

Trójkąty

Materiał opisuje projekt "Trójkąty" i składa się z sekcji: "Scenariusz projektu", "Przygotowujemy pierwszą aplikację", "Projektujemy drugą aplikację", "Sortujemy odcinki według długości", "Znajdujemy dwa najkrótsze odcinki i najdłuższy", "Podsumowanie – szacujemy złożoność obliczeniową", "Zadania uzupełniające".

Materiał zawiera 30 zrzutów ekranu, 5 filmów, 21 ćwiczeń, w tym 2 interaktywne.

Trójkąty

Scenariusz projektu

Projekt ten będzie się składał z dwóch aplikacji. Zadaniem użytkownika pierwszej z nich jest odpowiedzenie na proste pytanie, czy z trzech odcinków o podanych długościach (wylosowanych przez duszka) można zbudować trójkąt. Duszek powinien sprawdzić odpowiedź i wyświetlić odpowiedni komunikat.

> **Ćwiczenie 1**
>
> Uruchom poniższą aplikację i sprawdź swoją wiedzę.
>
>
>
>
> Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/b/P17dx7rmo
>
> Źródło: Janusz Wierzbicki, Maciej Borowiecki, licencja: CC BY 3.0.

W drugiej aplikacji duszek postawi przed użytkownikiem trudniejsze zadanie. Odcinków będzie więcej, trzeba odpowiedzieć na pytanie, czy z każdych trzech odcinków można zbudować trójkąt. Także twoje zadanie będzie trudniejsze, ponieważ musisz oprogramować działania duszka, który sprawdza odpowiedź użytkownika. Działanie przykładowej aplikacji możesz obejrzeć na poniższym filmie.



Film dostępny na portalu epodreczniki.pl

Źródło: Janusz Wierzbicki, Maciej Borowiecki, licencja: CC BY 3.0.

G3_1Film1

W obu aplikacjach wykorzystywane jest jedno proste tło składające się z dwóch części (w różnych kolorach). W górnej części znajdują się duszki – przyciski, w dolnej duszek, który sprawdza i informuje o poprawności odpowiedzi.

> **Ćwiczenie 2**
>
> Przygotuj tło do aplikacji. Może być bardziej rozbudowane graficznie, według twojego pomysłu.

> **Ćwiczenie 3**
>
> Przygotuj dwa duszki: przycisk **Tak** oraz przycisk **Nie**.
> Możesz skorzystać z szablonu przycisku z galerii Scratcha i uzupełnić go o odpowiedni napis.

Przygotowujemy pierwszą aplikację

> **Ćwiczenie 4**
>
> Załaduj przygotowane tło oraz duszki – przyciski. Domyślnie niech przyciski będą ukryte. Utwórz także trzy zmienne do pamiętania długości odcinków.

> **Ćwiczenie 5**
>
> Zastanów się, jakie działania powinny być wykonane po uruchomieniu aplikacji, czyli kliknięciu w zieloną flagę. Przygotuj właściwe skrypty.

Należy zadbać o komunikację pomiędzy duszkami. Duszek odpowiedzialny za sprawdzenie odpowiedzi musi wiedzieć, w który przycisk kliknął użytkownik. W Scratchu do komunikacji pomiędzy duszkami można wykorzystać komunikaty. Więcej o nadawaniu i odbieraniu komunikatów możesz się dowiedzieć tutaj.

> **Ćwiczenie 6**
>
> Przygotuj skrypty dla przycisków nadające odpowiednie komunikaty.

> **Ćwiczenie 7**
>
> Przygotuj skrypty dla duszka sprawdzającego odpowiedź, reagujące na otrzymanie komunikatów *tak* i *nie*. Po udzieleniu odpowiedzi należy ukryć przyciski, a więc nadać odpowiedni komunikat (np. *koniec*) informujący je.

> **Ćwiczenie 8**
>
> Przygotuj jeszcze skrypty ukrywające przyciski po otrzymaniu komunikatu *koniec*.

Projektujemy drugą aplikację

Przy większej liczbie odcinków niewygodnie będzie pamiętać ich długości na pojedynczych zmiennych. Użyjemy do tego celu listy. Więcej na temat list możesz się dowiedzieć tutaj.

> **Ćwiczenie 9**
>
> Utwórz listę do pamiętania długości odcinków (np. o nazwie *Odcinki*) oraz zmienną (np. o nazwie *n*), w której będzie przechowywana liczba losowanych odcinków.

Skrypty dla przycisków będą analogiczne jak w poprzedniej aplikacji. Zmienią się dla duszka sprawdzającego odpowiedź, ponieważ inny będzie algorytm sprawdzenia odpowiedzi użytkownika oraz zastosowane struktury danych (zamiast trzech pojedynczych zmiennych lista).

> **Ćwiczenie 10**
>
> Przygotuj skrypt, uruchamiany po kliknięciu w zieloną flagę, losujący najpierw liczbę odcinków, a potem ich długości.
> **Wskazówka:**
> Zakres losowania liczby odcinków dobierz tak, aby były co najmniej cztery i wszystkie elementy listy były równocześnie pokazane na scenie. Pamiętaj, aby przed losowaniem elementów listy usunąć jej poprzednią zawartość.

> **Ćwiczenie 11**
>
> Zastanów się nad algorytmem, jak sprawdzić, czy z każdych trzech odcinków można zbudować trójkąt. Przedyskutuj swoje propozycje z kolegami i koleżankami oraz nauczycielem. Spróbujcie podać więcej niż jedną propozycję rozwiązania. Porównaj następnie wasze propozycje z zawartymi w odpowiedzi.

Sortujemy odcinki według długości

Porządkowanie (sortowanie) to jeden z podstawowych problemów informatycznych. Istnieje wiele metod (algorytmów) sortowania, jedne działają szybciej, inne wolniej. W tym podrozdziale poznasz jedną z nich i wykorzystasz w tworzonej aplikacji. W przyszłości, jeśli będziesz kontynuować swoją przygodę z algorytmami, poznasz także inne i porównasz ich efektywność.

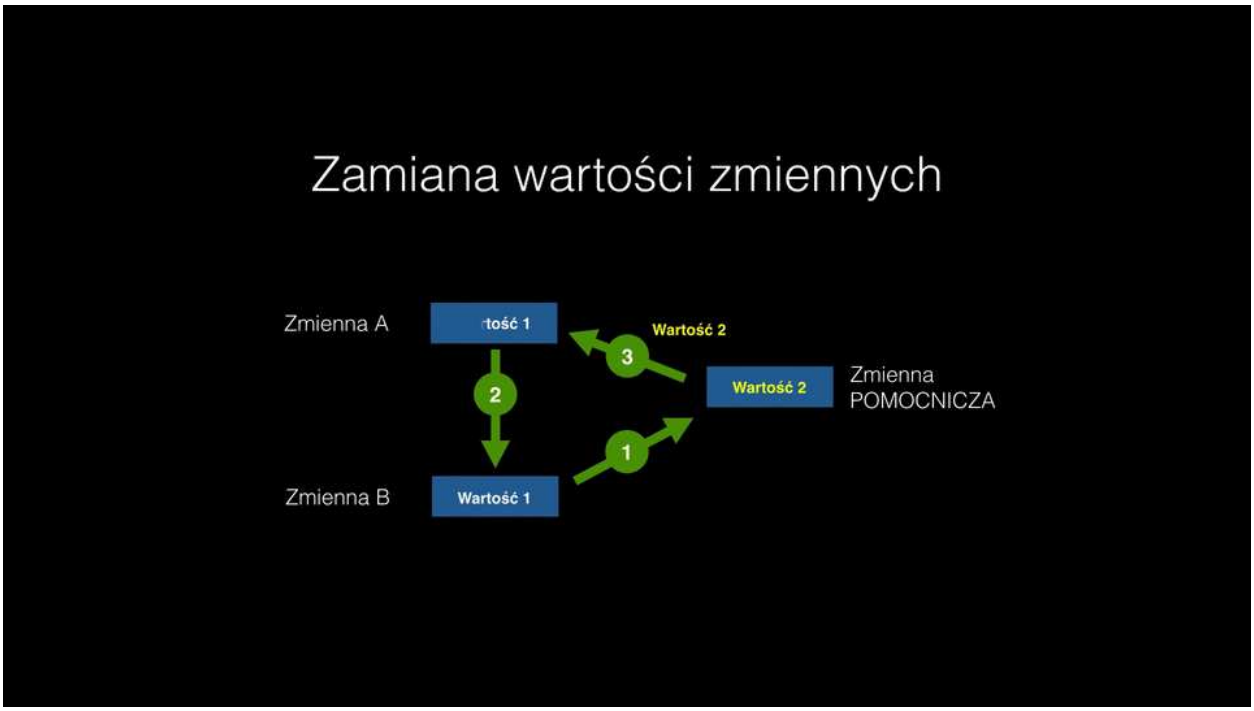
Najpierw wykonaj jednak dwa proste ćwiczenia.

Ćwiczenie 12

Zapisz w postaci listy kroków algorytm znajdowania najmniejszej liczby na liście. Możesz przygotować także prostą aplikację, w której jest losowana lista liczb, a następnie duszek znajduje najmniejszą liczbę na liście i wyświetla ją w dymku komiksowym.

Ćwiczenie 13

Popraw tak rozwiązanie poprzedniego zadania, aby najmniejsza liczba została ustawiona na pierwszym miejscu listy. Nie możesz jednak zgubić żadnego elementu (np. elementu, który pierwotnie był na pierwszym miejscu).
Wskazówka:
Możesz zamienić miejscami pierwszy element listy oraz element najmniejszy. Aby zamienić wartościami dwie zmienne (dwa elementy listy) należy użyć pomocniczej zmiennej.



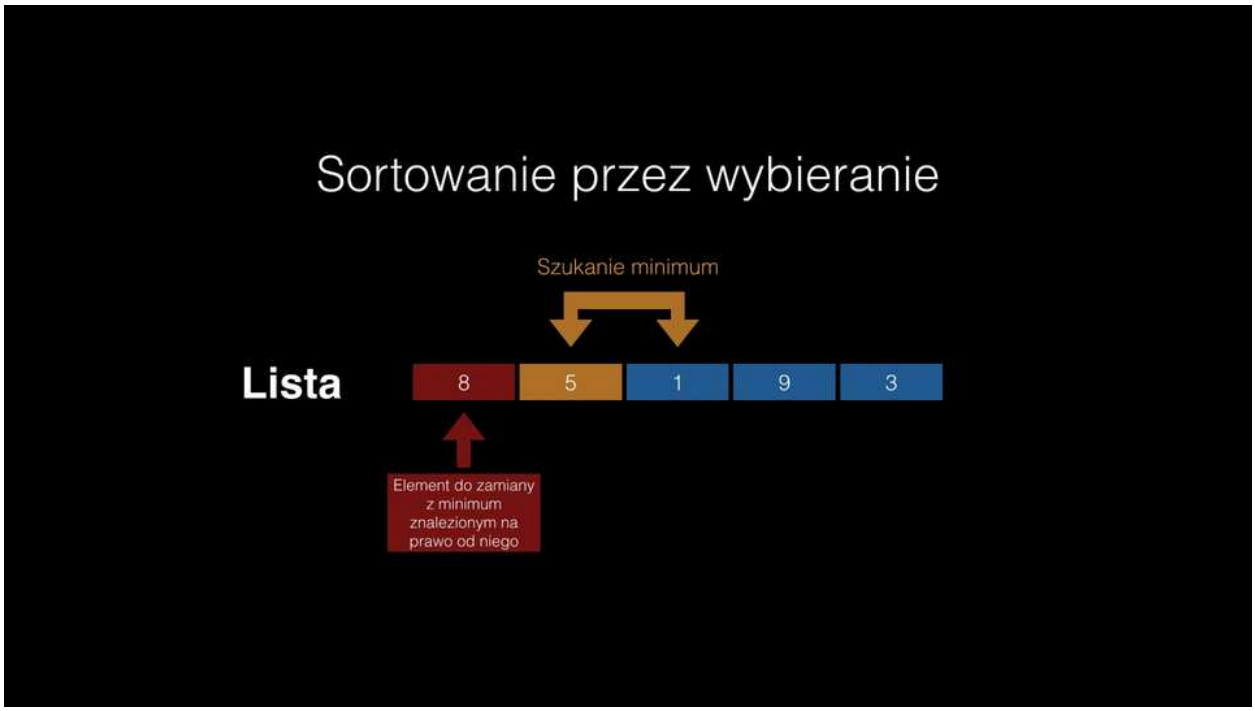
Film dostępny na portalu epodreczniki.pl

Źródło: Janusz Wierzbicki, Maciej Borowiecki, licencja: CC BY 3.0.

Animacja: Zamiana wartości zmiennych

Zwróć uwagę, że należy teraz poszukiwać miejsca (numeru elementu) najmniejszej liczby na liście, a nie jej wartości.

Algorytm sortowania przez wybieranie
Liczby na liście mają być ustawione od najmniejszej do największej, a więc w pierwszym kroku można znaleźć liczbę najmniejszą i zamienić ją z pierwszym elementem listy.
W kolejnym kroku postępujemy analogicznie poszukując liczby najmniejszej poczynając od drugiego elementu, następnie od trzeciego, itd. Powtarzamy więc wybieranie i ustawianie liczby najmniejszej n-1 razy (jak zostaje jedna liczba, to znajduje się już na właściwym miejscu, ciąg złożony z jednego elementu jest uporządkowany).
Przykład
W poniższej tabelce zaznaczone są kolejne kroki porządkowania następującego zestawu liczb: 8, 5, 1, 9, 3. Kolorem zielonym zaznaczone są liczby już ustawione, kolorem żółtym kolejne znalezione minimum, a czerwonym miejsce, na którym zostanie ustawione.



Film dostępny na portalu epodreczniki.pl

Źródło: Janusz Wierzbicki, Maciej Borowiecki, licencja: CC BY 3.0.

Animacja: Sortowanie przez wybieranie

Zapis algorytmu sortowania przez wybieranie w postaci listy kroków:
Dane:
n – liczba elementów listy,
L – lista liczb.
Wynik:
L – uporządkowana niemalejąco lista liczb.
1.Powtarzaj dla kolejnych wartości zmiennej i od 1 do n-1:
1.1.Ustaw wartość zmiennej m na i.
1.2.Powtarzaj dla kolejnych wartości zmiennej j od i+1 do n:

1.2.1.Jeżeli *L[j]* < *L[m]* to ustaw wartość zmiennej *m* na *j*.
1.3.Zamień elementy *L[i]* i *L[m]*.

> **Ćwiczenie 14**
>
> Utwórz skrypt (nowy blok) sortujący listę odcinków zgodnie z powyższym algorytmem. Więcej na temat tworzenia własnych bloków możesz się dowiedzieć tutaj.

Należy jeszcze zmodyfikować skrypty uruchamiane jako reakcja na nadanie komunikatów *tak* i *nie* (po kliknięciu w jeden z przycisków). Trzeba najpierw wywołać blok sortujący listę *Odcinki*, a następnie sprawdzić, czy suma dwóch pierwszych elementów jest większa od ostatniego. Na czas sortowania warto ukryć widoczność listy na scenie.

> **Ćwiczenie 15**
>
> Zmodyfikuj skrypty uruchamiane po otrzymaniu komunikatów *tak* i *nie*.

Znajdujemy dwa najkrótsze odcinki i najdłuższy

Żeby stwierdzić, czy z każdej trójki odcinków można zbudować trójkąt, nie trzeba porządkować odcinków według ich długości. Wystarczy znaleźć dwa najkrótsze odcinki oraz najdłuższy, czyli dwie liczby najmniejsze i największą. Możesz doprowadzić do sytuacji, że dwie najmniejsze liczby znajdą się na dwóch pierwszych miejscach, a największa na ostatniej. Możesz także znaleźć ich wartości na trzech pomocniczych zmiennych.

> **Ćwiczenie 16**
>
> Utwórz trzy pomocnicze zmienne, w których zapamiętasz poszukiwane liczby (np. o nazwach *min1*, *min2* i *max*).

> **Ćwiczenie 17**
>
> Zastanów się, jakie nadać wartości początkowe zmiennym *min1*, *min2* i *max*. Przygotuj pomocniczy skrypt nadający im wartości początkowe.
> **Wskazówka:**
> Najwygodniej nadać wartości początkowe wykorzystując liczby występujące na liście. Ponieważ potrzebujesz dwóch wartości minimalnych, możesz wykorzystać dwa pierwsze elementy listy *Odcinki*. Pamiętaj, aby był prawdziwy warunek *min1*<=*min2*.

> **Ćwiczenie 18**
>
> Zapisz w postaci listy kroków algorytm znajdowania dwóch minimów i maksimum.

> **Ćwiczenie 19**
>
> Utwórz nowy blok zgodnie z algorytmem z poprzedniego zadania.

Pamiętaj o dostosowaniu skryptów wywoływanych jako reakcja na nadanie komunikatów *tak* i *nie*.

Podsumowanie – szacujemy złożoność obliczeniową

W tym rozdziale zostały zaprezentowane trzy algorytmy rozwiązania tego samego problemu.

Spróbujmy dokonać oszacowania ich efektywności. Efektywność nazywamy złożonością obliczeniową i wyrażamy liczbą operacji charakterystycznych dla danego problemu. Przy sprawdzaniu, czy z każdych trzech odcinków można zbudować trójkąt, często były porównywane elementy listy. W związku z tym za operację charakterystyczną (dominującą) można przyjąć porównanie elementów listy. Lista ma określoną liczbę elementów (n), więc liczba porównań będzie zależeć od n. Nie musimy liczyć dokładnie, wystarczy podać wartość przybliżoną. Na przykład, dla wyrażenia $(n^2-n)/2$ możemy powiedzieć, że złożoność jest rzędu n^2. Przy dużych wartościach n decydujący wpływ na rząd wielkości ma najwyższa potęga. Co prawda w przykładowej aplikacji lista ma mało elementów, ale wyobraź sobie sytuację, że odcinków jest bardzo dużo.

Algorytm 1
Nie zapisywaliśmy go, ale z przedstawionej w trzecim podrozdziale tabelki wynika, że operacji jest wiele. Spróbujmy zapisać szkielet tego algorytmu (sprawdzanie wszystkich możliwych trójek liczb).

1.powtórz dla wartości i od 1 do n–2
1.1.powtórz dla wartości j od i+1 do n–1
1.1.1.powtórz dla wartości k od j+1 do n: Sprawdź trójkę *L[i]*, *L[j]*, *L[k]*.

Możesz stwierdzić, że złożoność tego algorytmu jest rzędu n^3.

Algorytm 2
Złożoność tego algorytmu to złożoność zastosowanego algorytmu sortowania. Przedstawiony algorytm sortowania przez wybieranie ma złożoność rzędu n^2 (kwadratową). Istnieją algorytmy sortowania efektywniejsze, o mniejszej złożoności. Jeśli będziesz kontynuować swoją przygodę z algorytmami, zapewne je poznasz i będziesz stosować.

Algorytm 3
Tym razem raz jest przeglądana lista (co prawda w jednym kroku pętli wykonywane jest więcej niż jedno porównanie). Możesz stwierdzić, że złożoność tego algorytmu jest rzędu n (liniowa).

Zadania uzupełniające

> **Ćwiczenie 20**
>
> Przygotuj aplikację, w której losowana jest lista liczb. Następnie niech duszek poprosi o podanie liczby i sprawdzi, czy ona występuje na liście. Zapisz najpierw algorytm sprawdzenia, czy liczba występuje na liście, w postaci listy kroków.

Ćwiczenie 21

Przygotuj podobną aplikację, jak w poprzednim zadaniu, ale po wylosowaniu posortuj listę. Zastanów się, czy wyszukując liczbę w posortowanej liście można zastosować inny algorytm, który wykona mniej operacji.
Wskazówka:
Uruchom poniższą aplikację, w której duszek zgaduje liczbę pomyślaną przez ciebie. Postaraj się wykorzystać algorytm stosowany przez duszka w rozwiązaniu zadania.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/b/P17dx7rmo

Źródło: Janusz Wierzbicki, Maciej Borowiecki, licencja: CC BY 3.0.