czy też interaktywnych opowieści, Blockly może znacznie więcej. Jak już wspomniałem, jest to środowisko programistyczne, a pod tym pojęciem rozumiem to, że cały interfejs jest tak naprawdę nakładką graficzną do prawdziwego języka programistycznego. Program stworzony w Blockly jest jednocześnie programem napisanym w języku JavaScript, zwanym także językiem internetu. Nie oznacza to, że możemy skorzystać z pełnych możliwości tego języka, ale jednak możemy tworzyć pełnoprawne programy i implementować rożne algorytmy. W związku z tym system ten można wykorzystać z powodzeniem także w liceum, co sam zresztą stosowałem. Z racji tego, że implementacje algorytmów są bardzo zbliżone do tych napisanych w C++, a różnice wynikają przede wszystkim z innego zarządzania tablicami, możemy zastąpić wybrany język poprzez Blockly nie tracąc przy tym na zrozumieniu tematu przez uczniów, a wręcz przeciwnie, zyskując znacznie.



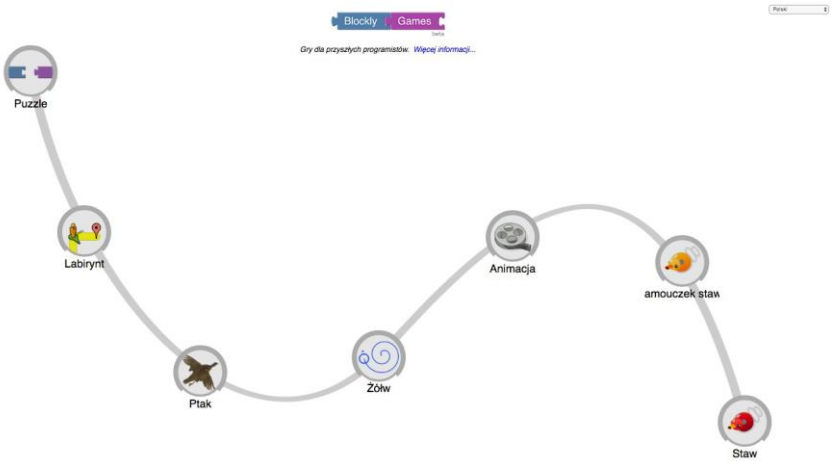
Rysunek 1 Sito Erastostenesa w Blockly

Bariera wybory Blockly jest znacznie mniejsza niż w przypadku faktycznego języka, co wynika z jego blokowego charakteru. Oczywiście nie nauczymy w ten sposób uczniów żadnego z maturalnych języków programowania, natomiast ułatwimy naukę i zrozumienie istoty programowania i algorytmiki tym, którzy nie wiążą swojej przyszłości z programowaniem i nie są tym zainteresowani. Należy tutaj zwrócić szczególną uwagę na fakt, że brak bezpośredniego związku Blockly z samą maturą z informatyki nie jest bynajmniej powodem, dla którego nauczyciel powinien zrezygnować z tego narzędzia. Zysk z jego stosowania, przynajmniej dla części klasy, wydaje się być na tyle istotny, by jednak rozważyć rozwiązanie problemu nauki konkretnego języka. Podejście które sam zastosowałem opierało się na jednoczesnym uczeniu języka C++ i korzystaniu z Blockly. Obserwując uczniów zainteresowanych programowaniem można zauważyć, że radzą oni sobie znacznie lepiej w rozwiązywaniu zadań programistycznych, więc można poświęcić im mniej uwagi w trakcie samej implementacji algorytmu. Tak było w przypadku mojej klasy, w której mniej niż 1/3 z uczniów zdecydowała się pozostać przy nauce C++. Pozostali uczniowie, którzy korzystali z Blockly, wymagali w większości większej uwagi i dokładnego wytłumaczenia implementacji krok po kroku.

Blockly można jednak z powodzeniem stosować nie tylko w liceum. Istnieje kilka „gier" Blockly, które polegają na rozwiązywaniu coraz to trudniejszych problemów algorytmicznych. Dobrym wprowadzeniem do programowania w klasie podstawowej może być np. gra Blockly Labirynt. Jak nazwa wskazuje, zadanie polega na przejściu labiryntu. Konkretne zadanie ucznia to „napisanie" (ułożenie) programu, który po uruchomieniu przemieści naszą postać z punktu początkowego do oznaczonego celu. Cała gra składa się z dziesięciu ćwiczeń, a w każdym z nich mamy do dyspozycji określone bloki (instrukcje). W początkowych labiryntach wystarczy zastosować instrukcje poruszenia się do przodu i skrętu, ale w kolejnych musimy już użyć pętli, a później także instrukcji warunkowej. Dużą zaletą gier Blockly jest ich stopniowo rosnący poziom trudności, różnorodność i proste wprowadzanie nowych konstrukcji programistycznych. W Libiryncie możemy nauczyć się korzystania z pętli i nstrukcji warunkowych do rozwiązywania prostych problemów, w grze Ptaki operujemy już na układzie współrzędnych i musimy rozróżniać kilka etapów gry, w rysowaniu żółwiem istotne są kąty, kolory i długości linii, a w Animacjach do tego wszystkiego dochodzi jeszcze czas. Tak więc do dyspozycji nauczyciela jest już przygotowany cały arsenał gotowych ćwiczeń o różnym poziomie trudności, wokół których można poprowadzić całą lekcję.

Jak wspomniałem wcześniej, w ostatnich latach Blockly zdobywa popularność. Jest to system otwarto źródłowy, a to oznacza, że możemy go dowolnie dostosowywać do własnych potrzeb. Co więcej, twórcy przygotowali specjalne narzędzie, które pozwala tworzyć własne bloki i m.in. dlatego Blockly znajdziemy także w wielu systemach sterowania robotami. Graficzne środowiska programistyczne takich

robotów jak Ozzobot opiera się właśnie na Blockly, widać więc, że jest to system praktycznie stosowany. Co więcej są plany połączenia Blockly ze Scratchem.



Rysunek 2 Blockly Games, https://blockly-games.appspot.com/

3. Języki programowania

Do nauki konkretnego języka programowania możemy przystąpić w różnym wieku, jednak lepsze efekty osiągniemy wtedy, gdy będzie to dopiero kolejny krok. Gdy zrozumieliśmy już na czym polega programowania, wiemy, jak działają różne instrukcje przepływu, czy też na podstawie Blockly, czy innego narzędzia, możemy już przystąpić do nauki konkretnego języka. Tutaj mamy dość duże możliwości wyboru, skupimy się jednak na językach uczonych w szkole, a konkretnie w liceum. Możemy więc wybierać pomiędzy C++, Javą, a Pythonem. Od 2016 roku pomijamy już język Pascal. W tym rozdziale opowiem po krótce o wadach i zaletach każdego z tych języków. Dodatkowo na końcu opowiem także o Ruby, który chociaż nie ma swojego miejsca na maturze, to znalazł ciekawe zastosowanie w pewnym narzędziu.

3.1. Java

Nad tym językiem nie będę się rozpisywał. Java jest językiem typowo obiektowym, jednym z najpopularniejszych wśród programistów []. Język ten służy przede wszystkim do tworzenia aplikacji okienkowych, serwisów webowych oraz aplikacji na platformę Android. Ma naprawdę szerokie zastosowania, ale raczej przy większych projektach. Jako pierwszy język, którego chcemy się nauczyć, ma bardzo dużą barierę wejścia. W zastosowaniu szkolnym zdecydowanie odradzam korzystania z Javy. Tak jak wspomniałem, jest to język obiektowy i wykorzystywanie go

jako języka strukturalnego jest zachowaniem nienaturalnym. Co za tym idzie nauka algorytmów na przykładzie implementacji w Javie nie jest rzeczą łatwą.

3.2. C++

Jako jeden z najpopularniejszych języków programowania, C++ znalazł swoje miejsce także w szkolnej edukacji. Nie należy się temu dziwić, jest to język, który każdy szanujący się programista powinien znać, przynajmniej jego podstawy. Ma bardzo szerokie zastosowania, od prostych aplikacji konsolowych, poprzez programowanie mikrokontrolerów do potężnych programów na komputery stacjonarne i silników graficznych. C++ jest uniwersalny i multi-platformowy. Można go nazwać językiem sztandarowym i wiele innych ma zbliżoną do niego składnię. Dlatego gdy już pozna się ten język, to nauczenie się innego jest zazwyczaj dość szybkim procesem. Nie jest to jednak język prosty do początkowej nauki.

Zacznijmy od zalet C++ i dlaczego uważam, że warto rozważyć uczenie go w swojej klasie. Po pierwsze jest to język strukturalny, w przeciwieństwie do Javy, która jest językiem obiektowym. Dzięki temu łatwiej jest się go nauczyć, ponieważ mniej rzeczy musimy zrozumieć na samym początku. Interesuje nas tylko ogólna struktura programu, funkcja main w której piszemy główny fragment naszego programu i dołączenie odpowiednich bibliotek. Jeden plik, na którym operujemy i nic więcej, przynajmniej w podstawowym zastosowaniu. Implementujemy nasz algorytm w funkcji main, albo własnej funkcji dedykowanej i o nic więcej się nie martwimy, ponieważ używamy języka zgodnie z jego przeznaczeniem. Poza tym mamy pełną kontrolę nad każdą zmienną, sami decydujemy o jej typie, a co za tym idzie rozmiarze. Jeśli potrzebujemy liczby całkowitej, piszemy int. Do liczb rzeczywistych użyjemy double lub float, a do tekstów typu string. Jeśli odpowiednio sformatujemy nasz kod i nazwiemy zmienne, to wystarczy jeden rzut oka by zrozumieć, co do czego służy. Instrukcje warunkowe i pętle stają się intuicyjne, jak tylko zrozumiemy ich działanie i łatwo je odzwierciedlić w schemacie blokowym. Dodatkowo podobieństwo do nich znajdziemy także w pseudokodzie, który bardzo często jest wzorowany na C++. W związku ze swoją popularnością i dość długą już obecnością w informatyce, łatwo jest znaleźć książkę, czy też tutorial dostępny w Internecie, z których możemy nauczyć się tego języka. Dostępne są także implementacje wielu algorytmów w C++, co jest wsparciem także dla nauczyciela. Dodatkowo należy wspomnieć, że język ten jest wykorzystywany w wielu serwisach z zadaniami programistycznymi, gdzie możemy wysłać nasze rozwiązanie do automatycznej oceny, a także na zawodach w programowaniu zespołowym, i oczywiście na olimpiadzie informatycznej.

Przejdźmy teraz do drugiej strony medalu, czyli wad, z których wiele wynika bezpośrednio z wyżej wymienionych przeze mnie zalet. Po pierwsze C++ ma bardzo sztywną strukturę, która chociaż zwiększa jego czytelność, to jednocześnie

sprawia, że musimy się jej nauczyć i zrozumieć. Podział programu na funkcje i konieczność implementacji funkcji main jest jednak dość ograniczający i na początku sprawia, że piszemy wiele linijek kodu, które tak naprawdę nas nie interesują. Po drugie musimy pamiętać o dołączeniu odpowiednich bibliotek, nawet jeżeli korzystamy z bardzo podstawowych funkcjonalności, jak np. obsługa wejścia i wyjścia. Wielu nauczycieli obecnie decyduje się na korzystanie z biblioteki iostream, a co za tym idzie dołączają także użycie przestrzeni nazw std. Jest to kolejna informacja, jaką trzeba przekazać uczniom i ją wytłumaczyć, która przecież nie jest bezpośrednio związana z samą implementacją algorytmów. Jednak za największą trudność w nauce C++ uważam zmienne i ich typy. Z jednej strony mamy nad nimi pełną kontrolę, z drugiej strony musimy je dobrze znać, wiedzieć jakie limity ma np. typ int, uważać, żeby użyć właściwego typu do danej wartości, i tym podobne. Jest to wiele informacji, które chociaż są później bardzo przydatne, to na samym początku znacznie zwiększają próg wejścia do języka. Uczeń musi nie tylko zrozumieć istotę zmiennej, jej zastosowanie, ale także pamiętać o zastosowaniu właściwego typu. Kontrola nad typami zmiennych jest bardzo ważna, ale jej praktyczne zastosowanie znajdujemy dopiero w późniejszym etapie tworzenia aplikacji w wybranym języku. Posunę się tutaj nawet do stwierdzenia, że w zastosowaniu szkolnym można ją praktycznie całkowicie pominąć.

C++ ma jeszcze jedno bardzo duże ograniczenie. Podczas gdy tworzenie w tym języku aplikacji konsolowych jest dość prostym zadaniem, to napisanie programu, który narysuje nam coś na ekranie, albo utworzy jakąś animację, nie jest już takie proste. W liceum np. poruszany jest w pewnym momencie temat fraktali i ich algorytmiczne generowanie. Jako nauczyciel i doświadczony programista miałem duży problem, by zaprezentować to uczniom przy użyciu C++. Z jednej strony biblioteki graficzne są zależne od używanego systemu, z drugiej nawet podstawowy program rysujący jedną linię musi zawierać wiele funkcji i linijek kodu, które wprowadzają zamieszanie wśród uczniów. W mojej opinii jest to duże ograniczenie, które zmniejsza atrakcyjność tego języka, ponieważ zazwyczaj jedynym wynikiem działania programu jest tekst wyświetlony w konsoli.

3.3. Python

Od roku 2016/17 na maturze pojawi się nowy język programowania w miejsce mającego już swoje lata Pascala. Mowa tu oczywiście o Pythonie. I chociaż zgadzam się z opinią, że Pascal jest świetnym językiem dydaktycznym, to jednak uważam, że usunięcie go z matury było słuszną decyzją. Jest to język, który obecnie staje się wymarłym, a rzeczy wymarłe należy zostawiać entuzjastom i nie zmuszać nikogo do uczenia się ich, szczególnie w dziedzinie informatyki. Z mojego doświadczenia równie dobrze można nauczyć się programowania w Pythonie (a może

nawet lepiej), co w Pascalu, chociaż w tym przypadku na nauczycielu ciąży większa odpowiedzialność odnośnie tego, w jaki sposób przekazać uczniom tą wiedzę.

Na początek zajmę się kwestią sporną, czyli co wybrać: C++ czy Python? Mowa tutaj przede wszystkim o klasach licealnych. Wybór ten jest rzeczywiście dość istotny i jak przy większości podejmowanych decyzji nie ma tutaj jedynej słusznej. Oba języki mają swoje wady i zalety, jeżeli spojrzymy na nie z dydaktycznego punktu widzenia. W poprzednim rozdziale opisałem po krótce język C++, tutaj więc skupię się na tym drugim. Python jest przede wszystkim językiem skryptowym, podobnie jak przytoczony wcześniej Ruby. Oznacza to przede wszystkim, że nie jest on kompilowany (chociaż może być), a jedynie interpretowany. W praktyce szkolnej znaczy to tyle, że niektóre błędy zobaczymy dopiero podczas działania programu, podczas gdy w C++ byłyby one widoczne na etapie kompilacji. Tak naprawdę podczas lekcji nie ma to aż tak dużego znaczenia, chociaż niektóre błędy uczniowie będą zgłaszali później, dlatego warto kłaść duży nacisk na częste testowanie swoich programów (nawet jeszcze nie dokończonych).

W porównaniu z C++ język Python jest prostszy w nauce. Dzieje się tak za sprawą wielu czynników, który wynikają głównie z jego skryptowej natury. Python nie narzuca tak sztywnej struktury kodu jak C++. Nie musimy pisać funkcji main, dołączać odpowiednich bibliotek (przynajmniej na początku), skupiamy się na tym co nas najbardziej interesuje. Poza tym w Pythonie nie występuje silne typowanie, co oznacza, że nie musimy zwracać szczególnej uwagi na różne typy zmiennych. Tak jak w C++ mamy int dla liczb całkowitych, czy też string dla tekstów, tak tutaj piszemy po prostu var i do naszej zmiennej przypisujemy dowolną wartość. Jest to zdecydowanie prostsze podejście, które sprawia, że uczniowie mają mniej do zapamiętania. W podobnie prosty sposób tworzymy tablice. Jeśli chodzi o pętle i instrukcje warunkowe to są one zbliżone do tych w C++. Podobnie jest w przypadku funkcji, które jednak nie wymagają podania typów parametrów i zwracanej wartości. Porównując te dwa języki warto wspomnieć o tym, że Python wymusza na programiście właściwe formatowanie. Kod wewnątrz pętli czy funkcji nie jest otoczony nawiasami, ale musi mieć odpowiednie wcięcia. Dzięki temu pisany program jest często znacznie czytelniejszy od tego pisanego przez uczniów w C++ i narzuca pewne dobre praktyki.

Czy jest jednak coś, co poza łatwością nauki daje językowi Python przewagę nad C++ w zastosowaniu szkolnym? Funkcjonalność. A nawet nie tyle sama funkcjonalność, co znacznie prostszy sposób tworzenia praktycznych narzędzi. Python posiada bardzo wiele bibliotek i zastosowań, dzięki czemu może być traktowany jako narzędzie do automatyzacji żmudnych zadań. W internecie znajdziemy wiele przykładów programów w Pythonie, które służą np. do przepisania danych z pliku na inny format, pobrania informacji z witryny internetowej, utworzenia ciekawej animacji, narysowania wykresu, a także wiele innych. Wiele z nich jest bardzo pro-

stych i możemy je wykorzystać do pokazania uczniom, że mogą pisać programy, które będą miały zastosowanie nie tylko na lekcjach informatyki, ale także na innych przedmiotach, czy też w życiu codziennym. Możemy ich w ten sposób łatwiej skłonić do samodzielnego rozwijania swoich umiejętności programistycznych.

3.4. Ruby – Sonic Pi

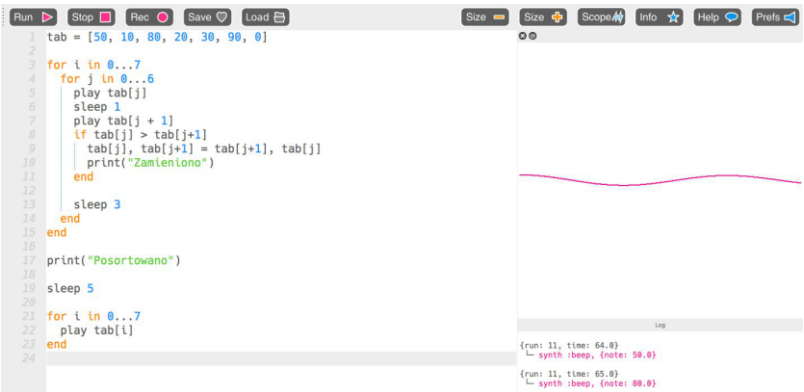
Urozmaicenie nauki programowania nie jest prostym zadaniem. Przede wszystkim proces twórczy programisty jest zazwyczaj długi i żmudny, a wyniki rzadko kiedy są ekscytujące. Podczas gdy ja, zaprawiony w boju autor wielu programów, potrafię się cieszyć jak dziecko, gdy po wielu godzinach ciężkiej pracy zobaczę wreszcie w konsoli prawidłowy wynik, to jednak większość nastolatków nie odczuje podobnej radości podczas pracy na lekcji. Obecnie tekst wypisany w konsoli, nawet jeżeli to za naszą sprawą się tam pojawił, nie jest niczym wyjątkowym ani zjawiskowym. Co w takim razie możemy zrobić, aby komputer pokazał nam coś więcej niż tylko literki na ekranie? A może tak napiszemy program, który liczby zamieni na dźwięk? Osoby znające C++ wiedzą pewnie, że generalnie nie jest to takie trudne. Za pomocą prostej komendy możemy kazać komputerowi wydać z siebie charakterystyczny dźwięk, tzw. 'beep'. Jak się postaramy, to możemy nawet uzyskać odpowiednią wysokość tego dźwięku. W ten sposób możemy jednak co najwyżej ułożyć piosenkę z gry Mario []. Bardziej ekscytująca byłaby możliwość tworzenia prawdziwej muzyki, a przynajmniej uzyskania brzmienia różnych instrumentów. Tutaj z pomocą przychodzi nam Sonic Pi.

Sonic Pi jest ciekawym narzędziem, które pozwala nam w sposób całkowicie programistyczny tworzyć muzykę, jednocześnie zachowując prostotę obsługi. Nie musimy więc tutaj wgłębiać się w zakamarki korzystania z karty dźwiękowej, tworzenia plików, czy też używania skomplikowanych protokołów. Wręcz przeciwnie, po prostu wybieramy instrument i podajemy odpowiednie parametry, takie jak wysokość dźwięku i długość jego trwania. Dodając do tego pętlę możemy uzyskać całkiem ciekawy bit. Przykładowy kod widoczny jest na obrazku [].

Kod w Sonic Pi napisany jest w języku Ruby. Jest to język skryptowy, kojarzony przede wszystkim z frameworkiem Ruby on Rails służącym do tworzenia aplikacji webowych. Założenia Ruby są bardzo zbliżone do założeń języka Python, w związku z czym można się go dość szybko nauczyć. Nie znaczy to jednak, że musimy od razu przechodzić cały kurs tego języka. Do tworzenia muzyki wystarczą nam jego podstawy. Nawet znając jedynie pętle i instrukcje wywołania danego dźwięku możemy już skomponować własny, przyjemny dla ucha utwór.

Jaka jednak korzyść płynie ze stosowania Sonic Pi dla nauczyciela, a przede wszystkim dla uczniów? Po pierwsze narzędzie to daje możliwość odbierania wyników działania napisanego programu na dwa sposoby: wzrokowy i słuchowy. Nie tylko widzimy, co robi nasz program, ale także to słyszymy. Co za tym idzie docho-

dzi do tego pewna estetyka i wyczucie. Warto zachęcać uczniów, aby pracowali nad swoimi programami, aby uzyskać ciekawy i satysfakcjonujący wynik, który w efekcie mogą wykorzystać np. jako dzwonek w telefonie. Poza tym programem, który generuje skomponowany przez nas utwór łatwiej jest się pochwalić niż takim, który dodaje dwie liczby. To pierwszy aspekt, moim zdaniem bardzo ważny, który zwiększa atrakcyjność programowania w oczach uczniów. Drugim powodem, dla którego nauczyciele powinni zastanowić się nad zastosowaniem tego narzędzia na swoich lekcjach, jest to, że nie jest ono zabawką. To jest dość potężny program, w którym wykorzystujemy prawdziwy, stosowany powszechnie język programowania. Oznacza to chociażby tyle, że umiejętności zdobyte przy korzystaniu z Sonic Pi uczniowie mogą z powodzeniem wykorzystać przy nauce C++, czy też Pythona. Ponadto Sonic Pi można wykorzystać także do zaprezentowania wybranych algorytmów, np. sortowania bąbelkowego.



Rysunek 3 Przykładowe sortowanie bąbelkowe w Sonic Pi

4. Podsumowanie

Powstaje coraz więcej narzędzi, które mają ułatwić naukę programowania i algorytmiki. Wiele z nich można z powodzeniem wykorzystać w szkole, na różnych etapach. Wymaga to oczywiście większego nakładu pracy nauczyciela, jako że nieraz będzie pionierem w użyciu danego programu, jednak zysk dla uczniów może być bardzo duży. Warto więc eksperymentować i nie bać się poznawać i wprowadzać nowych narzędzi, aby zwiększyć atrakcyjność i przyswajalność lekcji informatyki dla swojej klasy.