Przerywanie pętli

Źródło: Mitchell Luo, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Przeglądając zbiór danych (taki jak np. rozkład jazdy autobusów), szukasz zwykle konkretnej informacji. Kiedy ją znajdziesz – przerywasz analizę danych.

Podobnie bywa podczas programowania: nie zawsze musimy wykonywać operacje na każdym elemencie danego zbioru. Do sprawdzenia, czy zbiór ten zawiera poszukiwaną przez nas wartość, możemy wykorzystać pętlę. Gdy odnajdziemy taki element, nie musimy już kontynuować poszukiwań. Aby nie wykonywać niepotrzebnych instrukcji, pętlę należy zakończyć.

Znamy i umiemy zastosować w poszczególnych językach programowania pętle `for` oraz `while` czy `do-while`. W tym e-materiale omówimy dwie instrukcje: `break` i `continue`, które służą do sterowania działaniem pętli. Użycie ich pozwala przerwać pętlę albo sprawić, że część zapisanych w niej poleceń zostanie pominięta.

Implementację przerywania pętli w poszczególnych językach programowania przedstawiamy w e-materiałach:

- Przerywanie pętli w języku C++,
- Przerywanie pętli w języku Java,

- Przerywanie pętli w języku Python.

Więcej zadań? Sięgnij do Przerywanie pętli – zadania maturalne.

> **Twoje cele**
>
> - Zaimplementujesz różne typy pętli.
> - Scharakteryzujesz działanie instrukcji `break` oraz `continue`.
> - Przeanalizujesz przykłady zastosowania poleceń `break` i `continue`.

Przeczytaj

Przypomnijmy, co już wiemy na temat pętli. Służą one do wielokrotnego wykonywania jednej operacji lub całego zestawu poleceń. Dopóki spełniony jest pewien warunek logiczny, dopóty instrukcje umieszczane wewnątrz pętli są wykonywane. Oto budowa pętli `while` w języku C++ oraz Java:

Już wiesz

```
while (wyrażenie) {
    instr_1;
    instr_2;
    instr_3;
    ...
}
```

Polecenia `instr1`, `instr2`, `instr3` wykonywane będą tak długo, dopóki wyrażenie logiczne `wyrażenie` będzie prawdziwe. Podobnie wygląda pętla `do-while`. Różnica polega na tym, że warunek logiczny sprawdzany jest po wykonaniu instrukcji umieszczonej (lub umieszczonych) wewnątrz pętli.

W przypadku języka Python pętla `while` wygląda tak:

```
while wyrażenie:
    instr_1
    instr_2
    ...
```

Polecenia `instr_1`, `instr_2` wykonywane będą tak długo, dopóki `wyrażenie` będzie prawdziwe. W języku Python pętla `do-while` nie występuje.

Przypomnijmy jeszcze budowę pętli `for` w języku Java i C++:

```
for (deklaracja iteratora; warunek wykonania pętli; zmiana wart
    instr_1;
    instr_2;
    instr_3;
    ...
}
```

W tym przypadku używamy wyrażenia inicjującego (deklarację iteratora) jako licznika powtórzeń pętli. Wyrażenie dotyczące warunku pętli to warunek logiczny – jeżeli jest on spełniony, wykonywane są operacje w pętli. Z kolei zmiana wartości iteratora jest instrukcją zmieniającą wartość wyrażenia inicjującego (`deklaracja iteratora`), po każdej iteracji.

Natomiast w przypadku języka Python, pseudokod pętli `for` wygląda następująco:

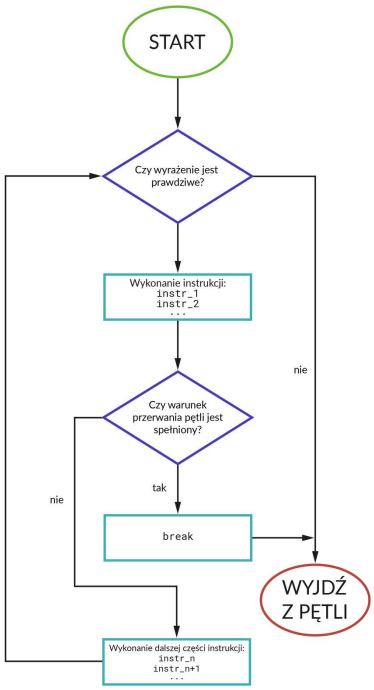
```
for i in range(zakres_początek, zakres_koniec, krok):
    instr_1
    instr_2
    ...
```

Dla każdej wartości z określonego zakresu (uwzględniając wartość kroku, czyli różnicy pomiędzy kolejnymi wartościami zakresu — domyślnie równej 1) wykonywane są instrukcje zawarte wewnątrz pętli. W zmiennej `i`, przy kolejnych iteracjach pętli, przechowywane są kolejno liczby całkowite z zakresu <`zakres_początek`, `zakres_koniec`).

Instrukcja `break`

Czasem zdarza się, że chcemy przerwać działanie pętli pomimo tego, że nie zostały wykonane wszystkie umieszczone wewnątrz niej instrukcje. W takiej sytuacji przydaje się polecenie `break`.

Instrukcji `break` może towarzyszyć warunek (wyrażenie logiczne). Jeżeli jest on spełniony, pętla zostanie opuszczona. Wyjaśnia to zaprezentowany schemat blokowy. Jeżeli instrukcji `break` nie towarzyszy warunek, to na pytanie zawarte w bloku „Czy warunek przerwania pętli jest spełniony?" odpowiedź zawsze brzmi „Tak".



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

Najpierw sprawdzamy, czy decydujące o wykonaniu pętli wyrażenie logiczne ma wartość `true`. Jeżeli nie, kończymy działanie pętli. Gdy natomiast wyrażenie jest prawdziwe, wykonywane są operacje opisane wewnątrz pętli. Jeżeli wśród poleceń znajdzie się instrukcja `break`, to pętla zostanie opuszczona bez wykonywania dalszych operacji. Jeżeli instrukcji `break` będzie towarzyszyło wyrażenie logiczne, to w przypadku przyjęcia wartości `true`, pętla także zostanie opuszczona. Jeżeli przyjmie wartość `false`, wykonywane będą kolejne instrukcje w pętli. Taki mechanizm często stosuje się w przypadku pętli zagnieżdżonych. Instrukcja `break` przerywa wówczas działanie pętli wewnętrznej.

Przykład 1

Przedstawiony program, przy użyciu schematu blokowego, wyświetla wszystkie słowa występujące w liście `zdanie` (kolejne słowa oddzielone są przecinkami), nie zawierające litery `znak`. W programie użyto pętli zagnieżdżonych. Przetestuj działanie programu dla następujących danych:

- `n = 4`
- `znak = "b"`
- `zdanie = "instrukcja,break,i,petle"`

Zwróć uwagę, że w programie użyto instrukcji `break`.

Specyfikacja problemu:

Dane:

- `znak` – litera alfabetu łacińskiego

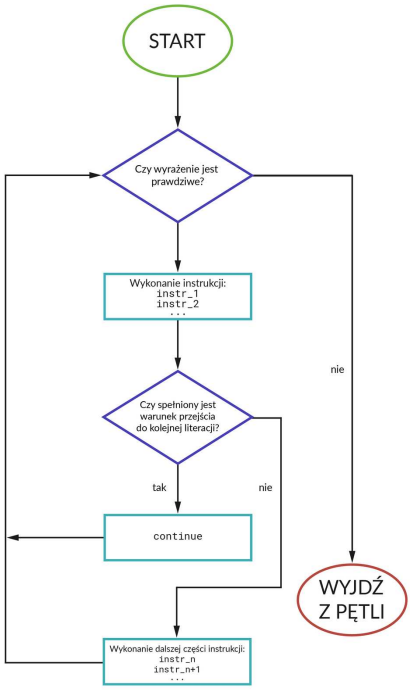
- `zdanie` – lista ciągów znaków zawierająca `n` słów różnej długości rozdzielonych przecinkami
- `n` – liczba naturalna; liczba słów w liście `zdanie`

Wynik:

- Program wypisze listę `zdanie` bez słów zawierających literę `znak`.

Instrukcja `continue`

Polecenie `continue` przypomina nieco działanie instrukcji `break`. Nie służy ono jednak do opuszczenia pętli, lecz do pominięcia kolejnych operacji w bieżącym cyklu. Jeżeli instrukcji `continue` dodatkowo towarzyszy warunek, schemat blokowy wygląda następująco:



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

W przypadku pojawienia się instrukcji `continue` pomijane są kolejne polecenia w pętli. Jeżeli wykonanie instrukcji `continue` jest zależne od warunku i jest on spełniony, rozpoczyna się następna iteracja. Natomiast w przypadku, gdy warunek nie jest spełniony, wykonywana jest dalsza część operacji w aktualnej iteracji.

Pętle nieskończone

W przypadku gdy warunek wykonania pętli zawsze jest spełniony, to działanie pętli trwa w nieskończoność. Przykładem jest pętla `for`, w której w ogóle nie podamy warunku (z punktu widzenia składni C++ nie jest to obowiązkowe), a także odpowiednio zbudowana pętla `while` lub `do-while`. Przykład w języku C++ lub Java:

```
do {
    instr_1;
    instr_2;
    ...
} while (true);
```

Pętla nieskończona w języku Python:

```
while True:
    instr_1
    instr_2
    ...
```

Jedynym sposobem zakończenia takiej pętli jest użycie instrukcji `break`. Warto wiedzieć, że pętli nieskończonych często używa się podczas programowania mikrokontrolerów.

Instrukcja wyboru wielokrotnego i `break`

Słowo kluczowe `break` pojawia się często w znanej ci już instrukcji `switch-case` w języku C++ oraz Java. W tym przypadku nie towarzyszy mu wyrażenie logiczne. Polecenie `break` ma po prostu zakończyć wykonywanie instrukcji wyboru wielokrotnego:

```
switch (wartość) {
    case przypadek1:
        instr_1;
        break;
    case przypadek2:
        instr_2;
        break;
    case przypadek3:
        instr_3;
        break;
    ...
}
```

Słownik

> **iteracja**

słowo pochodzące od łacińskiego *iteratio* (powtarzanie); oznacza powtarzanie w pętli tych samych instrukcji, aż do spełnienia pewnego warunku

zagnieżdżona pętla

pętla umieszczona wewnątrz innej pętli

Schemat interaktywny

Polecenie 1

Za pomocą schematu blokowego lub języka programowania zapisz algorytm sprawdzający, czy dana liczba naturalna $n > 1$ jest liczbą pierwszą. Przetestuj działanie swojego programu dla liczby `n = 997` oraz `n = 1024`. W programie należy użyć instrukcji `break`.

Specyfikacja problemu:

Dane:

- n – liczba naturalna; sprawdzamy czy jest liczbą pierwszą; $n > 1$

Wynik:

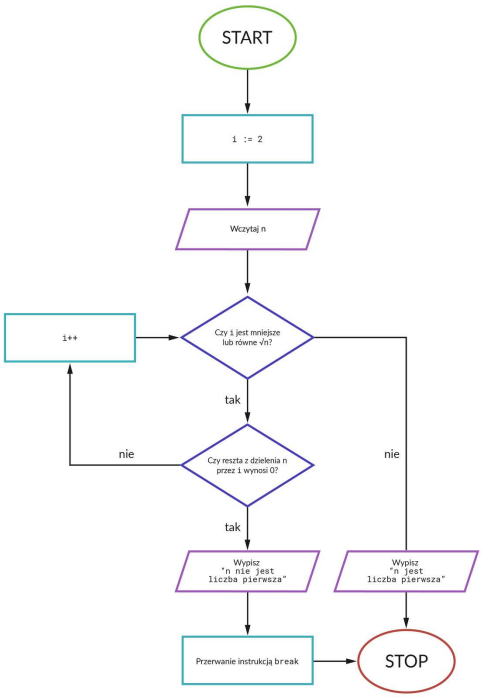
- Program, zależnie od wyniku, wyświetla komunikat: „n jest liczba pierwsza” lub „n nie jest liczba pierwsza”.

```
1
```

```
1
```

Polecenie 2

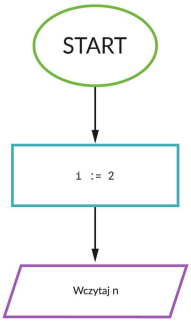
Porównaj swoje rozwiązanie z algorytmem przedstawionym w prezentacji.



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

Przed przystąpieniem do analizowania projektu przypomnij sobie zasady tworzenia pętli lub algorytmów iteracyjnych.

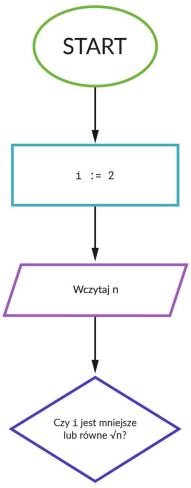
2



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

Rozpoczynamy schemat od bloku START. Następnie zmiennej `i` przypisujemy wartość 2 oraz pobieramy wartość zmiennej n.

3



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

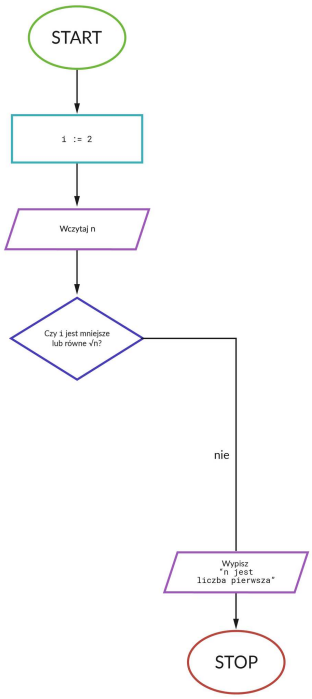
Teraz należy sprawdzić czy zmienna `i` jest mniejsza lub równa $\sqrt{n}$.

4

Dlaczego zmienna `i` ma być mniejsza lub równa $\sqrt{n}$? Otóż dlatego, że chcemy się dowiedzieć, czy liczba n ma jakiekolwiek dzielniki poza 1 i samą sobą. Sprawdzimy tylko liczby naturalne z przedziału od 2 do $\sqrt{n}$, ponieważ jeżeli znajdziemy tam dzielniki, to będzie ich tyle samo co w przedziale od $\sqrt{n}$ do n. W związku z tym,

jeżeli nie znajdziemy tam dzielnika, to nie znaleźlibyśmy go w całym przedziale. Posłużymy się przy tym pętlą.

5



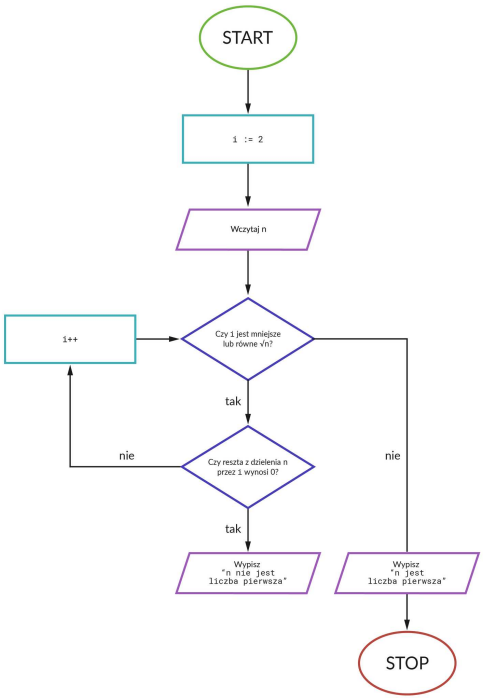
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W przypadku gdy warunek `i <=` $\sqrt{n}$ nie jest spełniony, wychodzimy z pętli. Jeżeli w zakresie od 2 do $\sqrt{n}$ nie znajdziemy żadnego całkowitego podzielnika, wyświetlimy komunikat, że n jest liczbą pierwszą. Natomiast gdy `i` jest mniejsze od $\sqrt{n}$, przechodzimy do wykonywania instrukcji w pętli.

6

Musimy sprawdzić, czy reszta z dzielenia liczby n przez `i` wynosi 0. Jeżeli znaleźlibyśmy taką wartość `i` w przedziale od 2 do $\sqrt{n}$, to n nie jest liczbą pierwszą.

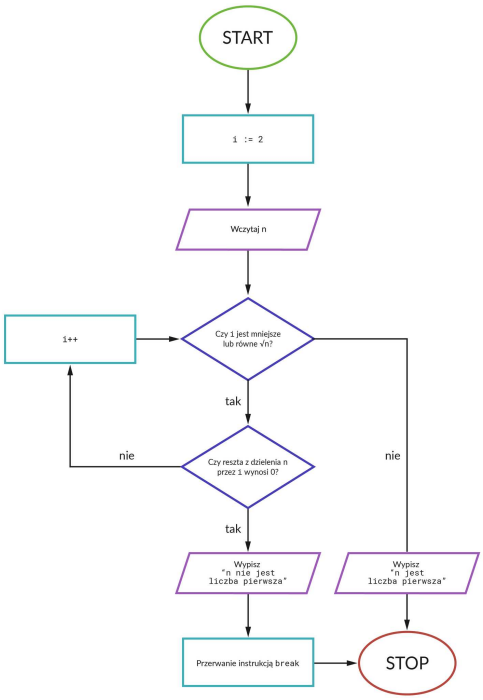
7



8

Jeżeli reszta z dzielenia liczby n nie wynosi 0, inkrementujemy zmienną `i`, a następnie zaczynamy wykonywać pętlę od nowa. Natomiast gdy reszta z dzielenia równa się 0, wyświetlamy tylko komunikat "Liczba n nie jest liczbą pierwszą".

9



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

Załóżmy, że w zakresie od 2 do $\sqrt{n}$ znaleźliśmy dzielnik liczby n. Czy musimy wykonywać działania w pętli po raz kolejny?

10

Nie musimy.

Skoro mamy kolejny (inny niż 1 i n) całkowity dzielnik liczby n, to wiemy, że nie jest ona liczbą pierwszą. Rezygnujemy zatem z wykonywania kolejnych instrukcji. Przerywamy pętlę, używając instrukcji `break`. Zaznaczyliśmy ją na schemacie. Po przerwaniu pętli kończy się działanie programu: przechodzimy do bloku o nazwie STOP.

11

Wykorzystując przedstawiony algorytm dla liczb wskazanych w poleceniu tj. 997 i 1024, otrzymamy następujące wyniki:

```
1) 997 jest liczba pierwsza
2) 1024 nie jest liczba pierwsza
```

Polecenie 3

Napisz program, który obliczy sumę liczb nieparzystych z przedziału $\langle a, b \rangle$ oraz wypisze wszystkie liczby będące składnikami sumy. Użyj instrukcji `continue` lub `break`.

Specyfikacja problemu:

Dane:

- a – początek przedziału, dla którego obliczamy sumę liczb nieparzystych; liczba całkowita
- b – koniec przedziału, dla którego obliczamy sumę liczb nieparzystych; liczba całkowita

Wynik:

- Wydrukowany komunikat tekstowy zawierający informację o obliczonej sumie oraz wypisaną listę wszystkich liczb będących jej składnikami.

Swój program przetestuj dla przedziału $\langle 1, 100 \rangle$.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zaznacz poprawną odpowiedź. Który z pseudokodów NIE przedstawia pętli nieskończonej?

- ◯ `dopóki(true) {instrukcja}`
- ◯ `dopóki(5) {instrukcja}`
- ◯ `dopóki(false) {instrukcja}`
- ◯ `dopóki(1) {instrukcja}`

Ćwiczenie 2

Zaznacz wszystkie poprawne odpowiedzi. W których pętlach można używać instrukcji `break` oraz `continue`?

- ☐ pętle nieskończone
- ☐ `while`
- ☐ `for`

Ćwiczenie 3

Zaznacz poprawną odpowiedź. Która instrukcja przerywa wykonywanie bieżącego cyklu pętli i rozpoczyna kolejny?

- ◯ `while`
- ◯ `break`
- ◯ `continue`

Ćwiczenie 4

Przedstawiony pseudokod prezentuje algorytm sumujący wszystkie liczby całkowite z przedziału [1 ; k]. Jak wykorzystać słowo kluczowe `continue`, aby sumowane były tylko liczby niepodzielne przez 3?

```
wczytaj k
suma ← 0
Dla i=1,...,k wykonuj
    suma ← suma + i
```

- ○
```
wczytaj k
suma ← 0
Dla i=1,...,k wykonuj
    Jeżeli i mod 3 != 0
        continue
    suma ← suma + i
```

- ○
```
wczytaj k
suma ← 0
Dla i=1,...,k wykonuj
    suma ← suma + i
    Jeżeli i mod 3 = 0
        continue
```

- ○
```
wczytaj k
suma ← 0
Dla i=1,...,k wykonuj
    Jeżeli i mod 3 = 0
        suma ← suma + i
    w przeciwnym razie
        continue
```

- ○
```
wczytaj k
suma ← 0
Dla i=1,...,k wykonuj
    Jeżeli i mod 3 = 0
        continue
    suma ← suma + i
```

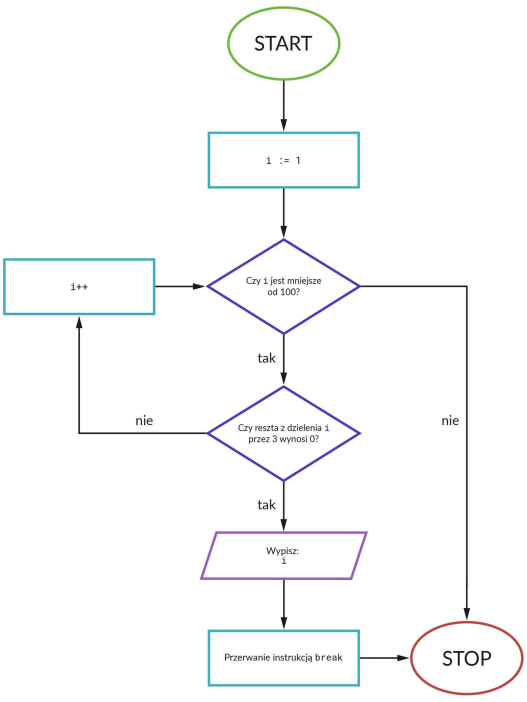
Ćwiczenie 5

Połącz w pary:

zagnieżdżona pętla	przerywa działanie pętli, w której się znajduje
iteracja	pętla wywołana wewnątrz innej pętli
instrukcja `break`	pojedyncze wykonanie wszystkich operacji wewnątrz pętli
instrukcja `continue`	pomija pozostałe instrukcje w danej iteracji i przechodzi do kolejnej

Ćwiczenie 6

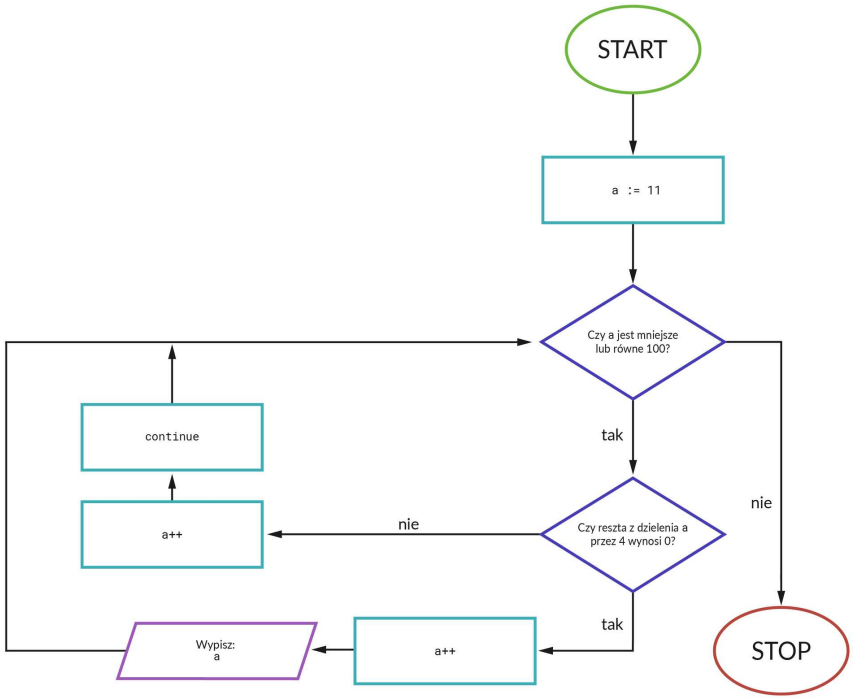
Zaznacz poprawną odpowiedź. Co zrobi program opisany w postaci następującego algorytmu?



- ◯ Wyświetli liczbę 3.
- ◯ Nie wyświetli żadnej liczby.
- ◯ Wyświetli wszystkie liczby podzielne przez 3, należące do przedziału od 1 do 100.

Ćwiczenie 7

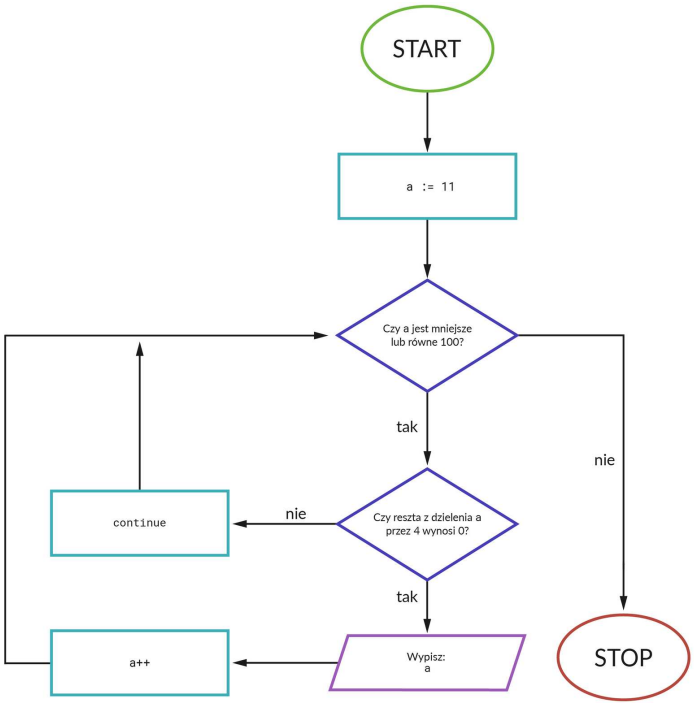
Zaznacz poprawną odpowiedź. Co zrobi program opisany w postaci następującego algorytmu?



- ◯ Wyświetli liczby niepodzielne przez 4 z zakresu od 11 do 100.
- ◯ Wyświetli jedną liczbę podzielną przez 4.
- ◯ Wyświetli jedną liczbę niepodzielną przez 4.
- ◯ Wyświetli liczby podzielne przez 4 z zakresu od 11 do 100.

Ćwiczenie 8

Zaznacz poprawną odpowiedź. Ile razy wykona się instrukcja `continue` według algorytmu przedstawionego za pomocą schematu blokowego?



- 23
- Będzie wykonywała się w nieskończoność
- 67
- 100

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Przerywanie pętli

Grupa docelowa:

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

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

> I. Rozumienie, analizowanie i rozwiązywanie problemów.
>
> Zakres podstawowy. Uczeń:
>
> 1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).
>
> 4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;
>
> 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);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

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):

- Zaimplementujesz różne typy pętli.
- Scharakteryzujesz działanie instrukcji `break` oraz `continue`.
- Przeanalizujesz przykłady zastosowania poleceń `break` i `continue`.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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: „Przerywanie pętli”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – do czego służą pętle?
   – co to jest iteracja?
   – w jaki sposób działają pętle zagnieżdżone?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny", czyta treść polecenia nr 1: „Za pomocą schematu blokowego lub języka programowania zapisz algorytm sprawdzający, czy dana liczba naturalna $n > 1$ jest liczbą pierwszą. Przetestuj działanie swojego programu dla liczby 957 oraz 1024. W programie należy użyć instrukcji `break`" i omawia kolejne kroki rozwiązania.
3. **Ćwiczenie umiejętności.** Nauczyciel przechodzi do sekcji „Sprawdź się". Uczniowie indywidualnie rozwiązują ćwiczenia nr 1-8 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".
2. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Podaj specyfikację problemu, który do rozwiązania będzie wymagał wykorzystania pętli zagnieżdżonej.

Wskazówki metodyczne:

- Treści w sekcji „Schemat interaktywny" można wykorzystać jako materiał służący powtórzeniu materiału.