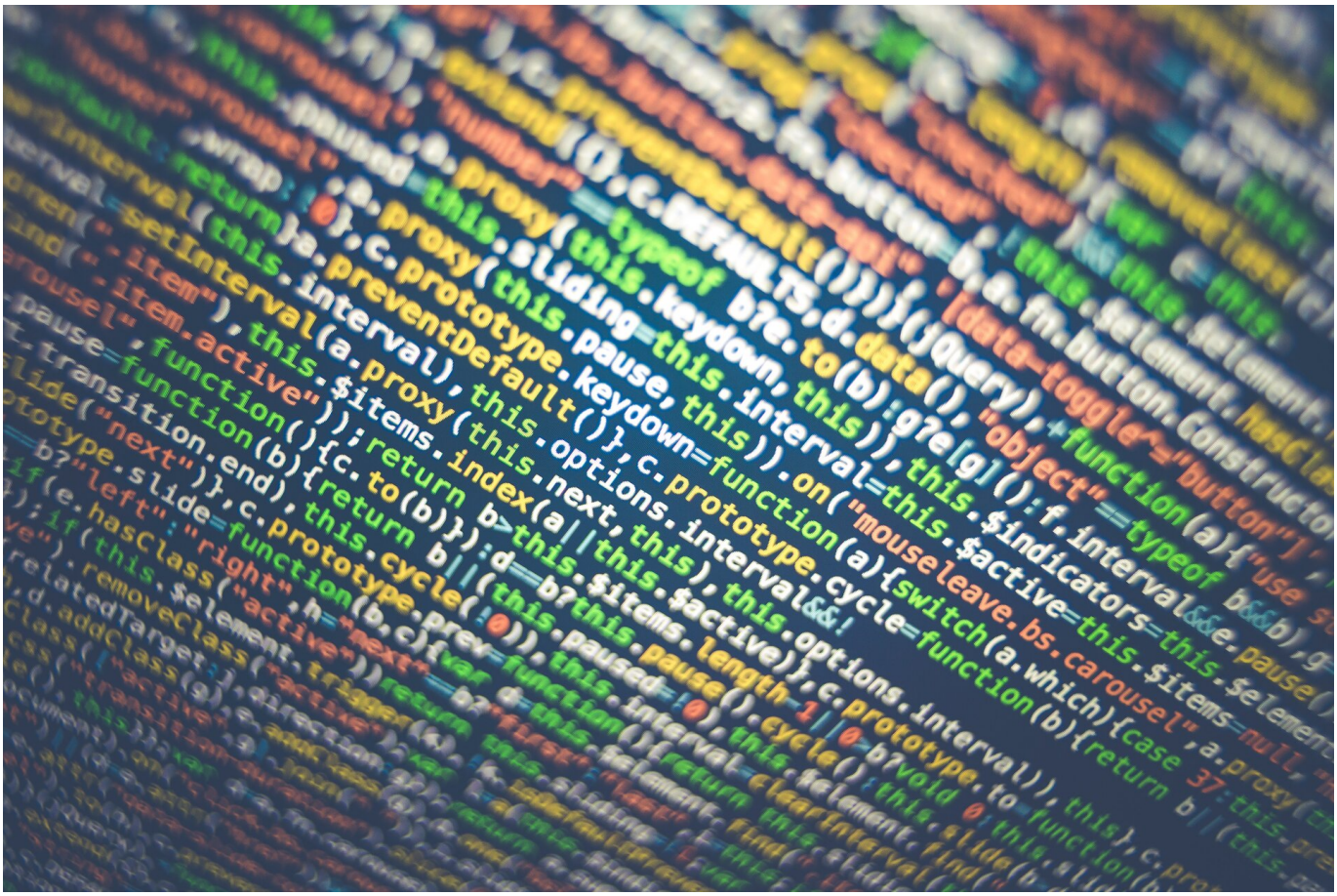

Paradygmaty programowania

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Markus Spiske, domena publiczna.

Sposób pisania programów zmieniał się na przestrzeni lat. Pierwsze programy komputerowe znacznie różniły się od współczesnych – u zarania epoki komputerów były one ciągami rozkazów wykonywanych jeden po drugim. Początkowo nie używano na przykład instrukcji warunkowych, zamiast tego konieczne było używanie instrukcji skoków, co utrudniało zrozumienie pisanego kodu.

Wraz z rozwojem technologii pojawiły się nowe wzorce, na podstawie których programiści pisali kod. W tym e-materiale skoncentrujemy się właśnie na współczesnych modelach programowania.

Twoje cele

- Wyjaśnisz, czym jest paradygmat programowania.
- Dobierzesz odpowiedni paradygmat programowania w celu rozwiązania postawionego problemu.
- Scharakteryzujesz, w jaki sposób wybierać paradygmat programowania w zależności od postawionego problemu.
- Przeanalizujesz zależności między paradygmatami programowania.

Przeczytaj

Wybrane paradygmaty programowania

Zanim przystąpimy do omawiania paradygmatów programowania, wyjaśnijmy co oznacza samo pojęcie paradygmat.

Greckie słowo *parádeigma* oznacza wzór lub przykład. Według filozofa Thomasa Kuhna paradygmaty są zbiorami pojęć i teorii składających się na podstawy określonej dziedziny nauki. W ujęciu programistycznym termin paradygmat jest używany w celu określenia zbioru mechanizmów, które są dostępne podczas pisania programu. Niektóre z takich mechanizmów programista może zdefiniować poprzez narzucone przez niego dobre praktyki oraz język programowania, który dostarcza konstrukcji ułatwiających tworzenie kodu. Wybierając dany paradygmat programowania, programista określa sposób realizacji przepływu sterowania i wykonywania programu komputerowego.

Początkowo programy wykorzystywały paradygmat programowania imperatywnego. Opisywały one proces wykonywania jako sekwencję niskopoziomowych instrukcji zmieniających stan programu. Programy z uwagi na stan zaawansowania technologicznego nie przypominały współczesnych aplikacji. Zadania, jakie miały zrealizować programy uruchamiane na pierwszych komputerach, sprowadzały się do wykonania sekwencji instrukcji i podania końcowego wyniku.

W miarę upływu czasu i rozwoju technologii zmieniały się zadania stawiane komputerom, a programy stawały się bardziej skomplikowane. Pojawiła się potrzeba pisania czytelnego kodu. Powstawały wzorce programowania, czyli paradygmaty – dostosowywano je do rozwiązywania coraz to nowych problemów.

Obecnie można wyróżnić wiele paradygmatów programowania. Omówimy kilka z nich:

- programowanie strukturalne,
- programowanie proceduralne,
- programowanie funkcyjne,
- programowanie obiektowe.

Programowanie strukturalne

Programowanie strukturalne polega na podziale kodu na hierarchicznie ułożone bloki z jednym punktem wejścia i jednym lub wieloma punktami wyjścia. Wprowadzone zostały konstrukcje blokowe (typu `if...then...else`, pętli: `while`, `for`, `repeat`) oraz podprogramy – zestawy instrukcji, których wykonanie można zlecić w dowolnym momencie. Wykorzystanie takich elementów pozwala zrezygnować z posługiwania się

instrukcjami skoków warunkowych lub bezwarunkowych (takich jak `goto` w języku C), których użycie jest konieczne w kodzie maszynowym.

Dzięki wprowadzeniu konstrukcji blokowych program lepiej odzwierciedla schemat blokowy, a w rezultacie staje się bardziej czytelny. Istotne jest także wyeliminowanie z kodu instrukcji skoków – ich nieprzemyślane użycie może powodować błędy w działaniu programu.

Strukturalność zakłócają instrukcje `break` oraz `continue`, które przerywają pętle. Działają one jak niezalecane instrukcje skoków. Używamy ich jednak, gdyż potrafią podnieść czytelność kodu. Należy jednak uważać przy ich stosowaniu, aby nie utworzyć ukrytych pułapek w programie. Zaleca się stosowanie tych instrukcji na początku lub końcu bloku, z którego mają nas wyprowadzić.

Programowanie proceduralne

Programowanie proceduralne jest rozwinięciem programowania strukturalnego. W kodzie pojawiają się wywołania procedur, które zastępują podprogramy. Procedura jest rozszerzeniem podprogramu – pozwala na przekazywanie mu parametrów. Dzięki temu rezygnuje się ze sztywno zdefiniowanych bloków kodu, zyskując większą elastyczność i możliwość odmiennego działania podprogramów, zależną od przekazanych parametrów.

Programowanie funkcyjne

Programowanie funkcyjne swoje początki zawdzięcza powstaniu rachunku lambda. Znaczącym krokiem w rozwoju programowania było zaimplementowanie w językach programowania funkcji matematycznych. Pojęcie funkcja oznacza procedurę mogącą zwracać pewną wartość. Ułatwiło to tworzenie funkcji rekurencyjnych. W rezultacie przestaje nas interesować, w jaki sposób działa podprogram (procedura/funkcja). Ważny jest rezultat działania, czyli zwracana wartość.

Programowanie obiektowe

W przypadku omówionych dotychczas technik programowania dane są oddzielone od funkcji (podprogramów). Funkcji służącej do dodawania dwóch liczb możemy przekazać dowolne dwie zmienne całkowite lub zmiennoprzecinkowe – i otrzymamy wynik.
W przypadku programowania obiektowego jest inaczej.

Samo pojęcie obiekt oznacza kontener na dane (zmienną lub zestaw zmiennych – mogą być one różnych typów), któremu towarzyszą funkcje zaprojektowane specjalnie w celu przetwarzania zawartości kontenera. Oficjalna definicja obiektu brzmi następująco: jest to element łączący stan (dane) z opisem ich zachowania (funkcjami).

Posłużmy się przykładem: kontener zawiera trzy liczby zmiennoprzecinkowe, a zdefiniowano dla niego funkcje obliczania sum i iloczynów liczb oraz wyznaczania wyróżnika równania kwadratowego z wszystkich elementów. Co istotne, żadna spośród trzech wymienionych funkcji nie może być wywołana dla dowolnych zmiennych, lecz wyłącznie dla liczb wchodzących w skład obiektu (czyli zapisanych w kontenerze). Definicja przytoczonego kontenera może być następująca:

```
Klasa TrzyLiczby
    zmienne całkowite a, b, c

    metoda suma()
        zwróć a + b + c

    metoda iloczyn()
        zwróć a * b * c

    metoda wyróżnik_równania_kwadratowego()
        zwróć b * b - 4 * a * c
```

Gdzie metoda jest funkcją, która zawsze jest własnością obiektu. Co oznacza, że nie może istnieć metoda bez przypisanego do niej obiektu.

Istotą programowania obiektowego jest budowanie aplikacji z komunikujących się ze sobą obiektów. Taka metoda sprawdza się podczas pisania współczesnych programów, wyposażonych zazwyczaj w interfejs graficzny. Składa się on z wielu współpracujących ze sobą elementów: okien, przycisków, list jedno- lub wielokrotnego wyboru itp. Dla potrzeb narzędzi służących do pisania programów zgodnie z paradygmatem obiektowym tworzone są obszerne biblioteki zawierające gotowe obiekty, które można dowolnie wykorzystywać.

Słownik

kod maszynowy

zestaw rozkazów, za których wykonanie odpowiada bezpośrednio procesor; składa się z ciągu zer i jedynek, które są interpretowane jako rozkazy i ich argumenty; generowany jest w procesie kompilacji lub w wyniku wykonywania kodu interpretowanego

makro assembler

niskopoziomowy język programowania; w odróżnieniu od klasycznego assemblera, w którym używamy tylko instrukcji procesora, makro assembler zawiera uproszczone konstrukcje (makra), które dopiero później są zamieniane na instrukcje procesora

rachunek lambda

formalny system logiki matematycznej wprowadzony w latach 30. XX w. przez matematyka Alonzo Churcha; służy do badania zagadnień związanych z podstawami matematyki; algorytmy zapisane w rachunku lambda można zaimplementować na maszynie Turinga

rejestr procesora

obszar pamięci znajdujący się wewnątrz procesora, służący do przechowywania danych tymczasowych; procesor kopiuje dane z pamięci do rejestru w celu ich przetwarzania

Prezentacja multimedialna

Polecenie 1

Zastanów się, jak mogłyby wyglądać proste programy (np. program wyświetlający napis *Hello world!*) zapisane z wykorzystaniem różnych paradygmatów programowania. Zapoznaj się z prezentacją.

1

Źródło: Clay Banks, unsplash.com, CC0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

W prezentacji przedstawimy dwa proste programy, które zapiszemy z wykorzystaniem różnych paradygmatów programowania.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Jak wygląda opracowany zgodnie z różnymi paradygmatami kod programu wypisującego komunikat `Hello world!`?

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie strukturalne

W przypadku programowania strukturalnego należałoby posłużyć się podprogramem. Przed laty wypisywanie komunikatów następowało po wywołaniu funkcji systemowych. Były to podprogramy, którym przekazywano tekst. Odbywało się to jednak w sposób odmienny niż w przypadku późniejszych procedur. Najpierw umieszczano w pamięci komputera tekst przeznaczony do wyświetlenia. Następnie zapisywano adres komunikatu w określonym miejscu (np. w rejestrze procesora).

Pseudokod programu mógł wyglądać następująco:

```
1
2 tekst ← "Hello world!"
3 załaduj adres zmiennej
  tekst do rejestru
4 wywołaj wypiszTekst
5
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie proceduralne

W przypadku programowania proceduralnego przekazywanie parametrów jest o wiele prostsze. Nie trzeba już umieszczać tekstu w specjalnym miejscu w pamięci --- odpowiada za to kompilator.

Oto przykładowy kod programu:

```

wypiszTekst("Hello
world!")

```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie funkcyjne

Programowanie funkcyjne nie wprowadza w tym przypadku znaczących zmian. Funkcja, która wyświetla tekst, może być typu `void`, ale może też na przykład zwrócić wartość 1 w przypadku, gdy wypisywanie komunikatu się nie powiodło.

W efekcie otrzymamy potwierdzenie wykonania zadania:

```

jeżeli wypiszTekst("Hello
world!") = 1
    // błąd
    zakończ działanie
programu
kontynuuj wykonywanie
programu

```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie obiektowe

Nowoczesne języki programowania obsługujące mechanizmy obiektowe zawierają cały zestaw funkcji wbudowanych --- na przykład związanych z wyświetlaniem komunikatów w konsoli. Odpowiednia funkcja (metoda) może zostać wywołana na rzecz urządzenia `Konsola`, które traktujemy w programie jako obiekt.

```

Konsola.wypisz_tekst("Hell
o world!")

```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Zapoznaj się z przykładowymi kodami programu sumującego dwie liczby.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie strukturalne

```

podprogram wczytaj_do_r1
    załaduj do rejestru r1
wartość od użytkownika

podprogram wypisz_z_r1
    wypisz na ekran
wartość rejestru r1

wczytaj_do_r1
a ← rejestr r1
```

```
wczytaj_do_r1
b ← rejestr r1

c ← a + b
rejestr r1 ← c

wypisz_z_r1

```

Ponieważ programowanie strukturalne bez zastosowania programowania proceduralnego lub funkcyjnego jest spotykane prawie wyłącznie w makro assemblerach, pominęliśmy kod podprogramów, które zostały użyte w powyższym przykładzie. W rzeczywistości podprogramy takie mogłyby zostać dostarczone przez system lub napisane z wykorzystaniem dostępu do pamięci urządzeń wejścia/wyjścia.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

9

Programowanie proceduralne

```

procedura
wczytaj(zmienna_docelowa)
    załaduj do
zmienna_docelowa wartość
od użytkownika

procedura wypisz(zmienna)
    wypisz na ekran
wartość zmienna

procedura
dodaj(zmienna_docelowa,
zmienna_dodawana)
    zmienna_docelowa ←
zmienna_docelowa +
zmienna_dodawana
```

```
10
11 wczytaj(a)
12 wczytaj(b)
13
14 dodaj(a, b)
15
16 wypisz(a)
17
```

Również w tym przypadku pominęliśmy kod procedur wczytujących oraz wypisujących dane. Zakładamy, że funkcjonalność taką dostarcza system. Zwróć uwagę, że w przypadku tego programu tracimy pierwotną wartość zmiennej a w momencie dodania do niej zmiennej b. Jeśli chcielibyśmy ją zachować, musielibyśmy utworzyć jej kopię, np. w nowej zmiennej c.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie funkcyjne

```
1
2 funkcja wczytaj()
3     a ← wprowadź liczbę
4     zwróć a
5
6 funkcja wypisz(zmienna)
7     wypisz na ekran
  wartość zmienna
8
9 a = wczytaj()
10 b = wczytaj()
11
12 c = dodaj(a,b)
13 wypisz(c)
14
```

Zwróć uwagę, że funkcja `wypisz` to nic innego jak procedura.

Źródło: Clay Banks, unsplash.com, CC0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P7aeMLY2a

Programowanie obiektowe

```
1
2  Klasa liczba
3      pole wartość
4
5      ustaw_wartość(zmienna)
6          wartość ← zmienna
7
8      zwróć_wartość()
9          zwróć wartość
10
11 Liczba a, b #Tworzymy dwie
   klasy, a oraz b
12 a.ustaw_wartość(Konsola.wc
   zytaj())
13 b.ustaw_wartość(Konsola.wc
   zytaj())
14
15 a.dodaj(b)
16 Konsola.wypisz(a.zwróć_war
   tość())
17
```

W przypadku programowania obiektowego podejść do takiego programu może być wiele, ponieważ istnieje wiele wzorców przepływu danych w programie. Powyższe rozwiązania są tylko przykładami.

Polecenie 2

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w prezentacji.

Polecenie 3

Przygotuj notatkę, w której opiszesz dwa inne wybrane przez siebie paradygmaty programowania.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż, który paradygmat najbardziej pasuje do zapisanego za pomocą pseudokodu programu.

```
1 Człowiek człowiek("Ala");
2 Kot kot("Puszek");
3
4 człowiek.pogłaszcz(kot);
```

- ◯ programowanie obiektowe
- ◯ programowanie proceduralne
- ◯ programowanie strukturalne
- ◯ programowanie funkcyjne

Ćwiczenie 2

Do paradygmatów programowania przyporządkuj charakterystyczne dla nich pojęcia.

programowanie proceduralne	procedura
programowanie strukturalne	podprogram
programowanie obiektowe	metoda
programowanie funkcyjne	funkcja

Ćwiczenie 3

Wskaż, która metoda programowania jako pierwsza wyeliminowała konieczność używania instrukcji skoku.

- ◯ programowanie funkcyjne
- ◯ programowanie proceduralne
- ◯ programowanie strukturalne
- ◯ programowanie obiektowe

Ćwiczenie 4

Uzupełnij tabelkę: wpisz P jeśli zdanie jest prawdziwe, F jeśli jest fałszywe.

Zdanie	P/F
W programowaniu obiektowym po raz pierwszy wprowadzono pętle i instrukcje warunkowe.	
Programowanie funkcyjne uniemożliwia pisanie podprogramów i procedur.	
W programowaniu funkcyjnym program składa się tylko z podprogramów i skoków.	
Istotą programowania obiektowego jest budowanie aplikacji z obiektów, które komunikują się ze sobą.	

Ćwiczenie 5

Przyporządkuj pojęciom odpowiednie wyjaśnienia.

procedura	umożliwia przekazanie parametrów
podprogram	określa zachowanie obiektu
funkcja	umożliwia grupowanie instrukcji
metoda	umożliwia przekazanie parametrów oraz zwraca wartość

Ćwiczenie 6

Wskaż, który paradygmat najbardziej pasuje do zapisanego za pomocą pseudokodu programu.

```
sumuj(a, b)
    zwróć a+b

wypisz(sumuj(10, 15))
```

- ◯ programowanie proceduralne
- ◯ programowanie funkcyjne
- ◯ programowanie strukturalne
- ◯ programowanie obiektowe

Ćwiczenie 7

Wskaż, które z poniższych instrukcji mogą zakłócać strukturalność programu, gdy nie są użyte prawidłowo.

- ☐ kontynuuj (continue)
- ☐ zwróć (return)
- ☐ jeżeli... w przeciwnym wypadku (if ... else)
- ☐ przerwij (break)

Ćwiczenie 8

Wskaż, dlaczego nie jest zalecane używanie instrukcji skoku.

- [ ] Ponieważ zmniejszają czytelność programu.
- [ ] Ponieważ łatwo stracić kontrolę nad poprawnym przepływem sterowania w programie.
- [ ] Ponieważ powodują one wycieki pamięci.
- [ ] Ponieważ zbyt łatwo się ich używa.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Paradygmaty programowania

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
>
> 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
>
> III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi.
>
> Zakres podstawowy. Uczeń:
>
> 1) zapoznaje się z możliwościami nowych urządzeń cyfrowych i towarzyszącego im oprogramowania;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wyjaśnisz, czym jest paradygmat programowania.
- Dobierzesz odpowiedni paradygmat programowania w celu rozwiązania postawionego problemu.
- Scharakteryzujesz, w jaki sposób wybierać paradygmat programowania w zależności od postawionego problemu.
- Przeanalizujesz zależności między paradygmatami programowania.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Paradygmaty programowania". Uczniowie zapoznają się z multimedium w sekcji „Prezentacja multimedialna".
2. Chętny lub wybrany uczeń przygotowuje rozwiązanie polecenia nr 1 z sekcji „Przeczytaj". Będzie pełnił rolę eksperta podczas zajęć.

Faza wstępna:

1. Prowadzący wyświetla na tablicy interaktywnej zawartość sekcji „Wprowadzenie” i omawia cele do osiągnięcia w trakcie lekcji.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj” wyświetlone na tablicy. Odnosząc się do treści zawartej w sekcji „Przeczytaj”, uczniowie w parach konstruują alternatywny przykład, rozwiązują go i ewentualnie wskazują możliwości jego zastosowania. Rezultaty omawiane są na forum klasy.
2. Uczniowie zapoznają się z treścią prezentacji multimedialnej, a później dyskutują i przedstawiają swoje pytania.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1–5. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie”. W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 6–8 z sekcji „Sprawdź się”.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.