PROGRAM KSZTAŁCENIA INFORMATYKI

dla zajęć pozalekcyjnych w szkole podstawowej

Anna Tatarczak

Spis treści

Wstęp

Obecnie uczniowie od wczesnych lat szkoły podstawowej mają styczność z komputerem zarówno w szkole jak i w domu, coraz powszechniejszy staje się również dostęp do Internetu. Celem zajęć dodatkowych informatyki w szkole podstawowej jest pogłębienie znajomości wybranej dziedziny zastosowań komputerów i informatyki jak również poszerzenie horyzontów, ukazanie młodemu człowiekowi możliwie pełnego obrazu tej dyscypliny.

Program dotyczy nauczania m informatyki w szkole podstawowej, w ramach zajęć dodatkowych - obejmuje 26 jednostki lekcyjne. Treści nauczania w programie są zapisane w postaci tabeli, która jednocześnie jest najprostszym przykładem rozkładu materiału. Program nauczania jest zbudowany tak , aby można było w nim łatwo odnaleźć potrzebne hasła i określić, jakie umiejętności powinni posiadać uczniowie. Pracujący z nim nauczyciel może dostosować go do potrzeb danej klasy oraz na jego bazie stworzyć własny program do indywidualnej pracy z uczniem o specyficznych bądź specjalnych potrzebach edukacyjnych.

Program przewiduje realizację części treści z podstawy programowej przedmiotu Informatyka oraz zakłada rozwijanie dodatkowych umiejętności, co znajduje odzwierciedlenie w zapisach dotyczących celów szczegółowych oraz monitorowania osiągnięć uczniów. Program kształcenia jako dokument opisuje zakres kształcenia informatycznego na zajęciach dodatkowych w szkole podstawowej. Ułatwieniem w planowaniu pracy nauczyciela jest orientacyjny przydział godzin przeznaczonych na realizację poszczególnych zajęć dodatkowych. Dodatkowy element programu stanowi odniesienie treści nauczania realizowanych w poszczególnych działach do podstawy programowej.

Poniższy program nauczania Informatyki w ramach zajęć dodatkowych jest zgodny z rozporządzeniem MEN z dnia 14 lutego 2017 r. w sprawie podstawy programowej wychowania przedszkolnego oraz podstawy programowej kształcenia ogólnego dla szkoły podstawowej, w tym dla uczniów z niepełnosprawnością intelektualną w stopniu umiarkowanym lub znacznym, kształcenia ogólnego dla branżowej szkoły I stopnia, kształcenia ogólnego dla szkoły specjalnej przysposabiającej do pracy oraz kształcenia ogólnego szkoły policealnej (DzU z 2017 r., poz. 59)

Metryczka

Typ szkoły: szkoła podstawowa, klasy I-VIII
Liczba jednostek lekcyjnych: 26h
Zagadnienia główne:

- algorytm,
- polecenia (kroki) składające się na algorytm,
- program – wykonywany algorytm,
- podstawowe bloki do budowy programów – odpowiedniki instrukcji w językach programowania,
- wykonywanie algorytmu, w szczególności kinestetyczna realizacja algorytmu,
- strategie połowienia, jako przykład podejścia dziel i zwyciężaj,
- algorytm wyznaczania binarnej reprezentacji liczby naturalnej,
- długość liczby w komputerze,
- symulacja algorytmu w arkuszu,
- funkcja, w tym funkcja rekurencyjna, i jej wykorzystanie w programie,
- testowanie programów,
- poprawianie (debagowanie) programu,
- zgodność komputerowej realizacji algorytmu ze specyfikacją problemu rozwiązywanego przez ten algorytm,
- komputer – urządzenie, tutaj uczeń, wykonujące program, będący realizacją algorytmu.

Cele edukacyjne

Osiąganie założonych celów edukacyjnych i wychowawczych jest możliwe dzięki stosowaniu na lekcjach różnorodnych form i metod nauczania. Ta różnorodność ma nie tylko sprawić że lekcje będą bardziej atrakcyjne, ale też zaktywizować uczniów w procesie uczenia się, zachęcić do rozwiązywania różnego rodzaju problemów, spowodować kształtowanie odpowiednich postaw.

Cele kształcenia

Nauczanie informatyki w sposób szczególny stymuluje rozwój intelektualny ucznia, poprzez:

- Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.
- Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.
- Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.
- Rozwijanie kompetencji społecznych, takich jak komunikacja i współpraca w grupie, w tym w środowiskach wirtualnych, udział w projektach zespołowych oraz zarządzanie projektami.
- Przestrzeganie prawa i zasad bezpieczeństwa. Respektowanie prywatności informacji i ochrony danych, praw własności intelektualnej, etykiety w komunikacji i norm współżycia społecznego, ocena zagrożeń związanych z technologią i ich uwzględnienie dla bezpieczeństwa swojego i innych.

Z ogólnych celów kształcenia informatycznego wynika, że szkoła powinna stymulować i wspomagać rozwój uzdolnień i zainteresowań uczniów. Osiągniemy to przez zainteresowanie uczniów rozwojem wiedzy informatycznej oraz nowymi możliwościami dostępu do informacji i komunikowania się:

- różnorodność problemów poruszanych na lekcjach;
- ukazywanie interesujących zastosowań informatyki;
- przedstawianie informatyki jako narzędzia niezbędnego do efektywnego wykonywania przyszłej pracy;
- prezentowanie perspektyw rozwoju informatyki.

Cele wychowania

Istotną część nauczania stanowi proces wychowania. W nauczaniu informatyki szczególnie eksponowane są cele wychowawcze, uczeń:

- przestrzega zasad regulaminu pracowni komputerowej,
- przestrzega zasad BHP w pracy na komputerze,
- dba o stanowisko swojej pracy przy komputerze,
- przestrzega zasad pracy w grupie,
- wykazuje potrzebę kreatywnego myślenia i działania,
- kształci wrażliwość estetyczną,
- uczy się szacunku do innych osób i ich pasji,
- kształci umiejętność programowania w wybranym języku programowania,
- kształci umiejętność myślenia algorytmicznego,
- wykazuje się dokładnością i starannością podczas pracy,
- stosuje się do zasad bezpieczeństwa podczas korzystania z internetu,
- kształci umiejętność myślenia algorytmicznego,
- postępuje zgodnie ze wskazaniami nauczyciela.

Cele dotyczące kształtowania postaw, należy realizować na każdej lekcji informatyki. Trzeba szczególnie wymagać od uczniów samodzielności w rozwiązywaniu problemów. Każdy uczeń powinien czuć się odpowiedzialny za powierzone mu zadania, ale jednocześnie powinien uczyć się współpracy z rówieśnikami.

Cele szczegółowe zajęć

Zajęcia	Osiągnięcia uczniów
1.	• 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.

Zajęcia	Osiągnięcia uczniów
2.	• 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.
3.	• 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.
4.	• 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).
5.	• 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,

Zajęcia	Osiągnięcia uczniów
	• transformuje program z Godziny Kodowania do programu w Pythonie, • uruchamia tak otrzymane programy i ewentualnie poprawia je, gdy go źle działają, • koncepcyjnie, operuje konstrukcjami programistycznymi, niezależnie od języka programowania.
6.	• orientuje się w strukturze programu napisanego 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.
7.	• 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.

Zajęcia	Osiągnięcia uczniów
8.	• formułuje specyfikację problemu do jego luźnego opisu, • stosuje metodę połowienia do wyszukiwania elementów 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.
9.	• 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.
10.	• 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.

Procedura osiągania celów

Do osiągnięcia zaprezentowanych celów kształcenia i wychowania najbardziej przydatne są następujące procedury:

1. Stworzenie odpowiedniego klimatu i bezpiecznej atmosfery na lekcji dodatkowej z informatyki.
2. Samodzielna praca z komputerem. Wykorzystanie podręcznika, Internetu, płyty CD dołączonej do podręcznika.
3. Rozwiązywanie ćwiczeń i zadań o rozmaitym stopniu trudności sprawdzających zrozumienie treści nauczania.
4. Korzystanie z wybranej platformy internetowej, poczty elektronicznej oraz różnymi programami komputerowymi m.in. Scratch, Excel.
5. Analiza struktury programu, analiza schematu algorytmem.
6. Stosowanie różnych form i metod pracy z uczniami: prezentacja multimedialna, praca metodą projektu, pokaz, pogadanka.
7. Rozwiązywanie zadań różnymi sposobami wyrabiając przy tym umiejętność poszukiwania najprostszych rozwiązań.
8. Zachęcanie uczniów do uczestnictwa w konkursach i projektach informatycznych.
9. Stopniowanie trudności alogorytmów.
10. Informowanie o postępach ucznia.

Ramowy rozkład materiału

Zgodnie z głównym celem zajęć z informatyki w szkole podstawowej, proponowane w tym programie treści dotyczą programowania i algorytmiki oraz możliwości wykorzystania komputerów, stosowania technologii informacyjnej, pracy zespołowej oraz etyki i bezpieczeństwa pracy.

Zebrane zostały treści przewidziane do realizacji w klasach 1-3, 4–6 i 7–8. Przedstawiam poniżej propozycję podziału treści programowych na poszczególne zagadnienia poruszane na zajęciach dodatkowych oraz orientacyjną liczbę godzin potrzebną na ich realizację.

Zajęcia	Temat w materiałach dydaktycznych	Godz.	Poziom kształcenia
1.	Pierwsze algorytmy poza komputerem	2	Klasa II
2.	Zaczynamy myśleć komputacyjnie	2	Klasa II
3.	Pierwsze kroki w programowaniu	2	Klasa III
4.	Pierwsze kroki w środowisku Scratcha	2	Klasa III-IV
5.	Przejście od Godziny Kodowania do Pythona	2	Klasa IV - VIII
6.	Przejście od Scratcha do Pythona	2	Klasa IV - VIII
7.	Wyszukiwanie i porządkowanie informacji	2	Klasa IV - VIII
8.	Wyszukiwanie przez połowienie	4	Klasa IV - VIII
9.	Reprezentacja liczb naturalnych w komputerze	4	Klasa VII – VIII
10.	Wydawanie reszty – algorytm w arkuszu kalkulacyjnym	4	Klasa IV – VI
Suma		26h	

Treści programowe

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, a od 2019 roku obejmie również szkoły ponadpodstawowe, a więc licea, technika i szkoły branżowe.

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 fundament:

- kolejność celów ogólnych kształcenia – kształcenie w zakresie logicznego, abstrakcyjnego i algorytmiczne myślenie 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 rozwiazywania 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

Proponowany program kształcenia uwzględnia powyższe fundamentalne założenia przyjęte w konstrukcji podstawy programowej informatyki i wynikające z nich wskazania dla realizacji kształcenia informatycznego w całym procesie kształcenia.

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
1.	Pierwsze algorytmy poza komputerem	1. Osiągnięcia w zakresie rozumienia, analizowania i rozwiązywania problemów. Uczeń: • tworzy polecenie lub sekwencje poleceń dla określonego planu działania prowadzące do osiągnięcia celu;
2.	Zaczynamy myśleć komputacyjnie	1. Osiągnięcia w zakresie rozumienia, analizowania i rozwiązywania problemów. Uczeń: • rozwiązuje zadania, zagadki i łamigłówki prowadzące do odkrywania algorytmów.

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
3.	Pierwsze kroki w programowaniu	1. Osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera […]. Uczeń: • programuje wizualnie: proste sytuacje lub historyjki […], pojedyncze polecenia, a także ich sekwencje sterujące obiektem na ekranie komputera […];
4.	Pierwsze kroki w środowisku Scratcha	1. Osiągnięcia w zakresie programowania i rozwiązywania problemów z wykorzystaniem komputera […]. Uczeń: • 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 […]; • tworzy proste rysunki, […] • zapisuje efekty swojej pracy we wskazanym miejscu.
5.	Przejście od Godziny Kodowania do Pythona	Z podstawy dla klas IV-VI: 1. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i zapisuje w wizualnym języku programowania: pomysły historyjek i rozwiązania problemów, w tym proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych oraz zdarzeń, • testuje na komputerze swoje programy pod względem zgodności z przyjętymi założeniami i

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
		ewentualnie je poprawia, objaśnia przebieg działania programów; • Z podstawy dla klas VII-VIII: 1. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, […].
6.	Przejście od Scratcha do Pythona	Z podstawy dla klas IV-VI: 1. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i zapisuje w wizualnym języku programowania: pomysły historyjek i rozwiązania problemów, w tym proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych oraz zdarzeń; 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: 1. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje:

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
		instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, […].
7.	Wyszukiwanie i porządkowanie informacji	Z podstawy dla klas IV-VI: 1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń: • tworzy i porządkuje w postaci sekwencji (liniowo) […] informacje […]; obiekty z uwzględnieniem ich cech charakterystycznych; • formułuje i zapisuje w postaci algorytmów polecenia składające się na: osiągnięcie postawionego celu, w tym znalezienie elementu w zbiorze nieuporządkowanym[…], znalezienie elementu najmniejszego i największego, • 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. 2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i zapisuje w […] języku programowania: […] proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych […], • testuje na komputerze swoje programy pod względem zgodności z przyjętymi założeniami i ewentualnie je poprawia, objaśnia przebieg działania programów;

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
		Z podstawy dla klas VII-VIII: 1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń: • 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; • stosuje przy rozwiązywaniu problemów podstawowe algorytmy: wyszukiwania i porządkowania: wyszukuje element w zbiorze […] nieuporządkowanym oraz porządkuje elementy w zbiorze metodą przez proste wybieranie […]; • rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów; • prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów. 2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • 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.

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
8.	Wyszukiwanie przez połowienie	Z podstawy dla klas IV-VI: 1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń: • tworzy i porządkuje w postaci sekwencji (liniowo) […] informacje […]; obiekty z uwzględnieniem ich cech charakterystycznych; • formułuje i zapisuje w postaci algorytmów polecenia składające się na: osiągnięcie postawionego celu, w tym znalezienie elementu w zbiorze uporządkowanym […], • 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. 2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • projektuje, tworzy i zapisuje w […] języku programowania: proste algorytmy z wykorzystaniem poleceń sekwencyjnych, warunkowych i iteracyjnych […], • 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: 1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
		• 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; • stosuje przy rozwiązywaniu problemów podstawowe algorytmy: wyszukiwania i porządkowania: wyszukuje element w zbiorze uporządkowanym […]; • rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów; • prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów. 2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • 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;
9.	Reprezentacja liczb naturalnych w komputerze	1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń: • przedstawia sposoby reprezentowania w komputerze […], liczb naturalnych (system binarny), […];

Zajęcia	Temat w materiałach dydaktycznych	Treści programowe
		2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń: • 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. […].
10.	Wydawanie reszty – algorytm w arkuszu kalkulacyjnym	1. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń: • formułuje i zapisuje w postaci algorytmów polecenia składające się na: rozwiązanie problemów z życia codziennego […] 2. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera […]. Uczeń: • przygotowuje i prezentuje rozwiązania problemów, posługując się podstawowymi aplikacjami ([…], arkusz kalkulacyjny, […]) na swoim komputerze […], wykazując się przy tym umiejętnościami: 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.

Planując cykl lekcji poświęconych konkretnemu zagadnieniu warto przeznaczyć czas na powtórzenie i usystematyzowanie omówionego wcześniej materiału dotyczącego danego hasła programowego. Materiał ten można rozszerzyć o ciekawsze i trudniejsze zadania. Aby przybliżyć uczniom wprowadzane pojęcia informatyczne, warto zwrócić uwagę na ich

powiązanie z życiem codziennym. Bardzo ważne jest, aby tematyka zadań ukazywała sposoby zastosowania informatyki w różnych dziedzinach życia.

Metody i formy pracy

Cele lekcji sugerują nauczycielowi zastosowanie odpowiedniej metod. Dobór metod do konkretnej lekcji powinien wynikać z tego jakie cele będą osiągane czy będzie to lekcja , na której nauczyciel wprowadza nowe treści, lekcja ćwiczeniowa, powtórzeniowa, czy też lekcja przeznaczona na sprawdzanie wiadomości i umiejętności uczniów. Niejednokrotnie na jednej lekcji wystąpią wszystkie te elementy, wówczas w każdej części nauczyciel może stosować różne metody nauczania. Warto pamiętać o tym, że każda metoda może zawierać elementy innej metody.

Założone cele są realizowane za pomocą następujących form pracy:

- podczas burzy mózgów prowadzonej przez całą klasę lub w grupach uczniów,
- przygotowywanie opisów algorytmów może odbywać się indywidualnie lub w parach uczniów,
- programy dla komputerów uczniowie piszą samodzielnie; testowanie programów może odbywać się w grupach uczniów,
- końcowym efektem pracy nad danym problemem jest prezentacja opisu algorytmu, programu komputerowego oraz wyników działania programu.

Ogólnie, jeśli na zajęciach dodatkowych z informatyki pojawia się problem, który ma być rozwiązany z pomocą komputera, to proces jego rozwiązywania powinien się składać z następujących sześciu etapów:

1. Opis, dyskusja i zrozumienie sytuacji problemowej;
2. Podanie specyfikacji problemów do rozwiązania;
3. Zaprojektowanie rozwiązania (algorytmu);
4. Implementacja (realizacja) rozwiązania w postaci programu komputerowego;
5. Testowanie i ewaluacja rozwiązania komputerowego;
6. Prezentacja sposobu otrzymania rozwiązania i samego rozwiązania

Wybór metod nauczania pozostawiamy nauczycielowi, niemniej jednak w realizacji tego programu proponujemy następujące metody pracy na poszczególnych zajęciach dodatkowych z informatyki.

Zajęcia	Temat w materiałach dydaktycznych	Metody pracy
1.	Pierwsze algorytmy poza komputerem	• **Słowna**: nauczyciel objaśnia cel zadań na macie, a uczniami 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 algorytmu, a następnie je wykonują.
2.	Zaczynamy myśleć komputacyjnie	• **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.
3.	Pierwsze kroki w programowaniu	• **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

Zajęcia	Temat w materiałach dydaktycznych	Metody pracy
		pracujący uczniowie przechodzą do bardziej zaawansowanych łamigłówek.
4.	Pierwsze kroki w środowisku Scratcha	• **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.
5.	Przejście od Godziny Kodowania do Pythona	• **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.
6.	Przejście od Scratcha do Pythona	• **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.

Zajęcia	Temat w materiałach dydaktycznych	Metody pracy
		• **Aktywizująca**: uczniowie mogą pracować (programować) w parach, aktywizując się nawzajem; ponadto uczniowie mogą modyfikować programy blokowe, by otrzymywać bardziej złożone programy w Pythonie.
7.	Wyszukiwanie i porządkowanie informacji	• **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.
8.	Wyszukiwanie przez połowienie	• **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.

Zajęcia	Temat w materiałach dydaktycznych	Metody pracy
9.	Reprezentacja liczb naturalnych w komputerze	• **Słowna i Oglądowa**: uczniowie wyprowadzają algorytm zamiany reprezentacji na tablicy przed cała 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?
10.	Wydawanie reszty – algorytm w arkuszu kalkulacyjnym	• **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.

Środki dydaktyczne

Do wszystkich zajęć wymagane są komputery, pozostałe środki dydaktyczne opisane są w tabeli poniżej. Dodatkowo szczegółowe przygotowanie klasy i materiałów dydaktycznych

opisane jest w sekcji Materiały dydaktyczne, jako element każdych zajęć. Uczniowie korzystają również z publikacji wymienionych w sekcji Materiały dodatkowe.

<table>
<tr><th>Zajęcia</th><th>Temat w materiałach dydaktycznych</th><th>Środki dydaktyczne</th></tr>
<tr><td>1.</td><td>Pierwsze algorytmy poza komputerem</td><td>

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. Faktycznie, mogą wystarczyć tylko karty ze strzałkami.
Łamigłówki dostępne pod adresem https://studio.code.org/courses, jako Kurs 1.
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.

</td></tr>
<tr><td>2.</td><td>Zaczynamy myśleć komputacyjnie</td><td>

Sudoku na papierze
Na zajęciach w pracowni komputerowej uczniowie będą korzystać z aplikacji pod adresami:

<table>
<tr><td>Zestaw nr 1</td><td>Sudoku I</td><td>http://www.mauthor.com/present/4643045024989184</td></tr>
<tr><td>Zestaw nr 2</td><td>Sudoku II</td><td>http://www.mauthor.com/present/5493604572463104</td></tr>
<tr><td>Zestaw nr 3</td><td>Sudoku III</td><td>http://www.mauthor.com/present/5287554153971712</td></tr>
<tr><td>Zestaw nr 4</td><td>Sudoku IV</td><td>http://www.mauthor.com/present/5568306823299072</td></tr>
</table>
</td></tr>
<tr><td>3.</td><td>Pierwsze kroki w programowaniu</td><td>

Kurs dostępny pod adresem https://studio.code.org/s/course1.

</td></tr>
</table>

Zajęcia	Temat w materiałach dydaktycznych	Środki dydaktyczne
		• Pierwsze kroki przy komputerze uczniowie mogą wykonać w aplikacji: http://www.mauthor.com/present/6494370079703040
4.	Pierwsze kroki w środowisku Scratcha	• Zainstalowane w pracowni informatycznej środowisko języka Scratch https://scratch.mit.edu/.
5.	Przejście od Godziny Kodowania do Pythona	• Łamigłówki dostępne na stronie https://studio.code.org/s/frozen/stage/1/puzzle/12.
6.	Przejście od Scratcha do Pythona	• Zainstalowane w pracowni informatycznej środowisko języka Scratch https://scratch.mit.edu/. • Zainstalowane w pracowni informatycznej środowisko języka Python
7.	Wyszukiwanie i porządkowanie informacji	• Zainstalowanie aplikacji *Maszyna Sortująca* na każdym komputerze w pracowni. Aplikacja ta pochodzi z podręcznika: E. Gurbiel, G. Hardt-Olejniczak, E. Kołczyk, H. Krupicka, M.M. Sysło, *Informatyka*, WSiP, Warszawa 2009. Ta aplikacja jest programem dydaktycznym, służącym do wykonywania eksperymentów z algorytmem znajdowania minimum w ciągu i algorytmem porządkowania przez wybór.
8.	Wyszukiwanie przez połowienie	• 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].
9.	Reprezentacja liczb naturalnych w komputerze	• Nauczyciel przygotowuje przykładowe liczby dziesiętne i binarne, dla których uczniowie będą tworzyli te drugie reprezentacje.
10.	Wydawanie reszty – algorytm w	Nie dotyczy

Zajęcia	Temat w materiałach dydaktycznych	Środki dydaktyczne
	arkuszu kalkulacyjnym	

Monitorowanie osiągnięć uczniów

Kontrolowanie i ocenianie uczniów powinno być spójne z tym, co było przedmiotem nauczania. Proponujemy następujący system oceniania:

- krótkie sprawdziany – mogą być niezapowiedziane, z 2–3 ostatnich tematów, 6–8 sprawdzianów w trakcie semestru
- Prace klasowe oraz testy oceniane są w skali 1–6, wg skali procentowej.

Ocena	Skala
niedostateczny	0% do 39%
dopuszczający	40% do 49%
dostateczny	50% do 74%
dobry	75% do 90%
bardzo dobry	91% do 100%
celujący	ocena bardzo dobry + zadanie dodatkowe

- W czasie semestru stawiane mogą być również plusy i minusy. Trzy plusy dają ocenę bardzo dobrą, trzy minusy – ocenę niedostateczną. Plusami i minusami oceniane mogą być: praca ucznia na lekcji, prace domowe, zadania dodatkowe.

Dodatkowo można zastosować stałe elementy oceniania:

- styl pracy ucznia podczas lekcji,
- efektywność i sposób pracy przy komputerze;
- umiejętność doboru narzędzia do realizowanego zadania;
- radzenie sobie z wykonywanym ćwiczeniem – poprawność jego wykonania;

- stopień biegłości w posługiwaniu się sprzętem i oprogramowaniem oraz słownictwem informatycznym,
- ogólny wynik wykonanej pracy widoczny na ekranie,
- estetyczny wygląd wykonanej pracy,
- zapisanie wykonanej pracy we właściwym miejscu na dysku,
- przygotowanie dokumentu do wydruku, korzystanie z podglądu wydruku,
- aktywność w trakcie lekcji,
- przestrzeganie zasad etycznych i prawnych związanych z korzystaniem z komputera i Internetu,
- aktywność w prowadzonej dyskusji,
- stopień zaangażowania w realizację projektu,
- sposób przygotowania i zaprezentowania wybranego tematu.

Ewaluacja programu

Celem ewaluacji jest uzyskanie informacji zwrotnej od uczniów na temat:

- treści programu (o ile wzrosła wiedza o danym zagadnieniu; czy treści pomogły zrozumieć problem; czy zakres treści okazał się wystarczający);
- form pracy (ich atrakcyjności, wpływu na aktywność podczas zajęć);
- metod pracy (czy były atrakcyjne);
- postaw (czy program miał wpływ na postawy odbiorców, jaki był wpływ programu na postawy odbiorców).

Proponuje się, by ewaluacja została przeprowadzona w formie ankiety skierowanej do ucznia. Wnioski płynące z ankiety wykorzystane zostaną w toku dalszej pracy dydaktycznej.

Materiał dydaktyczny

Zajęcia 1

Temat: Pierwsze algorytmy poza komputerem

Streszczenie

Zajęcia są wprowadzeniem do algorytmicznego myślenia. Odbywają się na pokratkowanej macie, na której zaznaczono lub można umieścić różne obiekty w zależności od interpretacji maty przez nauczyciela przekazanej uczniom. Uczniowie najpierw wykonują przykładowe ciągi poleceń (algorytmy) przekazane przez nauczyciela. Następnie, w grupach tworzą własne algorytmy służące do osiągnięcia z pozycji wyjściowej pewnego celu na macie. Algorytmu 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 do zajęć

Nie jest oczekiwane żadne specjalne przygotowanie uczniów w 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.

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. Faktycznie, 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.

Przebieg zajęć

1. *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.
2. *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 mapie zgodne z poleceniami. Po tym kinestetycznym wykonaniu algorytmu, uczniowie układają jego karty na mapie, by przekonać się, że algorytm został dobrze wykonany.
3. *Etap kreatywności uczniów*. Uczniowie, dla wybranych lokalizacji na macie, układają swoje algorytmy/programy z kart, a następnie wykonują je w parach.
4. *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.
5. *Indywidualna prac 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.
6. *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.
7. *Uwaga metodyczna.* KAŻDY uczeń powinien mieć szansę ułożenia algorytmu (programu) i wykonania go na planszy. Jest to ważny moment dla przyszłego myślenia algorytmicznego.
8. *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 zajęć

Podobny scenariusza 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

Zajęcia 2

Temat: Zaczynamy myśleć komputacyjnie

Streszczenie

Zajęcia są wprowadzeniem do myślenia, które ma cechy myślenia komputacyjnego. W preambule do podstawy programowej 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 z korzyścią dla innych dziedzin, 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 do zajęć

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].

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

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

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

Zajęcia są wprowadzeniem do komputacyjnego myślenia. Polecamy lekturę artykuły na ten temat w Załączniku 2. Proponowane zajęcia z łamigłówką Sudoku znakomicie umożliwiają uczniom pierwsze kroki w myśleniu abstrakcyjnym (nie ważne 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 kadym wierszu i w każdej kolumnie mają być umieszczone różne obrazki różne cyfry.

Uwaga. Wszystkie Sudoku w w zaproponowanych aplikacjach mają rozwiązania.

Przebieg zajęć

Cele zajęć są osiągane etapowo:

1. *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 podana zasadą, by się przekonać, czy dostatecznie jasno ją wyjaśnił uczniom.
2. *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.
3. *Indywidualna prac 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.
4. *Bardziej zaawansowane zajęcia.* Zestawy 3 i 4 są trudniejsze, nauczyciel podsuwa je uczniom, którzy szybko wykonali te pierwsze.
5. *Uwaga metodyczna.* KAŻDY uczeń powinien bez specjalnego kłopotu wykonać zestaw pierwszy. Ewentualnie, nauczyciel prosi innego ucznia o pomoc tym, którzy mają kłopoty.
6. *Dyskusja podsumowująca.* Zajęcia kończą się rozmową między uczniami, jak podobały się im łamigłówki, gdzie spotkali 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 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.

Zajęcia 3

Temat: Pierwsze kroki w programowaniu

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 do zajęć

Uczniowie mieli już zajęcia edukacji informatycznej, przynajmniej w rodzaju opisanych w Zajęciach 1, poświęcone pojęciom algorytm i program, np. układnych poza komputerem.

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, którzy nie czytają biegle – bloki, z których składane sa 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 Godzina Kodowania zalecamy nie wychodzić poza lekcje 1-10.

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

Przebieg zajęć

1. *Rozgrzewka*. Na początku zajęć uczniowie mogą wykonać kolejne zadania z:aplikacji http://www.mauthor.com/present/6494370079703040
2. *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.
3. *Indywidualna prac 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.
4. *Bardziej zaawansowane zajęcia.* Uczniom, którzy szybko wykonali te pierwsze łamigłówki, nauczyciel może polecić dalsze. To motywuje uczniów, wpływając na ocenę ich postępów.
5. *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.
6. *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 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 korzystani 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.

Zajęcia 4

Temat: Pierwsze kroki w środowisku Scratcha

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 nieco własny – liczy się kreatywność ucznia, a także współpraca, wymiana pomysłów, wspólne działania.

Przygotowanie do zajęć

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.

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 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.

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

1. *Rozgrzewka 1*. (5 min) Uczniowie wykonują wybraną łamigłówkę z Godziny Kodowania dla przypomnienia sobie, jak wygląda i funkcjonuje srodowisko tej inicjatywy.
2. *Wprowadzenie nauczyciela*. Następnie nauczyciel krótko przybliża budowę środowiska programowania w Scratchu.

3. *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.
4. *Indywidualna prac 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.
5. *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.
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. *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 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 Zajęcia 6 proponujemy wykonanie projektu w środowisku Scratcha, którego efektem są konstrukcje graficzne o pewnych własnościach matematycznych.

Zajęcia 5

Temat: Przejście od Godziny Kodowania do Pythona

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 znacznie.

Przygotowanie do zajęć

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

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.

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 znacznie. 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 prac 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.

   Uruchamiają programy z Pythonie i ewentualnie debagują je.
4. *Bardziej zaawansowane zajęcia.* Uczniom, którzy szybko wykonali zadanie proponujemy modyfikację według własnego uznania, na przykład uwzględniając inne figury do tworzenia płatków śniegu.
5. *Uwaga metodyczna.* KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie odpowiadającym programowi blokowemu.
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 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 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++. To 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.

Zajęcia 6

Temat: Przejście od Scratcha do Pythona

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. Te zajęcia 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 znacznie.

Przygotowanie do zajęć

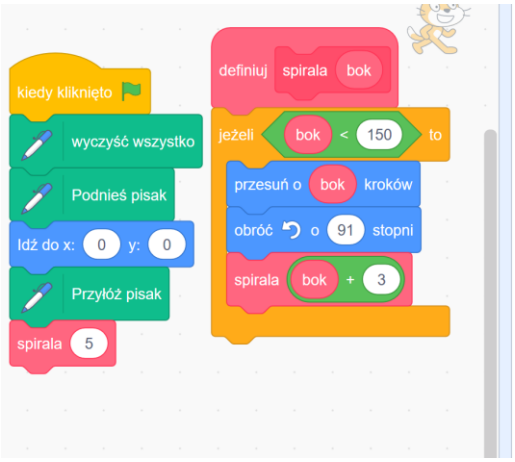
Uczniowie potrafią programować w Scratchu. Znają również środowisko programowania języka Python i potrafią również pisać proste programy w tym środowisku z wykorzystaniem modułu turtle.

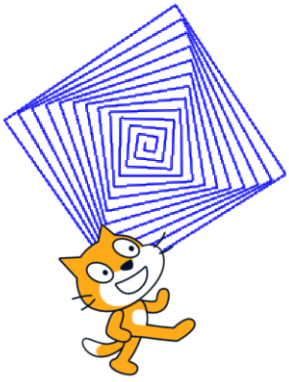
Nauczyciel wcześniej powinien dobrze poznać środowiska Scratcha i Pythona. Nauczyciel powinien mieć przygotowany przykład przejścia od instrukcji w blokach do tekstowego zapisu instrukcji z wykorzystaniem modułu turtle w Pythonie. Poniżej przykład takiego przejścia.

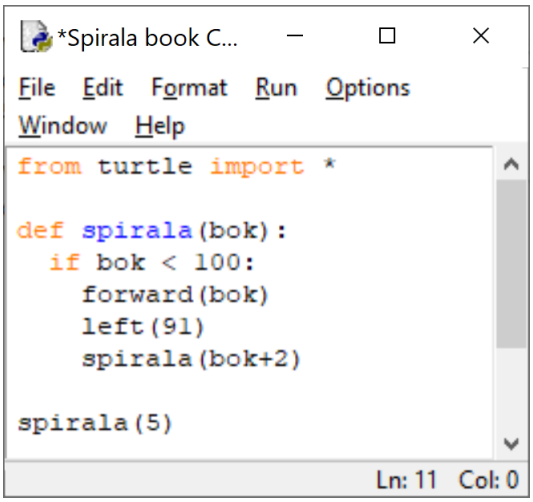
Programy w Scratchu i Pythonie służą do rysowania spirali – widać na rysunkach taki sam efekt. W obu programach zastosowano rekurencje końcową. Dodatkowy program w Pythonie wykonuje ten sam rysunek – rekurencję zastąpiono przez instrukcje warunkową.

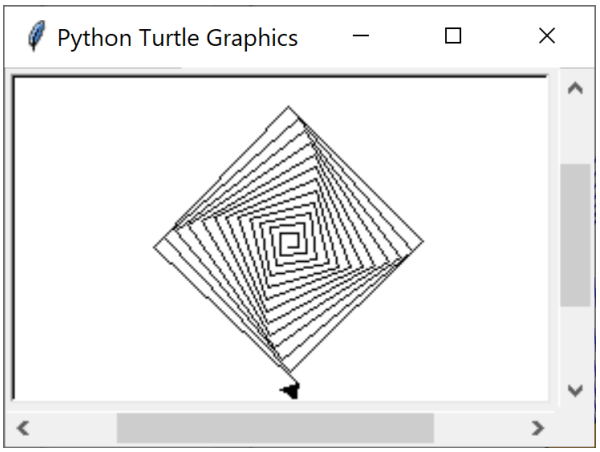
Programy te realizują ideę Paperta z 1980 roku, że nawet najmłodszym uczniom można przybliżyć pojęcie nieskończoności – po usunięciu z podanych programów warunku o zakończeniu rysowania, powstanie nieskończona spirala.

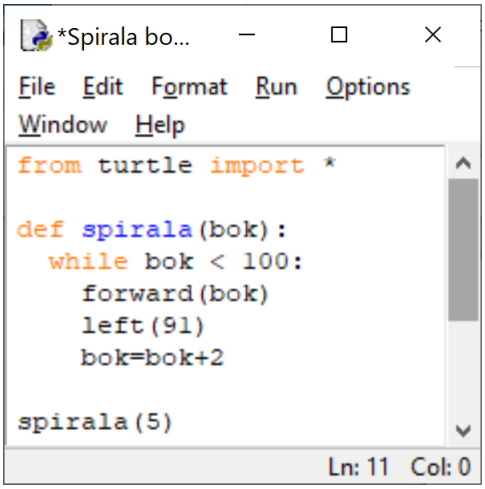
W obu programach rekurencyjnych występuje funkcja (definiuj i def), którą mają zdefiniować uczniowie.











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 znacznie. 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

1. *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.
2. *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, wprowadzić pojęcie funkcji. W Scratchu definiują to własny blok. Ponadto, dość naturalnie funkcja może przyjąć postać rekurencyjną. Więcej szczegółów na ten temat zawarto w książce 6.
3. *Bardziej zaawansowane zajęcia.* Uczniom, którzy szybko wykonali zadanie proponujemy modyfikację według własnego uznania, na przykład uwzględniając inne figury do tworzenia spiral.
4. *Uwaga metodyczna.* KAŻDY uczeń powinien zakończyć zajęcia programem w Pythonie odpowiadającym programowi blokowemu w Scratchu.
5. *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.
6. *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 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 te 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++. To 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.

Zajęcia 7

Temat: Wyszukiwanie i porządkowanie informacji

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 do zajęć

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

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.

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 komutacyjnego, 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, ze 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 porzadkuje 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 rozwiazywania 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 rozwiazywaniu problemów niemal w każdej dziedzinie, i nie tylko za pomocą komputerów.

Modyfikacja zajęć

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

Zajęcia 8

Temat: Wyszukiwanie przez połowienie

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. Ta strategia prowadzi do bardzo szybkich metod obliczeniowych.

Przygotowanie do zajęć

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

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.

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 określenia, jak dobra jest to metoda w porównaniu 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 za 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 nie znaleziona 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 realizacja tej metody.

   Wcześniej wspólnie opracowują specyfikację tego zadania, czyli określają, jakie mają być dane *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 opisane w jednej z jednostek tematycznych dla szkół ponadpodstawowych.

Modyfikacja zajęć

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

Zajęcia 9

Temat: Reprezentacja liczb naturalnych w komputerze

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 do zajęć

Uczniowie powinni posiadać przynajmniej podstawowe umiejętności programowania w języku Python.

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.

Zajęcia mogą się zacząć pytania, które może wydawać się 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):

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

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 $(a_n a_{n-1} ... a_1 a_0)_b$ zapisanej w tym systemie należą do zbioru $\{0, 1, ..., b-1\}$, a jej wartość dziesiętna a jest określona następującą równością:

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

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

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

a ciąg cyfr $(a_n a_{n-1} ... a_1 a_0)_2$ 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 $a_n \bullet 2^{n-1} + a_{n-1} \bullet 2^{n-2} + ... + a_1$, 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 poniższej tabeli 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:

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

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:

$$(1001101)_2 = 1\bullet 2^6 + 0\bullet 2^5 + 0\bullet 2^4 + 1\bullet 2^3 + 1\bullet 2^2 + 0\bullet 2^1 + 1$$
$$= (((((1\bullet 2 + 0)\bullet 2 + 0)\bullet 2 + 1)\bullet 2 + 1)\bullet 2 + 0)\bullet 2 + 1 = 77$$

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

2 * 2 = * 2 = + 1 = * 2 = + 1 = * 2 = * 2 = + 1 =

Dodatkowym ćwiczeniem dla uczniów może być obliczanie sumy liczb danych w postaci binarnej – pozostawmy to zadanie do samodzielnego wykonania. Zasada dodawania takich liczb nie równi 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, ze 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 jaka 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.

   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 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 przedstawianywch 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 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.

Zajęcia 10

Temat: Wydawanie reszty – algorytm w arkuszu kalkulacyjnym

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 która 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 do zajęć

Uczeń potrafi zapisać w arkuszu kalkulacyjnym proste obliczenia.

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

Dla własnej 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:	

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 znaleźć jest 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 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.

Materiały dodatkowe

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 do wersji wydawanej 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.