3.3. Trafiony, zatopiony

Jak wyszukać podany element w zbiorze?

Wiesz już, jak w programie Scratch utworzyć listę, a na liście wypełnionej liczbami odnaleźć najmniejszą lub największą liczbę. Teraz nauczysz się, jak przy użyciu programu sprawdzić, czy na liście znajdują się konkretne elementy, na przykład liczba 2 albo liczba 7.

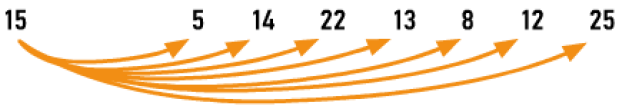
Wyobraź sobie grę, w której na początku trzeba podać siedem różnych liczb całkowitych z zakresu od 1 do 25, a następnie komputer z tego samego zakresu losuje siedem różnych liczb całkowitych. Grający otrzyma tyle bonusów do wykorzystania w grze, ile takich samych liczb pojawi się na obu listach. Aby program sprawdził, ile liczb powtarza się na listach gracza i komputera, trzeba go tego nauczyć. Na zajęciach przygotujesz projekt z przykładowym rozwiązaniem. Zanim zbudujesz odpowiednie skrypty, zastanów się, jaką strategię przyjąć, aby nie pominąć żadnego elementu.

W kolejnych etapach pracy:

1. ustalisz sposób, w jaki odszukasz elementy powtarzające się na listach,
2. zaprogramujesz rozwiązanie w języku Scratch.

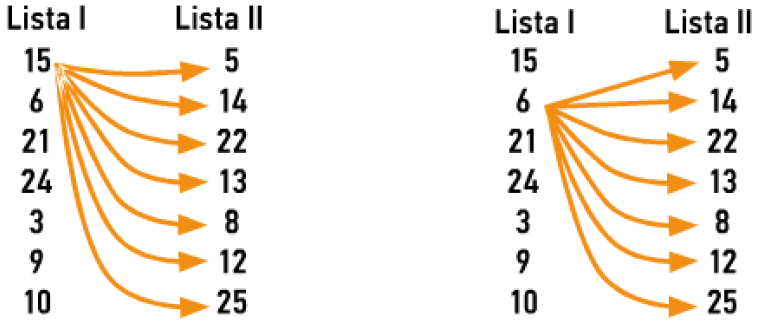
1 Ustalenie sposobu rozwiązania

Jeśli chcesz sprawdzić, czy na liście znajduje się dany element, wystarczy na przykład porównać go z kolejnymi elementami na liście. Poszukiwanie elementu możesz rozpocząć od początku lub końca listy. Przy liście siedmioelementowej będzie maksymalnie siedem porównań (rys. 1).



Rys. 1. Sprawdzenia w przypadku listy siedmioelementowej

Aby sprawdzić, które liczby z listy I są także na liście II, możesz postępować podobnie. W przypadku list I i II z rysunku 2 najpierw porównaj pierwszy element z listy I z elementami na kolejnych pozycjach listy II. Następnie sprawdź, czy drugi element z listy I znajduje się na liście II i tak dalej, aż uwzględnisz wszystkie elementy z listy I. Pamiętaj o tym, aby elementy na kolejnych pozycjach listy I porównywać z elementami na kolejnych pozycjach listy II – dzięki temu nie pominiesz żadnego z nich.



Rys. 2. Schemat postępowania dla pierwszych dwóch elementów z listy I

Zauważ, że w przypadku zestawów siedmioelementowych będzie aż 49 sprawdzeń – siedem elementów z listy I trzeba porównać z siedmioma elementami z listy II.

Liczby, które powtórzą się w obu zestawach, warto wyświetlić na trzeciej liście. Dzięki temu użytkownik zobaczy, które liczby dobrze wytypował. Długość listy będzie natomiast informowała, ile tych liczb jest.

2 Zaprogramowanie rozwiązania w języku Scratch

W nowym projekcie programu Scratch utwórz trzy listy o nazwach: „Podane", „Wylosowane" i „Trafione". Na pierwszej znajdą się liczby podane przez użytkownika, na drugiej liczby wylosowane przez komputer, a na trzeciej te, które powtórzą się w obu zestawach.

Teraz dla wybranego duszka lub tła zbuduj skrypt, dzięki któremu na liście „Podane" pojawią się liczby podane przez użytkownika. Skrypt powinien uruchomić się po naciśnięciu przycisku z zieloną flagą i spowodować, że:

- pojawi się pole do wpisania liczby,
- odpowiedź użytkownika zostanie dodana do listy „Podane", jeśli będzie to liczba z podanego zakresu i nie będzie jej jeszcze na tej liście,
- prośba o podanie liczby będzie się powtarzać do czasu, gdy długość listy będzie równa 7,
- zostanie nadany komunikat „Losuj", gdy na liście „Podane" znajdzie się siedem liczb (komputer po nadaniu komunikatu wylosuje liczby na listę „Wylosowane").

Odszukaj blok, dzięki któremu pojawi się pole do wpisania odpowiedzi. Zapisz na nim prośbę o podanie liczby. Aby program zapamiętał odpowiedź, musisz użyć zmiennej. Utwórz więc zmienną, na przykład o nazwie „Liczba", i ustaw odpowiedź użytkownika jako jej wartość (rys. 3).

Rys. 3. Prośba o podanie liczby i zapamiętanie odpowiedzi jako wartości zmiennej „Liczba"

Teraz program powinien sprawdzić, czy może dodać podaną przez użytkownika liczbę do listy. Zrobi to, jeżeli ta liczba mieści się we wskazanym zakresie i nie ma jej jeszcze na liście. Podane warunki powinny zostać spełnione jednocześnie, dlatego musisz połączyć je spójnikiem „i" – odpowiedni blok znajdziesz w kategorii **Wyrażenia**.

Aby gracz mógł podać liczbę całkowitą z zakresu od 1 do 25, określ w skrypcie, że liczba ma być jednocześnie większa od 0 i mniejsza od 26 (rys. 4).

Rys. 4. Fragment skryptu określający zakres liczb

Do sprawdzenia, czy wartość zmiennej „Liczba" znajduje się na liście „Podane", wystarczy użyć warunku z rysunku 5. Jeśli warunek ten umieścisz na bloku z napisem „nie", program odczyta to jako zaprzeczenie, czyli że wartości zmiennej „Liczba" nie ma na liście „Podane" (rys. 6).

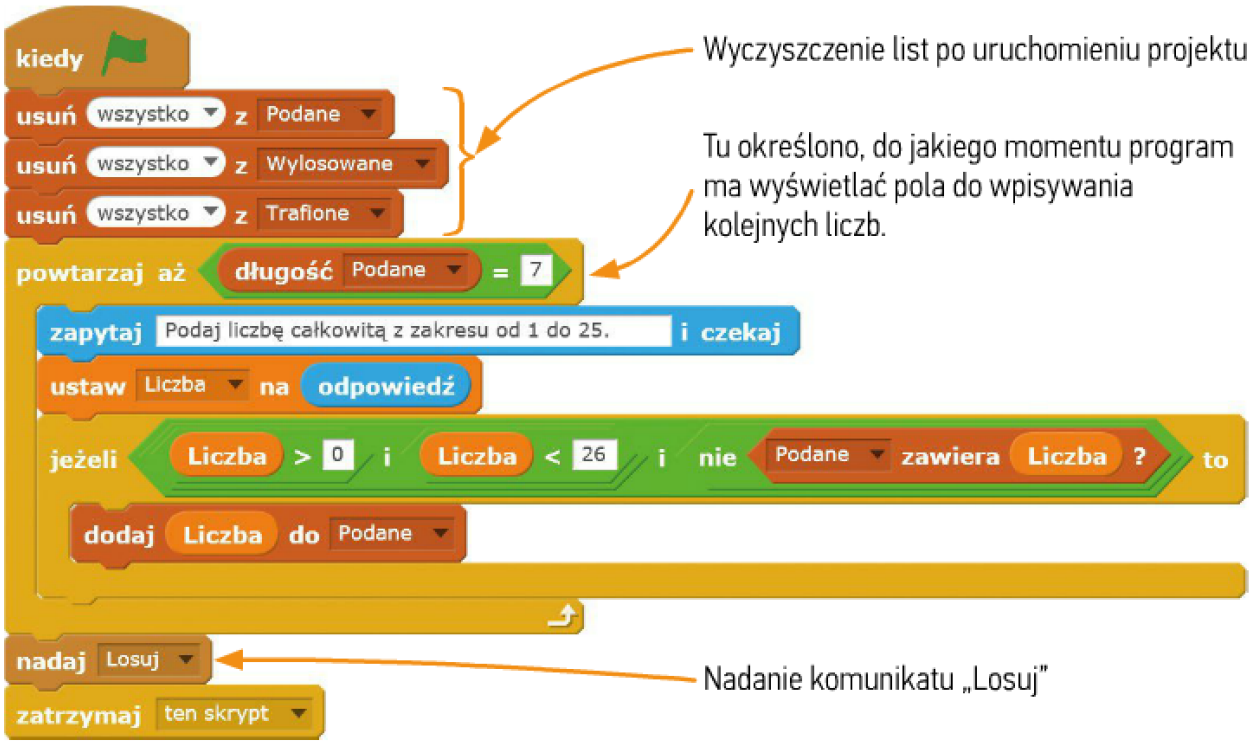
Rys. 5. Warunek określający, czy lista „Podane" zawiera element „Liczba"

Rys. 6. Zaprzeczenie warunku „Podane zawiera Liczba"

Dodaj do skryptu blok z napisami „jeżeli" i „to" i wstaw po tych napisach odpowiednie bloki (rys. 7).

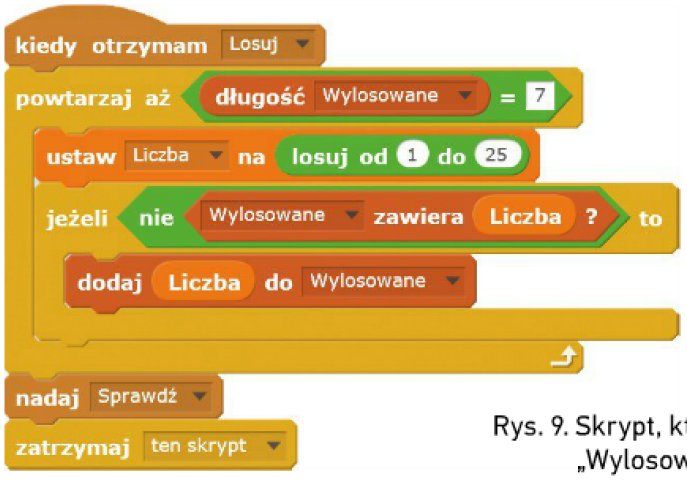
Rys. 7. Podanie liczby przez użytkownika i dodanie jej do listy, jeśli spełnia określone warunki

Teraz określ w skrypcie, że program ma ponawiać prośbę o podanie kolejnej liczby do momentu, gdy na liście będzie siedem elementów – zastosuj blok z napisem „powtarzaj aż". Po wypełnieniu listy nadaj komunikat „Losuj". Na początku skryptu umieść bloki, które spowodują wyczyszczenie list (rys. 8).



Rys. 8. Skrypt, dzięki któremu na liście „Podane" pojawi się siedem różnych liczb podanych przez użytkownika oraz zostanie nadany komunikat „Losuj"

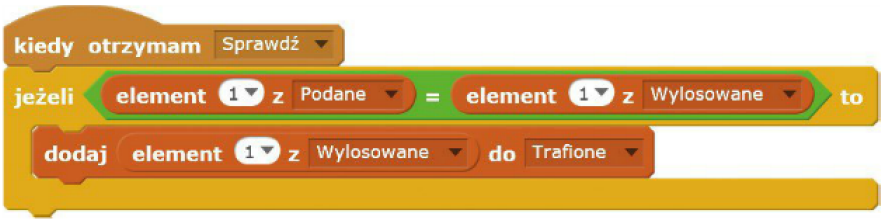
Zduplikuj skrypt powodujący wstawienie na listę „Podane" liczb użytkownika. Zmodyfikuj kopię tak, aby komputer wylosował z zakresu od 1 do 25 siedem różnych liczb i dodał je do listy „Wylosowane". Skrypt powinien uruchomić się po odebraniu komunikatu „Losuj". Po wylosowaniu liczb nadaj komunikat o treści „Sprawdź" – będzie to hasło dla programu, aby zaczął sprawdzać, czy na listach „Podane" i „Wylosowane" występują takie same liczby (rys. 9).



Rys. 9. Skrypt, który spowoduje dodanie liczb do listy „Wylosowane" i nadanie komunikatu „Sprawdź"

Skrypt zaczynający się od bloku z napisem „kiedy otrzymam Sprawdź" warto zbudować dla duszka. Później wystarczy niewielkie uzupełnienie skryptu, a duszek poinformuje o liczbie trafień.

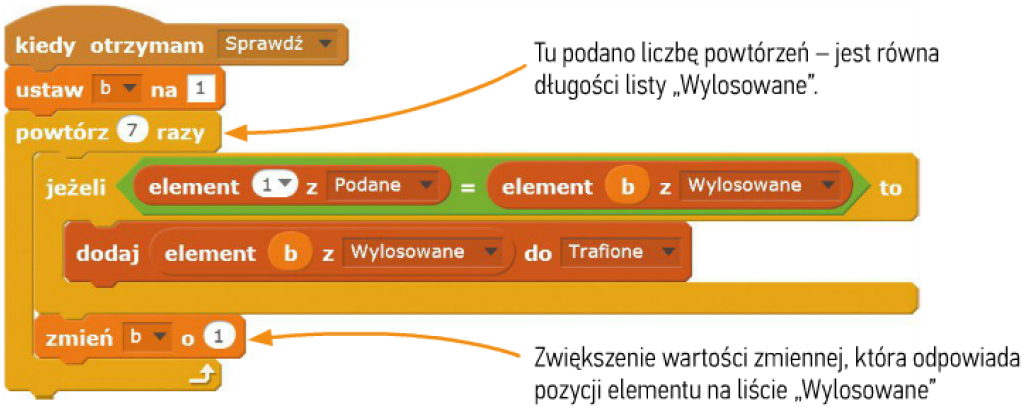
Teraz zbudujesz skrypt sprawdzający, czy liczby podane przez gracza znajdują się także na liście „Wylosowane". Zacznij od porównania pierwszego elementu z listy „Podane" z pierwszym elementem na liście „Wylosowane" – jeśli są równe, dodaj pierwszy element z listy „Wylosowane" do listy „Trafione" (rys. 10).



Rys. 10. Porównanie pierwszych elementów na listach – jeśli są takie same, program doda pierwszy element z listy „Wylosowane" do listy „Trafione"

Rozbuduj skrypt – określ w nim porównywanie pierwszego elementu z listy „Podane" z kolejnymi elementami na liście „Wylosowane". Ponieważ pozycje kolejnych elementów na liście „Wylosowane" będą się zwiększać o jeden, warto użyć zmiennej, na przykład o nazwie „b" – utwórz tę zmienną. Liczba powtórzeń (sprawdzeń) powinna być równa długości listy „Wylosowane", czyli w tym wypadku liczbie 7 (rys. 11).

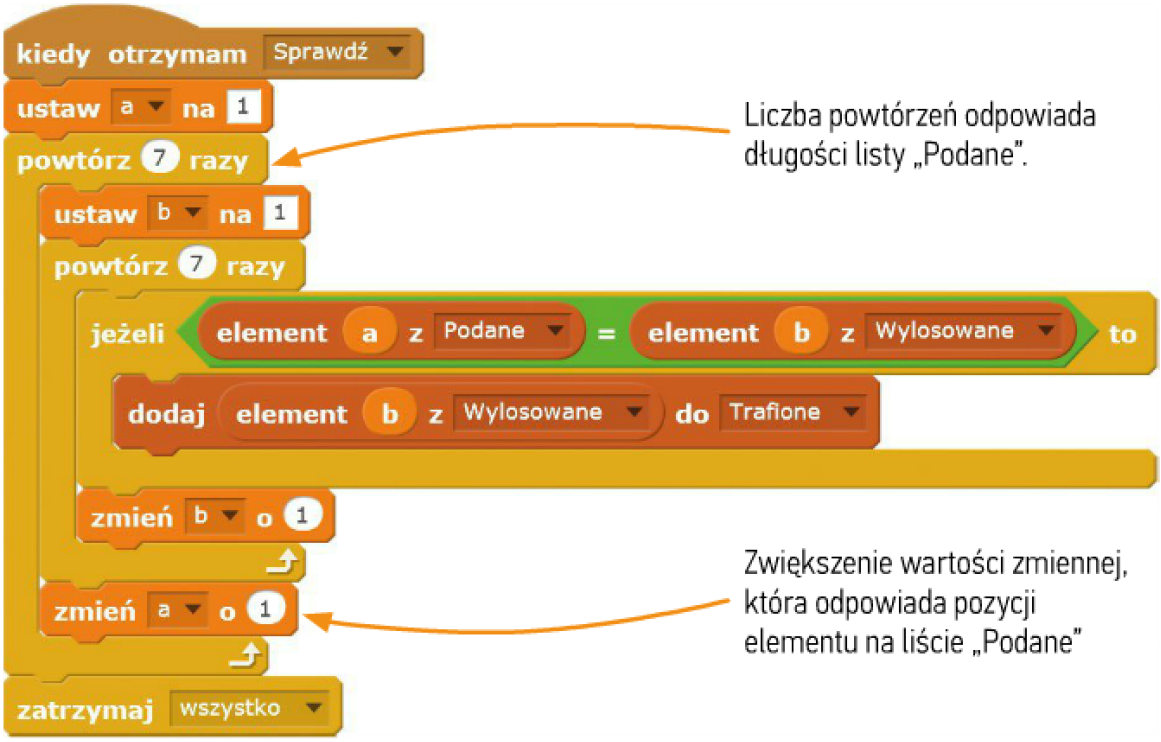
Rozbuduj skrypt – określ w nim porównywanie pierwszego elementu z listy „Podane" z kolejnymi elementami na liście „Wylosowane". Ponieważ pozycje kolejnych elementów na liście „Wylosowane" będą się zwiększać o jeden, warto użyć zmiennej, na przykład o nazwie „b" – utwórz tę zmienną. Liczba powtórzeń (sprawdzeń) powinna być równa długości listy „Wylosowane", czyli w tym wypadku liczbie 7 (rys. 11).



Rys. 11. Porównywanie pierwszego elementu z listy „Podane" z kolejnymi elementami z listy „Wylosowane"

Pozostało ci już tylko uwzględnić w skrypcie sprawdzanie, czy na liście „Wylosowane" znajdują się kolejne elementy z listy „Podane". Pozycje elementów z listy „Podane" będą się zmieniać, dlatego warto użyć nowej zmiennej, na przykład o nazwie „a". Utwórz ją.

Uzupełnij skrypt. Liczba na dodanym bloku z napisem „powtórz” powinna być taka jak długość listy „Podane”, czyli 7 (rys. 12).



Rys. 12. Skrypt porównujący elementy na listach „Podane” i „Wylosowane”

Możesz jeszcze tak zmodyfikować projekt, aby o liczbie trafień poinformował duszek. Pracę zapisz w *Teczce ucznia* pod nazwą *szczęśliwe_liczby*.

Znajdź na to sposób

Wykorzystany w temacie sposób poszukiwania konkretnego elementu w zbiorze jest poprawny. Zauważ jednak, że w przypadku odnalezienia szukanego elementu program nie musi już sprawdzać kolejnych elementów z tego zbioru. W zadaniu wykonanym na zajęciach, na przykład jeśli szukany element znajduje się na drugiej pozycji na liście, program wykona jeszcze pięć porównań, które nie mają wpływu na wynik. Zastanów się, jak zmodyfikować projekt utworzony na zajęciach, aby po znalezieniu elementu na liście program nie sprawdzał już kolejnych pozycji. Sprawdź w programie Scratch, czy twój sposób myślenia był właściwy.

Zapamiętaj

- **Zanim zaczniesz pracę nad projektem, ustal sposób rozwiązania.**
- **Aby sprawdzić, czy na liście znajduje się dany element, można na przykład porównać go z kolejnymi elementami na tej liście.**