Program koła informatycznego w zakresie algorytmiki i programowania, przygotowującego do startu w Olimpiadzie Informatycznej

Informacje o autorze:

- Imię i nazwisko autora: Ireneusz Bujnowski,
- Miejsce pracy: I Liceum Ogólnokształcące im. Adama Mickiewicza w Białymstoku
- Kontaktowy email: ibujnowski@gmail.com
- Telefon: 792 650 890

Krótki opis doświadczenia w pracy z uczniami uzdolnionymi informatycznie:

Jestem nauczycielem informatyki od 30 lat. Od ok. 16 lat pracuję z młodzieżą prowadząc zajęcia z programowania i algorytmiki przygotowując do startu w Olimpiadzie Informatycznej. Od 13 lat nieprzerwanie moi uczniowie uczestniczą w finale OI. W tym okresie zdobyli 27 tytułów laureata, 19 tytułów finalisty z wyróżnieniem oraz 39 tytułów finalisty Olimpiady Informatycznej.
W zawodach międzynarodowych zdobyli łącznie 19 medali: 4 medale w zawodach Międzynarodowej Olimpiady Informatycznej (1 złoty i 3 srebrne), 9 medali Olimpiady Krajów Bałtyckich (3 złote, 5 srebrnych i 1 brązowy), 6 medali Olimpiady Krajów Europy Środkowej (2 złote, 1 srebrny i 3 brązowe).

Główny cel kształcenia w ramach koła, opis głównych efektów uczenia się:

Celem koła jest przygotowanie się do startu w Olimpiadzie Informatycznej oraz lepsze przygotowanie do matury z informatyki. Z uwagi na różny poziom możliwości oraz pracowitości uczniów cel, którym jest start w OI, może być osiągany w ok. 50%. Zakładam też, że cała młodzież w klasach realizujących rozszerzony program informatyki powinna uczyć się programowania od początku klasy I. Efektem wczesnego kształcenia programowania i algorytmiki jest odsetek uczniów przystępujących do matury z informatyki. W mojej szkole jest to ok. 90%, co jest najwyższym wskaźnikiem w województwie podlaskim. Praca z młodzieżą na kołach i warsztatach w moim przypadku daje też inny efekt: prawie wszyscy absolwenci z klas informatycznych wybierają studia informatyczne.

Przedstawiony program nie jest skierowany do nauczycieli, którzy od lat mają finalistów Olimpiady Informatycznej, a raczej do tych nauczycieli którzy chcieliby uczyć programowania wchodząc razem z młodzieżą w trudną sferę algorytmiki.

Opis kompetencji uczniów przystępujących do programu i propozycja weryfikacji ich posiadania:

Proponuję nie wybierać uczniów do kół, każdy uczeń w z inną prędkością przełamuje trudności i w różnym czasie zaczyna programować. Znam przypadki, gdy uczeń w pierwszej klasie nie był w stanie przełamać się, a w trzeciej klasie zostawał finalistą OI. Program koła jest tak skonstruowany, by każdy mógł z sukcesem uczestniczyć co najmniej jeden rok. Daje to dobre podstawy zarówno dla „maturzystów” jak i „olimpijczyków”. Drugi rok, trudniejszy, generalnie przeznaczony jest dla osób najzdolniejszych, ale zachęcam do uczestnictwa wszystkich. Czasem wsłuchanie się w trudniejsze zagadnienia pozwala z innej perspektywy spojrzeć na wcześniej realizowane tematy. Z założenia koło ma aktywizować młodzież, mobilizować do

dodatkowego stałego wysiłku, co daje stały postęp dla wszystkich i umożliwia zdawanie informatyki na egzaminie maturalnym, gdzie ok. 40% punktów to algorytmika i programowanie.

Częstotliwość systematycznych zajęć, czas trwania zajęć:

Z mojego doświadczenia wynika, że w trakcie jednego roku szkolnego można przeprowadzić ok. 20 regularnych zajęć kół zainteresowań. W swoim projekcie zakładam więc pracę w cyklu 2-letnim (lub pracę na 2 poziomach zaawansowania). W każdym roku 20 obowiązkowych zajęć z wykładami realizowanych w 2-godzinnych blokach (zajęcia te powinny wyczerpać wszystkie zagadnienia z Sylabusa oraz pięć 4-godzinnych warsztatów w roku, które poświęcone mogą być na wyrównywanie poziomów uczniów, na wspólne rozwiązywanie zadań (to na początku np. po miesiącu zajęć), czy na zespołowe zawody programistyczne (np. przy okazji Nowego Roku, Dnia Wiosny, święta Szkoły, czy Dnia Dziecka). Daje to w sumie 60 godzin zajęć na każdym poziomie zaawansowania.
W mojej szkole dwa razy w roku odbywają się tygodniowe obozy naukowe oraz 4-godzinne warsztaty-zawody w każdą sobotę na kilku poziomach zaawansowania. Efekt współzawodnictwa i towarzyszących zawodom emocji są bardzo motywujące dla nauczycieli i uczniów. Zdaję sobie jednak sprawę z tego, że na początku bez widocznych efektów zarówno nauczyciele, jak i uczniowie nie poświęcą aż tyle dodatkowego czasu.

Warunki w tym infrastruktura niezbędne do realizacji zajęć (sprzęt, oprogramowanie, zasoby internetowe):

Prowadzenie koła nie wymaga specjalnego sprzętu, czy oprogramowania. Wystarczy dostęp do serwera, na którym nauczyciel ma uprawnienia wyższe niż uczniowie pozwalające na dodawanie zadań, tworzenie zawodów ew. oglądanie i porównywanie kodów uczniów. Jest kilka takich sprawdzarek (o których wiem), które udostępniają takie funkcje. Dobrym pomysłem, choć nie jest to niezbędne, jest posiadanie własnej sprawdzarki. Jest to kłopotliwe w utrzymaniu i nieustannym dostosowywaniu do zmieniającego się oprogramowania serwerowego. Własna sprawdzarka pozwala z czasem dostosować ją do naszych indywidualnych potrzeb i niejako przy okazji zbudować sobie zbiory problemów, zadań i materiałów dopasowanych do takiego serwisu.

Koncepcja i opis innych form kształcenia (praca samodzielna, obozy, grywalizacja itp.):

Nie trzeba chyba nikogo przekonywać jak bardzo istotnym elementem edukacji informatycznej jest rozwiązywanie zadań. Podczas zmagań z problemami młodzież doskonale utrwala podany materiał, ale też często sięga po nieznane dotąd algorytmy i omówienia. Stąd też w pierwszym roku treści zadań powinny odpowiadać wprowadzanym tematom, by w drugim roku nieco się mijać – czasem zadania powinny odnosić się do treści wcześniej realizowanych podczas wykładów, a czasem wręcz wyprzedzać materiał wykładowy, by chętni i najzdolniejsi samodzielnie zapoznawali się z nowymi treściami. Uzupełnieniem części stacjonarnej (wykładowej i warsztatowej) stanowią dwa tygodniowe obozy naukowe. Pierwszy z nich planowany jest na przełomie września i października – bezpośrednio przed pojawieniem się zadań I etapu Olimpiady Informatycznej. Dla klas I jest to sposób na wyrównanie umiejętności, ponieważ część uczniów uczyło się elementów programowania w szkole podstawowej, a także miejsce, gdzie bardzo pomaga się uczniom mającym problemy z początkowymi umiejętnościami związanymi z programowaniem. Dla klas II i III obóz jest

miejscem integracji przed olimpiadą oraz początkiem ciężkiej pracy po „rozleniwiających” wakacjach. Podczas obozów dobrze sprawdzają się tzw. konsultanci – to uczniowie klas starszych, którzy służą pomocą uczniom klas młodszych. Są konsultanci z klas III i ich zadaniem jest pomoc dla uczniów klas II, są też wybrani konsultanci z klas II, którzy z kolei pomagają uczniom klas I.

Drugi obóz organizowany jest przed II etapem OI. Zwykle na przełomie stycznia i lutego. Zwykle też od tego obozu pojawiają się zadania o wiele trudniejsze niż te, które rozwiązywali do tej pory.

Tematyka spotkań – grupa podstawowa i zaawansowana:

Proponowane tematy spotkań koła poziom podstawowy (lub pierwszy rok koła)

1. Zapoznanie ze środowiskiem CodeBlocks. Pierwsze programy w C++.
2. Instrukcje warunkowe oraz wyboru. Testowanie podzielności. Arytmetyka modulo.
3. Instrukcje pętli while oraz for.
4. Tablice jednowymiarowe – deklaracje, liniowe wyszukiwanie informacji w tablicach.
5. Tablice jednowymiarowe – Zliczanie, sumy prefiksowe – jako „pytania o przedziały”
6. Instrukcje „pętli w pętli” – sposoby rozwiązywania zadań z wieloma przypadkami w pojedynczym teście.
7. Tablice jednowymiarowe – sortowanie tablic, elementy złożoności obliczeniowej
8. Zmienne typu string i char – przetwarzanie tekstów.
9. Własna arytmetyka, implementacja wielkich liczb.
10. Funkcje w C++. Parametry funkcji oraz zasięg zmiennych – zmienne lokalne i globalne. Rekurencja, funkcje rekurencyjne.
11. Wyszukiwanie binarne – optymalne wyszukiwanie w tablicy z uporządkowanymi elementami. Biblioteka matematyczna (cmath)
12. Sortowanie z rekurencją, quicksort, mergesort. Metoda „dziel i zwyciężaj”.
13. Algorytm sortowania, funkcja sort z biblioteki algorithm, własna funkcja porównująca sortowane elementy.
14. Wyszukiwanie k-tego elementu w ciągu, najdłuższy podciąg rosnący.
15. Struktury danych w STL – pair, stack, queue, vector.
16. Przegląd algorytmów o różnej złożoności – elementy złożoności obliczeniowej. Koszt zamortyzowany, gąsienica - technika dwóch wskaźników.
17. Testy pierwszości, sito Eratostenesa.
18. Programowanie zachłanne i programowanie dynamiczne. Wybrane algorytmy i problemy rozwiązywane dynamicznie
19. Proste algorytmy geometryczne na płaszczyźnie, wzajemne położenie punktu i prostej (odcinka), punktu i figury płaskiej, przecinanie się odcinków, pole powierzchni wielokątów, otoczka wypukła.
20. Elementy teorii gier – gry kombinatoryczne, gry bezstronne, twierdzenie Spranque’a-Grundy’ego, gry minmax.

Proponowane tematy spotkań koła poziom zaawansowany (lub drugi rok koła):

1. Drzewa binarne, przeglądanie drzew metodami PRE-, IN- oraz POST- order.
2. Kopiec binarny – sortowanie przez kopcowanie.
3. Drzewa binarne implemetowane w tablicach – drzewo licznikowe, drzewo potęgowe.
4. Drzewo przedziałowe implementowane w tablicy.
5. Zaawansowane struktury danych w STL – set, multiset, mapa.

6. Zbiory rozłączne – FIND-UNION
7. Kolejka priorytetowa
8. Grafy – rodzaje i podstawowe własności, implementacja grafów. Przeglądanie grafu „wszerz”.
9. Grafy, przeglądanie grafu metodą „w głąb”, drzewo przeszukiwania DFS.
10. Spójność grafu, spójne składowe, sortowanie topologiczne.
11. Silna spójność grafów skierowanych – algorytmy Kosaraju oraz Tarjana, funkcja low, dwuspójność grafu.
12. Minimalne drzewa rozpinające – algorytm Kruskala.
13. Odległości w grafach I – BFS, algorytm Dijkstry,
14. Odległości w grafach II – algorytmy Belmana-Forda, SPFA,
15. Domknięcie przechodnie grafu, algorytm Floyda-Warshalla
16. Skojarzenia w grafach dwudzielnych.
17. LCA – najniższy wspólny przodek dwóch wierzchołków w drzewie – algorytm z wyszukiwaniem binarnym.
18. Range Minimum Query, algorytmy wyszukiwania wartości minimalnej w przedziale, zastosowanie RMQ do znajdowania LCA.
19. Przegląd algorytmów tekstowych – algorytm KMP, wyszukiwanie promieni palindromów w tekście.
20. Przegląd algorytmów tekstowych – rozwiązywanie wybranych problemów tekstowych przy pomocy hashowania.

Szczegółowy opisy poszczególnych zajęć w I roku działalności koła (40 godzin)

Zajęcia 1 (2 godziny)

Temat: Zapoznanie ze środowiskiem CodeBlocks. Pierwsze programy w C++.

Treści z sylabusa:

Matematyka:	Programowanie:	Algorytmika
Reprezentacja liczb w komputerze, zbiory	obsługa środowiska programistycznego, zmienne i typy zmiennych (całkowite i rzeczywiste), wejście i wyjście -formatowanie wyjścia, operatory arytmetyczne, działanie automatycznego systemu oceniającego	Algorytm liniowy

Czynności nauczyciela:

- prezentuje sposób zainstalowania środowiska do programowania w C++,
- opisuje strukturę programu w C++,
- omawia sposób wypisywania komunikatów tekstowych z poziomu C++,
- wyjaśnia typy danych oraz pojęcie zmiennej,
- pokazuje na ekranie (tablicy) sposoby wczytywania oraz wypisywania wartości liczbowych,
- opisuje sposób zapisu i działania operatorów arytmetycznych,
- demonstruje sposób działania systemu oceniającego rozwiązania uczniowskie.

Przykładowe zadania do rozwiązania na lekcji:

- Wypisz na ekranie napis „Hello World!”,
- Wczytaj trzy liczby i wypisz je na ekranie w zmienionej kolejności
- Wczytaj dwie liczby całkowite. Wypisz wynik dodawania, odejmowania, mnożenia, dzielenia całkowitoliczbowego oraz reszty z dzielenia

Przykładowe zadania domowe:

- Wczytaj liczbę n i wypisz sumę liczb od 1 do n, na lekcji podana jest anegdota z Gaussem i wzór sumy liczb naturalnych od 1 do 100,
- Wczytaj dwie liczby całkowite r oraz h. Oblicz i wypisz pole powierzchni całkowitej oraz objętość stożka, walca i kuli z dokładnością do 2 miejsc po przecinku.

Zadania na sprawdzarce:

Proste zadania sprawdzające umiejętność napisania bezbłędnego programu wypisującego teksty lub wartości liczbowe.

Uwagi do realizacji:

Zajęcia te są bardzo istotne. Młodzież nie może spotkać porażka zniechęcająca do udziału w dalszej części kursu programowania i algorytmiki, stąd ważne jest bardzo dokładne, wolne wyjaśnienie wszystkich problemów, które mogą zaistnieć przy instalacji, podczas pisania pierwszych programów, czy przy pierwszych zgłoszeniach rozwiązań do systemu sprawdzającego.