Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Wzorcowe materiały dydaktyczne w zakresie:

INFORMATYKA

POZIOM – **SZKOŁA PODSTAWOWA**

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

Spis treści

1. Wprowadzenie

Zaproponowane tutaj materiały szkoleniowe w zakresie informatyki są realizacją podstawy programowej informatyki, przy tym uwzględniono w nich fundamentalne założenia, które zostały przyjęte w konstrukcji i zawartości tej części podstawy i wynikające z nich wskazania metodyczne dla realizacji kształcenia informatycznego w całym procesie kształcenia, od pierwszej po ostatnią klasę w szkole. Te założenia są krótko przytoczone w rozdz. 2, a szczegółowo zostały opisane w Załączniku 2.

Przedstawiono po kilkanaście jednostek lekcyjnych dla poziomów kształcenia podstawowego i ponadpodstawowego. Odnoszą się one w większości do podstawowych działów kształcenia informatycznego – do **Algorytmiki** i do **Programowania**.

W każdej jednostce uwzględniono fundamentalne założenia opisane w rozdz. 3. W szczególności programowanie następuje po etapie analizy sytuacji problemowej i po zaprojektowaniu dla niej algorytmu – w tym procesie kształtowane jest myślenie komputacyjne u uczniów. Sytuacje problemowe odnoszą się do różnych dziedzin i różnych zastosowań. Kolejne jednostki tematyczne ilustrują również spiralny rozwój u uczniów: (1) rozumienia pojęć informatycznych, (2) znajomość algorytmów rozwiązywania coraz bardziej złożonych problemów oraz umiejętności wykorzystania coraz bardziej zaawansowanych możliwości technologii: (1) urządzeniami (komputerami, tabletami, robotami), (2) gotowym oprogramowaniem (sterującym robotami, oprogramowaniem edukacyjnym, pakietem biurowym), (3) systemami programowania (Godzina kodowania, Scratch, Python).

Niemal każda jednostka zawiera przynajmniej jedno zadanie programistyczne dla uczniów. Obecnie w sieci można znaleźć program niemal dla każdego zadania, uczniowie znakomicie o tym wiedzą, nauczyciele również. Proponuje się tutaj, by w przypadku zadań programistycznych nauczyciel motywował uczniów do własnego wysiłku, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań znalezionych w sieci lub zapożyczonych od innych uczniów. Podczas zajęć nauczyciel powinien bacznie diagnozować, w jaki sposób pracują uczniowie. Oczywiście to nie wyklucza innej umiejętności uczniów, którą także należy kształcić, i każdego informatyka – umiejętności korzystania z istniejących rozwiązań programistycznych. Dzisiaj nikt nie pisze profesjonalnych rozwiązań od zera.

2. Podstawowe założenia

Nowa podstawa programowa kształcenia informatycznego, czyli przedmiotu informatyka, i edukacji informatycznej w nauczaniu wczesnoszkolnym w klasach 1–3 została wprowadzona do szkół podstawowych od 2017 roku, a od 2019 roku obejmuje również szkoły ponadpodstawowe, a więc licea, technika i szkoły branżowe. W Załączniku 1 znajdują się wypisy z podstawy programowej dotyczące przedmiotu informatyka dla wszystkich etapów edukacyjnych.

Podstawa programowa informatyki jest oparta na kilku fundamentach, które powinny być uwzględnione w jej realizacji i stanowić jednocześnie fundamenty kształcenia informatycznego w polskich szkołach. Wynikają one m.in. z konstrukcji podstawy programowej, miejsca i roli programowania, powiązań informatyki z innymi dziedzinami oraz sugerowanej metodyki kształcenia. Oto te fundamenty:

- kolejność celów ogólnych kształcenia – kształcenie w zakresie logicznego, abstrakcyjnego i algorytmicznego myślenia zostało umieszczone w podstawie programowej w pierwszym punkcie jako najważniejsze, przed programowaniem i korzystaniem z aplikacji komputerowych,
- spiralność – w podstawie programowej przyjęto identyczne ogólne cele kształcenia dla wszystkich etapów edukacyjnych, sugerując w ten sposób spiralny rozwój uczniów wokół tych samych celów przez wszystkie lata w szkole, od pierwszej po ostatnią klasę,
- myślenie komputacyjne – jednym z głównych celów edukacji informatycznej jest rozwój sposobów myślenia angażowanego w formułowanie problemu i przedstawianie jego rozwiązania w taki sposób, aby komputer – człowiek lub maszyna – mógł skutecznie je wykonać; jest to sedno podejścia informatycznego do rozwiązywania problemów
- programowanie – etap kreatywnego rozwiązywania problemów – *learning by doing* – konstrukcjonizm – dialog z komputerem,
- informatyka w swoich zastosowaniach – nauczanie przez rozwiązywanie problemów z różnych dziedzin,

- metoda projektów – zalecana w podstawach wszystkich przedmiotów, praca w zespołach – informatyk nie pracuje dzisiaj sam.

Proponowane w tym dokumencie materiały szkoleniowe uwzględniają wspomniane fundamentalne założenia przyjęte w konstrukcji podstawy programowej informatyki i wynikające z nich wskazania dla realizacji kształcenia informatycznego w całym procesie kształcenia, od pierwszej po ostatnią klasę w szkole. Szczegółowe omówienie realizacji podstawy programowej informatyki można znaleźć w pracach autora prezentowanych na kolejnych konferencjach „Informatyka w Edukacji", X–XVI. Tekst pracy *Informatyka – fundamenty wdrożenia*, w której szczegółowo omówiono fundamenty wdrożenia, jest załącznikiem do tych materiałów.

Fundamentalne założenia przyjęte w konstrukcji i realizacji podstawy programowej informatyki w polskim systemie oświaty zostały dostrzeżone i docenione w międzynarodowym środowisku specjalistów w obszarze edukacji informatycznej skupionych w grupie roboczej IFIP TC 3, jak również w wielu krajach, stanowiąc propozycję dla lokalnych rozwiązań.

3. Dodatkowe materiały

W tym opisie materiałów szkoleniowych wykorzystano wiele zasobów, w większości utworzonych przez autora tych materiałów.

Książki, podręczniki

1. E. Gurbiel, G. Hardt-Olejniczak, E. Kołczyk, H. Krupicka, M.M. Sysło, *Nauka z komputerem. Książka dla ucznia gimnazjum*, WSiP, Warszawa 2001. Mogę udostępnić elektroniczną wersję tej książki wraz z poradnikiem dla nauczycieli.
2. E. Gurbiel, G. Hardt-Olejniczak, E. Kołczyk, H. Krupicka, M.M. Sysło, *Informatyka to podstawa. Zakres podstawowy dla szkół ponadgimnazjalnych*, WSiP, Warszawa 2012. Mogę udostępnić elektroniczną wersję tej książki wraz z poradnikiem dla nauczycieli.
3. M.M. Sysło, *Piramidy, Szyszki i inne konstrukcje programistyczne*, Helion, Gliwice 2015. Wydanie nieco zmienione w porównaniu z wersją wydawaną wcześniej przez WSiP.
4. M.M. Sysło, *Algorytmy*, Helion, Gliwice 2016. Zmienione i uzupełnione w stosunku do wersji wydawanej przez WSiP – zamiast schematów w programie ELI wprowadzono programy w języku Python obok programów w języku Pascal.
5. M.M. Sysło, *Myślenie komputacyjne w praktyce edukacyjnej*, PWN, Warszawa 2020 (w przygotowaniu).

Inne teksty

6. Wypisy z podstawy programowej informatyki – Załącznik nr 1.
7. M.M. Sysło, Jak myśleć komputacyjnie, Materiały Konferencji „Informatyka w Edukacji, XV", UMK, Toruń 2018 – Załącznik nr 2.
8. M.M. Sysło, Informatyka – fundamenty wdrożenia, Materiały Konferencji „Informatyka w Edukacji, XVI", UMK, Toruń 2019 – Załącznik nr 3.

Zasoby elektroniczne

9. *Maszyna sortująca*[1] – archiwum w załączeniu

Serwisy internetowe

10. Strona domowa autora: http://mmsyslo.pl.
11. Środowisko łamigłówek i kursów informatycznych – http://code.org[2]. Jest to bardzo bogaty portal, pełen materiałów do zajęć informatycznych z uczniami w każdym wieku. Nauczyciel powinien wcześniej zapoznać się z tym portalem i dokonać swojego wyboru. W tych materiałach odwołujemy się tylko do niektórych łamigłówek.
12. Środowisko programowania w języku Scratch – do zainstalowania po pobraniu z sieci. Może być wykorzystywane on-line i off-line. W tej drugiej wersji jest dostęp do milionów projektów.
13. Środowisko programowania w języku Python v. 3.x – do zainstalowania po pobraniu z sieci.

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

[2] Autor zajmuje się przekładem środowiska code.org na język polski.

4. Schemat opisu jednostki tematycznej

Wszystkie jednostki lekcyjne w dalszej części materiałów są opisane według jednolitego schematu przedstawionego poniżej.

Temat jednostki:

Poziom kształcenia

Poziom kształcenia (szkoła podstawowa, edukacja wczesnoszkolna, klasa). Klasa nie musi być pojedyncza – dopiero na podstawie dalszej części konspektu (przygotowania uczniów, zapisów podstawy) nauczyciel może wybrać te zajęcia dla konkretnej klasy.

Streszczenie

Krótkie streszczenie zajęć proponowanych w tej jednostce.

Przygotowanie uczniów

Oczekiwane i wymagane przygotowanie uczniów do zajęć proponowanych w tej jednostce.

Cele szczegółowe zajęć – osiągnięcia uczniów

Szczegółowe cele zajęć w kategoriach osiągnięć uczniów.

Uczeń:

Treści programowe – wypisy z podstawy programowej

Zapisy z podstawy programowej informatyki, których realizacją jest ta jednostka, mogą to być tylko fragmenty odpowiednich zapisów. Poszczególne zapisy są opatrzone numerem, pod jakim występują w podstawie. Dodatkowo można wymienić treści innych przedmiotów.

Uczeń:

Pojęcia i metody informatyczne

Pojęcia informatyczne kształcone i rozwijane na tych zajęciach oraz metody informatyczne stosowane przez uczniów, a także przez nauczyciela wspierającego uczniów. Pojęcia i metody mogą odnosić się również do komputerów i innego sprzętu.

Metody pracy w klasie

Przewidziane i sugerowane metody pracy uczniów wspieranych przez nauczyciela. Na ogół są to:

- **Słowna**: rozmowa, objaśnienia, dyskusja, rozmowy między uczniami.
- **Oglądowa**: pokaz/przykład na tablicy interaktywnej dla wszystkich uczniów.
- **Czynna**: uczniowie wykonują postawione im zadania, mogą wybrać.
- **Aktywizująca**: praca w grupie – zadania stawiane i wykonywane w grupie, testy.

Formy pracy uczniów

Opis, jak uczniowie pracują. Na ogół są to:

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami,
- praca indywidualna w zeszycie,
- praca indywidualna na komputerze z wykorzystaniem aplikacji komputerowej (lub na tablecie) z ewentualnym wsparciem innych uczniów,
- praca w grupie uczniów

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Opis, jak nauczyciel ma się przygotować do przeprowadzenia zajęć według tej jednostki, jakie ma przygotować pomoce dla uczniów, jakiego sprzętu będzie potrzebował.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Wskazówki metodyczne dla nauczyciela dotyczące realizacji celów nauczania, czyli osiągnięć uczniów, oraz realizacji zapisów podstawy programowej. Odnoszą się także do uwzględnienia siedmiu filarów wdrożenia.

Przebieg zajęć

W miarę szczegółowy opis przebiegu zajęć. Powinien zostawiać wiele swobody nauczycielowi, a zwłaszcza uczniom, pozwalając im często na realizację ich własnych pomysłów, ich kreatywności. Zajęcia uwzględniają również zaangażowanie i możliwości uczniów.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Ewentualna modyfikacja tej jednostki zajęć lub/i poszerzenie jej zakresu jako propozycja kontynuacji zajęć

5. Materiały szkoleniowe dla kształcenia informatycznego w szkole podstawowej

5.1 Pierwsze algorytmy poza komputerem

Temat jednostki

Pierwsze algorytmy poza komputerem

Poziom kształcenia

Szkoła podstawowa: edukacja wczesnoszkolna, klasy I–III. Zajęcia te powinny odbyć się w miarę wcześnie w tych klasach i stanowić pierwszy krok w edukacji informatycznej.

Streszczenie

Zajęcia są wprowadzeniem do algorytmicznego myślenia. Odbywają się na pokratkowanej macie, na której zaznaczono lub można umieścić różne obiekty w zależności od przekazanej uczniom przez nauczyciela interpretacji maty. Uczniowie najpierw wykonują przykładowe ciągi poleceń (algorytmy) udostępnione przez nauczyciela. Następnie w grupach tworzą własne algorytmy służące do osiągnięcia z pozycji wyjściowej pewnego celu na macie. Algorytmy jednej grupy uczniów wykonują uczniowie z innych grup. Algorytmy składają się z poleceń w postaci obrazkowej. Ciągi poleceń – to algorytm, przekazywanie algorytmu uczniowi to programowanie, a ten uczeń – to „komputer" wykonujący polecenia programu.

Ten typ zajęć informatycznych określa się mianem *unplugged*, bez komputera.

Przygotowanie uczniów

Nie jest oczekiwane żadne specjalne przygotowanie uczniów do wzięcia udziału w tych zajęciach. To są pierwsze zajęcia, na których uczniowie zetkną się z pojęciami informatycznymi, takimi jak algorytm, ale nie *explicite*, tylko w swoim działaniu.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- wykonuje polecenia innego ucznia związane z ruchem do przodu i do tyłu oraz skrętów w lewo i w prawo,
- odczytuje polecenia obrazkowe odnoszące się do ruchów i skrętów,
- tworzy ciąg poleceń obrazkowych do osiągnięcia określonego celu na macie,
- współpracuje z innym uczniem lub w grupie uczniów.

Treści programowe – wypisy z podstawy programowej edukacji informatycznej

1. Osiągnięcia w zakresie rozumienia, analizowania i rozwiązywania problemów. Uczeń:
   2) tworzy polecenie lub sekwencje poleceń dla określonego planu działania prowadzące do osiągnięcia celu;

Pojęcia i metody informatyczne

- algorytm,
- polecenia (kroki) składające się na algorytm,
- program – wykonywany algorytm,
- wykonywanie algorytmu, w szczególności kinestetyczna realizacja algorytmu,
- komputer – urządzenie, tutaj uczeń wykonujący program, będący realizacją algorytmu.

Metody pracy w klasie

- **Słowna**: nauczyciel objaśnia cel zadań na macie, a uczniowie w grupach rozmawiają przy tworzeniu swoich algorytmów i ich wykonywaniu.

- **Oglądowa**: nauczyciel może wyświetlić na tablicy interaktywnej ciekawe historyjki związane z przemieszczaniem się po macie, np. wzięte z Godziny Kodowania.
- **Czynna**: uczniowie najpierw tworzą, a następnie wykonują algorytmy.
- **Aktywizująca**: uczniowie w grupach układają algorytmy, a następnie je wykonują.

Formy pracy uczniów

- zajęcia z całą klasą – aktywne śledzenie demonstracji nauczyciela, z ewentualnymi pytaniami,
- praca w grupie uczniów,
- wykonywanie poleceń uczniów przez innych uczniów – symulacja komputera.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel przygotowuje pokratkowaną matę, wielkość kratki powinna umożliwić, by uczeń na niej stanął. Na macie mogą być umieszczone motywy (przyrodnicze, geograficzne itp.). Przygotowuje również wiele kart ze znakami oznaczającymi kierunki: do przodu, w lewo, w prawo. Wielkość kart powinna odpowiadać wielkości kratek na macie. Właściwie mogą wystarczyć tylko karty ze strzałkami.

W przebiegu zajęć proponuje się skorzystanie z prostych łamigłówek, które polegają na ustawianiu strzałek (kierunków), by osiągnąć określony cel. Takie łamigłówki są dostępne pod adresem **https://studio.code.org/courses** jako Kurs 1 (pokazane w nim jest, że karty ze strzałkami są wystarczające). Ten kurs jest przewidziany dla uczniów, którzy mogą jeszcze nie umieć biegle czytać. Zawiera bardzo wiele łamigłówek, dlatego nauczyciel powinien wcześniej zapoznać się z tym kursem i wybrać odpowiednie łamigłówki.

Jeśli szkoła dysponuje robotami, to zajęcia tej jednostki mogą być wzbogacone „programowaniem" robotów. W tym przypadku nauczyciel powinien przygotować roboty i odpowiednie zadania dla uczniów.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Zajęcia są wprowadzeniem do algorytmicznego myślenia. Algorytm na tych zajęciach jest ograniczony do ciągu kroków, które umożliwiają dotarcie do wybranego miejsca – to miejsce wskazuje nauczyciel lub wybierają uczniowie. Na macie jest oznaczony początek każdej drogi. Krokami są polecenia „idź do przodu", „zawróć", „skręć w lewo", „skręć w prawo".

Zajęcia mają na celu intuicyjne zapoznanie uczniów z takimi pojęciami, jak algorytm, krok algorytmu, program, komputer.

Przebieg zajęć

Cele zajęć są osiągane etapowo:

- *Wprowadzenie nauczyciela*. Na początku nauczyciel zaznajamia uczniów z matą i kartami. Pokazuje także przykład, jak ułożyć karty na macie, aby od wejścia przejść do wybranego obiektu na macie.
- *Ćwiczenia w grupach*. Uczniowie dobierają się w grupy po kilkoro i nauczyciel rozdaje grupom odpowiednio ułożone karty. W każdej grupie jeden z uczniów odczytuje kolejne strzałki na kartach (np. do przodu, w lewo, w prawo) a inny – przechodzi po macie zgodnie z poleceniami. Po tym kinestetycznym wykonaniu algorytmu uczniowie układają jego karty na macie, by przekonać się, że algorytm został dobrze wykonany.
- *Etap kreatywności uczniów*. Uczniowie, dla wybranych lokalizacji na macie, układają swoje algorytmy/programy z kart, a następnie wykonują je w parach.
- *Demonstracja z Godziny Kodowania.* Nauczyciel udostępnia uczniom na tablicy interaktywnej łamigłówki z **https://studio.code.org/s/course1**. Wybiera odpowiednie łamigłówki. Wszystkie polegają na układaniu programów, ale dla różnych sytuacji problemowych, np. do rysowania przez Artystę. Uczniowie mogą rozwiązywać te łamigłówki na tablicy interaktywnej.
- *Indywidualna praca uczniów.* Jeśli jest taka możliwość, uczniowie rozwiązują łamigłówki Godziny Kodowania na tabletach lub w pracowni komputerowej. Nauczyciel może polecić uczniom, by tymi łamigłówkami zainteresowali się w domu.
- *Zajęcia z robotami.* Jeśli szkoła dysponuje robotami, to zajęcia tej jednostki mogą być wzbogacone „programowaniem" robotów. W tym przypadku uczniowie wykonują zadania wcześniej przygotowane przez nauczyciela.

- *Uwaga metodyczna.* KAŻDY uczeń powinien mieć szansę ułożenia algorytmu (programu) i wykonania go na planszy. Jest to ważny moment dla przyszłego myślenia algorytmicznego.
- *Dyskusja podsumowująca.* Lekcję powinna zakończyć rozmowa między uczniami, stymulowana przez nauczyciela, w której uczniowie powinni spróbować określić własnymi słowami, co to jest krok, sekwencja (ciąg) kroków, kolejność, algorytm, program, wykonanie algorytmu lub programu, komputer.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Podobny scenariusz zajęć może mieć na celu realizację innego punktu z podstawy programowej – tworzenia przez uczniów ciągów poleceń (algorytmów), składające się na codzienne czynności, takie jak mycie zębów, ubieranie się, przygotowania do wyjścia do szkoły, przechodzenie przez jezdnię itp.:

1. Osiągnięcia w zakresie rozumienia, analizowania i rozwiązywania problemów. Uczeń:
   1) układa w logicznym porządku: obrazki, teksty, polecenia (instrukcje) składające się m.in. na codzienne czynności.

Mata może nie być potrzebna, karty natomiast powinny być oznaczone odpowiednimi ilustracjami czynności, które mają być uszeregowane.

Zajęcia powinny prowadzić do intuicyjnego poznania pojęcia „algorytm liniowy", czyli algorytm, w którym czynności/polecenia są wykonywane po kolei, w porządku liniowym.

Pod adresem **https://www.mauthor.com/present/4863796046987264** można znaleźć proste łamigłówki dla uczniów na tematy codziennych czynności.

Jeśli zajęcia mogą się odbyć w pracowni komputerowej, to polecamy „pierwsze kroki uczniów z komputerem w aplikacji: **http://www.mauthor.com/present/6494370079703040**

5.2 Zaczynamy myśleć komputacyjnie

Temat jednostki

Zaczynamy myśleć komputacyjnie

Poziom kształcenia

Szkoła podstawowa: edukacja wczesnoszkolna, klasy I–III.

Najlepszy efekt jest osiągany, gdy przynajmniej część tych zajęć odbywa się w pracowni komputerowej.

Streszczenie

Zajęcia są wprowadzeniem do myślenia, które ma cechy myślenia komputacyjnego, o którym szczegółowo piszę w pracy znajdującej się w Załączniku 2. W preambule do podstawy programowej (patrz Załącznik 1) czytamy:

„Podstawowe zadanie szkoły – alfabetyzacja w zakresie czytania, pisania i rachowania – wymaga poszerzenia o alfabetyzację w zakresie umiejętności rozwiązywania problemów z różnych dziedzin ze świadomym wykorzystaniem metod i narzędzi wywodzących się z informatyki oraz na lepsze zrozumienie, jakie są obecne możliwości technologii, komputerów i ich zastosowań".

Jest to nawiązanie do operacyjnej definicji **myślenia komputacyjnego** (ang. *computational thinking*), które określa procesy myślowe towarzyszące formułowaniu problemów i ich rozwiązań w postaci umożliwiającej ich efektywną realizację z wykorzystaniem komputera. Myślenie komputacyjne określa użyteczne postawy i umiejętności, jakie każdy, nie tylko informatyk, powinien starać się wykształcić i stosować. Dzięki takiemu szerokiemu spojrzeniu na kompetencje informatyczne informatyka nie jest ograniczana do nauki o komputerach, ale dostarcza metod dla działalności umysłowej, które mogą być wykorzystane w innych dziedzinach, jak i w codziennym życiu.

Myślenie komputacyjne w rozwiązywaniu problemów cechuje się stosowaniem:

1. **abstrakcji**, by skupić uwagę na głównych cechach rozważanej sytuacji problemowej,
2. **dekompozycji**, gdy problem można rozłożyć na mniejsze części dla ułatwienia rozwiązywania,
3. **myślenia algorytmicznego**, by zaproponować sposób rozwiązywania, który może być przeznaczony dla komputera,
4. **uogólnień**, by znane rozwiązanie zastosować w innej sytuacji, być może bardziej złożonej.

W tej jednostce tematycznej zajęć proponujemy, by uczniowie rozwiązywali ciąg łamigłówek typu Sudoku, od najprostszych, po bardziej złożone. W tym celu korzystają z kilku aplikacji z cyklu **Informatyka dla smyka**.

Przygotowanie uczniów

Uczniowie powinni w miarę sprawnie posługiwać się myszą przy klikaniu przycisków i przenoszeniu ikon. Powinni również umieć zaznaczać wybrane pola na ekranie i wpisywać w nich cyfry z przedziału [1, 9].

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- postępuje zgodnie z przyjętymi zasadami, jakie powinno spełniać rozwiązanie łamigłówki Sudoku,
- wydziela fragmenty łamigłówki (wiersze, kolumny, fragmenty kwadratowe), by do nich stosować zasadę Sudoku,
- podejmuje decyzje w kolejnych krokach, tworząc w ten sposób algorytm,
- potrafi abstrahować – stosować tę samą zasadę do różnych obiektów i do obiektów o różnej wielkości.

Treści programowe – wypisy z podstawy programowej edukacji informatycznej

1. Osiągnięcia w zakresie rozumienia, analizowania i rozwiązywania problemów. Uczeń:
   3) rozwiązuje zadania, zagadki i łamigłówki prowadzące do odkrywania algorytmów.

Pojęcia i metody informatyczne

- abstrakcyjne myślenie – nie ma znaczenia, co uczniowie umieszczają w Sudoku: owoce, zwierzęta czy cyfry, ważna jest zasada,
- rozkład problemu na mniejsze części – uczniowie stosują zasadę Sudoku do wierszy, kolumn i wydzielonych kwadratów,
- algorytm – dla każdego Sudoku uczniowie tworzą algorytm jego wypełnienia poprawnie,
- uogólnienia – kolejne Sudoku jest coraz bardziej złożone, ale zasada pozostaje taka sama.

Uczniowie poznają te pojęcia, w pewnym sensie nieświadomie, wykonując ćwiczenia, w których te pojęcia występują.

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel podaje cel zajęć i na wybranym przykładzie wyświetlonym na tablicy interaktywnej objaśnia, na czym polega Sudoku.
- **Czynna**: uczniowie pracują nad rozwiązaniami łamigłówek.
- **Aktywizująca**: coraz trudniejsze Sudoku mogą aktywizować uczniów.

Formy pracy uczniów

- zajęcia z całą klasą – aktywne śledzenie demonstracji nauczyciela, z ewentualnymi pytaniami,
- indywidualne, w parach lub w grupie rozwiązywanie łamigłówek Sudoku.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Głównym celem zadań jest rozwiązywanie łamigłówek typu Sudoku. Zanim uczniowie zaczną rozwiązywać te łamigłówki na komputerze, proponujemy, by nauczyciel przygotował proste Sudoku o wielkościach podobnych jak w programach, ale do wykonania na papierze. Zestaw może być o rozmiarach 4 x 4 i zawierać figury geometryczne. Potrzebna jest pokratkowana plansza 4 x 4 i po 4 figury każdego rodzaju: kwadraty, trójkąty, koła, prostokąty. Można dać uczniom czysty zestaw do ułożenia od początku, albo częściowo wypełnić, by uzu-

pełnili. Należy zadbać, by każde takie ćwiczenie dla uczniów miało rozwiązanie, w tym celu można zacząć od ułożonego i usunąć z niego niektóre karty. Ćwiczenia poza komputerem uczniowie mogą wykonać w zeszycie, przerysowując propozycje nauczyciela.

Na zajęciach w pracowni komputerowej uczniowie będą korzystać z aplikacji pod adresami

Zestaw nr 1	Sudoku I	**http://www.mauthor.com/present/4643045024989184**
Zestaw nr 2	Sudoku II	**http://www.mauthor.com/present/5493604572463104**
Zestaw nr 3	Sudoku III	**http://www.mauthor.com/present/5287554153971712**
Zestaw nr 4	Sudoku IV	**http://www.mauthor.com/present/5568306823299072**

– Nauczyciel powinien wcześniej przejrzeć wszystkie aplikacje i zawarte w nich ćwiczenia, by ewentualnie pomagać później uczniom.

Pierwsze dwa zestawy mogą być przystępne dla uczniów w klasach 1–3, dwa ostatnie są bardziej złożone – to typowe zestawy 9 x 9. Te trudniejsze można polecić uczniom, którzy szybciej uporają się z pierwszymi dwoma zestawami.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Zajęcia są wprowadzeniem do komputacyjnego myślenia. Polecamy lekturę artykułu na ten temat w Załączniku 2. Proponowane zajęcia z łamigłówką Sudoku znakomicie umożliwiają uczniom pierwsze kroki w myśleniu abstrakcyjnym (nieważne jest, co uczniowie układają, ważna jest zasada), starają się uzupełniać planszę w mniejszych fragmentach, czyli dekomponują ją, w ten sposób tworzą algorytm rozwiązania konkretnej łamigłówki i w kolejnych krokach przechodzą do coraz bardziej złożonych łamigłówek. W taki sposób należy kierować pracą uczniów.

Pierwsze Sudoku to tak zwane **kwadraty łacińskie** – w każdym wierszu i w każdej kolumnie mają być umieszczone różne obrazki różne cyfry.

Uwaga. Wszystkie Sudoku w zaproponowanych aplikacjach mają rozwiązania.

Przebieg zajęć

Cele zajęć są osiągane etapowo:

- *Wprowadzenie nauczyciela*. Na początku nauczyciel objaśnia na przykładzie, na czym polega rozwiązanie łamigłówki Sudoku. Może to zrobić na tablicy interaktywnej, także na zwykłej tablicy, z pomocą papierowego zestawu lub posługując się aplikacją komputerową. Może poprosić wybranego ucznia, by ułożył obrazki zgodnie z podaną zasadą, by się przekonać, czy dostatecznie jasno ją wyjaśnił uczniom.
- *Ćwiczenia w grupach*. Uczniowie dobierają się w grupy po kilkoro, nauczyciel rozdaje grupom zestawy do Sudoku na papierze i uczniowie znajdują rozwiązanie. Na ogół te proste Sudoku, jak w zestawie w pierwszej aplikacji, mają wiele rozwiązań – różni uczniowie mogą otrzymać różne rozwiązania.
- *Indywidualna praca uczniów.* Uczniowie pracują z aplikacjami, do których adresy otrzymują od nauczyciela. Powinni zacząć od pierwszej aplikacji – nie powinna sprawić kłopotów żadnemu uczniowi. Druga aplikacja zaczyna się od przypomnienia pierwszej, a dalej następują trudniejsze przykłady.
- *Bardziej zaawansowane zajęcia.* Zestawy 3 i 4 są trudniejsze, nauczyciel podsuwa je uczniom, którzy szybko wykonali te pierwsze.
- *Uwaga metodyczna.* KAŻDY uczeń powinien bez specjalnego kłopotu wykonać zestaw pierwszy. Ewentualnie nauczyciel prosi innego ucznia o pomoc tym, którzy mają kłopoty.
- *Dyskusja podsumowująca.* Zajęcia kończą się rozmową między uczniami, jak podobały się im łamigłówki, gdzie napotkali trudności, czy ukończyli wszystkie łamigłówki, czy łatwiejsze były Sudoku z obrazkami, czy z cyframi. Nauczyciel może zapytać, jaką obrali strategię rozwiązywania – można tutaj przewidzieć żywą dyskusję, którą dobrze jest podsumować, że faktycznie w tych łamigłówkach **nie ma gotowego algorytmu**, gotowej i tej samej kolejności wypełniania, kolejność zależy od wypełnienia i indywidualnych decyzji.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Kontynuacja tych zajęć może mieć miejsce w starszych klasach z zestawami trudniejszymi wśród zaproponowanych.

W sieci jest dostępnych wiele aplikacji Sudoku dla różnych poziomów trudności. Najtrudniejsze zestawy często dobrze jest rozwiązywać na papierze, gdzie dla ułatwienia kolejnych kroków można zapisywać częściowe decyzje do podjęcia.

5.3 Pierwsze kroki w programowaniu

Temat jednostki:

Pierwsze kroki w programowaniu

Poziom kształcenia

Szkoła podstawowa: edukacja wczesnoszkolna, klasy I–III, raczej późniejsze klasy.

Streszczenie

Zajęcia mają na celu pierwszy kontakt uczniów z programowaniem komputera. Odbywają się w gotowym środowisku **Godziny Kodowania**, w którym programy są układane z bloków i wykonywane w ramach gotowych łamigłówek. Uczniowie nie muszą zwracać uwagi na składniową (syntaktyczną) poprawność programów, a jedynie na to, co programy mają robić.

Zajęcia są pierwszym etapem programowania wizualno-blokowego – programy składają się z gotowych bloków, a ich efekt działania jest widoczny w postaci graficznej (wizualnej).

Przygotowanie uczniów

Uczniowie mieli już zajęcia edukacji informatycznej, przynajmniej w rodzaju opisanych w 5.1, poświęcone pojęciom algorytm i program, np. układnych poza komputerem.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- orientuje się w strukturze obrazu na ekranie – gdzie znajdują się bloki, obszar roboczy i plansza z łamigłówką,
- tworzy program złożony z bloków (przez ich przeciąganie i upuszczanie), aby osiągnąć cel danej łamigłówki,
- uruchamia program i śledzi jego działanie na planszy łamigłówki,
- ewentualnie określa złe działanie swojego programu, poprawia go więc.

Treści programowe – wypisy z podstawy programowej edukacji informatycznej

2. Osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera […]. Uczeń:
    1. programuje wizualnie: proste sytuacje lub historyjki […], pojedyncze polecenia, a także ich sekwencje sterujące obiektem na ekranie komputera […];

Pojęcia i metody informatyczne

- algorytm dla danego celu,
- podstawowe bloki do budowy programów – odpowiedniki instrukcji w językach programowania,
- program – algorytm dla komputera,
- wykonanie programu,
- poprawianie (debagowanie) programu

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel objaśnia i ilustruje na tablicy interaktywnej, jaka jest struktura obrazu na ekranie i jak tworzy się program.
- **Czynna**: uczniowie pracują przy komputerach.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, aktywizując się nawzajem; ponadto sprawniej pracujący uczniowie przechodzą do bardziej zaawansowanych łamigłówek.

Formy pracy uczniów

- praca przy komputerze, indywidualna lub w parach – większe grupy przy jednym komputerze nie są wskazane

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel wcześniej powinien zapoznać się ze środowiskiem Godziny Kodowania **https://code.org/**. Aktualne informacje o tym środowisku są publikowane w blogu na stronie autora **http://mmsyslo.pl**. W przypadku tej konkretnej jednostki tematycznej zalecane jest skorzystanie z Kursu 1, który jest przewidziany dla najmłodszych uczniów nie czytających biegle – bloki, z których składane są programy, nie zawierają poleceń słownych. Strona Kursu 1: **https://studio.code.org/s/course1**. Kurs zawiera wiele łamigłówek i nauczyciel powinien wybrać łamigłówki, uwzględniając preferencje uczniów. Na pierwszych zajęciach z Godziną Kodowania zalecamy nie wychodzić poza lekcje 1–10.

Pierwsze kroki przy komputerze uczniowie mogą wykonać w aplikacji:
http://www.mauthor.com/present/6494370079703040

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Przyjęto podejście do nauki programowania, w którym uczniowie najpierw poznają podstawowe instrukcje/polecenia dla komputera w postaci bloków i układają z nich programy do z góry przygotowanych sytuacji problemowych (łamigłówek). Godzina Kodowania umożliwia wprowadzenie nawet bardzo zaawansowanych konstrukcji programistycznych.

Przebieg zajęć

- *Rozgrzewka*. Na początku zajęć uczniowie mogą wykonać kolejne zadania z: aplikacji **http://www.mauthor.com/present/6494370079703040**
- *Wprowadzenie nauczyciela*. Na początku nauczyciel objaśnia, na czym polega rozwiązywanie łamigłówek w Godzinie Kodowania – **https://studio.code.org/s/course1** Na komputerach uczniowskich jest już otwarta ta strona. Jest to najprostszy zestaw łamigłówek dla początkujących uczniów. Wykonanie jednak łamigłówki nauczyciel może zademonstrować na tablicy interaktywnej lub poprosić ucznia, by zademonstrował.
- *Indywidualna praca uczniów.* Uczniowie pojedynczo lub w parach rozwiązują kolejne łamigłówki, z różnymi motywami. W tej jednostce tematycznej nie powinni wyjść poza lekcje 1–10 w Kursie nr 1.
- *Bardziej zaawansowane zajęcia.* Uczniom, którzy szybko wykonali te pierwsze łamigłówki, nauczyciel może polecić dalsze. To motywuje uczniów, wpływając na ocenę ich postępów.
- *Uwaga metodyczna.* KAŻDY uczeń powinien bez specjalnego kłopotu wykonać łamigłówki w lekcjach 1–10. Ewentualnie nauczyciel prosi innego ucznia o pomoc tym, którzy mają kłopoty.
- *Dyskusja podsumowująca.* Zajęcia kończą się rozmową między uczniami, jak podobały się im łamigłówki na komputerze – wiele podobnych wykonywali na matach. Nauczyciel kieruje dyskusję w kierunku znaczenia programu dla komputera i możliwości jego korygowania, gdy zawiera błędy.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Godzina Kodowania zawiera olbrzymi zasób łamigłówek, które mogą stanowić punkt startowy dla zajęć związanych z programowaniem, a poświęconych niemal każdej konstrukcji czy sytuacji programistycznej.

Przy częstym korzystaniu z tej inicjatywy zalecamy, by nauczyciel się zarejestrował i zarejestrował całą klasę, będzie mógł wtedy obserwować postępy swoich uczniów. Wiele z tych łamigłówek można polecić uczniom do wykonania w domu – nauczyciel będzie mógł wtedy obserwować ich aktywność (gdy będą zalogowani).

Wiele zestawów łamigłówek w Godzinie Kodowania jest przewidzianych właśnie na godzinę kodowania **https://code.org/hourofcode/overview** – taki był pierwotny cel inicjatorów tego środowiska programowania. Te zestawy na godzinę na ogół są związane z motywami dobrze znanymi uczniom z opowieści, gier, filmów i przeróżnych gadżetów. Zalecamy te zestawy, zawierają bowiem wiele konstrukcji przydatnych później przy programowaniu w Scratchu. Wśród nich są: Gwiezdne wojny, Minecraft, Kraina lodu, Angry Birds, Potańcówka (znakomita!) itd.

5.4 Pierwsze kroki w środowisku Scratcha

Temat jednostki:

Pierwsze kroki w środowisku Scratcha

Poziom kształcenia

Szkoła podstawowa: edukacja wczesnoszkolna, klasy I–III, raczej późniejsze klasy lub klasy IV–VI – wcześniejsze klasy.

Streszczenie

Zajęcia są kontynuacją nauki programowania w środowisku programowania wizualno-blokowego. Pierwsze zajęcia na ten temat odbyły się w środowisku Godziny Kodowania. W środowisku języka Scratch uczniowie tworzą programy dla własnych historyjek, w Godzinie Kodowania były to łamigłówki na wybrane tematy, polegające na ułożeniu programu dla danej sytuacji problemowej. Teraz zadaniem uczniów jest utworzenie programów (zwanych skryptami) dla własnych tematów/celów. Pewnym natchnieniem dla uczniów może być przegląd skryptów dostępnych w sieci – jest ich ponad milion. Ideą twórców Scratcha jest dzielenie się pomysłami, zrealizowanymi projektami. Można wziąć dostępny skrypt i utworzyć z niego własny – liczy się kreatywność ucznia, a także współpraca, wymiana pomysłów, wspólne działania.

Przygotowanie uczniów

Uczniowie są już po zajęciach w środowisku Godziny Kodowania, zatem jest im znane programowanie wizualno-blokowe. Znają też podstawowe bloki do budowania programów. Nieco inaczej wygląda okno programu, ale tylko zmienił się układ, a główne części są podobne – uczniowie nie powinni mieć kłopotu z przystosowaniem się do różnic.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- orientuje się w strukturze obrazu na ekranie: po lewej stronie znajdują się pogrupowane bloki – to przybornik, na środku jest obszar roboczy – miejsce na układane programy/skrypty, wreszcie po prawej stronie jest scena, na której duszek tworzy zaprogramowaną przez ucznia historyjkę,
- wymyśla historyjkę, którą chciałby zaprogramować, może przy tym skorzystać z przykładów innych projektów,
- tworzy program złożony z bloków (przez ich przeciąganie i upuszczanie), aby zbudować program/skrypt według własnego pomysłu,
- uruchamia program/skrypt i śledzi jego działanie na scenie,
- ewentualnie określa złe działanie swojego programu/skryptu, poprawia go więc (debaguje).

Treści programowe – wypisy z podstawy programowej edukacji informatycznej

2. Osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera […]. Uczeń:
   1) programuje wizualnie: proste sytuacje lub historyjki według pomysłów własnych i pomysłów opracowanych wspólnie z innymi uczniami, pojedyncze polecenia, a także ich sekwencje sterujące obiektem na ekranie komputera […];

2) tworzy proste rysunki, […]
3) zapisuje efekty swojej pracy we wskazanym miejscu.

Pojęcia i metody informatyczne

algorytm dla własnego pomysłu:

- podstawowe bloki do budowy programów – odpowiedniki instrukcji w językach programowania,
- program – realizacja algorytmu dla komputera,
- wykonanie programu,
- poprawianie (debagowanie) programu.

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel objaśnia widoczną budowę środowiska Scratcha, przypominając środowisko Godziny Kodowania i wymieniając główne różnice. Dodatkowo wskazuje na miejsca, gdzie można znaleźć samouczki i przykładowe projekty.
- **Czynna**: uczniowie pracują przy komputerach nad swoimi pomysłami.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, realizując wspólny pomysł projektu. Projekty uczniów mogą czerpać z dyskusji między nimi, jak również z sugestii nauczyciela. Najciekawsze projekty są w dużym stopniu interaktywne, dzięki uwzględnieniu w nich zdarzeń.

Formy pracy uczniów

- praca przy komputerze, indywidualna lub w parach, ewentualnie większe grupy pracujące nad realizacją większego, bardziej ambitnego projektu.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel powinien wcześniej zapoznać się ze środowiskiem języka Scratch i zainstalować je na komputerach w pracowni **https://scratch.mit.edu/**. W tym środowisku można pracować w trybie off-line lub on-line. W tym drugim trybie uczniowie (po zarejestrowaniu się i zalogowaniu) mają dostęp do wielu materiałów dodatkowych i do projektów wykonanych przez inne osoby.

Przed pierwszymi zajęciami w środowisku Scratcha warto, by nauczyciel wybrał przykładowe projekty i zademonstrował i polecił uczniom, jakie to środowisko ma możliwości realizacji niemal nieograniczonych pomysłów uczniów.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Przyjęto, że tworzenie projektów w środowisku Scratcha następuje po wcześniejszych zajęciach w Godzinie Kodowania – oba te środowiska służą do rozwijania umiejętności programowania wizualno-blokowego. A zatem należy wykorzystać umiejętności uczniów nabyte podczas pracy z Godziną Kodowania. Jedną z propozycji projektów zgłoszoną uczniom przez nauczyciela może być zakodowanie w Scratchu łamigłówki, którą uczniowie rozwiązywali w Godzinie Kodowania.

Przebieg zajęć

- *Rozgrzewka 1*. (5 min) Uczniowie wykonują wybraną łamigłówkę z Godziny Kodowania dla przypomnienia sobie, jak wygląda i funkcjonuje środowisko tej inicjatywy.
- *Wprowadzenie nauczyciela*. Następnie nauczyciel krótko przybliża budowę środowiska programowania w Scratchu.
- *Rozgrzewka 2*. Dla zapoznania się z funkcjonowaniem tego środowiska uczniowie uruchamiają wybrany projekt utworzony w tym środowisku. Taki projekt może zasugerować nauczyciel, dobierając go tematycznie i funkcjonalnie do celów swoich zajęć.
- *Indywidualna praca uczniów.* Uczniowie pojedynczo lub w parach pracują w środowisku Scratcha nad realizacją swoich pomysłów, projektów. Nic nie stoi na przeszkodzie, by wymieniali się swoimi pomysłami oraz sugestiami, w jaki sposób otrzymać w tym środowisku pewne efekty.

- *Uwaga metodyczna.* KAŻDY uczeń powinien ukończyć nawet najprostszy projekt w Scratchu dla pomysłu wcześniej zgłoszonego do wykonania. Efekt swojej pracy może oddać po dopracowaniu projektu po zajęciach.
- *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
- *Podsumowanie zajęć.* Na podsumowanie zajęć uczniowie prezentują swoje projekty, jednocześnie wyjaśniając, w jaki sposób uzyskali ciekawe efekty w swoich projektach. Taka dyskusja może przynieść wiele pomysłów wszystkim uczniom, które następnie mogą zrealizować, rozwijając dalej swoje projekty.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Środowisko Scratcha ma niemal nieograniczone możliwości kodowania pomysłów uczniów. W zależności od liczby godzin lekcyjnych przeznaczonych na programowanie w tym środowisku nauczyciel może zaproponować uczniom bardziej ambitne tematy projektów. Mogą to być na przykład gry.

Ciekawe tematy projektów w Scratchu mogą pochodzić z innych przedmiotów, na przykład z matematyki, fizyki, czy historii, dzięki czemu uczniowie mogą pogłębić swoją wiedzę z tych przedmiotów.

W jednostce tematycznej 5.6 proponujemy wykonanie projektu w środowisku Scratcha, którego efektem są konstrukcje graficzne o pewnych własnościach matematycznych.

5.5 Przejście od Godziny Kodowania do Pythona

Temat jednostki:

Przejście od Godziny Kodowania do Pythona

Poziom kształcenia

Szkoła podstawowa: informatyka, klasa IV–VIII

Streszczenie

Zajęcia mają na celu „łagodne" przeprowadzenie uczniów ze środowiska Godziny Kodowania do środowiska programowania w języku Python, czyli przejście ze środowiska programowania wizualno-blokowego do środowiska programowania tekstowego. W tym pierwszym środowisku uczniowie tworzą programy dla gotowych łamigłówek, a w tym drugim – mają nieograniczoną swobodę. Jednak znajomość bloków z tego pierwszego środowiska pozwala im łatwo przenieść programy w postaci bloków do czysto tekstowego języka, gdyż faktycznie w każdym bloku jest również zapisany tekst tego, do czego służy blok.

Zajęcia w tej jednostce mają na celu wykształcenie u uczniów abstrakcyjnego myślenia o programowaniu – program to obiekt abstrakcyjny sytuacji, którą chcemy zaprogramować. W tym sensie postać zapisu programu ma drugorzędne znaczenie.

Przygotowanie uczniów

Uczniowie potrafią rozwiązywać łamigłówki Godziny Kodowania. Znają również środowisko programowania języka Python i potrafią pisać proste programy w tym środowisku z wykorzystaniem modułu `turtle`.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- orientuje się w strukturze programu napisanego w Godzinie Kodowania, jak i w języku Python,
- potrafi wyabstrahować tekst programu z bloków Godziny Kodowania,

- stosuje instrukcje z modułu `turtle` w Pythonie,
- transformuje program z Godziny Kodowania do programu w Pythonie,
- uruchamia tak otrzymane programy i ewentualnie poprawia je, gdy źle działają,
- koncepcyjnie operuje konstrukcjami programistycznymi, niezależnie od języka programowania.

Treści programowe – wypisy z podstawy programowej informatyki

Z podstawy dla klas IV–VI:

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i zapisuje w wizualnym języku programowania:
      a. pomysły historyjek i rozwiązania problemów, w tym proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych oraz zdarzeń,
   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:

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 […].

Uwaga. W podstawie programowej dla szkół podstawowych jest zalecenie, by rozpocząć programowanie w języku tekstowym. Takie środowisko jest niezbędne dla realizacji algorytmów podanych w podstawie w części I.

Pojęcia i metody informatyczne

- algorytm dla danego celu,
- podstawowe bloki i instrukcje do budowy programów,
- program – algorytm dla komputera,
- wykonanie programu,
- poprawianie (debagowanie) programu.

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel ilustruje na tablicy interaktywnej przykładowe powiązanie bloku z instrukcją tekstową, posługując się przy tym tekstem, który jest w bloku.
- **Czynna**: uczniowie pracują przy komputerach.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, aktywizując się nawzajem; ponadto uczniowie mogą modyfikować programy blokowe, by otrzymywać bardziej złożone programy w Pythonie.

Formy pracy uczniów

- praca przy komputerze, indywidualna lub w parach – większe grupy przy jednym komputerze nie są wskazane.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel wcześniej powinien zapoznać się ze środowiskiem Godziny Kodowania **https://code.org/**. Aktualne informacje o tym środowisku są publikowane w blogu na stronie autora **http://mmsyslo.pl**.

W przypadku tej jednostki, zaleca się skorzystanie z zestawu łamigłówek Kraina lodu z Anną i Elzą, a w szczególności z łamigłówki: **https://studio.code.org/s/frozen/stage/1/puzzle/12**.

Nauczyciel powinien mieć przygotowany przykład przejścia od instrukcji w blokach do tekstowego zapisu instrukcji z wykorzystaniem modułu `turtle` w Pythonie.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Zajęcia w tej jednostce mają na celu wykształcenie u uczniów abstrakcyjnego myślenia o programowaniu – program to obiekt abstrakcyjny sytuacji, którą chcemy zaprogramować. W tym sensie sposób czy język zapisu

programu ma drugorzędne znaczenie. W tej jednostce tematycznej uczniowie doświadczają takiego podejścia przy przejściu od blokowych programów w Godzinie Kodowania do programów czysto tekstowych w Pythonie.

Zajęcia tej jednostki tematycznej są też okazją do wykorzystania wiedzy z matematyki na temat kątów, długości odcinków i tworzenia figur oraz ich kombinacji.

Przebieg zajęć

1. *Krok początkowy*. Uczniowie rozwiązują jedną z łamigłówek Godziny Kodowania, proponuje się: **https://studio.code.org/s/frozen/stage/1/puzzle/12**.
2. *Wprowadzenie nauczyciela*. Nauczyciel wyjaśnia, na czym będzie polegała lekcja – pokazuje, jak „wyciągnąć" teksty z bloków i utworzyć z nich program w Pythonie.
3. *Indywidualna praca uczniów.* Z programu blokowego ukończonej łamigłówki z Godziny Kodowania uczniowie wypisują na papierze teksty z kolejnych bloków. Następnie z tych tekstów tworzą program w Pythonie z wykorzystaniem modułu `turtle`.
4. Uruchamiają programy w Pythonie i ewentualnie debagują je.
5. *Bardziej zaawansowane zajęcia*. Uczniom, którzy szybko wykonali zadanie proponujemy modyfikację według własnego uznania, na przykład uwzględniając inne figury do tworzenia płatków śniegu.
6. *Uwaga metodyczna*. KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie odpowiadającym programowi blokowemu.
7. *Indywidualna praca uczniów i ich ocena*. W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
8. *Dyskusja podsumowująca.* Zajęcia kończą się dyskusją na temat znaczenia języka programowania w programowaniu. Uczniowie znają już środowisko programów blokowych (Godzina Kodowania, Scratch) i tekstowych (Python), powinni więc po tych zajęciach umieć myśleć konstrukcjami programistycznymi (jak przypisanie, pętla, instrukcja warunkowa) w oderwaniu od ich konkretnej realizacji i zapisu w danych środowisku programowania.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Poszerzenie zakresu tych zajęć może polegać na modyfikacji programów w języku Python, by zastosować inne figury geometryczne do tworzenia płatków śniegu.

Innym poszerzeniem może być zastąpienie w rozważaniach języka Python przez C++, jeśli ten drugi język jest przedmiotem zajęć. Odradza się jednak zajmowanie uczniów zbyt wieloma środowiskami programowania. Wystarczy wprowadzenie do programowania wizualno-blokowego w Godzinie Kodowania, następnie programowanie w Scratchu i na końcu jeden z języków, ale nie oba, Python lub C++. Nie wykluczam, że niektórzy uczniowie mogą pisać programy w obu tych środowiskach tekstowego programowania. Językiem C++ mogą być zainteresowani uczniowie startujący w konkursach i olimpiadach informatycznych.

5.6 Przejście od Scratcha do Pythona

Temat jednostki:

Przejście od Scratcha do Pythona

Poziom kształcenia

Szkoła podstawowa: informatyka, klasa IV–VIII

Streszczenie

Zajęcia mają na celu „łagodne" przeprowadzenie uczniów ze środowiska Scratcha do środowiska programowania w języku Python, czyli przejście ze środowiska programowania wizualno-blokowego do środowiska programowania tekstowego. W obu środowiskach uczniowie mają nieograniczoną swobodę w realizacji swoich pomysłów. Zajęcia te mają zilustrować łatwość przenoszenia programów w postaci blokowej do czysto tekstowego języka, gdyż faktycznie w każdym bloku jest również zapisany tekst tego, do czego służy blok.

Zajęcia w tej jednostce tematycznej mają na celu wykształcenie u uczniów abstrakcyjnego myślenia o programowaniu – program to obiekt abstrakcyjny sytuacji, którą chcemy zaprogramować. W tym sensie zapis programu ma drugorzędne znaczenie.

Przygotowanie uczniów

Uczniowie potrafią programować w Scratchu. Znają również środowisko programowania języka Python i potrafią pisać proste programy w tym środowisku z wykorzystaniem modułu `turtle`.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- orientuje się w strukturze programu napisanego zarówno w Scratchu, jak i w Pythonie,
- tworzy własny blok (funkcję) w Scratchu i wykorzystuje go w programie, w tym w sposób rekurencyjny,
- stosuje instrukcje z modułu `turtle` w języku Python,
- potrafi wyabstrahować tekst programu z bloków Scratcha,
- transformuje program ze Scratcha do programu w Pythonie,
- uruchamia tak otrzymane programy i ewentualnie poprawia je, gdy źle działają,
- koncepcyjnie operuje konstrukcjami programistycznymi, niezależnie od języka programowania.

Treści programowe – wypisy z podstawy programowej informatyki

Z podstawy dla klas IV–VI:

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   1) projektuje, tworzy i zapisuje w wizualnym języku programowania:
      a. pomysły historyjek i rozwiązania problemów, w tym proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych oraz zdarzeń,
   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:

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 […].

Uwaga. W podstawie programowej dla szkół podstawowych jest zalecenie, by rozpocząć programowanie w języku tekstowym. Takie środowisko jest niezbędne dla realizacji algorytmów podanych w podstawie w części I.

Pojęcia i metody informatyczne

- algorytm dla danego celu,
- podstawowe bloki i instrukcje do budowy programów,
- funkcja, w tym funkcja rekurencyjna, i jej wykorzystanie w programie,
- rekurencja,
- program – algorytm dla komputera,
- wykonanie programu,
- poprawianie (debagowanie) programu.

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel ilustruje na tablicy interaktywnej przykładowe powiązanie bloku z instrukcją tekstową, posługując się przy tym tekstem, który jest w bloku.
- **Czynna**: uczniowie pracują przy komputerach.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, aktywizując się nawzajem; ponadto uczniowie mogą modyfikować programy blokowe, by otrzymywać bardziej złożone programy w Pythonie.

Formy pracy uczniów

- praca przy komputerze, indywidualna lub w parach – większe grupy przy jednym komputerze nie są wskazane.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

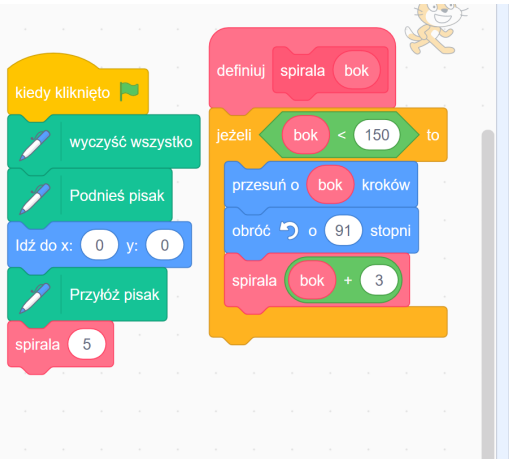
Nauczyciel wcześniej powinien dobrze poznać środowiska Scratcha i Pythona.

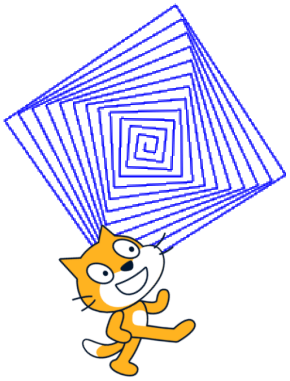
Nauczyciel powinien mieć przygotowany przykład przejścia od instrukcji w blokach do tekstowego zapisu instrukcji z wykorzystaniem modułu `turtle` w Pythonie. Poniżej przykład takiego przejścia.

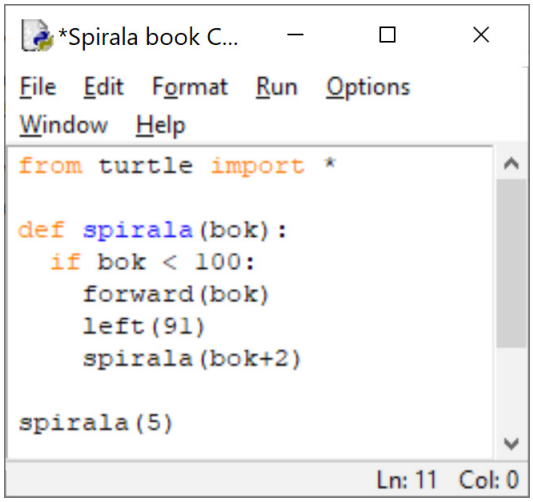
Programy w Scratchu i Pythonie służą do rysowania spirali – widać na rysunkach taki sam efekt. W obu programach zastosowano rekurencję końcową. Dodatkowy program w Pythonie wykonuje ten sam rysunek – rekurencję zastąpiono przez instrukcję warunkową.

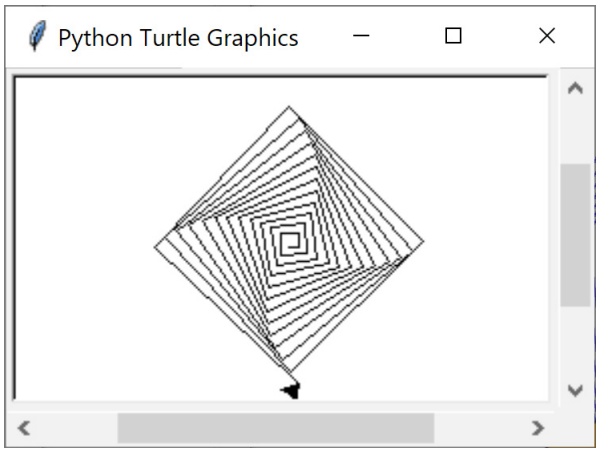
Programy te realizują ideę Paperta z 1980 roku, że nawet najmłodszym uczniom można przybliżyć pojęcie nieskończoności – po usunięciu z podanych programów warunku o zakończeniu rysowania powstanie nieskończona spirala.

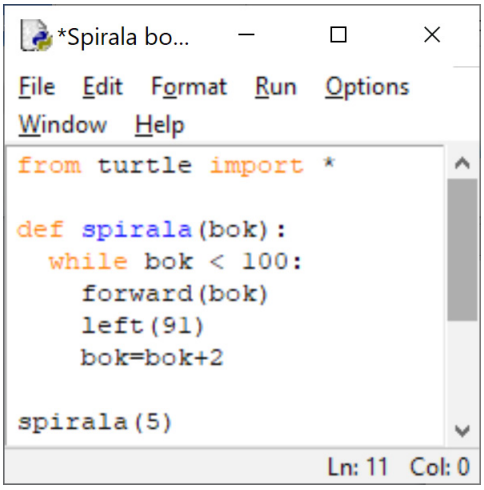
W obu programach rekurencyjnych występuje funkcja (`definiuj` i `def`), którą mają zdefiniować uczniowie.











Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Zajęcia w tej jednostce mają na celu wykształcenie u uczniów abstrakcyjnego myślenia o programowaniu – program to obiekt abstrakcyjny sytuacji, którą chcemy zaprogramować. W tym sensie sposób czy język zapisu programu ma drugorzędne znaczenie. W tej jednostce uczniowie doświadczają takiego podejścia przy przejściu od blokowych programów w Scratchu do programów czysto tekstowych w Pythonie.

Dodatkowe uwagi metodyczne są zamieszczone w poprzednim punkcie. Odnoszą się one do wykorzystania funkcji w programowaniu oraz użycia funkcji w zaprogramowaniu rekurencyjnego rozwiązania.

Zajęcia tej jednostki tematycznej są też okazją do wykorzystania wiedzy z matematyki na temat kątów, długości odcinków i tworzenia figur oraz ich kombinacji.

Przebieg zajęć

- *Wprowadzenie nauczyciela*. Nauczyciel wyjaśnia, na czym będzie polegała lekcja – pokazuje, jak „wyciągnąć" teksty z bloków Scratcha i utworzyć z nich program w Pythonie. Może w tym posłużyć się przykładem, który jest zamieszczony powyżej.
- *Indywidualna praca uczniów.* Najpierw uczniowie tworzą program w Scratchu. Następnie wypisują na papierze teksty z kolejnych bloków tego programu, a na końcu z tych tekstów tworzą program w Pythonie z wykorzystaniem modułu `turtle`.

Uruchamiają programy z Pythonie i ewentualnie debagują je.
Uwaga. Ta lekcja jest okazją, jeśli nie było o tym mowy wcześniej aby wprowadzić pojęcie funkcji. W Scratchu definiuje to własny blok. Ponadto dość naturalnie funkcja może przyjąć postać rekurencyjną. Więcej szczegółów na ten temat zawarto w książce 6.

- *Bardziej zaawansowane zajęcia.* Uczniom, którzy szybko wykonali zadanie, proponujemy modyfikację według własnego uznania, na przykład uwzględniając inne figury do tworzenia spiral.
- *Uwaga metodyczna.* KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie odpowiadającym programowi blokowemu w Scratchu.
- *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
- *Dyskusja podsumowująca.* Zajęcia kończą się dyskusją na temat znaczenia języka programowania w programowaniu. Uczniowie znają już środowisko programów blokowych (Godzina Kodowania, Scratch) i tekstowych (Python), powinni więc po tych zajęciach umieć myśleć konstrukcjami programistycznymi (jak przypisanie, pętla, instrukcja warunkowa) w oderwaniu od ich konkretnej realizacji i zapisu w danych środowisku programowania.
  Innymi tematami do dyskusji są pojęcia funkcji i rekurencji, którym powinno być poświęconych wiele innych lekcji.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Poszerzenie zakresu tych zajęć może polegać na modyfikacji programów w języku Python, by zastosować inne figury geometryczne do tworzenia spiral. Różnorodne postacie spiral można obejrzeć w Internecie, wyszukując je po haśle spirale.

Dodatkowe zajęcia należy poświęcić na utrwalenie pojęcia funkcji w językach programowania jako konstrukcji, którą uczeń sam buduje. Ponadto osobne zajęcia powinny być poświęcone rekurencji jako podejściu do rozwiązywania problemów i zapisywania ich rozwiązań w postaci programów rekurencyjnych. Więcej na ten temat jest w książkach 3–6.

Innym poszerzeniem może być zastąpienie w rozważaniach języka Python przez C++, jeśli ten drugi język jest przedmiotem zajęć. Odradza się jednak zajmowanie uczniów zbyt wieloma środowiskami programowania. Wystarczy wprowadzenie do programowania wizualno-blokowego w Scratchu (lub w Godzinie Kodowania), a następnie programowanie w Pythonie lub C++. Nie wykluczam, że niektórzy uczniowie mogą pisać programy w obu tych środowiskach tekstowego programowania. Językiem C++ mogą być zainteresowani uczniowie startujący w konkursach i olimpiadach informatycznych.

5.7 Wyszukiwanie i porządkowanie informacji

Temat jednostki:

Wyszukiwanie i porządkowanie informacji
Ta jednostka tematyczna jest przewidziana na kilka lekcji.

Poziom kształcenia

Szkoła podstawowa: informatyka, klasy IV–VI i VII–VIII

Streszczenie

Wyszukiwanie i porządkowanie informacji 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. Ta jednostka jest wprowadzeniem do tej tematyki. Jednocześnie przejście od prostej metody wyszukiwania do porządkowania jest **ilustracją spiralnego rozwoju** umiejętności uczniów.

Przygotowanie uczniów

Powinni umieć stosować instrukcje iteracyjne w programach, jednocześnie zajęcia te będą okazją do pogłębienia tych umiejętności. Ponadto powinni umieć zaprojektować algorytm, posługując się w tym celu schematem blokowym.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- tworzy specyfikację problemu do jego luźnego opisu,
- posługuje się schematem blokowym, by zaprojektować algorytm,
- korzysta z instrukcji iteracyjnych w Pythonie,
- znajduje najmniejszy lub największy element w zbiorze,
- porządkuje ciąg, stosując metodę przez wybór,
- 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.

Treści programowe – wypisy z podstawy programowej informatyki

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 osiągnięcia, 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:
      b. […] proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych […],
      c. […],
   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:
2) 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

- specyfikacja sytuacji problemowej, dla której ma być zaprojektowany algorytm,
- algorytm dla danej specyfikacji,
- algorytm 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.

Metody pracy w klasie

- **Słowna i Oglądowa**: nauczyciel dostarcza różnych przykładów 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.
- **Czynna**: uczniowie najpierw dyskutują nad rozwiązaniem postawionego problemu, a później tworzą algorytm i programują go w Pythonie.
- **Aktywizująca**: uczniowie mogą pracować (programować) w parach, dzielą zadania do wykonania między sobą, aktywizują się nawzajem.

Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna nad stworzeniem algorytmu i na komputerze, by zaprogramować algorytm.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

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.

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.

Polecamy zainstalowanie aplikacji *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.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Wyszukiwanie i porządkowanie to jedne z najczęściej wykonywanych operacji w komputerze, w tym również w programach użytkowników. Każdy uczeń powinien poznać idee podstawowych metod/algorytmów realizacji tych operacji nie tylko na potrzeby zajęć informatycznych, ale także jako metody stosowane poza obliczeniami komputerowymi.

Ta jednostka tematyczna jest wprowadzeniem do tematyki wyszukiwania i porządkowania. Następna jednostka jest poświęcona wyszukiwaniu wśród informacji uporządkowanych – to również bardzo ważny obszar zastosowań metod informatycznych, które stanowią bazę dla myślenia komputacyjnego, stosowanego w różnych sytuacjach praktycznych i życiowych.

Przebieg zajęć

1. *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 elementu w ciągu (liczb), a na końcu – do porządkowania liczb, jeśli wiemy, jak znaleźć element najmniejszy.
2. *Indywidualna, grupowa i w całej klasie praca uczniów*. Najpierw uczniowie piszą (formalną) specyfikację problemów, dla których będą tworzyli algorytm, a później program.
   Kolejny etap to zaprojektowanie algorytmu znajdowania najmniejszego elementu. Zapewne większość uczniów poda algorytm liniowy, czyli od lewej do prawej. Dyskusja, czy to jest najlepszy algorytm, może doprowadzić do stosowania algorytmu turniejowego. Może być zaskoczeniem dla uczniów, że oba te algorytmy, liniowy i turniejowy, wykonują tyle samo porównań, co więcej – nie ma algorytmu, który wykonywałby mniej porównań.
   Kolejny etap dyskusji to jak użyć algorytmu znajdowania najmniejszego elementu, by uporządkować ciąg liczb? Tutaj wystarczy pytanie nauczyciela mające formę podpowiedzi – a jaki element znajduje się na początku ciągu uporządkowanego?
   Dyskusję nad dwoma problemami tych zajęć uczniowie ilustrują, posługując się aplikacją *Maszyna Sortująca*, która jest w załączniku do tych materiałów.
3. *Indywidualna praca uczniów*. Po tym etapie wymyślania i projektowania algorytmów uczniowie programują je w języku Python.
   Uruchamiają swoje programy z Pythonie, ewentualnie debagują je i testują ich działanie na różnych ciągach danych.
   Jeden program powinien znajdować najmniejszy element w ciągu danych, a drugi porządkować ciąg danych.
4. *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.
   Jeszcze bardziej zaawansowanym zadaniem dla uczniów jest policzenie, ile porównań i przestawień elementów wykonują ich programy dla ciągu danych, których jest *n*. (Odsyłamy tutaj do punktu 6.2 w książce *Algorytmy*).
5. *Uwaga metodyczna*. KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie, który znajduje najmniejszy element w ciągu, oraz innym programem, który porządkuje ciąg.
6. *Indywidualna praca uczniów i ich ocena*. W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
7. *Dyskusja podsumowująca*. Zajęcia kończą się dyskusją na 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.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Naturalnym poszerzeniem tematyki tej jednostki jest zagadnienie wyszukiwania elementów w ciągach uporządkowanych. Zajmujemy się tym w następnej jednostce tematycznej.

5.8 Wyszukiwanie przez połowienie

Temat jednostki:

Wyszukiwanie przez połowienie

Poziom kształcenia

Szkoła podstawowa: informatyka, klasy IV–VI i VII–VIII

Streszczenie

Wyszukiwanie i porządkowanie informacji 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. Ta jednostka tematyczna jest kontynuacją poprzedniej. Poświęcona jest wyszukiwaniu danego elementu w zbiorach uporządkowanych. W tym celu jest stosowna metoda połowienia (zbioru), która jest prostym przykładem bardzo ważnej strategii rozwiązywania problemów **dziel i zwyciężaj**, stosowanej w różnych sytuacjach. Strategia ta prowadzi do bardzo szybkich metod obliczeniowych.

Przygotowanie uczniów

Wcześniej uczniowie powinni poznać metody wyszukiwania elementów w dowolnych ciągach, niekoniecznie uporządkowanych. W komputerowej realizacji wyszukiwania przez połowienie może być przydatna znajomość rekurencji. Oczekuje się, że uczniowie w miarę sprawnie programują w Pythonie.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- formułuje specyfikację problemu do jego luźnego opisu,
- stosuje metodę połowienia do wyszukiwania elementów w zbiorze uporządkowanym,
- wyjaśnia ideę metody dziel i zwyciężaj w projektowaniu algorytmów,
- posługuje się schematem blokowym, by zaprojektować algorytm,
- korzysta z instrukcji iteracyjnych w Pythonie,
- 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.

Treści programowe – wypisy z podstawy programowej informatyki

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 uporządkowanym […],
      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 uporządkowanym […],
   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

- specyfikacja sytuacji problemowej, dla której ma być zaprojektowany algorytm,
- algorytm dla danej specyfikacji,
- algorytm znajdowania elementu w zbiorze uporządkowanym,
- strategie połowienia, jako przykład podejścia dziel i zwyciężaj,
- komputerowa realizacja algorytmu,
- debagowanie i testowanie programu,
- zgodność komputerowej realizacji algorytmu ze specyfikacją problemu rozwiązywanego przez ten algorytm.

Metody pracy w klasie

- **Słowna i Oglądowa**: dyskusja z uczniami, w jaki sposób szybko znajdujemy informacje w encyklopediach, słownikach. Konkluzja: nie jest to wyszukiwanie liniowe, od któregoś końca.
- **Czynna**: uczniowie grają najpierw w grę w znajdowanie ukrytej liczby, dochodzą do algorytmu, a następnie piszą program do jego realizacji.
- **Aktywizująca**: próba odpowiedzi na pytanie, ile prób trzeba wykonać w metodzie przez połowienie, by znaleźć szukany element? Inne pytanie – jak zmodyfikować metodę przez połowienie, by uwzględniać, że niektóre poszukiwane w słowniku słowa zaczynają się od liter, które są blisko początku lub blisko końca alfabetu.

Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- gra w parach w zgadywanie liczby,
- praca indywidualna nad stworzeniem algorytmu i na komputerze, by zaprogramować algorytm.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel przygotowuje tabele (po jednej dla pary uczniów), które uczniowie będą wypełniali w trakcie gry w zgadywanie liczby. Wzór takiej tabeli jest podany w rozdz. 10 w książce [3]. Wyniki zgromadzone w tej tabeli przez różne pary uczniów posłużą do wyciągnięcia wniosków dotyczących własności wyszukiwania przez połowienie.

Problemy wyszukiwania i porządkowania są szczegółowo omówione w książkach [3] – rozdz. 10 i [4] – rozdz. 5, 9 i 12.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Wyszukiwanie informacji jest jedną z najczęściej wykonywanych operacji w komputerze. Znając trudność prostego, liniowego wyszukiwania w zbiorze nieuporządkowanym, wszystkie informacje w komputerze są uporządkowane. Podobnie czyni się w programach użytkowników. Wtedy można zastosować wyszukiwanie metodą przez połowienie. Zajęcia te mają na celu przybliżenie tej metody uczniom i określenie, jak dobra jest to metoda w porównaniu z wyszukiwaniem liniowym.

Nauczyciel, komentując wyszukiwanie przez połowienie, powinien naszkicować ogólniejszą technikę projektowania algorytmów, znaną jako dziel i zwyciężaj. Warto przy okazji dodać, że siła tej techniki bierze się z dekompozycji problemu na mniejsze podproblemy, jednej ze strategii myślenia komputacyjnego.

Przebieg zajęć

1. *Wprowadzenie do zajęć*. Nauczyciel opisuje grę dwuosobową w zgadywanie ukrytej liczby: jedna osoba ukrywa liczbę w znanym przedziale liczb naturalnych, a druga ma odgadnąć, zadając możliwie najmniej pytań „czy ta liczba jest mniejsza od x". Uczniowie dobierają się w pary, a nauczyciel rozdaje im tabele do wypełnienia w trakcie gry. Wyniki z tych tabel posłużą później do analizy zastosowanych w grze strategii.
2. *Dyskusja w całej klasie*. Zapewne większość zespołów wypracowała tę samą metodę, polegającą na połowieniu przedziału, w którym znajduje się jeszcze nieznaleziona liczba.
   Nauczyciel stawia teraz trudniejsze pytanie: ile należało zadać pytań, by znaleźć poszukiwaną liczbę? Jak ta liczba pytań zależy od długości przedziału, w którym znajduje się poszukiwana liczba? Zebrane w tabelach wyniki powinny ułatwić znalezienie odpowiedzi na to pytanie. Nauczyciel kieruje uwagę uczniów na przebieg metody: na początku jest pewien zbiór liczb, w każdym kroku z tego zbioru pozostaje połowa, aż na końcu otrzymujemy szukaną liczbę. A zatem jeśli w przedziale jest 100 liczb, to w kolejnych krokach będzie ich (połowy zaokrąglamy do góry):
   100, 50, 25, 13, 7, 4, 2, 1
   Należało więc zadać pytanie 7 razy. Informatyk w tym miejscu zauważy, ze 7 jest najmniejszą potęgą liczby 2 większą od 100, $2^7=128$.
3. *Indywidualna praca uczniów*. Po tym etapie analizy metody połowienia uczniowie przystępują do pisania w Pythonie programu, będącego realizacją tej metody.
   Wcześniej wspólnie opracowują specyfikację tego zadania, czyli określają, jakie mają być *Dane* i co ma być *Wynikiem*. Wśród wyników powinna się znaleźć liczba zadanych pytań.
4. *Indywidualnie* uruchamiają swoje programy i testują je na różnych ciągach danych i poszukiwanych liczbach. Warto uprzedzić uczniów, że napisanie poprawnego programu dla wyszukiwania przez połowienie nie jest łatwe.
5. *Bardziej zaawansowane zadanie*. Uczniom, którzy uporali się z programem dla metody połowienia, proponujemy zaprogramowanie interpolacyjnej metody wyszukiwania – jest opisana w książce [4], punkt 9.3.
6. *Indywidualna praca uczniów i ich ocena*. W przypadku zadań programistycznych nauczyciel motywuje uczniów wyżej, oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
7. *Dyskusja podsumowująca*. W tej dyskusji nauczyciel ma okazję naszkicować ogólne podejście do rozwiązywania problemów, zwane metodą dziel i zwyciężaj. Może przy tym posłużyć się na przykład porządkowaniem przez scalanie, używając do tego potasowanej talii kart. Ta metoda jest opisana w jednej z jednostek tematycznych dla szkół ponadpodstawowych.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Naturalnym poszerzeniem tematyki tej jednostki jest ogólna technika rozwiązywania problemów zwana metodą dziel i zwyciężaj.

5.9 Reprezentacja liczb naturalnych w komputerze

Temat jednostki:

Reprezentacja liczb naturalnych w komputerze

Poziom kształcenia

Szkoła podstawowa, informatyka, matematyka, klasy VII–VIII

Uczniowie mogą być jednak wcześniej przygotowani do tych zajęć, natomiast skorzystanie ze schematu Hornera musi być przesunięte do szkoły ponadpodstawowej.

Streszczenie

Znajomość reprezentacji informacji w komputerze, w szczególności reprezentacji liczb, ma duże znaczenie dla zrozumienia, jak działają komputery, jak wykonują obliczenia i inne operacje. Wiele algorytmów korzysta z postaci liczb w komputerze. Ta jednostka tematyczna dotyczy podstawowych algorytmów – znajdowania reprezentacji binarnej liczb oraz obliczania wartości dziesiętnej liczb binarnych. Dodatkowo zawiera materiał, który wymaga posłużenia się pewnymi wiadomościami ze szkoły ponadpodstawowej.

Przygotowanie uczniów

Posiadają przynajmniej podstawowe umiejętności programowania w języku Python.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- formułuje specyfikację problemu do jego luźnego opisu,
- zamienia postać dziesiętną liczby naturalnej na postać binarną i na odwrót,
- projektuje algorytm, posługując się schematem blokowym,
- korzysta z instrukcji iteracyjnych i warunkowych w Pythonie,
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych.

Treści programowe – wypisy z podstawy programowej informatyki, klasy VII–VIII

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
   3) przedstawia sposoby reprezentowania w komputerze […], liczb naturalnych (system binarny), […];

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. 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 […].

Pojęcia i metody informatyczne

- bit, bajt,
- reprezentacja binarna liczby naturalnej,
- algorytm wyznaczania binarnej reprezentacji liczby naturalnej,
- obliczanie dziesiętnej wartości liczby binarnej,
- długość liczby w komputerze,
- komputerowe realizacje algorytmów zamiany reprezentacji liczb.

Metody pracy w klasie

- **Słowna i Oglądowa**: uczniowie wyprowadzają algorytm zamiany reprezentacji na tablicy przed całą klasą; wspólnie uzasadniają poprawność reprezentacji i sposobów jej otrzymywania.
- **Czynna**: uczniowie ćwiczą wykonywanie algorytmów na papierze, zanim zaprogramują je na komputerze. Algorytmy zamiany reprezentacji mogą być również łatwo wykonywane na kalkulatorze.

- Dysponując binarnymi reprezentacjami liczb, uczniowie wykonują na nich podstawowe operacje arytmetyczne, począwszy od dodawania.
- **Aktywizująca**: uczniów mogą aktywizować do dalszych działań pytania postawione przez nauczyciela: jaka jest długość liczby w komputerze, tzn. ile bitów zajmuje reprezentacja binarna liczby o danej wartości?

Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna – odręczne na papierze otrzymywanie binarnych reprezentacji liczb oraz dziesiętnych wartości liczb binarnych; wykonywanie działań na liczbach binarnych,
- praca indywidualna nad stworzeniem algorytmu i na komputerze, by zaprogramować algorytm.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel przygotowuje przykładowe liczby dziesiętne i binarne, dla których uczniowie będą tworzyli te drugie reprezentacje. Liczby w postaci binarnej mogą posłużyć również do wykonywania na nich działań arytmetycznych.

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Mogą się zacząć pytania, które będą się wydawać uczniom dziwne. Co to znaczy, że na przykład 73051 jest liczbą dziesiętną? Jesteśmy tak przyzwyczajeni do automatycznego wykonywania rachunków w systemie dziesiętnym, że niemal nigdy nie korzystamy świadomie z tego, że poszczególne cyfry liczb w tym systemie spełniają następującą równość (podajemy ją dla przykładowej liczby):

$$7 \cdot 10^4 + 3 \cdot 10^3 + 0 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0 = 73051.$$

Liczba 10 w tym zapisie nazywa **się podstawą systemu liczenia**. Ogólnie za **podstawę systemu** reprezentacji liczb można wybrać dowolną liczbę naturalną b (nawet większą niż 10, jak 16), wtedy cyfry liczby $(a_n a_{n-1} \ldots a_{1a0})_b$ zapisanej w tym systemie należą do zbioru $\{0, 1, \ldots, b-1\}$, a jej wartość dziesiętna a jest określona następującą równością:

$$a = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b + a_0$$

Jeśli $b = 2$, to mamy system **binarny**, liczba a wyraża się wzorem

$$a = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 2 + a_0$$

a ciąg cyfr $(a_n a_{n-1} \ldots a_{1a0})_2$ w powyższym wzorze nazywa się **binarnym rozwinięciem** liczby a lub po prostu **liczbą binarną**. W tym przypadku cyfry a_i nazywa się **bitami** i mogą one przyjmować wartości 0 lub 1. Liczby binarne stanowią podstawę arytmetyki komputerowej, ponieważ elektroniczne elementy pamięci oraz logiki komputerów mogą się znajdować w dwóch stanach, które są identyfikowane z cyframi 0 i 1.

Interesuje nas teraz algorytm wyznaczania binarnej reprezentacji naturalnej liczby dziesiętnej a, czyli w jaki sposób można znaleźć kolejne bity tego rozwinięcia? Aby odpowiedzieć na to pytanie, zauważmy, że najmniej znaczący bit rozwinięcia binarnego liczby jest równy reszcie z dzielenia tej liczby przez 2. I rzeczywiście, jeśli podzielimy liczbę a w rozwinięciu binarnym przez 2, to iloraz jest równy $a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \ldots + a_1$, a reszta wynosi a_0 – jest to najmniej znaczący bit w reprezentacji liczby a. Następne bity rozwinięcia znajdujemy w podobny sposób, dzieląc kolejne ilorazy przez 2 i to postępowanie kończymy, gdy iloraz wynosi 0. Ten proces (algorytm) jest zilustrowany w tabeli poniżej dla liczby $29 = (11101)_2$. Uczniowie powinni zauważyć, że w tym algorytmie reprezentacja liczby jest tworzona od końca, czyli od najmniej znaczącego bitu. Należy o tym pamiętać, gdyż prowadzi to czasem do nieporozumień i błędów.

Dzielenie	Iloraz	Reszta
29 | 2	14	**1**
14 | 2	7	**0**
7 | 2	3	**1**
3 | 2	1	**1**
1 | 2	0	**1**

W obliczeniach ilorazu i reszty z dzielenia liczb całkowitych przez siebie będą potrzebne uczniom dwie operacje wykonywane na liczbach całkowitych, których wyniki są również liczbami całkowitymi. Dla dwóch liczb całkowitych k i l wartością k **mod** l jest reszta z dzielenia k przez l, czyli jest to liczba r spełniająca nierówności $0 \le r < l$. Z kolei wartością k **div** l jest iloraz całkowity z dzielenia k przez l, czyli wynik dzielenia k przez l obcięty do liczby całkowitej. W szczególnym przypadku, jeśli $l = 2$, to k mod 2 ma wartość 0 — gdy k jest liczbą parzystą, i wartość 1 — gdy k jest liczbą nieparzystą. Z kolei jeśli k jest liczbą parzystą, to k div 2 jest równe $k/2$, a jeśli k jest liczbą nieparzystą, to k div 2 ma wartość $(k - 1)/2$. W języku Python operacja mod ma symbol %, a operacja div – ma symbol //, patrz program poniżej.

Z drugiej strony, jeśli mamy liczbę daną w postaci binarnej $a = (a_n a_{n-1} \ldots a_1 a_0)_2$, to jej wartość dziesiętną obliczamy ze wzoru:

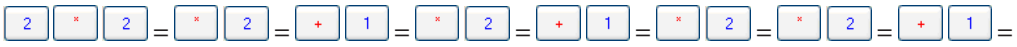
$$a = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 2 + a_0$$

Ten wzór to nic innego jak wielomian o współczynnikach 0 lub 1, w którym zmienna x ma wartość 2. Zatem, mając liczbę a daną w postaci binarnej $(a_n a_{n-1} \ldots a_1 a_0)_2$, można posłużyć się schematem Hornera, by policzyć wartość dziesiętną a. Schemat Hornera jest przedmiotem zajęć w szkole ponadpodstawowej, a tutaj wyjaśnimy na przykładzie, np. liczby binarnej $(1001101)_2$, jak obliczyć jej dziesiętną wartość. W poniższym przekształceniu kilkakrotnie wyłączamy 2 przed nawias:

$$(1001101)_2 = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1$$

$$= (((((1 \cdot 2 + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1 = 77$$

Tutaj ciekawostka. Zapis binarnej reprezentacji liczby w postaci schematu Hornera może posłużyć do obliczenia jej dziesiętnej wartości za pomocą prostego kalkulatora – demonstrujemy to poniżej. Zauważmy, że zera z rozwinięcia są pomijane w obliczeniach.



Dodatkowym ćwiczeniem dla uczniów może być obliczanie sumy liczb danych w postaci binarnej – pozostawmy to zadanie do samodzielnego wykonania. Zasada dodawania takich liczb nie różni się od dodawania długich liczb dziesiętnych, oczywiście nie można zapomnieć o przenoszeniu cyfr.

Przebieg zajęć

1. *Dyskusja w całej klasie.* Zajęcia są poświęcone reprezentacji liczb w komputerze. Uczniowie zapewne wiedzą lub przynajmniej domyślają się, że w takiej reprezentacji liczby są ciągami cyfr 0 i 1. Aby naprowadzić uczniów, jak taka reprezentacja może wyglądać, nauczyciel pyta, co to znaczy, że dana liczba jest w reprezentacji dziesiętnej. Jak wyjaśniamy powyżej, rolę liczby 10 w komputerowej reprezentacji liczby przyjmuje liczba 2.
2. *Indywidualna praca uczniów.* Po wstępnym wyjaśnieniu, jak interpretować postać liczby w reprezentacji o danej podstawie 10 lub 2, uczniowie znajdują na papierze reprezentacje binarne dla kilku liczb dziesiętnych – te liczby należy przedstawić jako sumy potęg liczby 2.
3. *Ponownie wspólnie*. Widząc, jak wyglądają binarne reprezentacje liczb, uczniowie dyskutują, jak zautomatyzować proces otrzymywania takich reprezentacji, czyli jaką postać powinien mieć algorytm dla tego zadania. W dyskusji, być może stymulowanej przez nauczyciela, dochodzą do algorytmu polegającego na wielokrotnym dzieleniu liczby dziesiętnej i ilorazów z dzielenia przez 2 i zapisywaniu reszty.
4. *Nauczyciel*. Wyjaśnia, że w algorytmie będą potrzebne dwie operacje – dzielenia całkowitego i brania reszty z dzielenia całkowitego. Uczniowie wykonują przykładowe obliczenia z użyciem tych operacji.
5. *Indywidualna praca uczniów.* Uczniowie są już przygotowani, by napisać program w Pythonie, który dla danej liczby dziesiętnej będzie generował jej postać binarną – należy ich uprzedzić, że kolejne bity są generowane w algorytmie od końca reprezentacji.
6. *Bardziej zaawansowane zadanie.* Bardziej zaawansowanym zadaniem jest obliczanie dziesiętnej wartości liczby danej w postaci binarnej. Nauczyciel może zachęcić uczniów do zapoznania się z metodą bazującą na schemacie Hornera, a zwłaszcza z jej wykorzystaniem w obliczeniach na prostym kalkulatorze.
   Innym zadaniem, niezbyt skomplikowanym, jest dodawanie liczb binarnych. Uczniowie otrzymują dwie liczby w postaci binarnej, dodają je, a następnie sprawdzają poprawność wyniku, odwołując się do dziesiętnej postaci składników i wyniku.

7. *Indywidualna praca uczniów i ich ocena*. W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
8. *Dyskusja podsumowująca*. W tej dyskusji nauczyciel powinien uzasadnić, dlaczego ten temat jest ważny dla informatyków i nie tylko. Reprezentacja binarna liczb jest bowiem wykorzystywana w wielu algorytmach. Ponadto użytkownik komputerów i sieci Internet może spotkać wiele informacji przedstawianych w postaci binarnej, np. adresy w sieci.
   Z kolei uczniowie mogą podzielić się opiniami, jak trudne według nich jest operowanie na liczbach binarnych, a w ogólności – na ciągach zer i jedynek.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Do binarnej reprezentacji liczb wraca się wielokrotnie w informatyce szkolnej, np. przy szybkim potęgowaniu, co jest przedmiotem jednej z jednostek tematycznych dla szkół ponadpodstawowych.

5.10 Wydawanie reszty – algorytm w arkuszu kalkulacyjnym

Temat jednostki:

Wydawanie reszty – algorytm w arkuszu kalkulacyjnym

Poziom kształcenia

Szkoła podstawowa, informatyka, klasy IV–VI

Streszczenie

Problem reszty można uznać za problem życia codziennego – chcemy, aby ani portfel, ani kieszenie nie wypełniało nam zbyt wiele banknotów i monet. W szczególnym przypadku może to być reszta, którą otrzymujemy w sklepie lub w restauracji, lub którą mamy wydać.

Matematycznie jest to trudny problem, ale okazuje się, że algorytm zachłannie działa całkiem dobrze. Ten algorytm bazuje na prostej strategii – jeśli reszta ma się składać z najmniejszej liczby banknotów i monet, to powinna być wydawana od największych nominałów.

Tę strategię bardzo prosto można zapisać w arkuszu kalkulacyjnym, co ilustruje jednocześnie, że arkusz może być wykorzystywany do zapisywania i wykonywania algorytmów. Ma ponadto dużą zaletę – można w nim bardzo prosto wykonywać eksperymenty obliczeniowe.

Przygotowanie uczniów

Uczeń potrafi zapisać w arkuszu kalkulacyjnym proste obliczenia.

Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- analizuje praktyczny problem i podejmuje próby jego rozwiązania,
- wypełnia arkusz odpowiednimi danymi,
- dobiera formuły obliczeniowe w arkuszu odpowiednio do zaplanowanych obliczeń,
- wykonuje symulacje obliczeń zapisanych w arkuszu i ewentualnie koryguje arkusz,
- proponuje i uzasadnia strategię zachłanną dla rozwiązania problemu optymalizacji,

Treści programowe – wypisy z podstawy programowej informatyki, klasy IV–VI

I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
   2) formułuje i zapisuje w postaci algorytmów polecenia składające się na:
      a. rozwiązanie problemów z życia codziennego […]

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń:
   3) przygotowuje i prezentuje rozwiązania problemów, posługując się podstawowymi aplikacjami ([…], arkusz kalkulacyjny, […]) na swoim komputerze […], wykazując się przy tym umiejętnościami:
      c. korzystania z arkusza kalkulacyjnego w trakcie rozwiązywania zadań związanych z prostymi obliczeniami: wprowadza dane do arkusza, formatuje komórki, definiuje proste formuły i dobiera wykresy do danych i celów obliczeń.

Pojęcia i metody informatyczne

- problem reszty,
- podejście zachłanne do rozwiązywania problemu,
- algorytm w arkuszu,
- symulacja algorytmu w arkuszu.

Metody pracy w klasie

- **Słowna i Oglądowa**: uczniowie wyprowadzają algorytm wydawania reszty w dyskusji całej klasy.
- **Czynna**: uczniowie indywidualnie zapisują w arkuszu kalkulacyjnym dane dotyczące polskiej waluty oraz algorytm zachłanny.
- **Aktywizująca**: uczniów aktywizują kolejno stawiane przez nauczyciela pytania dotyczące różnych sytuacji wydawania reszty.

Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionym przez nauczyciela problemem,
- praca indywidualna – zapisanie algorytmu w arkuszu kalkulacyjnym, wykonanie eksperymentów obliczeniowych.

Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Polecamy nauczycielowi p. 12.1 w książce [3], w którym przedstawiono różne aspekty problemu reszty. Znajdują się tam zarówno przykłady, jak i różne warianty tego problemu.

Z ciekawości nauczyciel powinien utworzyć własne rozwiązanie w arkuszu. Przykładowy arkusz jest pokazany poniżej. Zwracamy uwagę na użycie specjalnych funkcji, które zaokrąglają kwoty do dwóch miejsce po przecinku (do groszy). Ich dobór jest istotny, by w obliczeniach nie zgubić nawet jednego grosza.

	A	B	C	D
1				
2		**Kwota do wydania**		**1 234,19 zł**
3	**Nominały**	**Liczba nominałów**	**Kwota**	**Pozostało**
4				1 234,19 zł
5	200,00 zł	**6**	1 200,00 zł	34,19 zł
6	100,00 zł	**0**	- zł	34,19 zł
7	50,00 zł	**0**	- zł	34,19 zł
8	20,00 zł	**1**	20,00 zł	14,19 zł
9	10,00 zł	**1**	10,00 zł	4,19 zł
10	5,00 zł	**0**	- zł	4,19 zł
11	2,00 zł	**2**	4,00 zł	0,19 zł
12	1,00 zł	**0**	- zł	0,19 zł
13	0,50 zł	**0**	- zł	0,19 zł
14	0,20 zł	**0**	- zł	0,19 zł
15	0,10 zł	**1**	0,10 zł	0,09 zł
16	0,05 zł	**1**	0,05 zł	0,04 zł
17	0,02 zł	**2**	0,04 zł	- zł
18	0,01 zł	**0**	- zł	- zł
19		**Razem**	**1 234,19 zł**	

	A	B	C	D
1				
2		**Kwota do wydania**		**1234,19**
3	**Nominały**	**Liczba nominałów**	**Kwota**	**Pozostało**
4				=D2
5	200	**=ZAOKR.DÓŁ(D4/A5;0)**	=A5*B5	=ZAOKR(D4-C5;2)
6	100	**=ZAOKR.DÓŁ(D5/A6;0)**	=A6*B6	=ZAOKR(D5-C6;2)
7	50	**=ZAOKR.DÓŁ(D6/A7;0)**	=A7*B7	=ZAOKR(D6-C7;2)
8	20	**=ZAOKR.DÓŁ(D7/A8;0)**	=A8*B8	=ZAOKR(D7-C8;2)
9	10	**=ZAOKR.DÓŁ(D8/A9;0)**	=A9*B9	=ZAOKR(D8-C9;2)
10	5	**=ZAOKR.DÓŁ(D9/A10;0)**	=A10*B10	=ZAOKR(D9-C10;2)
11	2	**=ZAOKR.DÓŁ(D10/A11;0)**	=A11*B11	=ZAOKR(D10-C11;2)
12	1	**=ZAOKR.DÓŁ(D11/A12;0)**	=A12*B12	=ZAOKR(D11-C12;2)
13	0,5	**=ZAOKR.DÓŁ(D12/A13;0)**	=A13*B13	=ZAOKR(D12-C13;2)
14	0,2	**=ZAOKR.DÓŁ(D13/A14;0)**	=A14*B14	=ZAOKR(D13-C14;2)
15	0,1	**=ZAOKR.DÓŁ(D14/A15;0)**	=A15*B15	=ZAOKR(D14-C15;2)
16	0,05	**=ZAOKR.DÓŁ(D15/A16;0)**	=A16*B16	=ZAOKR(D15-C16;2)
17	0,02	**=ZAOKR.DÓŁ(D16/A17;0)**	=A17*B17	=ZAOKR(D16-C17;2)
18	0,01	**=ZAOKR.DÓŁ(D17/A18;0)**	=A18*B18	=ZAOKR(D17-C18;2)
19		**Razem**	**=SUMA(C5:**	

Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Podejście heurystyczne jest jednym ze sposobów myślenia o rozwiązywaniu problemów, zwłaszcza z codziennego otoczenia uczniów i rzeczywistych zastosowań. Problem reszty jest jednym z najprostszych problemów, dla rozwiązania którego uczniowie są w stanie sami wymyśleć metodę rozwiązywania, która faktycznie okazuje się najlepszą.

Przebieg zajęć

1. *Dyskusja w całej klasie.* Nauczyciel przedstawia problem reszty, a uczniowie dyskutują, jaki powinien być sposób wydawania reszty, by została utworzona z najmniejszej liczby banknotów i monet.
   Wkładem do dyskusji mogą być obserwacje uczniów poczynione w różnych miejscach, w sklepie czy w restauracji.
   Zapewne szybko pojawi się propozycja, by wydawać resztę od największego nominału.
   Pada pytanie od nauczyciela: czy zawsze będzie to najmniejsza ilość banknotów i monet? Odpowiedź na to pytanie nie jest łatwa. W przypadku polskiej waluty trudno jest znaleźć kontrprzykład. Jednak, gdyby pojawiła się dodatkowa moneta, np. o nominale 21 groszy, dla hazardzistów, kontrprzykład byłoby łatwo znaleźć. A co w sytuacji, gdy w kasie braknie niektórych banknotów lub monet? Uczniowie żywo dyskutują.
2. *Indywidualna praca uczniów.* Każdy z uczniów zapisuje w arkuszu kalkulacyjnym algorytm zachłanny i testuje jego poprawność dla różnych kwot reszty. Warto sprawdzić dla kwot kończących się parzystą i nieparzystą liczbą groszy, by upewnić się, że poprawnie działa formuła zaokrąglania liczb do dwóch miejsc po przecinku. Nauczyciel sugeruje uczniom takie testowanie.
3. *Nauczyciel*. Sugeruje znalezienie w sieci waluty wybranego kraju i odpowiednie zmodyfikowanie arkusza dla tego kraju.
4. *Indywidualna praca uczniów.* Jako dodatkowe wyzwanie, nauczyciel proponuje zapisanie algorytmu zachłannego dla problemu reszty w Pythonie.
5. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych (także w arkuszu) nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program (arkusz) został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
6. *Dyskusja podsumowująca.* Dyskusja może dotyczyć przydatności arkusza do zapisywania w nim algorytmów. Uczniowie podają, które ze znanych im algorytmów potrafiliby zapisać w arkuszu, a z którymi mieliby kłopot. Inna kwestia, to jakie według nich zalety i wady mają rozwiązania w arkuszu, a jakie rozwiązania w języku programowania.

Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Wiele modyfikacji problemu reszty można uzyskać, rozważając waluty innych państw lub przyjmując dodatkowe założenia o postaci banknotów i monet, a także o ich ilościach, np. ograniczonych w danej chwili.