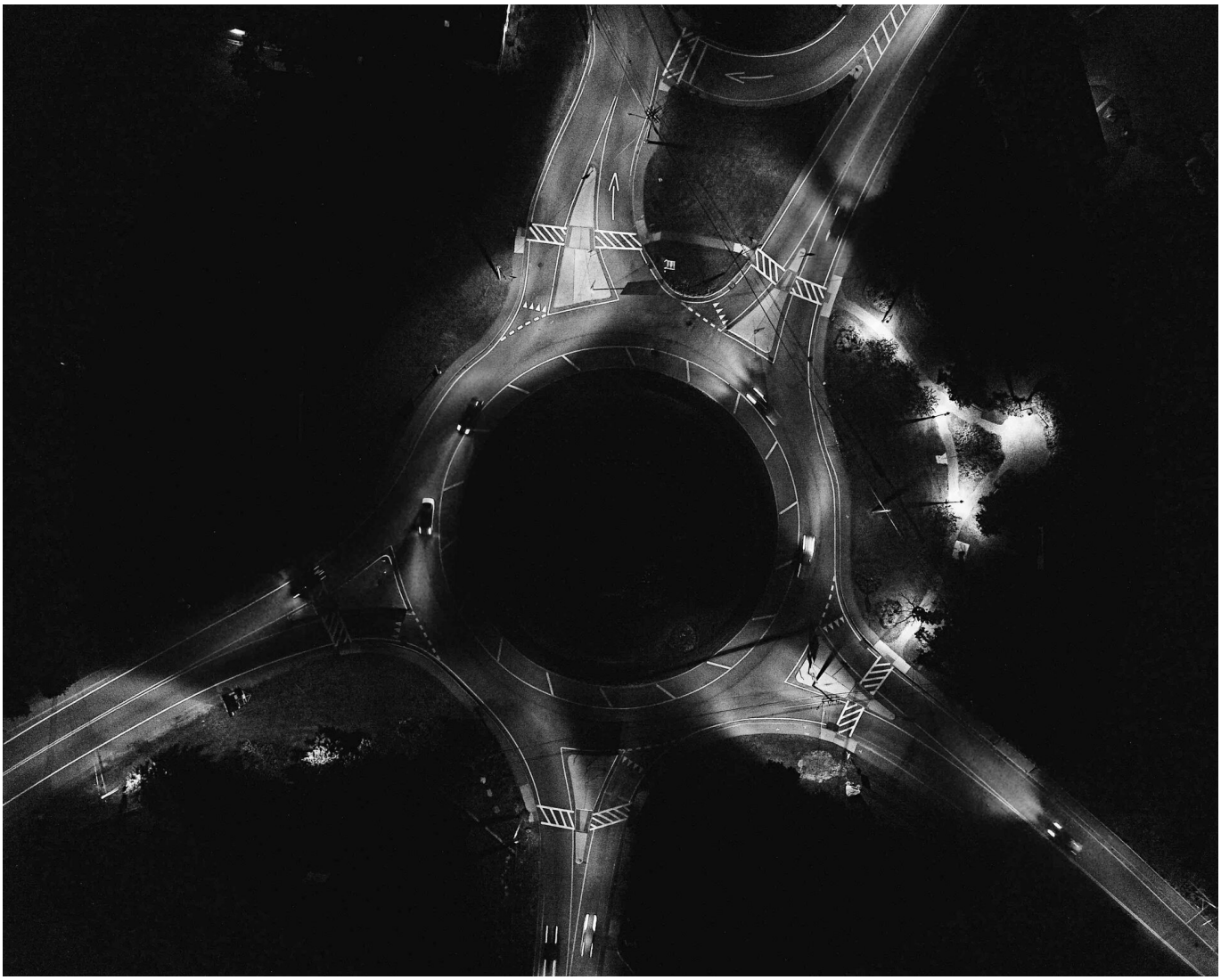


Instrukcja warunkowa

Źródło: Osman Rana, domena publiczna.

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

Decyzje są zazwyczaj uzależnione od wielu różnych czynników. Zakładamy, że dopiero wtedy, gdy dane warunki zostaną spełnione, zdecydujemy i wybierzemy odpowiednie rozwiązanie.

Decyzje muszą podejmować także programy komputerowe. W zależności od zaistniałej sytuacji muszą odpowiednio reagować. Dlatego wykorzystywanie instrukcji warunkowych to podstawa sterowania programami.

W tym e-materiale poznamy najważniejsze informacje dotyczące instrukcji warunkowych.

Implementacje instrukcji warunkowych w wybranych językach programowania znajdziesz w e-materiałach:

- Instrukcja warunkowa w języku C++,
- Instrukcja warunkowa w języku Java,
- Instrukcja warunkowa w języku Python.

Więcej zadań? Sięgnij do: Instrukcja warunkowa – ćwiczenia.

> **Twoje cele**

- Przeanalizujesz budowę instrukcji warunkowej.
- Dowiesz się, czym jest zagnieżdżanie warunków.
- Skonstruujesz własne instrukcje warunkowe.

Przeczytaj

Do czego służą instrukcje warunkowe?

Programy cały czas podejmują jakieś decyzje, w jaki sposób to robią?

My, podejmując decyzje, kierujemy się emocjami i rozumem – programy wykorzystują instrukcje warunkowe. Są to elementy w programowaniu, które mają za zadanie sprawdzić, czy podany warunek lub warunki zostały spełnione oraz odpowiednio na to zareagować.

Oto kilka przykładów sytuacji, w których wykorzystywane są instrukcje warunkowe:

- Sprawdzenie, czy podany login i hasło się zgadzają.
- Zablokowanie dostępu do zawartości PREMIUM, jeśli subskrypcja wygasła.
- Wysłanie powiadomienia, jeśli na konto bankowe przyszedł przelew.
- Sprawdzenie daty ważności ubezpieczenia.
- Przyznawanie rabatów podczas zakupów.
- Wysyłanie newsletterów.

Jak zbudowana jest instrukcja warunkowa?

Najpopularniejszą instrukcją warunkową jest tzw. instrukcja warunkowa if, która jest zbudowana z 3 elementów:

1. Sprawdzenie określonych warunków.
2. Reakcja, jeśli warunki są spełnione.
3. (opcjonalnie) Reakcja, jeśli warunki nie zostały spełnione.

```
jeśli te warunki są spełnione (warunek_1, warunek_2, ...)
{
    -> Wykonaj te operacje
}
a jeśli nie są spełnione
{
    -> Wykonaj te operacje
}
```

W wielu językach programowania instrukcja warunkowa wygląda podobnie. Dlatego tak ważne jest, by wiedzieć, do czego się jej używa i jak jest zbudowana. Oto przykłady tej samej instrukcji warunkowej napisanej w 3 różnych językach:

Instrukcja warunkowa `if` napisana w języku C++:

```
if (warunek_1, warunek_2, ...)
{
    // Operacje wykonywane, jeśli warunki są spełnione
}
else
{
   // Operacje wykonywane, jeśli warunki nie są spełnione
}
```

Instrukcja warunkowa `if` napisana w języku Java:

```
if (warunek_1, warunek_2, ...)
{
    // Operacje wykonywane, jeśli warunki są spełnione
}
else
{
   // Operacje wykonywane, jeśli warunki nie są spełnione
}
```

Instrukcja warunkowa `if` napisana w języku JavaScript:

```
if (warunek_1, warunek_2, ...)
{
    // Operacje wykonywane, jeśli warunki są spełnione
}
else
{
   // Operacje wykonywane, jeśli warunki nie są spełnione
}
```

Wszystkie powyższe implementacje instrukcji `if` są identyczne. Potwierdza to fakt, że instrukcja `if` w wielu językach wygląda i działa podobnie lub tak samo.

> **Ważne!**
>
> Powyższe fragmenty kodów nie będą działać prawidłowo, ponieważ zawierają pewne uproszczenie: instrukcje warunkowe oddzielone zostały znakiem przecinka. Na

> przyszłych lekcjach poznasz tzw. operatory logiczne, które zastąpią wyżej użyte znaki przecinka.

Czym jest zagnieżdżanie warunków?

Bardzo często okazuje się, że po sprawdzeniu jednego warunku trzeba sprawdzić jeszcze kolejne. Co więcej, bardzo często należy to zrobić **tylko wtedy**, gdy poprzednie warunki zostały spełnione. Zachodzi pytanie: **jak to zrobić?**

Odpowiedź jest prosta: można ponownie użyć kolejnej instrukcji warunkowej (wewnątrz poprzedniej). Taki mechanizm nazywa się zagnieżdżeniem instrukcji.

```
jeśli te warunki są spełnione (warunek_1, warunek_2, ...)
{
    jeśli kolejne warunki są spełnione (warunek_3, warunek_4, ...
    {
        -> Wykonaj te operacje
    }
}
a jeśli nie są spełnione
{
    -> Wykonaj te operacje

    jeśli kolejne warunki są spełnione (warunek_5, warunek_6, ...
    {
        -> Wykonaj te operacje
    }
}
```

Przyjrzyjmy się temu na przykładzie: chcemy napisać instrukcję warunkową, która sprawdzi, czy danemu klientowi przysługuje zniżka 10% na zakupy. Kupujący otrzymuje ją, jeśli posiada kartę stałego klienta oraz wartość jego zakupów przekracza 500 zł.

Najpierw sprawdźmy pierwszy warunek -> czy kupujący posiada kartę stałego klienta?

```
jeśli te warunki są spełnione ("klient posiada kartę stałego klie
{

}
jeśli te warunki nie są spełnione
{
```

```
7
8 }
```

Jeśli tak, możemy kontynuować sprawdzanie kolejnych warunków, jeśli nie, wypisujemy komunikat o niespełnieniu warunku.

```
1 jeśli te warunki są spełnione ("klient posiada kartę stałego klie
2 {
3     jeśli te warunki są spełnione ("wartość zakupów klienta przek
4     {
5
6     }
7     jeśli te warunki nie są spełnione
8     {
9
10    }
11 }
12 jeśli te warunki nie są spełnione
13 {
14    Niestety nie posiadasz karty stałego klienta.
15 }
```

Skoro klient posiada kartę stałego klienta, można sprawdzić kolejny warunek -> czy wartość zakupów przekroczyła 500 zł?

Jeśli nie, wyświetlmy komunikat o niespełnieniu warunku, a jeśli tak, to wyświetlmy komunikat o przyznanym rabacie.

```
1 jeśli te warunki są spełnione ("klient posiada kartę stałego klie
2 {
3     jeśli te warunki są spełnione ("wartość zakupów klienta przek
4     {
5         Gratulacje, dostałeś rabat 10 % na zakupy.
6     }
7     jeśli te warunki nie są spełnione
8     {
9         Niestety wartość Twoich zakupów nie przekracza 500 zł.
10    }
11 }
12 jeśli te warunki nie są spełnione
```

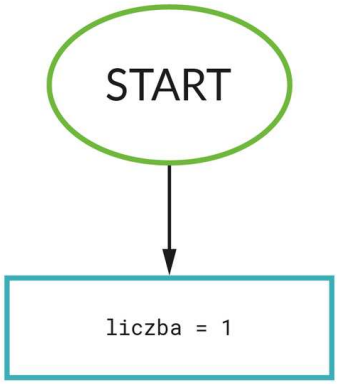
```
{
    Niestety nie posiadasz karty stałego klienta.
}
```

Łączenie instrukcji warunkowych z innymi elementami programowania

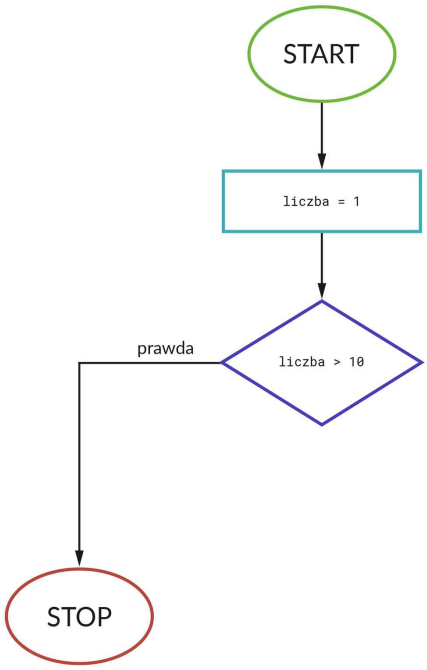
Prawdziwy potencjał sprawdzania warunków ujawnia się, gdy łączy się je z innymi elementami programowania, np. iteracją. Omówmy przykład wypisania wszystkich liczb parzystych z zakresu 1, 2, ..., 10.

Zastanówmy się, jaki warunek musi być spełniony, aby liczba była parzysta? Oczywiście musi się dzielić przez 2, ale jak to zapisać? Pomyśl nad tym, zaraz do tego wrócimy. Najpierw zacznijmy budować schemat blokowy.

Tworzymy zmienną, która przechowa nam sprawdzaną liczbę. Skoro zaczynamy od 1, to przypisujemy zmiennej `liczba` wartość 1:

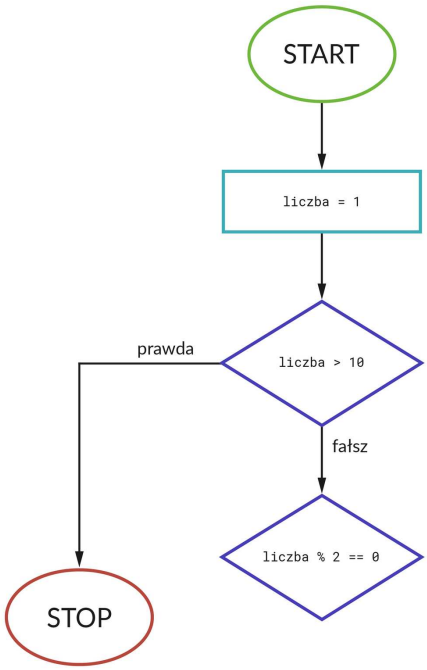


Potem kontrolujemy, czy sprawdzana liczba nie przekroczyła wartości 10 (sprawdzamy liczby z zakresu od 1 do 10) – jeśli tak, to kończymy algorytm:



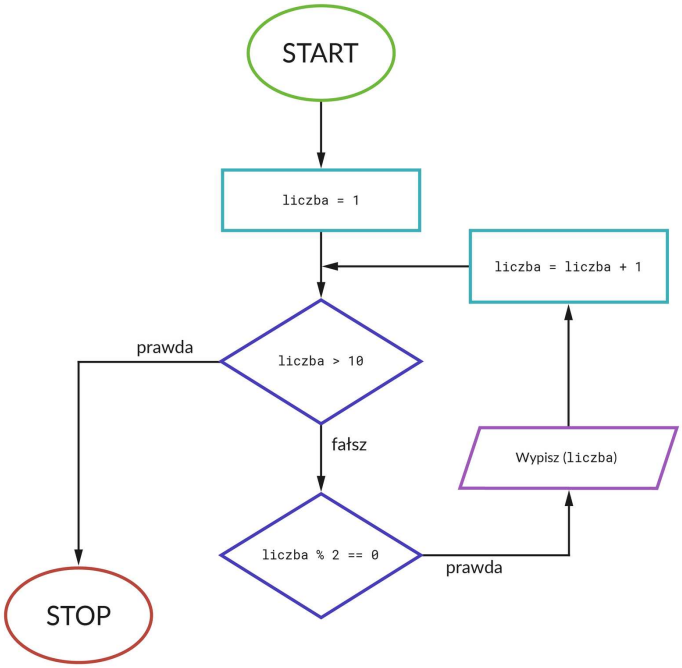
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W przeciwnym wypadku kontrolujemy, czy wartość zmiennej `liczba` jest podzielna przez 2. Wystarczy sprawdzić, czy reszta z dzielenia danej liczby przez 2 wynosi 0 (to warunek parzystości). Służy do tego operator `%`, który oznacza resztę z dzielenia (w naszym przypadku to `liczba % 2`), a operatorem porównania jest symbol `==`.



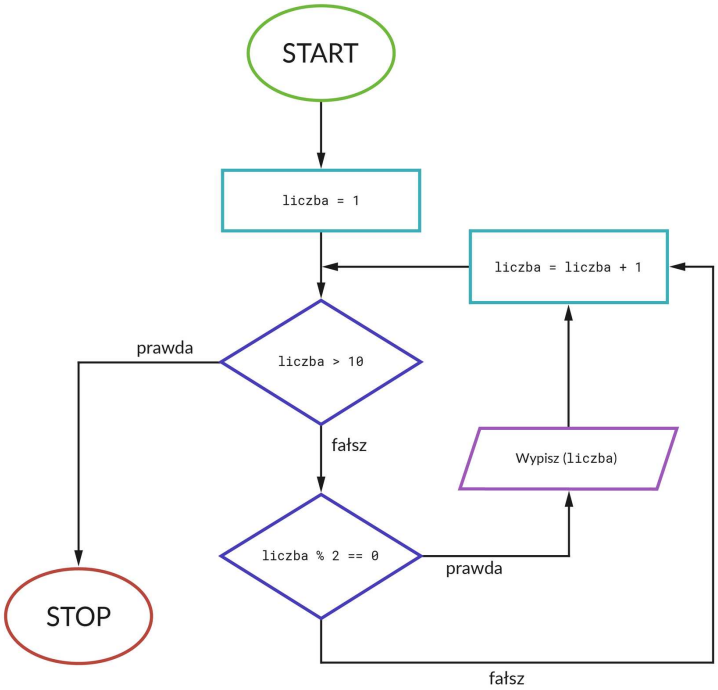
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Jeśli liczba jest parzysta, to wyświetlamy ją na ekranie i sprawdzamy kolejną liczbę.



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

Pozostało nam jedynie poprowadzić z drugiej instrukcji warunkowej, która sprawdza podzielność sprawdzanej liczby przez 2, strzałkę oznaczającą działanie algorytmu, jeżeli `liczba` nie będzie podzielna przez 2.



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

W ten oto sposób udało się nam stworzyć prosty algorytm wykorzystujący instrukcje warunkowe.

Słownik

> **instrukcja warunkowa `if`**

podstawowa instrukcja sprawdzająca dowolną liczbę warunków, zwracająca wynik TRUE (prawda) lub FALSE (fałsz)

instrukcja zagnieżdżona

instrukcja podrzędna znajdująca się wewnątrz podobnej instrukcji macierzystej

Schemat interaktywny

Polecenie 1

Za pomocą pseudokodu napisz program, który sprawdzi, czy z trzech odcinków o długościach: `bok_1`, `bok_2`, `bok_3` można zbudować trójkąt.

Specyfikacja:

Dane:

- *bok_1*, *bok_2*, *bok_3* – długości poszczególnych boków trójkąta; liczby rzeczywiste

Wynik:

W zależności od wyniku na standardowym wyjściu wyświetla się komunikat: *To jest trójkąt*, gdy z odcinków o zadanych długościach można zbudować trójkąt; w przeciwnym przypadku na konsoli wyświetla się komunikat: *To nie jest trójkąt*.

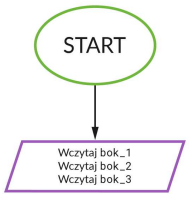
```
1
```

Jeśli chcesz powtórzyć wiadomości ze szkoły podstawowej, za pomocą bloków stwórz program, który sprawdzi, czy z trzech odcinków podanej długości można zbudować trójkąt.

Polecenie 2

Porównaj swoje rozwiązania z prezentacją.

1



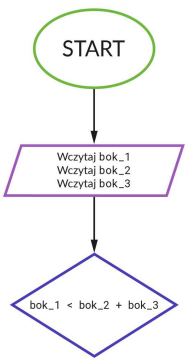
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0

Materiał audio dostępny pod adresem:

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

Najpierw należy wczytać długości 3 odcinków.

2



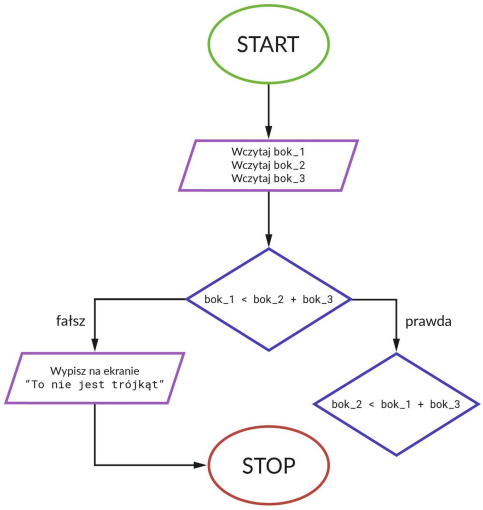
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0

Materiał audio dostępny pod adresem:

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

Teraz możemy zacząć sprawdzać warunki. Na początek sprawdźmy, czy długość odcinka bok_1 jest mniejsza od sumy odcinków bok_2 i bok_3.

3



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

Materiał audio dostępny pod adresem:

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

Jeśli warunek nie jest spełniony, wyświetlamy stosowny komunikat i kończymy program.

Jeśli natomiast jest spełniony, sprawdzamy czy długość odcinka bok_2 jest mniejsza od sumy odcinków bok_1 i bok_3.

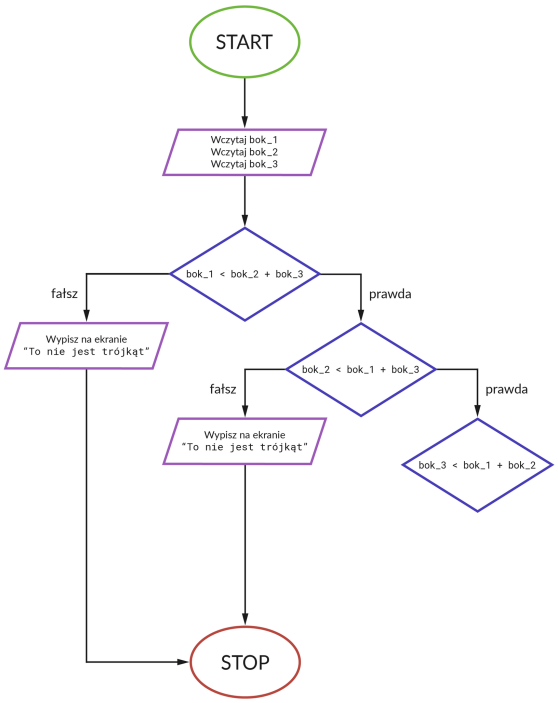
4

Materiał audio dostępny pod adresem:

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

Sprawdzamy kolejny warunek.

5



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

Materiał audio dostępny pod adresem:

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

Jeśli warunek nie jest spełniony, wyświetlamy stosowny komunikat i kończymy program.

Jeśli jest spełniony, sprawdzamy, czy długość odcinka `bok_3` jest mniejsza od sumy odcinków `bok_1` i `bok_2`.

6

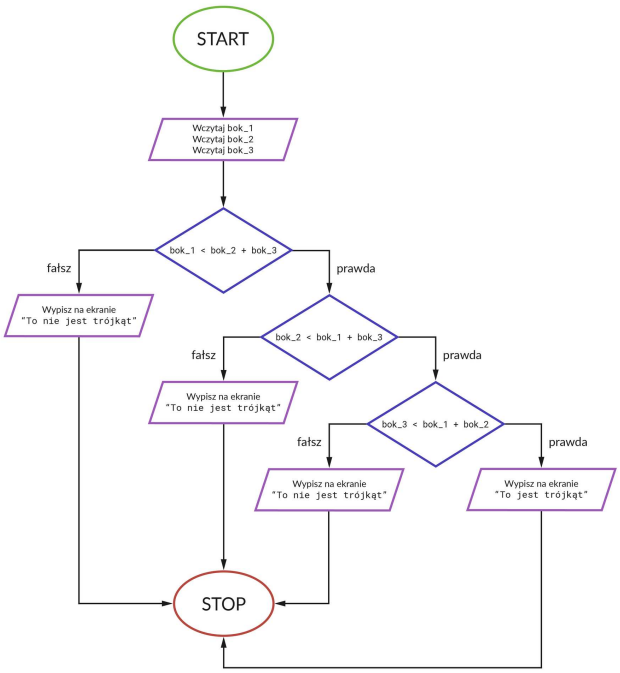
Materiał audio dostępny pod adresem:

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

Jeśli ostatni warunek nie został spełniony, robimy to samo, co poprzednio.

Jeśli jednak kontrola ma wynik pozytywny, oznacza to, że wszystkie warunki zostały spełnione.

7



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

Materiał audio dostępny pod adresem:

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

Na tym etapie już wiemy, że z podanych odcinków można utworzyć trójkąt. Wyświetlamy wynik działania naszego algorytmu i kończymy program.

8

Materiał audio dostępny pod adresem:

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

Po zbadaniu tego przykładu widzimy, jak wielki potencjał mają instrukcje warunkowe. Są one wykorzystywane zarówno w prostych programach, takich jak ten (sprawdzanie warunku trójkąta), jak i w tych bardzo skomplikowanych.

Materiał audio dostępny pod adresem:

9

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

Jeśli mimo wszystko dalej nie masz pewności, czy dokładnie rozumiesz działanie instrukcji warunkowych, radzimy przejrzeć ten materiał raz jeszcze. Niestety bez dobrego opanowania sprawdzania warunków uczenie się kolejnych elementów programowania może okazać się niemożliwe.

10

Materiał audio dostępny pod adresem:

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

Być może zastanawiasz się nad tym, czy nie można by uprościć tego algorytmu?

Owszem, można – potrzebna jednak będzie wiedza z łączenia warunków i operacji logicznych. Poruszymy te kwestie w przyszłych lekcjach.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż poprawną odpowiedź.

Czy bez instrukcji warunkowej można myśleć o napisaniu bardziej zaawansowanej aplikacji?

- ◯ tak
- ◯ nie

Ćwiczenie 2

Wskaż poprawną odpowiedź.

Czy instrukcje warunkowe można zagnieżdżać?

- ◯ nie
- ◯ tak

Ćwiczenie 3

Wskaż, do czego nie służy instrukcja warunkowa.

- ☐ Odpowiada za gromadzenie i układanie danych.
- ☐ Sprawdza, czy podane warunki są spełnione.
- ☐ Powtarza określone operacje podaną ilość razy.

Ćwiczenie 4

Wskaż poprawną odpowiedź.

Czy zawsze trzeba określać, co ma się stać w wypadku niespełnienia warunku?

- ◯ tak
- ◯ nie

Ćwiczenie 5

Czy instrukcje warunkowe można łączyć z innymi elementami programowania, np. pętlami?

- ◯ nie
- ◯ tak

Ćwiczenie 6

Wskaż poprawną odpowiedź.

Czy istnieje znaczna różnica w implementacji instrukcji warunkowej C++, Java i JavaScript?

- ◯ tak
- ◯ nie

Ćwiczenie 7

Uzupełnij zdanie.

Instrukcja warunkowa może mieć [ ] [ ].

możliwy rezultat | możliwych rezultatów | pięć | jeden | wiele | możliwe rezultaty | trzy | 2 | nieskończenie

Ćwiczenie 8

Wyjaśnij, czym jest instrukcja zagnieżdżona.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Instrukcja warunkowa

Grupa docelowa:

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

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.

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

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.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

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):

- Przeanalizujesz budowę instrukcji warunkowej.
- Dowiesz się, czym jest zagnieżdżanie warunków.
- Skonstruujesz własne instrukcje warunkowe.

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;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Instrukcja warunkowa". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel inicjuje rozmowę wprowadzającą w temat lekcji. Przedstawia cele zajęć oraz kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. Odnosząc się do treści zawartej w sekcji „Przeczytaj", uczniowie w parach konstruują alternatywny przykład, definiując samodzielnie problem, rozwiązanie i ewentualną implementację. Rezultaty omawiane są na forum klasy.
2. Uczniowie w dwuosobowych zespołach próbują rozwiązać Polecenie 1 zaprezentowane w sekcji „Schemat interaktywny". Zapoznają się z prezentacją.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-7 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

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.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 8 z sekcji „Sprawdź się".

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.
- W ramach przypomnienia wiadomości ze szkoły podstawowej uczniowie mogą przygotować pseudokod wskazanego przez nauczyciela algorytmu (np. szukania wartości minimalnej oraz maksymalnej w zbiorze).