Zapisywanie algorytmów za pomocą pseudokodu

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Rodion Kutsaev, dostępny w internecie: unsplash.com, domena publiczna.

Zapisywanie algorytmów za pomocą schematu blokowego nie zawsze jest wygodne, zwłaszcza gdy algorytm ma wiele rozgałęzień i jest długi albo przeciwnie – gdy ma mało rozgałęzień i cały schemat byłby bardzo wysoki. Z tego względu często stosuje się zapis w postaci listy kroków. Do problemów algorytmicznych można jednak wykorzystać jeszcze inny rodzaj zapisu – pseudokod. I właśnie nim zajmiemy się w tym e-materiale.

Więcej informacji i ćwiczeń dotyczących sposobów zapisywania algorytmów znajdziesz w e-materiałach:

- Zapisywanie algorytmów za pomocą listy kroków,
- Zapisywanie algorytmów za pomocą schematu blokowego,
- Zapisywanie algorytmów za pomocą schematu blokowego – ćwiczenia,
- Zapisywanie algorytmów – zadania maturalne.

Twoje cele

- Prześledzisz sposób zapisu algorytmu w postaci pseudokodu.
- Zapiszesz algorytmy za pomocą pseudokodu.
- Przeanalizujesz najważniejsze elementy tworzące składnię pseudokodu.

Przeczytaj

Zapisywanie algorytmów za pomocą pseudokodu

Poznaliśmy metodę zapisywania algorytmów za pomocą schematów blokowych oraz listy kroków. Pseudokod, czyli kolejny sposób zapisu, ma podobną strukturę, jednak pomijane są w nim kwestie semantyczne oraz składniowe na rzecz prostoty i czytelności.

Pseudokod przypomina języki programowania – można go więc konwertować na kod. Należy przy tym pamiętać, że pseudokod powinien być pisany bez wskazania na konkretny język. Można w nim stosować formuły matematyczne lub zdania w języku naturalnym, jeśli nie zaburza to czytelności.

Pseudokod nie ma jednego standardu zapisu – zazwyczaj korzysta się ze składni, która opiera się na składni istniejących języków programowania. W tym e-materiale posłużymy się zapisem, który pojawia się w arkuszach maturalnych.

> **Ważne!**
>
> Wszystkie formy zapisu pseudokodu (z wyjątkiem tablic) przedstawione w tym materiale, są zaczerpnięte ze sprawozdania z egzaminu maturalnego Centralnej Komisji Egzaminacyjnej

Wprowadzanie zmiennych

W pseudokodzie nie deklaruje się zmiennych i nie alokuje się jawnie żadnej pamięci, dlatego nowe zmienne mogą pojawiać się bez wcześniejszych zapowiedzi.

W arkuszach maturalnych przypisanie oznaczone jest odwróconą strzałką. Zdarza się też zapis przypisania za pomocą dwukropka i znaku równości.

> **Przykład 1**
>
> ```
> wprowadź a
> b ← 2 - a
> ```

Operatory arytmetyczne, relacyjne, logiczne, wartości logiczne

W pseudokodzie możliwe jest zapisywanie operacji arytmetycznych. W tym celu stosuje się następujące operatory arytmetyczne:

- $+$ operator dodawania,
- $-$ operator odejmowania,

- * operator mnożenia,
- / operator dzielenia,
- `div` dzielenie liczb całkowitych,
- `mod` dzielenie modulo (reszta z dzielenia), wyłącznie dla liczb całkowitych.

Operatorami porównania (relacyjnymi) używanymi przy zapisie pseudokodu są:

- $<$ logiczny operator relacji między dwoma wyrażeniami, zwraca wartość prawdziwą, gdy wartość wyrażenia po lewej stronie operatora jest mniejsza od wartości wyrażenia po jego prawej stronie,
- $>$ logiczny operator relacji między dwoma wartościami, zwraca wartość prawdziwą, gdy wartość po lewej stronie operatora jest większa od wartości po jego prawej stronie,
- $\leq$ logiczny operator relacji między dwoma wartościami, zwraca wartość prawdziwą, gdy wartość wyrażenia po lewej stronie operatora jest mniejsza lub równa wartości wyrażenia po jego prawej stronie,
- $\geq$ logiczny operator relacji między dwoma wartościami, zwraca wartość prawdziwą, gdy wartość wyrażenia po lewej stronie operatora jest większa lub równa wartości wyrażenia po jego prawej stronie,
- $=$ logiczny operator relacji między dwoma wartościami, zwraca wartość prawdziwą, gdy wartości wyrażeń po obu jego stronach są sobie równe,
- $\neq$ logiczny operator relacji między dwoma wartościami, zwraca wartość prawdziwą, gdy wartości wyrażeń po obu jego stronach są różne od siebie.

Operatory logiczne używane są przy zapisie pseudokodu w formie słów:

- `i` – iloczyn logiczny, zwraca wartość prawdziwą, gdy wyrażenia po obu stronach tego operatora są prawdziwe,
- `lub` – suma logiczna, zwraca wartość prawdziwą, gdy prawdziwe jest co najmniej jedno z wyrażeń, na których użyto operatora,
- `jest/nie jest` – operatory pozwalające na tworzenie wyrażenia, często wykorzystywane są do określenia stanu wyrażenia razem z wartościami logicznymi lub do tworzenia warunków opartych na stanie obiektu.

Wartości logiczne używane w pseudokodzie:

- `prawda` – wartość logiczna określająca prawdziwość zdania logicznego,
- `fałsz` – wartość logiczna określająca fałszywość zdania logicznego.

Instrukcje warunkowe

Podstawowe słowa kluczowe używane w pseudokodzie przy zapisie instrukcji warunkowych:

- `jeżeli` – odpowiednik `if` w językach programowania,

- `w przeciwnym wypadku` – odpowiednik `else` w językach programowania.

Słowo kluczowe `jeżeli` sprawia, że blok kodu, który następuje po tym słowie, będzie wykonany raz, **jeżeli** warunek występujący po tym słowie będzie spełniony.

> **Przykład 2**
>
> ```
> 1 ...
> 2 jeżeli a > b wykonuj
> 3     wypisz "początek bloku"
> 4     wypisz "a większe od b"
> 5     wypisz "koniec bloku"
> ```
>
> Wcięcia w linijkach 3-5 wyznaczają zakres bloku, który podporządkowany jest słowu kluczowemu `jeżeli`. Kod tam się znajdujący zostanie wykonany, jeśli będzie spełniony warunek przy słowie kluczowym.

Zwrot `w przeciwnym wypadku` oznacza alternatywę dla kodu wykonywanego w przypadku spełnienia warunku zapisanego przy słowie kluczowym `jeżeli`. Jest wykonywany w przypadku niespełnienia tego warunku. Analogicznie jak instrukcja `else` jest umieszczana po instrukcji `if`, tak instrukcja `w przeciwnym razie` musi być umieszczona w instrukcji `jeżeli`.

> **Przykład 3**
>
> ```
> 1 ...
> 2 jeżeli a > b
> 3     wypisz "początek bloku"
> 4     wypisz "a większe od b"
> 5     wypisz "koniec bloku"
> 6 w przeciwnym razie
> 7     wypisz "początek bloku"
> 8     wypisz "a nie większe od b"
> 9     wypisz "koniec bloku"
> ```
>
> Wcięcia w linijkach 3-5 wyznaczają zakres bloku, który podporządkowany jest słowu kluczowemu `jeżeli`. Kod tam się znajdujący zostanie wykonany, jeśli spełniono warunek przy słowie kluczowym.
>
> Wcięcia w linijkach 7-9 wyznaczają zakres bloku, który podporządkowany jest zwrotowi `w przeciwnym razie`. Kod ten zostanie wykonany, gdy nie będzie spełniony warunek po słowie kluczowym `jeżeli`.

Instrukcje warunkowe, zgodnie z nazwą, sugerują zawieranie warunku, od którego zależy ich wykonanie. Te warunki to zdania logiczne, wykorzystujące operatory porównania i operatory logiczne.

Pętle

Pętla jest to konstrukcja pozwalająca na cykliczne wykonywanie zawartego w niej fragmentu kodu, tak długo jak długo spełniony jest zawarty w niej warunek.
W pseudokodzie podstawowymi słowami występującymi przy zapisie pętli są:

- `dopóki` – instrukcja powtarzania, odpowiednik polecenia `while` w językach programowania, w połączeniu z warunkiem i słowem kluczowym `wykonaj` pozwala na powtarzalne wykonywanie sparowanego z nim kodu, tak długo jak długo spełniony jest warunek,
- `wykonuj` – instrukcja powtarzania, może być wykorzystana jak polecenie `do` w językach programowania, wraz ze słowem kluczowym `dopóki` i warunkiem pozwala na zapisanie pętli odpowiadającej pętli `do ... while` w językach programowania; instrukcja służy do sygnalizowania, jaki fragment kodu jest zawarty w pętli,
- `dla` – instrukcja iteracji, słowo kluczowe do stworzenia pętli odpowiadającej pętli `for` w językach programowania, pozwala na wykonanie zawartego w pętli kodu określoną liczbę razy.

Słowo kluczowe `dopóki` powoduje, że blok, który po nim następuje, będzie wykonywany, **dopóki** warunek zapisany po słowie kluczowym pozostaje spełniony.

> **Przykład 4**
>
> ```
> 1 ...
> 2    dopóki a > b wykonuj
> 3        wypisz "początek bloku"
> 4        wypisz "a większe od b"
> 5        wypisz "koniec bloku"
> ```
>
> Wcięcia w linijkach 3-5 wyznaczają zakres bloku kodu, który podporządkowany jest słowu kluczowemu `dopóki` i który będzie wykonywany tak długo, jak długo pozostanie spełniony warunek podany w tej konstrukcji.

Słowo kluczowe `wykonuj` może być wykorzystane do stworzenia kolejnego rodzaju pętli, która zawsze wykona się co najmniej raz i będzie powtarzana tak długo, jak długo będzie spełniony warunek (w językach programowania to pętla `do ... while`). Jej konstrukcja jest podobna do poprzedniej z przedstawianych pętli, jednak blok kodu znajduje się między słowami kluczowymi `wykonuj` oraz `dopóki`.

> **Przykład 5**

```
1 ...
2    wykonuj
3        wypisz "a większe od b"
4        a ← a - b
5    dopóki a > b
```

Blok kodu zawarty w pętli (linie 3-4) zawsze wykona się co najmniej raz, niezależnie od wartości logicznej zawartego w niej warunku. Od spełnienia warunku zależeć będą kolejne powtórzenia pętli

Słowo kluczowe `dla` pozwala na stworzenie pętli pozwalającej na powtórzenie zawartego w niej kodu określoną liczbę razy (pętla `for` w językach programowania). Pętla ta zamiast warunku zawiera zmienną, której przypisujemy dozwolony zakres wartości i ustalamy, w jaki sposób ma zmieniać swoje wartości (np. rosnąć o 1, wzrastać dwukrotnie i tak dalej). Warto zwrócić uwagę, że w instrukcji `dla` wartość przypisujemy znakiem `=`.

Przykład 6

```
1 ...
2    dla n = 1, 2, ..., k wykonuj
3        wypisz n
```

Funkcje

Kolejną istotną notacją jest sposób zapisu funkcji. Funkcje i procedury to wydzielone części programu wykonujące jakieś operacje, które mogą zostać wywołane podczas działania programu. Mogą one przyjmować argumenty i zwracać wartości. W pseudokodzie będziemy je zapisywać następująco:

```
1 funkcja nazwa(argument1, argument2...., argumentN):
2     ....
3     zwróc ...
```

W miejsce elementu `nazwa` wpisujemy nazwę funkcji, której chcemy użyć. Argumentów może być nieskończenie wiele – taka notacja stanowi alternatywę dla pisania `wprowadź a, wprowadź b, wprowadź...`. Oto przykładowa funkcja zapisana w pseudokodzie:

```
1 funkcja przykład(a, b, c):
2     jeżeli a > b
```

```
3         zwróć c
4     w przeciwnym razie
5         zwróć b
```

Załóżmy, że jej argumentami są liczby naturalne:

```
1 a = 1
2 b = 2
3 c = 3
```

Funkcję wywołamy następująco:

```
1 funkcja przykład(a, b, c):
2     jeźeli a > b
3         zwróć c
4     w przeciwnym razie
5         zwróć b
6
7 przykład(1, 2, 3)
```

W pierwszym kroku sprawdzamy, czy 1 jest większe od 2. Nie jest, więc pomijamy instrukcję zawartą w **linijce 3**.

Od razu przechodzimy do **linijek 4** oraz **5**, zwracamy zatem liczbę 2.

Tablice

Tablica jest to kontener przechowujący dane tego samego typu, w których dane dostępne są za pomocą indeksów, najczęściej przyjmujące wartości liczbowe. Zapisując pseudokod przy nazwie tablicy w nawiasach kwadratowych, musimy zapisać zakres indeksów, czyli od jakiego indeksu zaczyna się numeracja elementów tablicy i na jakim się kończy. Odczyt elementu z tablicy zapisujemy w postaci nazwy tablicy i wybranego indeksu w nawiasach kwadratowych.

W pseudokodzie indeksy tablicy zazwyczaj zaczynają się od 1, ale coraz częściej numeruje się od 0, co jest zgodne z językami programowania, których wykorzystanie jest dopuszczone na egzaminie maturalnym.

```
1 tablica[1..200]
```

```
wypisz tablica[8]
```

W ten sam sposób można też w pseudokodzie zapisać inne struktury o dostępie swobodnym, w razie potrzeby zastępując indeks wybranym kluczem.

> **Ważne!**
>
> Konstrukcje przedstawione są zgodnie ze sprawozdaniem z egzaminu maturalnego sporządzonym przez CKE, jednak nie są to wszystkie polecenia potrzebne do zapisania bardziej złożonych algorytmów. Problem ten jest rozwiązywany dzięki temu, że w pseudokodzie rezygnuje się ze ścisłych reguł składniowych i zezwala na opisywanie bardziej skomplikowanych instrukcji (które nie zostały przedstawione w tym e-materiale) oraz kroków algorytmów w języku naturalnym.

Przykłady

Potrafimy już zapisać algorytmy za pomocą listy kroków oraz schematu blokowego, pora więc na przeprowadzenie ich konwersji do pseudokodu.

Algorytm liniowy

Zapoznajmy się ze specyfikacją algorytmu, który oblicza czas potrzebny na pokonanie drogi `s2` na podstawie tego, ile czasu (`t1`) zajęło pokonanie drogi `s1`.

Zakładamy, że droga pokonywana jest z taką samą prędkością.

Specyfikacja problemu:

Dane:

- `t1` – liczba rzeczywista; czas przebycia poprzedniej drogi wyrażony w godzinach
- `s1` – liczba rzeczywista; długość wcześniejszej drogi wyrażona w kilometrach
- `s2` – liczba rzeczywista; długość drogi wyrażona w kilometrach

Wynik:

- `t2` – liczba rzeczywista; czas przebycia drogi wyrażony w godzinach

Gdy podamy niezbędne dane, musimy obliczyć prędkość, z jaką poruszano się przy pokonywaniu drogi `s1`. Prędkość obliczamy, dzieląc pokonaną drogę przez czas potrzebny na jej pokonanie. Korzystamy tu ze wzoru na prędkość:

$$V = \frac{s}{t}$$

gdzie:

- V to prędkość wyrażona w kilometrach na godzinę
- t to czas wyrażony w godzinach
- s to długość drogi wyrażona w kilometrach

Przekształcamy ten wzór, aby uzyskać wzór na czas potrzebny na pokonanie danej drogi:

$$t = \frac{s}{V}$$

Podstawiamy odpowiednie wartości do wzoru i obliczamy wynik.

Lista kroków

1. Podaj `t1`.
2. Podaj `s1`.
3. Podaj `s2`.
4. Przypisz zmiennej `V` wynik dzielenia `s1` przez `t1`.
5. Przypisz zmiennej `t2` wynik dzielenia `s2` przez `V`.
6. Wyświetl wartość zmiennej `t2`.
7. Zakończ działanie algorytmu.

Pseudokod

```
wprowadź t1
wprowadź s1
wprowadź s2
V ← s1 / t1
t2 ← s2 / V
wypisz t2
```

Algorytm rozgałęziony

Zapoznajmy się ze specyfikacją algorytmu sprawdzającego, czy dana liczba jest liczbą dodatnią.

Specyfikacja problemu:

Dane:

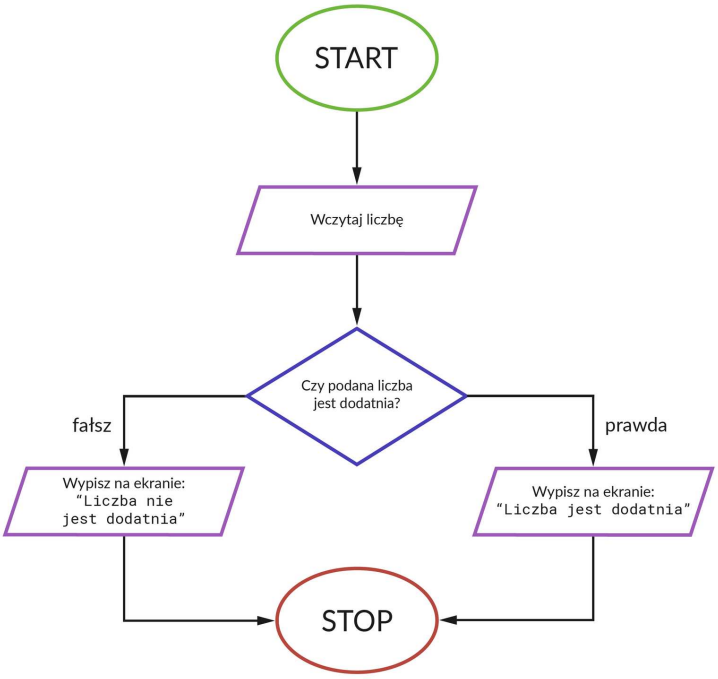
- `liczba` – liczba rzeczywista

Wynik:

- komunikat `Liczba nie jest dodatnia` lub `Liczba jest dodatnia`

> **Ważne!**
>
> Zakładamy, że 0 nie jest liczbą dodatnią.

Schemat blokowy



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

Pseudokod

```
wprowadź liczba
jeżeli liczba > 0
    wypisz "Liczba jest dodatnia"
w przeciwnym razie
    wypisz "Liczba nie jest dodatnia"
```

Algorytm z pętlą

Zapoznajmy się z przykładowym algorytmem obliczania sumy liczb naturalnych w danym przedziale. Zakładamy, że `liczba1` jest mniejsza od `liczba2`.

Specyfikacja problemu:

Dane:

- `liczba1` – liczba naturalna; początek przedziału; `liczba1 < liczba2`
- `liczba2` – liczba naturalna; koniec przedziału; `liczba2 > liczba1`

Wynik:

- `suma` – liczba naturalna; suma liczb naturalnych w danym przedziale

Lista kroków

1. Podaj `liczba1`.
2. Podaj `liczba2`.
3. Przypisz zmiennej `i` wartość `liczba1`.
4. Przypisz zmiennej `suma` wartość `liczba1`.
5. Zwiększ wartość zmiennej `i` o 1.
6. Dodaj do zmiennej `suma` wartość zmiennej `i`.
7. Jeżeli:
    - `i` jest równa `liczba2`
        - przejdź do kroku 8.
    - `i` jest różna od `liczba2`
        - wróć do kroku 5.
8. Wyświetl wartość zmiennej `suma`.
9. Zakończ działanie algorytmu.

Pseudokod

```
wprowadź liczba1
wprowadź liczba2
i ← liczba1
suma ← liczba1

dopóki i != liczba2
    i ← i + 1
    suma ← suma + i

wypisz suma
```

Algorytm z tablicą

Zapoznajmy się z przykładowym algorytmem sortującym liczby przechowywane w tablicy niemalejąco. Algorytm wykorzystuje również pętle i instrukcje warunkowe. Więcej o tym algorytmie możesz przeczytać w materiale dotyczącym sortowania bąbelkowego.

Specyfikacja problemu:

Dane:

- `n` – liczba naturalna; liczba elementów tablicy `tablica`

- `tablica` – `n`-elementowa tablica nieposortowanych liczb rzeczywistych

Wynik:

- `tablica` – `n`-elementowa tablica posortowanych niemalejąco liczb rzeczywistych

Pseudokod

```
n ← 200
tablica[1..n]

dla i = 1, 2, ..., n - 1 wykonuj
    dla j = 2, 3, ..., n - (i - 1) wykonuj
        jeżeli tablica[j - 1] > tablica[j]
            tymczasowy ← tablica[j - 1]
            tablica[j - 1] ← tablica[j]
            tablica[j] ← tymczasowy

wypisz tablica
```

Słownik

> **funkcja i procedura**
>
> inaczej podprogram, czyli ciąg instrukcji zawartych w bloku, który może być wielokrotnie wykorzystywany w różnych miejscach programu; funkcja zwyczajowo zwraca wartość, a procedura nie zwraca
> **notacja**
>
> ustalony sposób zapisu
> **semantyka**
>
> dziedzina nauki zajmująca się znaczeniem wyrazów oraz badaniem związków zachodzących między wyrażeniami języka a przedmiotami, do których się odnoszą
> **składnia**
>
> nauka zajmująca się budową wypowiedzeń (zdań); bada funkcje elementów zdania i zależności między nimi
> **specyfikacja algorytmu**
>
> (specyfikacja problemu) opisanie problemu przez podanie danych, z których korzysta algorytm oraz określenie wyniku, który ma być efektem działania algorytmu

Prezentacja multimedialna

Pseudokod – rozwiązanie przykładowego problemu

Polecenie 1

W tej sekcji przedstawimy proces tworzenia pseudokodu na podstawie prostego algorytmu. Zapoznaj się z nim i wykonaj ćwiczenia.

Specyfikacja problemu:

Dane:

- `dzien` – liczba naturalna z przedziału $[1, 7]$; numer dnia tygodnia; w programie zakładamy, że liczba 1 odpowiada poniedziałkowi

Wynik:

- program wypisuje nazwę odpowiedniego dnia tygodnia

1

Pseudokod wykorzystuje elementy języka naturalnego. Jest jednak bardziej usystematyzowany niż lista kroków.
Źródło: Pixabay, pixabay.com, dostęp 18.11.2022, CC 0.

Materiał audio dostępny pod adresem:

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

Napiszmy algorytm, który będzie pobierał od użytkownika numer dnia tygodnia i zwracał nazwę podanego dnia. W programie zakładamy, że liczba 1 odpowiada poniedziałkowi, a kolejne liczby następującym po nim dniom tygodnia.

2

Materiał audio dostępny pod adresem:

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

Zanim przystąpimy do pisania pseudokodu, przeanalizujmy zadania, które program będzie wykonywał.

1. W algorytmie chcemy pobrać od użytkownika wartość liczbową (cyfry od 1 do 7). W zależności od podanej wartości, algorytm ma wyświetlić nazwę odpowiedniego dnia tygodnia. Nie chcemy, by kod kończył działanie po pobraniu i odkodowaniu pierwszej wartości, wykorzystamy zatem pętlę.
2. Żeby wiedzieć, jaki komunikat wyświetlić użytkownikowi w odpowiedzi na podawane przez niego wartości, powinniśmy je w jakiś sposób od siebie odróżnić. Zastosujemy zatem instrukcje warunkowe i wartości wzorcowe.
3. Damy użytkownikowi możliwość zakończenia działania pętli.

3

Materiał audio dostępny pod adresem:

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

Po określeniu zadań, możemy przystąpić do zapisania algorytmu. Operacja pobierania danych od użytkownika i ich przetwarzania ma

być powtarzana do momentu otrzymania od użytkownika sygnału do przerwania pracy, co w praktyce oznacza użycie pętli. Biorąc pod uwagę, że chcemy, aby ciąg poleceń wykonał się co najmniej raz (żeby np. dostać od użytkownika sygnał do końca pracy), to odpowiednią pętlą będzie pętla `wykonuj ... dopóki warunek`. Warunek zakończenia pętli dodamy w następnych krokach.

```
wykonuj
...
dopóki warunek
```

4

Materiał audio dostępny pod adresem:

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

Kolejnym krokiem algorytmu będzie pobranie danych od użytkownika. W pseudokodzie rezygnujemy ze ścisłych reguł składniowych, co pozwala na użycie języka naturalnego do opisu kroków algorytmu. Opis pobierania danych od użytkownika zapiszemy jako polecenie `wprowadź`.

Potrzebna będzie zmienna, do której zapiszemy podane przez użytkownika dane.

```
wykonuj
    wprowadź dzien
dopóki warunek
```

5

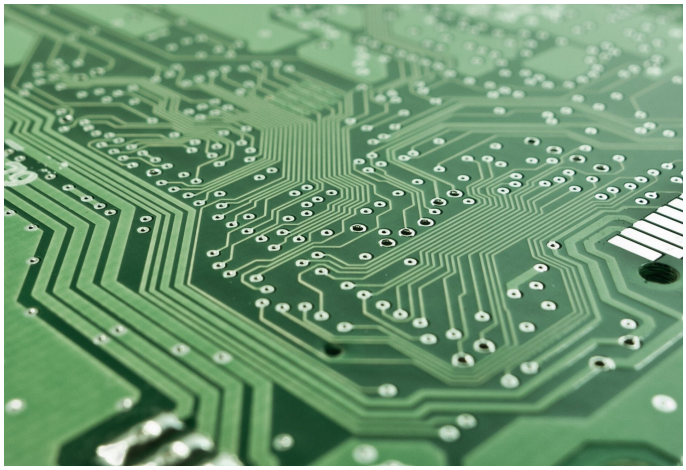
Materiał audio dostępny pod adresem:

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

Kolejny krok to obsługa otrzymanych danych. Program ma zwracać różne komunikaty w zależności od wartości wczytanych danych. W tym celu konieczne będzie rozróżnianie wczytanych danych i dostosowanie komunikatu do ich wartości. Zrobimy to za pomocą instrukcji warunkowych i polecenia `wypisz`. Podejście to wymaga zastosowania instrukcji `jeżeli`, dla każdej obsługiwanej wartości pobieranych danych (wartości liczbowe od 1 do 7) – przyrównania wartości pobranej od użytkownika do wartości pożądanych i komunikatów wyświetlanych w każdym z tych przypadków. Żeby nie tworzyć osobnej instrukcji warunkowej dla każdej z obsługiwanych wartości, instrukcję `jeżeli` wykorzystamy w połączeniu z instrukcją `w przeciwnym razie` do stworzenia jednej rozbudowanej instrukcji warunkowej.

```
1  wykonuj
2      wprowadź dzien
3      jeżeli dzien = 1
4          wypisz "poniedziałek"
5      w przeciwnym razie jeżeli dzien = 2
6          wypisz "wtorek"
7      w przeciwnym razie jeżeli dzien = 3
8          wypisz "środa"
9      w przeciwnym razie jeżeli dzien = 4
10         wypisz "czwartek"
11     w przeciwnym razie jeżeli dzien = 5
12         wypisz "piątek"
13     w przeciwnym razie jeżeli dzien = 6
14         wypisz "sobota"
15     w przeciwnym razie jeżeli dzien = 7
16         wypisz "niedziela"

17 dopóki warunek
```



Choć pseudokod swoją konstrukcją może przypominać język programowania, komputer nie jest w stanie go przetworzyć.
Źródło: Pixabay, pixabay.com, dostęp 18.11.2022, CC 0.

Materiał audio dostępny pod adresem:

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

Spełniliśmy już dwa z trzech założeń. Pozostaje do zaimplementowania możliwość przerwania działania programu przez użytkownika. Wykorzystamy do tego niezapisany jeszcze warunek pętli. Przyjmijmy, że program będzie kończył swoje działanie, gdy użytkownik poda mu liczbę 0. Oznacza to, że pętla będzie wykonywać się dopóty, dopóki użytkownik za pomocą klawiatury nie wprowadzi wartości 0.

```
1 wykonuj
2     wprowadź dzien
3     jeżeli dzien = 1
4         wypisz "poniedziałek"
5     w przeciwnym razie jeżeli dzien = 2
6         wypisz "wtorek"
7     w przeciwnym razie jeżeli dzien = 3
8         wypisz "środa"
```

```
9       w przeciwnym razie jeżeli dzien = 4
10          wypisz "czwartek"
11      w przeciwnym razie jeżeli dzien = 5
12          wypisz "piątek"
13      w przeciwnym razie jeżeli dzien = 6
14          wypisz "sobota"
15      w przeciwnym razie jeżeli dzien = 7
16          wypisz "niedziela"

17  dopóki dzien ≠ 0
```

Po tym etapie otrzymujemy kompletny kod, spełniający wszystkie początkowe założenia.

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

Polecenie 2

Zmodyfikuj pseudokod przedstawiony w prezentacji tak, by wyświetlał komunikat `Podaj poprawny dzień tygodnia`, jeśli użytkownik wprowadzi liczbę spoza przedziału `[0,7]`. Program powinien kończyć swoje działanie, kiedy użytkownik poda liczbę `0`.

Specyfikacja problemu:

Dane:

- `dzien` – liczba naturalna; numer dnia tygodnia; w programie zakładamy, że liczba 1 odpowiada poniedziałkowi

Wynik:

- program wypisuje nazwę odpowiedniego dnia tygodnia lub komunikat `Podaj poprawny dzień tygodnia`

```
1
```

Polecenie 3

Dodaj do swojego pseudokodu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż zdanie prawdziwe.

- Pseudokod może zawierać pewne niejasności w opisie realizacji algorytmu.
- W pseudokodzie musimy dbać o alokowanie i zwalnianie pamięci.
- Pseudokod tworzony jest zazwyczaj na podstawie schematu blokowego.
- Pseudokod jest formą zapisu algorytmu, która przypomina języki programowania.

Ćwiczenie 2

Zdecyduj, które elementy są istotne, a które nieistotne dla pseudokodu, i przyporządkuj je do właściwych kategorii.

Elementy istotne

Elementy pomijalne

czytelność | semantyka | poprawność zapisu | poprawność algorytmu | dostosowanie do języka programowania | jasność przekazu | syntaktyka | jednoznaczność

Ćwiczenie 3

Oznacz każde pojęcie właściwym kolorem.

Sposoby zapisu algorytmu | Rodzaj algorytmu | Obiekt matematyczny

pseudokod
algorytm geometryczny
język naturalny
średnia geometryczna
schemat blokowy
kwadrat

Ćwiczenie 4

Uzupełnij tekst właściwymi fragmentami.

W pseudokodzie istotne jest, aby był on przede wszystkim _____.

W pseudokodzie _____ tworzenie tablic.

_____ jeden standard zapisu pseudokodu.

Jeżeli pseudokod zostanie napisany w języku, którego nie znamy, wówczas pseudokod ten będzie _____.

użyteczny | poprawny semantycznie | nie jest możliwe | Istnieje | bezużyteczny | jest możliwe | zrozumiały | Nie istnieje

Ćwiczenie 5

Zapisz za pomocą pseudokodu algorytm, który oblicza napięcie elektryczne.

Wzór na obliczenie napięcia elektrycznego:

$$U = \frac{W}{q}$$

gdzie:

- U – napięcie elektryczne
- W – praca
- q – ładunek elektryczny

Specyfikacja problemu:

Dane:

- `W` – liczba rzeczywista; praca podana w dżulach
- q – liczba rzeczywista; ładunek elektryczny podany w kulombach

Wynik:

- U – liczba rzeczywista; napięcie elektryczne podane w woltach

```
1
```

Ćwiczenie 6

Za pomocą pseudokodu zapisz algorytm, który rozwiązuje równanie liniowe (znajduje miejsce zerowe funkcji liniowej).

Wzór funkcji liniowej:

$$f(x) = ax + b$$

gdzie:

- a – współczynnik kierunkowy prostej
- b – wyraz wolny

Specyfikacja problemu:

Dane:

- a – liczba rzeczywista; współczynnik przy niewiadomej x
- b – liczba rzeczywista; wyraz wolny równania

Wynik:

- x – liczba rzeczywista, wartość niewiadomej, dla której równanie jest oznaczone (podana funkcja liniowa przyjmuje wartość 0), komunikat `Równanie tożsamościowe`, gdy równanie jest spełnione dla wszystkich liczb rzeczywistych (współczynniki a i b równe 0), lub komunikat `Równanie sprzeczne`, gdy równanie nie ma rozwiązań (współczynnik a równy 0 i b różny od 0)

```
1
```

Ćwiczenie 7

Zapisz za pomocą pseudokodu algorytm, który wypisuje liczby parzyste z przedziału zamkniętego od 0 do n.

Specyfikacja problemu:

Dane:

- n – liczba naturalna

Wynik:

- liczby naturalne parzyste z przedziału `[0, n]`

```
1
```

Ćwiczenie 8

Zapisz za pomocą pseudokodu algorytm, który wyznacza naturalne dzielniki pewnej liczby naturalnej.

Specyfikacja problemu:

Dane:

- `liczba` – liczba naturalna

Wynik:

- wypisane dzielniki naturalne podanej liczby

```
1
```

Ćwiczenie 9

Za pomocą pseudokodu zapisz algorytm obliczania średniej n liczb rzeczywistych podanych przez użytkownika. Wartość n również podawana jest przez użytkownika.

Specyfikacja problemu:

Dane:

- n – liczba naturalna; informacja, z ilu liczb obliczana będzie średnia
- a_1, a_2…, a_n – liczby rzeczywiste podane przez użytkownika

Wynik:

- `srednia` – liczba rzeczywista; średnia arytmetyczna

```
1
```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Zapisywanie algorytmów za pomocą pseudokodu

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.

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).

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;

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):

- Prześledzisz sposób zapisu algorytmu w postaci pseudokodu.
- Zapiszesz algorytmy za pomocą pseudokodu.
- Przeanalizujesz najważniejsze elementy tworzące składnię pseudokodu.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- burza mózgów;
- mapa myśli;
- 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. Uczniowie przypominają sobie najważniejsze informacje na temat sposobów zapisu algorytmów poznanych na wcześniejszych lekcjach.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Zapisywanie algorytmów za pomocą pseudokodu". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. Metodą burzy mózgów uczniowie przypominają najważniejsze informacje na temat sposobów zapisu algorytmów poznanych na wcześniejszych lekcjach.
3. Chętna lub wybrana osoba zapisuje informacje na tablicy za pomocą mapy myśli.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel sprawdza przygotowanie uczniów do lekcji. W razie potrzeby prosi uczniów o zapoznanie się z sekcją „Przeczytaj". Jeśli część klasy zapoznała się z nią w domu, nauczyciel prosi o przygotowanie propozycji uzupełnienia mapy myśli stworzonej na początku lekcji o informacje dotyczące pseudokodu.
2. Nauczyciel wyświetla uczniom algorytmy zapisane za pomocą listy kroków oraz schematów blokowych, zamieszczone w wybranych e-materiałach. Prosi uczniów, aby w parach przygotowali zapis tych algorytmów za pomocą pseudokodu.
3. Chętne lub wybrane osoby prezentują przygotowany pseudokod. Nauczyciel omawia go na forum klasy.
4. **Praca z multimedium.** Nauczyciel prosi uczniów, aby w parach zapoznali się z treścią „Prezentacji multimedialnej" i wykonali polecenia nr 1 oraz nr 2.
5. Uczniowie indywidualnie wykonują ćwiczenia 5–9 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność rozwiązań.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 1–4 z sekcji „Sprawdź się" .
2. Uczniowie wykonują polecenie nr 3 z sekcji „Prezentacja multimedialna".

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Prezentacja multimedialna”, „Sprawdź się” jako materiał do lekcji powtórkowej.