Czy algorytm jest poprawny?

- Wprowadzenie
- Przeczytaj
- Schemat interaktywny
- Sprawdź się
- Dla nauczyciela

Źródło: Rodion Kutsaev, unsplash.com, domena publiczna.

Poznając wybrany język oprogramowania, zgłębiamy też reguły algorytmiki. Weryfikacja poprawnego działania stworzonego algorytmu pozwala nam oszacować, czy dla prawidłowych danych algorytm zwróci oczekiwany przez nas wynik. Bezbłędnie zaprojektowany algorytm ułatwia napisanie poprawnie działającego kodu.

Więcej informacji na temat algorytmów i algorytmiki znajdziesz w e-materiałach:

- Algorytmika,
- Algorytmy na co dzień,
- Własności algorytmów.

Twoje cele

- Wyjaśnisz pojęcia: niezmiennik pętli, częściowa poprawność algorytmu oraz własność stopu.
- Przeanalizujesz, jak wygląda poprawny algorytm i podasz jego cechy.
- Ocenisz poprawność działania kilku prostych algorytmów.

Przeczytaj

Co oznacza poprawność algorytmu?

To, co intuicyjnie rozumiemy jako poprawność algorytmu, formalnie określamy mianem **całkowitej poprawności**. Składają się na nią dwa elementy: własność stopu oraz częściowa poprawność.

Aby wyjaśnić znaczenie tych terminów, rozważmy najpierw pojęcie specyfikacji algorytmu. Określamy tak ścisły opis danych wejściowych, danych wyjściowych oraz relacji, jakie między nimi zachodzą.

Algorytm powinien działać w sposób poprawny dla danych, które spełniają warunek początkowy – w tej sytuacji zwróci on dane wyjściowe spełniające warunek końcowy, czyli wynik osiągnięty w sposób opisany przez specyfikację. Nie oznacza to, że algorytm nie będzie działał dla danych niespełniających warunku początkowego - po prostu nie jest on do nich przystosowany i nie można na nim polegać.

Częściowa poprawność oznacza, że jeżeli algorytm zakończy się wykonywać, a dane wejściowe spełniały warunek początkowy, to zwrócony przez niego wynik będzie poprawny. **Własność stopu** oznacza, że dla każdych danych wejściowych spełniających warunek początkowy algorytm skończy się wykonywać po skończonej liczbie kroków.

Algorytm jest całkowicie poprawny, jeżeli jest jednocześnie częściowo poprawny oraz ma własność stopu. Oznacza to, że jeżeli otrzymał on poprawne dane, zwróci poprawny wynik po skończonej liczbie kroków.

Zagadnienie poprawności omówmy najpierw na podstawie następującego algorytmu.

Specyfikacja problemu:

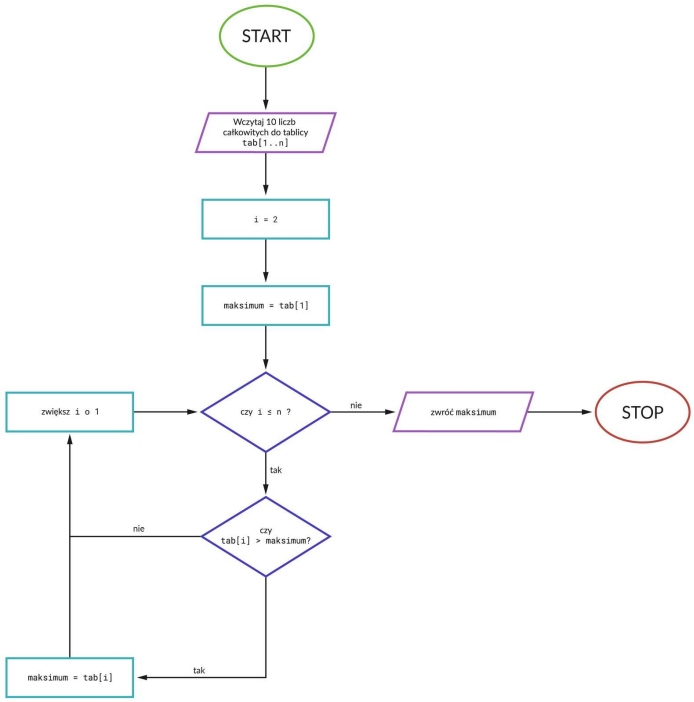
Dane:

- `n` – liczba elementów w tablicy; liczba naturalna
- `tab` – tablica `n` liczb całkowitych

Wynik:

Algorytm zwraca największą liczbę występującą w tablicy.

Przedstawiony na schemacie blokowym algorytm wygląda następująco:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Ten sam algorytm możemy przedstawić również za pomocą pseudokodu:

```
tab[1..n] ← wprowadź n liczb całkowitych do tablicy
i ← 2
maksimum ← tab[1]
dopóki i ≤ n wykonuj:
    jeżeli tab[i] > maksimum:
        maksimum ← tab[i]
    i ← i + 1
zwróć maksimum
```

Dowodzenie poprawności

Przeprowadzenie dowodu na całkowitą poprawność sprowadza się do udowodnienia częściowej poprawności oraz własności stopu.

By sprawdzić poprawność algorytmu, dzielimy go na fragmenty. Za pomocą zapisów matematycznych sprawdzamy, czy dane wejściowe dają odpowiednie dane wyjściowe. Uzyskany końcowy wynik powinien być zgodny ze specyfikacją.

Aby przybliżyć istotę tych dowodów, przeanalizujmy działanie podanego wyżej algorytmu dla przykładowych danych:

iteracja	tab	i	maksimum
przed iteracją 1	`[1,2,3,9,10,6,4,1,2,7]`	2	1
po iteracji 1	`[1,2,3,9,10,6,4,1,2,7]`	3	2
po iteracji 2	`[1,2,3,9,10,6,4,1,2,7]`	4	3
po iteracji 3	`[1,2,3,9,10,6,4,1,2,7]`	5	9
po iteracji 4	`[1,2,3,9,10,6,4,1,2,7]`	6	10
po iteracji 5	`[1,2,3,9,10,6,4,1,2,7]`	7	10
po iteracji 6	`[1,2,3,9,10,6,4,1,2,7]`	8	10
po iteracji 7	`[1,2,3,9,10,6,4,1,2,7]`	9	10
po iteracji 8	`[1,2,3,9,10,6,4,1,2,7]`	10	10
po iteracji 9	`[1,2,3,9,10,6,4,1,2,7]`	11	10

Na koniec zostaje zwrócona wartość zmiennej `maksimum`.

Możemy zauważyć, że zwrócona przez algorytm wartość jest poprawna – liczba 10 jest rzeczywiście największa spośród przykładowych danych wejściowych.

> Nic nie stoi na przeszkodzie, abyśmy sprawdzili działanie tego algorytmu dla innego zestawu danych. Nieskończenie wiele razy moglibyśmy sprawdzać, co zostanie zwrócone przez algorytm, a i tak nie mielibyśmy stuprocentowej pewności, że ten algorytm jest poprawny. Zawsze bowiem może okazać się, że pominęliśmy pewien skrajny przypadek – sytuację, w której nasz algorytm zwróci niepoprawny wynik, lub (co gorsze) nigdy nie przestanie się wykonywać. Właśnie dlatego przywiązuje się dużą uwagę do poprawności algorytmów i przeprowadza się dowód na całkowitą poprawność uwzględniający wszystkie przypadki zgodne ze specyfikacją algorytmu.

Zachodzi tu pewna prawidłowość – wartość zmiennej `maksimum` nigdy się nie zmniejsza. Co więcej, możemy stwierdzić, że jej wartość w każdym wykonaniu pętli będzie równa

największej liczbie napotkanej **do tej pory**, czyli w elementach tablicy o indeksach od 1 do `i` -1.

W momencie, w którym pętla kończy się wykonywać, zmienna `maksimum` zawiera największą wartość spośród elementów tablicy o indeksach od 1 do 10, czyli największą wartość w całej tablicy.

Taką zależność nazywamy **niezmiennikiem pętli**. Jest to warunek, który jeśli jest spełniony przed wykonaniem iteracji, będzie spełniony również po niej. Może zostać on użyty w tzw. metodzie niezmienników, która jest jednym ze sposobów na udowodnienie częściowej poprawności algorytmu.

> Znalezienie niezmiennika pętli może być problematyczne nawet w relatywnie prostych algorytmach. Wynika to z faktu, że nie jest on oczywisty na pierwszy rzut oka. Często znalezienie takiego niezmiennika wymaga przeanalizowania działania algorytmu na kilku zestawach danych wejściowych, a następnie wyciągnięcie odpowiednich wniosków.
> **Ciekawostka**
>
> Użycie niezmienników pętli to forma dowodu indukcyjnego. Składa się on z dwóch kroków: kroku bazowego oraz kroku indukcyjnego. W kroku bazowym udowadniamy, że pewna teza T jest prawdziwa dla 0, tj. $T\left(0\right)$ jest prawdziwe. Krok indukcyjny polega na wykazaniu, że jeżeli teza $T\left(k\right)$ jest prawdziwa dla pewnej liczby naturalnej k, to teza $T\left(k+1\right)$ również musi być prawdziwa.

Wiemy już, że wartość zmiennej `maksimum`zawiera w każdej iteracji pętli największą wartość w pewnym przedziale, więc po wyjściu z pętli będzie ona zawierała największą wartość w całej tablicy. Pytanie brzmi: czy pętla kiedykolwiek się zakończy?

W przypadku omawianego algorytmu własność stopu możemy udowodnić w następujący sposób:

1. Algorytm skończy się wykonywać, kiedy `i > n`.
2. `n` ma określoną wartość.
3. W każdej iteracji pętli wartość zmiennej `i` zwiększa się o 1.
4. Zatem algorytm zakończy się po skończonej liczbie iteracji – tym samym udowodniliśmy własność stopu.

> **Ważne!**
>
> Zwróć szczególną uwagę na dokładność naszego dowodu. Samo zwiększanie zmiennej nie wystarczy, aby dowód był prawidłowy, musi się ona zwiększać o stałą wartość. Tak samo liczba `n`, użyta w krokach pierwszym i drugim, musi być stała i skończona. Inaczej nasz dowód nie będzie wystarczający.

Sprawdziliśmy, czy nasz algorytm posiada własność stopu – zatrzymuje się w momencie, w którym wartość zmiennej `i` przekracza wartość `n`. Korzystając z niezmiennika pętli,

wykazaliśmy również jego częściową poprawność. Możemy zatem stwierdzić, że nasz algorytm jest całkowicie poprawny – za tym stwierdzeniem stoi pewna forma dowodu matematycznego.

Dowody na całkowitą poprawność bardziej skomplikowanych algorytmów będą oczywiście odpowiednio dłuższe, lecz na poziomie szkoły ponadpodstawowej analiza na takim poziomie jest wystarczająca.

Przykład algorytmu – obliczanie silni

Silnię liczby naturalnej `n` definiujemy jako iloczyn wszystkich liczb naturalnych od 1 do `n`, zaś zapisujemy jako `n!`. Aby obliczyć jej wartość dla pewnego `n`, możemy wykorzystać następujący algorytm:

Specyfikacja problemu:

Dane:

- `n` – liczba, której silnię obliczamy; liczba naturalna

Wynik:

Algorytm zwraca wartość silni liczby `n`.

Zapis algorytmu za pomocą pseudokodu:

```
n ← wprowadź liczbę naturalną n
wynik ← 1
i ← 2
dopóki i ≤ n wykonuj:
    wynik ← wynik * i
    i ← i + 1
zwróć wynik
```

Rozważmy działanie tego algorytmu dla liczby `n = 5`.

iteracja	**n**	**wynik**	**i**
przed iteracją 1	5	1	2
po iteracji 1	5	2	3
po iteracji 2	5	6	4
po iteracji 3	5	24	5

iteracja	n	wynik	i
po iteracji 4	5	120	6

Na końcu zostaje zwrócony wynik 120, co rzeczywiście jest silnią z liczby 5.

W przypadku tego algorytmu, własność stopu jest analogiczna, jak w algorytmie wyszukiwania największej liczby w tablicy, tj. zmienna `i` zwiększa się o stałą wartość, więc po skończonej liczbie iteracji przekroczy zmienną `n`, która również jest wartością stałą i skończoną.

Kluczem do niezmiennika pętli w tym algorytmie jest zmienna `wynik`. Jest to bowiem silnia z liczby `i`. Możemy na podstawie tych stwierdzeń udowodnić poprawność tego algorytmu, jak również zademonstrować ją na przykładzie. Możemy też wyjaśnić krok po kroku, dlaczego nasz algorytm zwraca poprawne wyniki.

Inne cechy algorytmów

Poza opisanymi, wynikającymi z definicji, właściwościami całkowitej poprawności, możemy wymienić jeszcze kilka cech poprawności algorytmu.

Aspektem algorytmu, który przyjmujemy za oczywisty, jest jego **określoność**. Oznacza to, że czytelnik musi być w stanie zrozumieć i dokładnie odwzorować kolejne etapy wykonywania algorytmu. Aby algorytm był niepoprawnie określony, wystarczy jedna błędna instrukcja. Przykładowo: „wczytaj liczby" nie mówi nam nic o tym, jakie liczby mamy wczytać – całkowite, rzeczywiste, zespolone...? Ile liczb trzeba wczytać? W tej instrukcji powinniśmy zawrzeć konkrety, przez co mogłaby ona przyjąć postać, np. „wczytaj liczbę całkowitą `X` oraz liczbę naturalną `N`".

> Określoność algorytmu umożliwia wykorzystanie komputera, aby zweryfikować, czy faktycznie zwraca on poprawne wyniki dla konkretnych danych wejściowych. Musimy się jednak liczyć z tym, że zapis algorytmu w języku programowania wiąże się z ryzykiem wprowadzenia błędów składniowych, które uniemożliwiają uruchomienie programu, dopóki nie zostaną naprawione, oraz błędów logicznych, wynikających z niepoprawnego przełożenia algorytmu na wybrany język programowania.

Inną ważną cechą algorytmu jest jego **uniwersalność**. Oznacza ona, że algorytm powinien działać dla dowolnych danych spełniających warunki specyfikacji. Dotyczy to również liczby danych wejściowych, np. algorytm ma działać dla ciągu `n`–wyrazowego, a nie dla ciągu zawierającego z góry określoną liczbę wyrazów.

Na szczególną uwagę zasługuje wspomniana wcześniej skończoność algorytmu. W praktyce nie wystarczy, że algorytm **kiedyś** skończy się wykonywać – chcemy, aby stało się to jak najszybciej, przy jak najmniejszej liczbie kroków lub zużywając jak najmniej zasobów takich

jak pamięć, w której przechowywane są zmienne. Dobrze zaprojektowane algorytmy wykonują tylko niezbędne operacje – innymi słowy są **efektywne**.

Słownik

> **algorytm całkowicie poprawny**
>
> algorytm, który jest zarówno częściowo poprawny, jak i posiada własność stopu
> **częściowa poprawność algorytmu**
>
> własność algorytmu, która mówi, że dla poprawnych danych wejściowych, jeżeli algorytm zakończy się wykonywać po skończonej liczbie kroków, to zwróci on poprawne dane wyjściowe
> **niezmiennik pętli**
>
> zdanie logiczne, które jeśli jest prawdziwe przed wykonaniem iteracji pętli, będzie prawdziwe również po jej wykonaniu
> **własność stopu**
>
> cecha algorytmu, która mówi, że dla poprawnych danych wejściowych algorytm zatrzymuje się w skończonym czasie

Schemat interaktywny

Polecenie 1

Zaprojektuj algorytm, który obliczy sumę wszystkich liczb zawartych w tablicy. Tablica ta powinna zawierać n liczb naturalnych. Sprawdź poprawność jego działania dla przykładowych danych, a następnie przeanalizuj jego poprawność.

Specyfikacja problemu:

Dane:

- n – liczba elementów w tablicy; liczba naturalna
- `lista` – tablica zawierająca n liczb naturalnych

Wynik:

Na wyjściu standardowym wyświetlona zostanie liczba przedstawiająca sumę liczb zawartych w tablicy.

Wyjście dla przykładowych danych:

```
35
```

> **Polecenie 2**
>
> Zapisz algorytm z polecenia 1 za pomocą listy kroków.

```
1
```

Polecenie 3

Rozstrzygnij, czy zaprojektowany algorytm jest poprawny. Uzasadnij swoją odpowiedź.

Polecenie 4

Zapisz zaprojektowany algorytm i go przetestuj dla -elementowej tablicy zawierającej wartości `[1, 2, 6, 7, 8, 11]`.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Dokończ zdanie.

Dowód na częściową poprawność algorytmu przeprowadzony przy użyciu niezmienników pętli to przykład...

- dowodu konstruktywnego.
- dowodu przez indukcję.
- dowodu nie wprost.

Ćwiczenie 2

Dokończ zdanie.

Niezmiennik pętli to...

- fragment algorytmu, którego nie można zapisać za pomocą pętli.
- to samo, co warunek wykonywania pętli.
- zdanie logiczne, które jest prawdziwe po wykonaniu iteracji pętli – pod warunkiem, że było prawdziwe przed nią.

Ćwiczenie 3

Zapoznaj się z algorytmem i wykonaj ćwiczenie.

```
1 n ← 2
2 l ← 10
3 dopóki n < l wykonuj:
4     l ← n * l
5 zwróć l
```

Rozstrzygnij, czy zaprezentowany algorytm posiada własność stopu. Odpowiedź uzasadnij.

Ćwiczenie 4

Uzupełnij zdanie, wpisując odpowiednie pojęcie.

[ ] poprawność algorytmu oznacza, że dla poprawnych danych wejściowych algorytm, jeżeli zakończy się wykonywać, zwróci poprawne dane wyjściowe.

Ćwiczenie 5

Zapoznaj się z algorytmem i wykonaj ćwiczenie.

```
wartość ← 1
i ← 1
dopóki i <= n wykonuj:
    wartość ← wartość * i
    i ← i + 1
zwróć wartość
```

Określ, jaki jest rezultat działania algorytmu dla pewnego n będącego dodatnią liczbą całkowitą. Podaj wartość zwróconą przez algorytm dla n = 5, n = 6, n = 8. Czy jesteś w stanie powiedzieć, jaka będzie jego wartość dla n = 51?

Ćwiczenie 6

Uzupełnij zdania odpowiednimi z podanych pojęć.

Dowód na własność stopu [ ] być pisany językiem naturalnym. [ ] on wymagany do udowodnienia całkowitej poprawności algorytmu. Własność stopu [ ], że algorytm jest częściowo poprawny.

Nie jest | nie może | Jest | oznacza | nie oznacza | może

Ćwiczenie 7

Zapoznaj się z algorytmem i wykonaj ćwiczenie.

```
tab[1..10] ← wprowadź liczby naturalne do tablicy
licznik ← 0
i ← 1
dopóki i <= długość(tab) wykonuj:
    jeżeli tab[i] mod 2 = 0
        licznik ← licznik + 1
zwróć licznik
```

Zaprezentowany algorytm podaje, ile jest liczb parzystych w tablicy danych. Jest on jednak niepoprawny. Napraw błąd w algorytmie i sprawdź jego poprawność dla przykładowych danych, a następnie dokonaj analizy jego poprawności.

Dla zainteresowanych

Ćwiczenie 8

Podaj przykład algorytmu, w którym do udowodnienia częściowej poprawności nie można wykorzystać niezmienników pętli.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Czy algorytm jest poprawny?

Grupa docelowa:

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

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

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

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

a) na liczbach: badania pierwszości liczby, zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, działań na ułamkach z wykorzystaniem NWD i NWW,

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

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 pojęcia: niezmiennik pętli, częściowa poprawność algorytmu oraz własność stopu.
- Przeanalizujesz, jak wygląda poprawny algorytm i podasz jego cechy.
- Ocenisz poprawność działania kilku prostych algorytmów.

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: „Czy algorytm jest poprawny?". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów zajęć, przejście do wspólnego ustalenia kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj" wyświetlone na tablicy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny", czyta treść polecenia nr 1 „Zaprojektuj algorytm, który obliczy sumę wszystkich liczb zawartych w tablicy. Tablica ta powinna zawierać n liczb naturalnych. Sprawdź poprawność jego działania dla przykładowych danych, a następnie przeanalizuj jego poprawność. Przetestuj algorytm dla 6-elementowej tablicy zawierającej wartości [1, 2, 6, 7, 8, 11]" i omawia kolejne kroki rozwiązania.
3. **Ćwiczenie umiejętności.** Nauczyciel przechodzi do sekcji „Sprawdź się". Uczniowie indywidualnie rozwiązują ćwiczenia nr 1-4 na czas. Osoba, która poprawnie rozwiąże zadania jako pierwsza, wygrywa, a nauczyciel może nagrodzić ją oceną za aktywność.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

Praca domowa:

1. Uczniowie wykonują polecenia 5-8 z sekcji „Sprawdź się". Następnie sprawdzają poprawność własnych rozwiązań z odpowiedziami.
2. Uczniowie wykonują polecenie 2 z sekcji „Schemat interaktywny".

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedium w sekcji „Schemat interaktywny" do przygotowania się do lekcji powtórkowej.