Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

INFORMATYKA

Lekcja pokazowa/otwarta – SP

Scenariusz lekcji

Temat:
Wyszukiwanie i porządkowanie informacji

Maciej M. Sysło
syslo@ii.uni.wroc.pl
http://mmsyslo.pl

Wrocław miasto spotkań
DOLNY ŚLĄSK
Dolnośląska Biblioteka Pedagogiczna we Wrocławiu
Uniwersytet Wrocławski
GMINA DŁUGOŁĘKA
Wrocławskie Centrum Doskonalenia Nauczycieli
Dolnośląska Szkoła Ćwiczeń

1. Obszar informatyki i klasa, w której będą prowadzone zajęcia

Obszar informatyki: Algorytmika i programowanie – szkoła podstawowa

Klasa: wybór klasy zależy od nauczyciela, który powinien wziąć pod uwagę, czy uczniowie są przygotowani do zajęć na zaproponowany temat.

Przygotowanie uczniów do lekcji: uczniowie wiedzą, co to jest algorytm, układali już bowiem algorytmy dla różnych sytuacji problemowych. Przydatna też będzie umiejętność budowania opisów algorytmu w postaci schematów blokowych. Uczniowie znają już środowisko programistyczne języka Python lub C++, potrafią stosować instrukcje iteracyjne i warunkowe.

Wyszukiwanie i porządkowanie to jedne z najczęściej wykonywanych operacji w komputerze, w tym również w programach użytkowników. Każdy uczeń powinien więc poznać idee podstawowych metod/algorytmów realizacji tych operacji, nie tylko na potrzeby zajęć informatycznych, ale także jako metody stosowane poza obliczeniami komputerowymi. Ta lekcja jest wprowadzeniem do tematyki wyszukiwania i porządkowania. Następne lekcje powinny być poświęcone wyszukiwaniu wśród informacji uporządkowanych oraz innym bardziej efektywnym algorytmom porządkowania.

2. Zagadnienia metodyczne stanowiące podstawę przygotowania lekcji w zakresie rozwijania kompetencji metodycznych

Metodyczną podstawę realizacji lekcji na zaplanowany temat, a faktycznie każdej lekcji realizującej obowiązującą podstawę programową informatyki, stanowią fundamenty kształcenia informatycznego w polskich szkołach wynikające z konstrukcji podstawy programowej, miejsca i roli programowania, powiązań informatyki z innymi dziedzinami oraz metodyki kształcenia. Tymi fundamentami są:

1. **kolejność** celów ogólnych kształcenia – kształcenie w zakresie abstrakcyjnego i algorytmicznego myślenia powinno poprzedzać naukę programowania i korzystanie z aplikacji komputerowych
2. **spiralność** kształcenia wokół tych samych informatycznych pojęć, metod i celów przez wszystkie lata w szkole od pierwszej po ostatnią klasę
3. **myślenie komputacyjne** – jednym z głównych celów kształcenia informatycznego jest rozwój sposobów myślenia angażowanego w formułowanie problemu i przedstawianie rozwiązań w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać; jest to sedno podejścia informatycznego do rozwiązywania problemów
4. **programowanie** – etap kreatywnego rozwiazywania problemów z wykorzystaniem komputera – *learning by doing,* konstrukcjonizm
5. **informatyka w swoich zastosowaniach** – nauczanie przez rozwiązywanie problemów z różnych dziedzin
6. **metoda projektów** – zalecana w podstawach wszystkich przedmiotów praca w zespołach – informatyk nie pracuje dzisiaj sam, samodzielnie, tylko w zespole

Te ogólne wskazania zostały uwzględnione w tej propozycji scenariusza i powinny zostać uwzględnione i zrealizowane przez nauczyciela, który ma poprowadzić pokazową/otwartą lekcję według tego scenariusza.

Wyszukiwanie i porządkowanie informacji, w szczególności liczb, to jedne z najczęściej wykonywanych operacji w komputerze i ściśle ze sobą powiązane, gdyż szybkość wyszukiwania w komputerze zawdzięcza się przede wszystkim temu, że informacje w komputerze są uporządkowane. Algorytmy porządkowania należą do klasyki informatyki i kształcenia informatycznego na każdym poziomie edukacji. Ta lekcja jest wprowadzeniem do tej tematyki, jednocześnie przejście od prostej metody wyszukiwania do porządkowania jest **ilustracją spiralnego rozwoju** umiejętności uczniów.

Zgodnie z zalecaną powyżej metodyką uczniowie najpierw poznają odpowiednie algorytmy – znajdowania najmniejszego lub największego elementu i zastosowania tej operacji w metodzie porządkowania – a następnie, gdy zrozumieją, na czym polegają te algorytmy, o czym mogą się przekonać, wykonując je odręcznie na papierze, czyli postępując zgodnie z zasadą metodyczną

Zanim zaczniesz programować,
wpierw pomyśl komputacyjnie.

Następnie przystępują do ich programowania dla komputera. Na końcu testują poprawność programów i ewentualnie poprawiają znalezione błędy.

3. Temat lekcji: Wyszukiwanie i porządkowanie informacji

4. Treści nauczania

Z podstawy dla klas IV–VI:

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
   1) tworzy i porządkuje w postaci sekwencji (liniowo) […] informacje […]
      a) […]
      b) obiekty z uwzględnieniem ich cech charakterystycznych;
   2) formułuje i zapisuje w postaci algorytmów polecenia składające się na:
      a) […]
      b) osiągnięcie postawionego celu, w tym znalezienie elementu w zbiorze nieuporządkowanym[…], znalezienie elementu najmniejszego i największego,
      c) […]
   3) w algorytmicznym rozwiązywaniu problemu wyróżnia podstawowe kroki: określenie problemu i celu do osiągniecia, analiza sytuacji problemowej, opracowanie rozwiązania, sprawdzenie rozwiązania problemu dla przykładowych danych, zapisanie rozwiązania w postaci schematu lub programu.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i zapisuje w […] języku programowania:
      a) […] proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych […],
      b) […]
   2) testuje na komputerze swoje programy pod względem zgodności z przyjętymi założeniami i ewentualnie je poprawia, objaśnia przebieg działania programów;

Z podstawy dla klas VII–VIII:

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
   1) formułuje problem w postaci specyfikacji (czyli opisuje dane i wyniki) i wyróżnia kroki w algorytmicznym rozwiązywaniu problemów. Stosuje różne sposoby przedstawiania algorytmów, w tym w języku naturalnym, w postaci schematów blokowych, listy kroków;
   2) stosuje przy rozwiązywaniu problemów podstawowe algorytmy:
      a) […]
      b) wyszukiwania i porządkowania: wyszukuje element w zbiorze […] nieuporządkowanym oraz porządkuje elementy w zbiorze metodą przez proste wybieranie […];
   3) […]
   4) rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów;
   5) prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice. W szczególności programuje algorytmy z działu I pkt 2;

Pojęcia i metody informatyczne, którym są poświęcone zajęcia:

- specyfikacja sytuacji problemowej, dla której ma być zaprojektowany algorytm
- algorytm dla danej specyfikacji
- algorytmy znajdowania najmniejszego lub największego elementu w ciągu
- porządkowanie elementów ciągu przez wybór
- komputerowa realizacja algorytmu

- testowanie programów
- zgodność komputerowej realizacji algorytmu ze specyfikacją problemu rozwiązywanego przez ten algorytm

5. Cele/osiągnięcia ucznia

Uczeń:

- tworzy specyfikację problemu do jego luźnego opisu
- posługuje się schematem blokowym w trakcie projektowania algorytmu
- znajduje najmniejszy lub największy element w zbiorze/ciągu
- porządkuje ciąg, stosując metodę przez wybór
- korzysta z instrukcji iteracyjnych i warunkowych w Pythonie
- programuje funkcję i stosuje ją w innych programach
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych
- operuje konstrukcjami programistycznymi potrzebnymi do realizacji wybranych kroków algorytmu

6. Metody pracy z uczniem, formy pracy uczniów

- **Słowna i oglądowa**: dyskusja w klasie, nauczyciel dostarcza różne przykłady sytuacji problemowych, w których należy znajdować szczególne elementy i/lub je porządkować. Jest to wprowadzenie do dyskusji między uczniami, jak znajdować rozwiązania w poszczególnych sytuacjach, moderowana przez nauczyciela ma doprowadzić do ogólnego sformułowania odpowiednich algorytmów; nauczyciel posługuje się różnymi artefaktami, związanymi z konkretnymi sytuacjami.
- **Czynna**: uczniowie najpierw dyskutują z nauczycielem i między sobą nad rozwiązaniami stawianych problemów, później indywidualnie ćwiczą i testują poznane algorytmy odręcznie na wielu przykładach, a na końcu piszą odpowiednie programy w języku Python, debagują je i testują na tych samych przykładach, które stosowali w obliczeniach odręcznych; programowanie może przebiegać w parach.
- **Aktywizująca**: nauczyciel może przydzielać uczniom wcześniej przygotowane zestawy danych dla algorytmów i programów, które mogą służyć do głębszego prześledzenia działania algorytmów; uczniowie mogą aktywizować się nawzajem, np. pracując w parach – programowanie w parach jest powszechnym podejściem przy tworzeniu oprogramowania, pojawia się również w szkołach; w parach uczniowie mogą nawzajem się aktywizować.

7. Środki dydaktyczne wykorzystywane przez nauczyciela i przez uczniów

Nauczyciel powinien przygotować opisy różnych sytuacji problemowych, w których należy znaleźć wyróżnione elementy i je uporządkować. Powinien być również przygotowany do dyskusji na temat możliwych metod i algorytmów wyszukiwania i porządkowania.

Poleca się zainstalowanie aplikacji[1] *Maszyna Sortująca* na każdym komputerze w pracowni, której archiwum znajduje się w załączeniu do tego dokumentu. Ta aplikacja jest programem dydaktycznym, służącym do wykonywania eksperymentów z algorytmem znajdowania minimum w ciągu i algorytmem porządkowania przez wybór. Uczeń może konfigurować tę aplikację do własnych celów, wybierać różne dane, tryb pracy (krokowy lub ciągły), rejestrować przebieg działania algorytmu.

Do demonstracji działania porządkowania przez wybór nauczyciel i uczniowie mogą posłużyć się aplikacją w sieci pod adresem https://www.toptal.com/developers/sorting-algorithms – należy wybrać kolumnę *Selection*, w której z kolei znajdują się cztery rodzaje danych do uporządkowania (te dane to odcinki porządkowane co do długości, od najkrótszego u góry).

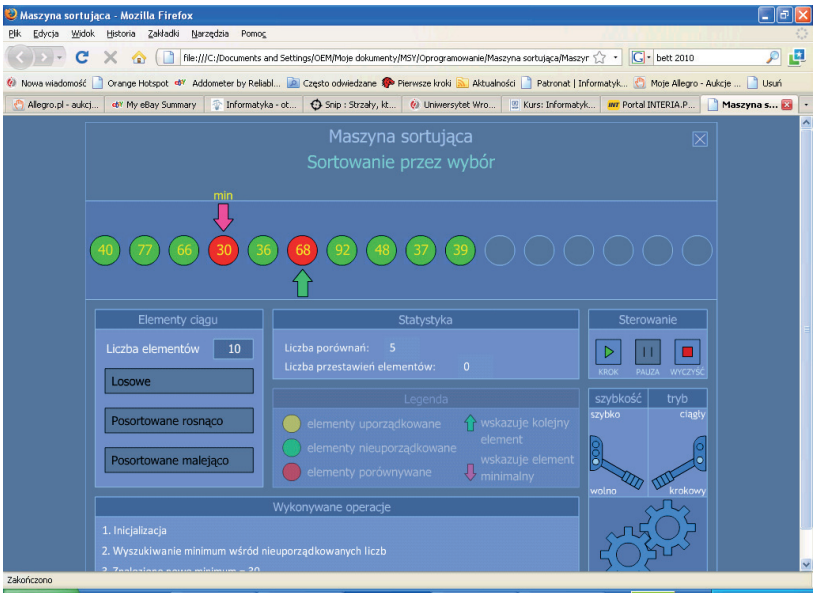
[1] Aplikacja ta pochodzi z podręcznika: E. Gurbiel, G. Hardt-Olejniczak, E. Kołczyk, H. Krupicka, M.M. Sysło, Informatyka, WSiP, Warszawa 2009.

W załączeniu jest również krótka prezentacja dotycząca wyszukiwania i porządkowania – polecamy ją w opisie przebiegu lekcji.

Już nie jako środek dydaktyczny, ale jako źródło wzorcowych rozwiązań polecamy książkę M.M. Sysło *Algorytmy* [4], a w niej cały rozdz. 5 poświęcony wyszukiwaniu, w szczególności znajdowaniu najmniejszego/największego elementu w ciągu, a także rozdz. 6, punkt 6.2 poświęcony porządkowaniu przez wybór.

8. Przebieg lekcji

1. *Nauczyciel* informuje uczniów, jaki jest temat lekcji i co będzie celem ich działań: poznanie algorytmów szukania elementu najmniejszego i wykorzystanie tego algorytmu do uporządkowania wszystkich elementów. Końcowym efektem mają być programy komputerowe realizujące te algorytmy.
2. *Wprowadzenie do zajęć*. Nauczyciel inicjuje dyskusję na temat wyszukiwania i porządkowania informacji, podając wiele przykładów z życia i z otoczenia uczniów. Uczniowie proponują sposoby znajdowania wyróżnionych elementów. Następnie dyskusja jest sprowadzona do sposobu znajdowania najmniejszego, największego, najlepszego, naj- elementu w ciągu (liczb), w zbiorze.
   W prezentacji na slajdzie nr 2 są wymienione różne sytuacje problemowe, w których należy znaleźć wyróżniony element. Dyskusja powinna skupić się na sposobach porównywania elementów i jak wykonać to porównanie, np. za pomocą komputera. Nie wszystkie porównania da się tak wykonać.
   Dla ciekawskich – co to jest *metoda spaghetti* znajdowania najwyższego ucznia w klasie? Czy może być przydatna, by znaleźć najniższego ucznia w klasie?
3. *Wyjaśnienie nauczyciela*. Nauczyciel wyjaśnia, co to jest specyfikacja sytuacji problemowej. Jest to etap abstrakcyjnego myślenia uczniów – nie jest istotne, jaki charakter mają elementy zbioru, mamy je porównywać i znaleźć ten naj-. **Specyfikacja** składa się z opisu danych i opisu wyniku, jaki mamy otrzymać na podstawie tych danych. Zatem w rozważanych sytuacjach danymi są elementy wybranego charakteru, a wynikiem element naj-, który należy znaleźć wśród nich. Jak widać, należy jeszcze określić kryterium, według którego będą porównywane elementy – wzrost uczniów, data urodzenia, wielkość liczby.
4. *Indywidualna praca uczniów*. Każdy uczeń pisze specyfikację problemu dla wybranego rodzaju elementów.
5. *Wspólna praca uczniów*. Kolejny etap, to zaprojektowanie algorytmu znajdowania elementu naj- w ciągu. Można zacząć od przykładu: w szeregu, twarzą do sali, staje grupa uczniów w losowej kolejności. Jeden z uczniów, stosując systematyczne przeglądanie uczniów, ma znaleźć najniższego wśród nich. Podchodzi do pierwszego, prosi go, by wystąpił i zaczął przeglądanie kolejnych uczniów. Wybrany uczeń podchodzi do kolejnego ucznia i jeśli jest on niższy od niego, to wraca na swoje miejsce, a ten niższy od niego kontynuuj to postępowanie aż do osiągnięcia końca szeregu.
   Podobne postępowanie proponujemy, by znaleźć najmłodszego ucznia. W tym przypadku ten wybrany uczeń przechodzi szereg uczniów, pytając kolejnych o datę urodzenia. Na kartce koryguje datę, gdy pojawia się wcześniejsza, jednocześnie zapisuje imię ucznia przy wpisywanej dacie urodzenia.



6. *Indywidualna praca uczniów.* Teraz każdy uczeń otwiera na komputerze aplikację *Maszyna Sortująca* (patrz slajd nr 4) i wielokrotnie wykonuje znajdowanie najmniejszego elementu w ciągu liczb[2]. Dla lepszego skupienia uwagi wybiera tryb krokowy algorytmu. Demonstrację przegląda dla różnych typów ciągów: losowych, uporządkowanych i odwrotnie uporządkowanych – algorytm powinien być uniwersalny i działać poprawnie na dowolnym ciągu liczb.
7. *Indywidualna praca uczniów.* Po zastosowaniu w tych przykładach liniowego algorytmu wyszukiwania najmniejszego elementu w ciągu uczniowie nie powinni mieć kłopotu z naszkicowaniem schematu blokowego tego algorytmu (patrz slajd nr 5 w załączonej prezentacji), zanim przejdą do jego programowania. Wykonują ten schemat odręcznie w zeszycie. Testują schemat algorytmu na wielu danych.
8. *Wspólna praca uczniów. Intelektualny relaks* – nauczyciel inicjuje dyskusję na temat efektywności liniowego algorytmu.
   Pierwsze pytanie: ile porównać wykonuje ten algorytm? Uczniowie nie powinni mieć kłopotów z policzeniem, że wykonywanych jest n–1 porównań, gdy ciąg ma długość n – z wyjątkiem pierwszego elementu, od którego zaczynamy szukanie, porównywany jest każdy inny.
   Drugie pytanie. Czy tego nie można wykonać szybciej, czy najmniejszego elementu w ciągu n liczby nie można znaleźć za pomocą mniej niż n–1 porównań?
   Podpowiedź nauczyciela. Zamiast szukania minimum wśród liczb, zorganizujmy turniej tenisowy (lub w ping-ponga), by znaleźć najlepszego gracza. Zwykle taki turniej jest rozgrywany metodą pucharową – przegrywający odpada. Na slajdzie nr 7 w prezentacji jest pokazany przykładowy turniej i okazuje się, że w turnieju 8 graczy trzeba rozegrać 7 meczów, by wyłonić zwycięzcę.
   Nauczyciel prosi uczniów, by potwierdzili tę obserwację, rysując drzewo turnieju dla innej liczby graczy w turnieju.
   Podsumowanie tego wątku. Nauczyciel przytacza przepiękny argument Hugona Steinhausa (jest zapisany na slajdzie nr 8), że rzeczywiście potrzeba przynajmniej n–1 porównań (meczów), by znaleźć najmniejszą liczbę wśród n liczb (zwycięzcę w turnieju n graczy).
9. *Indywidualna praca uczniów.* Uczniowie zapisują teraz w Pythonie poznany algorytm znajdowania najmniejszego elementu w ciągu. Dane do tego programu mogą być albo umieszczone wcześniej w liście, albo wczytywane z klawiatury.

Porządkowanie przez wybór

Druga część zajęć jest poświęcona wyprowadzeniu metody porządkowania przez wybór. Jednocześnie ma to być przykład spiralnego rozwinięcia umiejętności z pierwszej części i posłużenia się tym wcześniej stworzonym fragmentem oprogramowania.

10. *Dyskusja, zainicjowana przez nauczyciela pytaniem*: W jaki sposób użyć algorytmu znajdowania najmniejszego elementu, by uporządkować ciąg liczb?
    Ewentualna podpowiedź: Jaki element znajduje się na początku ciągu uporządkowanego?
    Odpowiedź na to pytanie uczniowie znajdą szybko, kontynuując obserwowanie w działaniu aplikacji *Maszyna Sortująca*, po znalezieniu przez nią pierwszego najmniejszego elementu – polecamy slajdy nr 9 i 10.
    Polecamy również prześledzić działanie porządkowania przez wybór w aplikacji: https://www.toptal.com/developers/sorting-algorithms dla wszystkich rodzajów porządkowanych ciągów (należy wybrać kolumnę *Selection*). Widać, jak kolejne coraz większej długości odcinki są przenoszone ku górze.
11. *Indywidualna praca uczniów.* Uczniowie przechodzą do programowania algorytmu porządkowania przez wybór. Uruchamiają swoje programy w Pythonie, ewentualnie debagują je i testują ich działanie na różnych ciągach danych.

[2] Znajdowanie minimum w ciągu jest pierwszym etapem działania tej aplikacji, która w kolejnych etapach porządkuje ciąg metodą przez wybór, co jest dalszym ciągiem lekcji.

12. *Bardziej zaawansowane zadanie.* Większość uczniów nie powinna mieć problemu z napisaniem programu porządkowania ciągu przez wybór, w którym jako funkcja jest wykorzystany algorytm znajdowania najmniejszego elementu w ciągu. Świadczyć to będzie o spiralnym rozwijaniu swoich możliwości programistycznych.
13. Jeszcze bardziej zaawansowanym zadaniem dla uczniów jest policzenie, ile porównań i przestawień elementów jest wykonywanych w porządkowaniu przez wstawianie dla ciągu *n* danych, których jest *n*. Obliczenia są zilustrowane na slajdzie nr 11, odsyłamy także do punktu 6.2 w książce M.M. Sysło. *Algorytmy*).
14. *Uwaga metodyczna.* KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie, który znajduje najmniejszy element w ciągu, oraz innym programem, który porządkuje ciąg przez wybór.
15. *Dyskusja podsumowująca.* Zajęcia kończą się dyskusją na temat wyszukiwania i porządkowania informacji oraz znaczenia tych operacji w codziennych sytuacjach problemowych.
    Innym wątkiem w dyskusji może być strategia rozwiązywania problemów, w której jeden algorytm jest użyty jako istotny krok w innym algorytmie. Warto, by uczniowie byli tego świadomi, gdyż ta strategia jest powszechnie stosowana w rozwiązywaniu problemów niemal w każdej dziedzinie i nie tylko za pomocą komputerów.

Ocena pracy uczniów

Ocenianie powinno być motywujące dla uczniów kolejno przydzielanymi zadaniami do wykonania odręcznie na etapie poznawania algorytmów. Zadania są przydzielane wszystkim uczniom, ale zakres ich wykonania może być różny, może wymagać pomocy innego ucznia lub wsparcia przez nauczyciela.

Ocena postępów w programowaniu powinna uwzględniać zakres samodzielności w programowaniu, ale o tym uczniowie powinni być informowani na każdych zajęciach informatycznych. Taki system oceniania może dopingować uczniów do podejmowania coraz odważniejszych prób samodzielnego programowania, bez pomocy ucznia w parze, bez wsparcia nauczyciela i bez korzystania z pomocy „z sieci", czyli korzystania z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych.

Uwaga! Niestety, zmorą zadań programistycznych (np. domowych) jest obecnie nagminne korzystanie przez uczniów z kodów dostępnych w sieci (niezliczona liczba kodów w każdym języku programowania dla każdego zadania). Nauczyciel powinien wziąć to pod uwagę i wypracować własny system oceniania zadań programistycznych.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Naturalnym poszerzeniem tematyki tej lekcji są zajęcia poświęcone innym algorytmom porządkowania – w podstawie programowej wymienia się jeszcze porządkowanie przez zliczanie, przez (binarne) wstawianie, przez scalanie i porządkowanie szybkie.