# RÓŻNE PODEJŚCIA W PRACY Z UCZNIAMI UZDOLNIONYMI INFORMATYCZNIE

*Krzysztof Diks*
*Jakub Radoszewski, Jakub Łącki*
*Marek Cygan, Tomasz Idziaszek*

Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

{diks, jrad, cygan, j.lacki, idziaszek}@mimuw.edu.pl

W tym artykule zamieszczamy przemyślenia osób związanych z Olimpiadą Informatyczną na temat pracy z uczniem uzdolnionym informatycznie. Przemyślenia są osobiste. Czasami zawierają różne spojrzenie na ten sam problem. Ich największą zaletą jest to, że przedstawiają je osoby, które niewątpliwie są utalentowane, osiągnęły sukces i starają się teraz same docierać do młodzieży uzdolnionej informatycznie i pracować z nią systematycznie.

## 1. Cztery elementy w pracy z uczniem zdolnym (*Krzysztof Diks*)

Moje przemyślenia w pracy z uczniem zdolnym, to przemyślenia nauczyciela akademickiego, który nigdy nie uczył w szkole i którego kontakt z uczniami uzdolnionymi informatycznie ma miejsce tylko poprzez Olimpiadę Informatyczną. Z drugiej strony jako nauczyciel akademicki na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego miałem do czynienia z kilkuset studentami, o których śmiało można powiedzieć, że stanowili (i stanowią) śmietankę polskiej informatyki. Moje przemyślenia sformułuję w czterech punktach.

### 1. Talenty rodzą się wszędzie, zarówno w wielkich miastach, jak i malutkich osadach

Talent to diament, który należy dostrzec, wydobyć i oszlifować. W początkach tego procesu nic nie zastąpi rodziców i nauczycieli. W okresie wczesno szkolnym szkodliwa jest wszelkiego rodzaju specjalizacja. Dzieci przejawiające zdolności w naukach ścisłych należy zachęcać do czytania popularnych książek z matematyki, fizyki, chemii itp. Należy też zachęcać je do rozwiązywania zagadek logicznych i nauki gry w warcaby, szachy, Go. Niezmiernie ważna jest już w tym okresie możliwość szlachetnej rywalizacji. Służą temu różnego rodzaju konkursy, takie jak Kangur i Bóbr. Udział w konkursach umożliwia kontakty z innymi uczniami o podobnych zainteresowaniach i zdolnościach i tworzeniu grup rówieśniczych, w których wybitni uczniowie nie czują się wyobcowani.

### 2. Rola nauczyciela (rodzica) zmienia się wraz z rozwojem talentu ucznia

W początkowym etapie należy ucznia wprowadzić w dziedzinę. Przekazać mu minimum wiedzy i umiejętności niezbędnych w dalszym rozwoju. W przypadku informatyki polega to głównie na nauczeniu programowania w małej skali i podstaw algorytmiki. Według mnie programowania nie należy uczyć w oderwaniu od algorytmiki. Program, to przecież formalny zapis algorytmu. Początkujący programista powinien poznać tylko podstawowe instrukcje sterujące (if, while), proste typy danych i tablice oraz umieć korzystać z procedur i funkcji. Uczenie pełnego języka programowania, ze wszystkimi jego zawiłościami, jest na tym etapie zupełnie niepotrzebne. W późniejszym okresie

pracy z uczniem rola opiekuna powinna się ograniczyć do przyjaznego stymulowania, podsuwaniu stosownej lektury oraz problemów do rozwiązywania. Najlepsi są w stanie pracować samodzielnie i szybko się uczyć, ale potrzebują odpowiednich bodźców – życzliwego zainteresowania, docenienia ich pracy.

### 3. Konkursy informatyczne motorem rozwoju talentu

Wspomniałem już o pożytkach płynących z udziału w różnego rodzaju konkursach. Według mnie olimpiady przedmiotowe są jednym z najważniejszych elementów rozwoju uczniów utalentowanych. W tym roku szkolnym (2010/2011) odbywa się XVIII edycja Olimpiady Informatycznej. We wszystkich edycjach Olimpiady wzięło udział kilkanaście tysięcy uczniów. Zmieniają oni oblicze naszego kraju. Dla przykładu portal społecznościowy Nasza Klasa stworzyli byli olimpijczycy, podobnie coraz popularniejszy portal Codility.com jest też dziełem olimpijczyków.

Dlaczego udział w Olimpiadzie jest ważny? Żeby osiągnąć w niej sukces należy wykształcić cechy, które są niezbędne w osiągnięciu sukcesu w życiu szkolnym, naukowym, czy też zawodowym. Należą do nich między innymi: pracowitość, wytrwałość, samodzielność, dociekliwość, chęć rywalizacji, otwartość. Ludzie zdolni chcieliby rozwiązywać zadania (a w przyszłości problemy) na miarę ich zdolności i umiejętności. Rozwiązanie zadania powinno sprawiać satysfakcję, a jego nierozwiązanie pobudzać do dalszej nauki i rozwijania talentu. Zadania stawiane uczniom powinny być tak układane, żeby pozwalały uczniom zrobić kolejny krok na drodze ich rozwoju – poznać i zrozumieć nową technikę, odkryć nowy algorytm.

Olimpiada to też miejsce tworzenia się wspólnoty zainteresowań. Niezmiernie ważną rolę w Olimpiadzie Informatycznej odgrywa forum dostępne w Systemie Informatycznym Olimpiady. Jest to miejsce dyskusji nad zadaniami, wymiany doświadczeń, dzielenia się testami, nawiązywania znajomości. Udział w Olimpiadzie uczy też profesjonalizmu. Organizatorzy starają się przygotować profesjonalny serwis i wymagają profesjonalizmu od uczestników, w szczególności etycznego zachowania.

### 4. Rola autorytetów

Młodzi ludzie, a szczególnie ludzie utalentowani potrzebują autorytetów, które mogą stanowić wzorzec dla rozwoju ich karier. Dlatego z młodymi ludźmi powinni pracować starsi koledzy, którzy osiągnęli sukces i mogą być wzorem do naśladowania. W Olimpiadzie Informatycznej wiele prac wykonują (układają zadania, piszą wzorcowe rozwiązania, odpowiadają za oprogramowanie, prowadzą warsztaty) byli olimpijczycy, którzy osiągnęli sukcesy na skalę światową. Są to zwycięzcy Olimpiady Informatycznej oraz medaliści prestiżowych konkursów międzynarodowych, najlepsi studenci, doktoranci i cenieni pracownicy polskich uczelni. Ich wiedza i umiejętności są doceniane w skali globalnej. Największe firmy branży IT starają się przyciągnąć ich do siebie. Wielu byłych olimpijczyków osiąga też sukcesy naukowe w skali świata. Talent wymaga docenienia. Rozumieją to znani naukowcy-informatycy, którzy z pasją uczestniczą w pracach Olimpiady i dzielą się swoją wiedzą z olimpijczykami.

## 2. Wyławianie talentów informatycznych i budzenie zainteresowań uczniów informatyką, w szczególności algorytmiką (*Jakub Radoszewski*)

Istnieją różne drogi, którymi można młodzież zainteresować informatyką, w szczególności programowaniem i algorytmiką. Bodaj najprostszym pomysłem jest systematyczne prowadzenie podopiecznych: najpierw przez bardziej zaawansowane korzystanie z komputera (można z dużą dozą pewności założyć u młodzieży podstawową znajomość komputera), dalej przez składnię wybranego

języka programowania (względnie także przez prostszy język pośredni), by wreszcie dotrzeć do świata ciekawszych zagadnień algorytmicznych. Jest to w dużej mierze metoda małych kroków, dobra dla podopiecznych lubiących odkrycia natury technicznej, stymulująca wizualnie. Niestety, używając tej metody, może być trudniej zachęcić do informatyki osoby wybitnie utalentowane, lubiące przede wszystkim rozwiązywać ambitne łamigłówki, a nie przedzierać się przez gąszcz zagadnień technicznych. Takie osoby mogły już wcześniej odkryć swoje zainteresowania innymi przedmiotami ścisłymi.

Dobrą próbą wzbudzenia zainteresowania utalentowanych uczniów informatyką może być pokazanie im ciekawych zagadnień algorytmicznych z pominięciem uprzedniego wprowadzania całej warstwy technicznej. Dzięki temu podejściu uczniowie mogą antycypować to, z jakimi problemami będą mogli zmierzyć się po przyswojeniu sobie niezbędnego materiału natury technicznej. Innymi słowy, można ich zainteresować programowaniem, zainteresowawszy ich uprzednio algorytmiką. W tym duchu można, na przykład, zacząć od wprowadzenia schematów blokowych i na nich prezentować pierwsze przykłady algorytmów. Jest to ciekawy pomysł, choć takie schematy mogą wydawać się uczniom nieco sztuczne i przez to mniej interesujące.

Nasz pomysł[1] polega na przedstawieniu uczniom łamigłówek sformułowanych w języku matematyki (z użyciem bardzo prostych pojęć), których rozwiązania wymagają jedynie użycia kartki i długopisu, ale zawierają istotne elementy myślenia algorytmicznego. Przykłady takich zadań można znaleźć w moich artykułach, zamieszczonych w czasopiśmie *Delta*: Zadanka (nie)informatyczne, *Delta* 8/2009 i Liczba liczb, *Delta* 6/2010[2]. Zadania wspomnianego typu można tworzyć na podstawie konkursowych zadań informatycznych, biorąc konkretne dane wejściowe, których rozmiar nie może być zbyt mały (trywialne zadanie), ani zbyt duży (zadanie zbyt męczące, przez to niezbyt interesujące). Nie każdy problem algorytmiczny można sformułować w ten sposób – trzeba wziąć pod uwagę to, że rozwiązanie natury algorytmicznej, które chcemy wymusić, powinno być optymalne wśród możliwych rozwiązań pod względem łącznego czasu jego wymyślania, powiększonego o czas jego ręcznego wykonywania. W przeciwnym przypadku rozwiązujący chętniej wybiorą, albo mniej ciekawą metodę siłową, pozwalającą wyznaczyć wynik za pomocą większej, choć wciąż rozsądnej liczby operacji, albo metody heurystyczne, które działają dokładnie dla wskazanych danych. Innymi słowy, w przypadku takich zadań stosujemy inną definicję złożoności obliczeniowej rozwiązania.

Podobne do powyższego pomysły występują w niektórych konkursach informatycznych. W Polsce najbardziej znany jest konkurs Bóbr (http://www.bobr.edu.pl/), w którym uczestnicy rozwiązują test złożony z krótkich i stosunkowo prostych zadań-łamigłówek. Innymi przykładami są: Olimpiada Informatyczna w RPA[3] oraz Australijski Konkurs Informatyczny.[4] Jednym z głównych celów tych konkursów jest popularyzacja informatyki, w czym korzystają one z paradygmatu *competitive learning*. Nasz pomysł różni się od nich tym, że celuje głównie w popularyzację informatyki wśród uczniów zainteresowanych naukami ścisłymi lub szczególnie utalentowanych w tym kierunku. Ponadto, rozważane przez nas zadania nie mają charakteru konkursowego: mogą być bardziej złożone, wymagać dłuższego kombinowania, zawierać wskazówki. Rezygnacja z formatu konkursowego jest celowa – opieramy się na naturalnym zainteresowaniu uczniów zagadkami i łamigłówkami (na przykład ostatnio coraz bardziej popularnymi łamigłówkami Sudoku i obrazkami logicznymi).

[1] Kubica M. Radoszewski J., Algorithms without Programming, *Olympiads in Informatics* 4, 2010, str. 52-66.

[2] Polecamy portal *Delty* pod adresem http://www.deltami.edu.pl/, zawierający bardzo ciekawe materiały dla uczniów szukających ciekawych zadań i ich rozwiązań, jak również prezentacji zagadnień z obszaru ciekawej matematyki i informatyki.

[3] Merry B., Gallotta M., Hultquist C., Challenges in Running a Computer Olympiad in South Africa, *Olympiads in Informatics* 2, 2008, str. 105-114.

[4] Burton B. A., Encouraging Algorithmic Thinking Without a Computer, *Olympiads in Informatics* 4,2010, str. 3-14.

## 3. Uczenie algorytmiki i programowania od podstaw (*Jakub Łącki*)

Choć pytanie o metody pracy z uczniami wybitnie zdolnymi wydaje się bardzo naturalne, trudno sformułować kategoryczne opinie na ten temat. Po pierwsze, uczniowie wybitnie zdolni, są, niejako z definicji, grupą nieliczną i wszelkie wnioski wyciągane na podstawie doświadczeń w ich nauczaniu są wielce niepewne. Po drugie, uczenie młodzieży algorytmiki rozpoczęło się stosunkowo niedawno i nawet najstarsi nauczyciele mają za sobą zaledwie kilkanaście lat doświadczeń. Wydaje się zatem, że wszelkie metody można zaproponować jedynie na podstawie niewielkiego doświadczenia, popartego intuicją i zdrowym rozsądkiem. Jednak i to może nie być skuteczną metodą. Mimo wszystko spróbuję zaprezentować swój punkt widzenia w tej sprawie, zastrzegam jednak, że będzie to zaledwie efekt moich przemyśleń i skromnych doświadczeń. Oprę się głównie na swoich doświadczeniach z czasów, gdy byłem uczniem. Wprawdzie prowadziłem także zajęcia z młodzieżą w trakcie studiów, jednak były to raczej krótkie obozy lub pojedyncze zajęcia i nie potrafię wyciągnąć zbyt wielu wniosków z kilku takich spotkań.

Zainteresowanie zdolnych uczniów algorytmiką nie jest trudne. Młodzi i utalentowani uczniowie są z reguły chętni do nauki. Programowanie i algorytmika stoją dodatkowo na uprzywilejowanej pozycji, bo stanowią nie tylko wyzwanie naukowe, lecz również poszerzają bardzo praktyczne umiejętności uczniów. Programowanie może być dla uczniów narzędziem, które umożliwi im realizowanie ich kreatywności. Za to algorytmika z pewnością spodoba się zarówno teoretykom, jak i praktykom, bo, szczególnie na podstawowym poziomie, pokazuje, jak w prosty sposób można rozwiązać stosunkowo złożone problemy przy użyciu komputera. Nie sądzę więc, by jakiekolwiek wysublimowane metody były konieczne do zachęcenia uczniów do nauki informatyki. Poważniejszą kwestią wydaje się umożliwienie nauki większej liczbie uczniów. Obecnie taką szansę dostają oni tylko w nielicznych szkołach, dość powiedzieć, że każdego roku w finałach Olimpiady Informatycznej reprezentowane są szkoły z zaledwie około 13 województw, za to z wielu szkół jest po kilku finalistów. W grupie zdolnych uczniów nie powinno być więc problemów ze znalezieniem chętnych do nauki.

Zanim rozpocznie się uczenie algorytmiki, uczniowie muszą opanować umiejętność programowania.[5] Dla utalentowanych uczniów nie powinno być to zbyt trudne. Nauka algorytmiki jest jednak trudniejsza i na niej chciałbym się skoncentrować.

Książki o tematyce algorytmicznej są pisane zazwyczaj z myślą o studentach i zawierają mnóstwo wiedzy teoretycznej. Prawdopodobnie nie trafią do wielu młodszych czytelników i mogą wręcz wytworzyć wrażenie, że cała dziedzina jest bardzo trudna. Zajęcia dla młodzieży powinny rozpocząć się od zupełnie innych tematów niż podręczniki akademickie. Może to być po prostu kontynuacja zajęć o programowaniu, na których po zapoznaniu uczniów z językiem programowania przejść można do rozwiązywania prostych problemów za pomocą komputera. Podręczniki akademickie mogą zaś służyć jako źródło encyklopedyczne.

Początkowa część zajęć będzie prawdopodobnie dla uczestników szczególnie wciągająca. Uczniów powinny zainteresować wyjątkowo proste rozwiązania problemów, których nawet najlepsi z nich nie będą zapewne w stanie rozwiązać samodzielnie. Jako przykład mogą posłużyć: problem plecakowy oraz zadanie znajdowania fragmentu tablicy o maksymalnej sumie. Nie oznacza to, że polecam te tematy na pierwsze zajęcia z algorytmiki. Lepiej rozpocząć od metod, które wszyscy uczniowie w zasadzie znają, czyli np. dodawania pisemnego wielocyfrowych liczb albo prostych sposobów sortowania.

[5] Nie sądzę, by uczenie algorytmiki bez wcześniejszej nauki programowania było skuteczne, bo tym sposobem pomniejszy się grupę potencjalnie zainteresowanych o tych, którzy nastawiają się na praktyczne zastosowania. Jeśli ktoś interesuje się naukami ścisłymi, to niemal na pewno umiejętność programowania będzie mu w życiu przydatna.

Dobierając tematy na zajęcia warto kierować się kryterium praktyczności omawianych metod. Do takich z pewnością należą metody przeszukiwania grafów, których liczne zastosowania można bez trudu dostrzec. Dużo trudniej jest przekonać uczniów o praktyczności stosowania wyrafinowanych struktur danych, choć mogą one być pomocne w praktycznym zrozumieniu złożoności obliczeniowej algorytmu.[6]

Podczas nauki zdolnych uczniów należy unikać podawania im gotowych rozwiązań, warto wyrabiać w nich samodzielność. Przed pokazaniem rozwiązania problemu, można dać uczniom czas na pomyślenie tak, by sami zrozumieli na czym polega trudność, a po zapoznaniu się z rozwiązaniem – by wiedzieli, w jaki sposób „zaatakować" problem. Same algorytmy również lepiej przedstawiać na przykładach, pozostawiając ich implementacje jako pożyteczne ćwiczenie. W szczególności złą praktyką jest pokazywanie gotowych kodów źródłowych. Kod źródłowy to po prostu zapis metody w formalnym języku i nie ułatwia on zrozumienia zawartego w nim pomysłu.

W trakcie rozwoju młodych talentów trzeba też dbać o to, by w miarę poznawania nowych porcji wiedzy, poszerzali oni również umiejętności logicznego myślenia i samodzielnego rozwiązywania problemów. Bardzo ważną rolę odgrywa tu nauczyciel. Mimo, że w Internecie znaleźć można wiele serwisów z zadaniami programistycznymi, początkujący uczeń nie ma możliwości znalezienia czegoś na odpowiednim dla siebie poziomie. Co gorsza, zadania rozwijające pomysłowość giną wśród dziesiątek problemów, które wymagają przede wszystkim dobrych umiejętności programistycznych. Bardzo ważne więc, by nauczyciel zadbał o odpowiedni dobór zadań dla swoich podopiecznych, a najlepiej prowadził własny serwis z zadaniami.

Znalezienie chętnych na zajęcia z informatyki z reguły nie stwarza trudności, jeśli tylko takie zajęcia dobrze rozreklamujemy. Jednak w miarę upływu czasu, każdy następny etap rozwoju wymaga od ucznia dużego wysiłku i wielu godzin poświęconych na naukę. Trzeba o tym pamiętać i zwracać uwagę na motywację uczniów. Ponownie, w przypadku informatyki nie powinno to być trudne. Istnieje całe mnóstwo konkursów programistycznych i algorytmicznych, począwszy od konkursów internetowych, a skończywszy na Olimpiadzie Informatycznej, w której bodaj najważniejszą dla uczniów nagrodą, poza osobistą satysfakcją, jest wolny wstęp na dowolną uczelnię w kraju. Niektóre zawody odbywają się co kilka tygodni i osiąganie w nich coraz lepszych rezultatów to dobry cel dla początkujących programistów i algorytmików. Poza tym, nauka informatyki stwarza świetne szanse zawodowe, które nie są zarezerwowane jedynie dla małej grupy wybitnie uzdolnionych. Z naszego kraju każdego roku dziesiątki osób dostają się na praktyki wakacyjne do renomowanych, światowych firm. Z samego Uniwersytetu Warszawskiego na praktyki do czołowych firm w Stanach Zjednoczonych pojechało w roku 2010 ponad 20 osób. Wiele z nich to byli olimpijczycy. Takie perspektywy z pewnością działają motywująco na młodych ludzi.

## 4. Doskonalenie umiejętności (*Marek Cygan*)

Bazą do poniższych przemyśleń było przygotowywanie studentów UW do konkursów programistycznych, jak również prowadzenie kółka informatycznego w VI LO w Bydgoszczy.

Po osiągnięciu pewnego poziomu wiedzy i umiejętności programistycznych zdecydowanie zwiększają się możliwości samodzielnego rozwoju ucznia. Z uwagi na rozwój technologiczny wiedza wcześniej niedostępna, lub też trudno dostępna w postaci literatury anglojęzycznej, staje się coraz bardziej powszechna za sprawą Internetu. W sieci znaleźć można bardzo wiele serwisów z zadaniami bądź też materiałami, podzielonymi na sekcje tematyczne dotyczące poszczególnych zagadnień algorytmicznych lub ogólniej informatycznych. Należy jednak zaznaczyć, iż większość materiałów,

[6] Uwaga o strukturach danych pochodzi od Krzysztofa Diksa.

które można znaleźć w Internecie, jest dostępna w języku angielskim. Obecnie narzędzia do automatycznego tłumaczenia, pomimo powszechnej dostępności, nie rozwiązują problemu bariery językowej, gdyż automatycznie generowane tłumaczenia nie są wystarczająco dobre przy tłumaczeniu bardziej zaawansowanych tekstów matematycznych i informatycznych. Dlatego warto zachęcić uczniów do nauki języka obcego, co z pewnością będzie przydatną umiejętnością w ich życiu, a jednocześnie umożliwi dostęp do zdecydowanie większej ilości materiałów, które mogą ich zainteresować.

Przy samodzielnej pracy ucznia ważną kwestią jest możliwość uzyskania pomocy w rozwiązywaniu zadań, których uczeń nie umie samodzielnie rozwiązać. Być może zadania te wymagają przysposobienia nieznanej partii materiału, a może uczeń nie zauważył kluczowej w danym zadaniu obserwacji. W przypadku zbioru zadań bez rozwiązań i bez podpowiedzi trudno będzie się uczniowi rozwijać. W takim wypadku ważna jest pomoc nauczyciela lub rozmowa z innymi uczniami, co zwiększa zaangażowanie grupy w dążeniu do wspólnego celu doskonalenia swoich umiejętności.

Kolejnym istotnym elementem nauki jest możliwość sprawdzenia się w konfrontacji z presją oraz innymi uczniami. Rozwiązywanie zadania w warunkach konkursowych istotnie różni się od spokojnego analizowania problemu, dlatego należy umożliwić uczniom sprawdzenie się i nabycie doświadczenia w warunkach zbliżonych do konkursowych. Może się to odbyć za pomocą uczestnictwa w zawodach internetowych bądź poprzez przygotowanie przez nauczyciela zestawu zadań, które uczniowie będą rozwiązywać na czas. Bardzo często zdarza się, że uczniowie z niewielkim (bądź z żadnym) doświadczeniem konkursowym popełniają podstawowe błędy w warunkach konkursowych – zaczynają implementację nieprzemyślanych rozwiązań, nie potrafią kontrolować czasu, przez co zamiast jednego dobrze rozwiązanego zadania, kończą konkurs z dwoma prawie rozwiązanymi zadaniami, co oznacza bardzo małą (bądź też zerową) liczbę punków.

Praca z grupą uczniów różni się od nauczania indywidualnego. W przypadku pracy z grupą uczniów warto rozważyć różne metody zachęcające uczniów do współpracy – wymiany wiedzy i doświadczeń. Jedną z takich metod jest praca w parach nad wspólnym rozwiązaniem zadania oraz jego implementacją. Jeśli uczeń musi wytłumaczyć swój nieczytelny program komputerowy, to najprawdopodobniej następnym razem będzie starał się poprawić strukturę swojego programu oraz wyeliminować złe nawyki, co pozytywnie wpłynie na jego umiejętności programistyczne. Jednakże należy dokonywać podziału na podgrupy w sposób przemyślany. Warto, aby uczeń słabszy miał możliwość współpracy z osobami dużo lepszymi, co pozwoli mu zauważyć swoje braki, jednakże częsta współpraca z uczniami dużo lepszymi może być zniechęcająca dla obu stron. Ponadto w przypadku pracy z grupą uczniów o zróżnicowanym poziomie lub też wieku należy zadbać o zróżnicowany stopień trudności materiału tak, aby mniej doświadczeni uczniowie mieli szansę zrozumieć podstawowe zagadnienia, natomiast najlepsi powinni mieć też możliwość rozwiązywania zadań dodatkowych.

## 5. Prosty język maszynowy pierwszym językiem programowania? (*Tomasz Idziaszek*)

Obecnie większość konkursów algorytmicznych wymaga od uczestników nie tylko umiejętności projektowania efektywnych algorytmów, ale również programowania. Co więcej, biegłe programowanie jest punktem wyjścia do nauki algorytmiki – o ile bowiem sprawne zapisanie mniej efektywnego algorytmu jest zwykle premiowane częściowymi punktami, to nieumiejętność zapisania optymalnego algorytmu już nie. Niestety dość często zakłada się, że skoro uczniowie potrafią napisać kompilujący się program, to jest to równoznaczne z tym, że umieją programować. Nie jest to prawdą w ogólności.

Nauczyciel chcący wykształcić umiejętność programowania u uczniów stoi przed niełatwym wyborem pierwszego języka programowania, z którym uczniowie zetkną się na jego zajęciach.

Poniżej przedstawimy wybór, jakiego dokonał Andrzej Gąsienica-Samek, kiedy prowadził olimpijskie kółko informatyczne w roku 2001, i wymienimy zalety takiego podejścia. Wybór Andrzeja był dosyć niestandardowy, gdyż językiem, którego używał na zajęciach do wprowadzenia młodych adeptów w świat informatyki, był język maszynowy.

Warto zauważyć, że takie podejście ma pewne uzasadnienie historyczne. Zanim pojawiły się kompilatory języków wyższego poziomu, programowanie siłą rzeczy musiało odbywać się w wewnętrznym języku maszyny liczącej. Maszyna taka była w gruncie rzeczy dosyć prosta, tak więc i jej język udostępniał nieduży zbiór prostych operacji. Powodowało to, że złożone obliczenia wymagały długich programów, które były trudne w utrzymaniu. Naturalna skłonność człowieka do myślenia abstrakcyjnego spowodowała opracowanie różnych technik i paradygmatów programowania, które pozwalały mu zwięźlej wyrażać swoje myśli, oraz narzędzi przekształcających programy wyższego poziomu na język maszynowy. Jednak zawsze fundament, na którym opierały się konstrukcje wyższego poziomu, był niezmienny.

Język używany na kółku olimpijskim był wewnętrznym językiem prostej maszyny zaprojektowanej przez Andrzeja. Miała ona niewielką pamięć złożoną z kolejnych komórek, w których można było zapisać liczby całkowite. Kod programu był zapisany w osobnym miejscu. Maszyna udostępniała raptem pięć instrukcji (ustawianie zawartości komórek pamięci, skoki warunkowe i bezwarunkowe, operacje czytania liczb z wejścia i wypisywania liczb na wyjście). Jedynymi udogodnieniami były: możliwość pisania wyrażeń arytmetycznych w postaci znanej uczniom z lekcji matematyki (z dodatkowym operatorem dostępu do komórki pamięci) oraz możliwość definiowania nazwanych stałych (dzięki temu dostęp do zmiennych mógł odbywać się poprzez podanie ich nazwy zamiast numeru komórki pamięci).

Jedną z podstawowych zalet tego języka była jego prostota. Zdecydowana większość współczesnych języków programowania nie ma tej zalety. Chcąc napisać choćby najprostszy program w języku C lub Java, jesteśmy zmuszeni do tłumaczenia dość zawiłej składni i technicznych elementów związanych z dołączaniem potrzebnych bibliotek czy nieintuicyjną obsługą wejścia/wyjścia. Nawet w przypadku języka Pascal musimy wyjaśnić znaczenie typów przy deklaracji zmiennych i niefortunne zasady używania średnika. Próba prześlizgnięcia się nad tymi problemami (mówiąc, że to są rzeczy, których nie trzeba jeszcze rozumieć) może spowodować u uczniów frustrację.

Drugą zaletą było oswajanie uczniów z architekturą komputera. Dzięki temu nie traktowali komputera jako magicznej czarnej skrzynki, ale jako narzędzie, które rozumieją, jak działa. Przy okazji pisania bardziej złożonych programów, uczniowie sami dostrzegali ograniczenia języka maszynowego i to był właściwy moment na wspólne wprowadzenie mechanizmów, które pozwalały im ułatwić sobie pracę. Jeden z takich mechanizmów, którego zrozumienie jest kluczowe w nauce programowania, to rekurencja. Wiele osób rozpoczynających naukę od języka wysokiego poziomu ma kłopot ze zrozumieniem idei rekurencji, która wydaje się im jakąś czarną magią. Na kółku uczniowie pod kierunkiem nauczyciela od podstaw budowali mechanizm stosu używanego do pamiętania informacji związanych z wywoływanymi podprogramami. Dało im to pełne zrozumienie idei tego mechanizmu, który bez kłopotu odkrywali w językach wyższego poziomu.

Dopiero gdy uczestnicy kółka nabrali biegłości w posługiwaniu się językiem maszynowym i poznali implementację podstawowych konstrukcji programistycznych, które udostępniają języki wyższego poziomu, mogli zacząć naukę algorytmiki. Ta była już prowadzona z użyciem języka Pascal, który był jednym z dostępnych języków na Olimpiadzie Informatycznej.

Może nasunąć się pytanie, czy rzeczywiście do sprawnego programowania w języku Pascal jest potrzebna znajomość języka maszynowego? Z przykrością można stwierdzić, że spora liczba studentów informatyki ma mglistą wiedzę na temat działania komputera na poziomie niskim (procesor, programowanie w języku maszynowym), czy nawet średnim (implementacja mechanizmów

dostarczanych przez języki wysokiego poziomu, tj. odśmiecanie pamięci, mechanizmy języków obiektowych). Jednym z efektów tej niewiedzy jest pisanie programów, które są nieefektywne czasowo lub pamięciowo. Studenci bowiem wiedzą, że dany mechanizm działa, ale nie wiedzą jak, wskutek czego nie są w stanie oszacować, jak długo działa i ile pamięci komputera będzie potrzebować. W przypadku uczniów i studentów, którzy startują w konkursach programistycznych może to być poważna przeszkoda. We współczesnym komputerze istnieje bardzo wiele czynników, które mogą zaważyć na czasie wykonania programu, który skądinąd może być oparty na teoretycznie optymalnym algorytmie (wymienić można chociażby koszt różnych mechanizmów wejścia/wyjścia, dostęp do pamięci uwzględniający wpływ pamięci podręcznej i dyskowej, koszt instrukcji skoku w procesorze czy też koszt używanych funkcji bibliotecznych). Aby być w stanie w pełni zrozumieć te zagadnienia, jest potrzebny czas na ich poznanie i oswojenie się z nimi w praktyce. Uczniowie, którzy nie znają podstaw działania komputera, a rekurencję traktują jak „czarną skrzynkę", będą mieć z tym kłopoty. Zwłaszcza uczniów, którzy w przyszłości chcieliby sięgnąć po laury w Olimpiadzie Informatycznej, czy innych konkursach programistycznych, należy od początku rozwijać w kierunku dogłębnego zrozumienia zasad działania komputera.

Materiały z kółka olimpijskiego, o którym piszemy w ostatniej części, są dostępne na stronie http://web.archive.org:
http://web.archive.org/web/20020610104156/kolko.ofek.waw.pl/kasmv1/podstawy.html
http://web.archive.org/web/20030916184656/kolko.ofek.waw.pl/kasmv1/spec.html
http://web.archive.org/web/20031123071251/kolko.ofek.w