Instrukcja warunkowa

SPIS TREŚCI

Wstęp

Programy komputerowe realizowane są "po kolei"[1] — od "początku" do "końca", jedno polecenie po drugim tak długo, aż nie ma już żadnego polecenia do wykonania. W jakimś sensie przypomina to działanie klasycznego projektora filmowego wyświetlającego klatki filmu z celuloidowej taśmy — od początku, klatka po klatce, do końca taśmy.

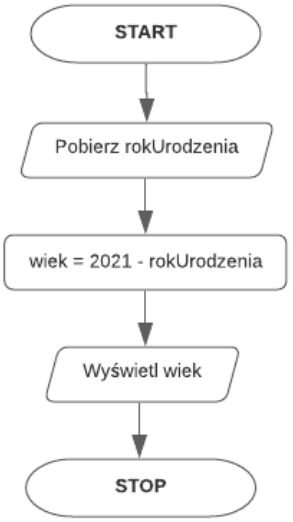
Programy które pisaliśmy dotychczas funkcjonowały dokładnie w ten sam, podany powyżej sposób. Po ich rozpoczęciu deklarowaliśmy i inicjowaliśmy jakieś zmienne, następnie wyświetlaliśmy jakiś komunikat na ekranie prosząc użytkownika o wprowadzenie jakiś danych, które następnie stawały się przedmiotem przetwarzania. Na koniec wynik obliczeń wraz z tekstem przewodnim wyświetlaliśmy na ekranie.

Jednak programy z których korzystamy w codziennym życiu działają raczej inaczej. Niezależnie od tego czy jest to strona internetowa czy aplikacja z której korzystamy za pomocą komputera czy smartfona jest ona interaktywna. Użytkownik decyduje co chce zobaczyć na ekranie, czy jakiego rodzaju obliczenia przeprowadzić. Program nie działa na zasadzie od początku do końca i od góry do dołu, tylko daje użytkownikowi możliwość decydowania o przebiegu programu, o tym jakie informacje mają być wyświetlane na ekranie, czy jakie obliczenia mają zostać wykonane.

[1] Uściślenie tego określenia można przeczytać w Dywagacjach pod koniec tej instrukcji.

Po kolei, czyli od góry do dołu

Poniższy schemat blokowy prezentuje sekwencję poleceń dzięki której obliczany jest wiek użytkownika komputera na podstawie roku urodzenia.



Na początku pobierany jest rok urodzenia użytkownika, następnie obliczany jest wiek jako różnicą między rokiem bieżącym (2021) a podanym rokiem urodzenia. Ostatecznie wartość ta wyświetlana jest na ekranie.

Tak napisany program zawsze działa tak samo — po każdym uruchomieniu realizowana jest zawsze ten sama sekwencja poleceń, a rola użytkownika ogranicza się ewentualnie do wprowadzenia z klawiatury danych potrzebnych do wykonania np. obliczeń.

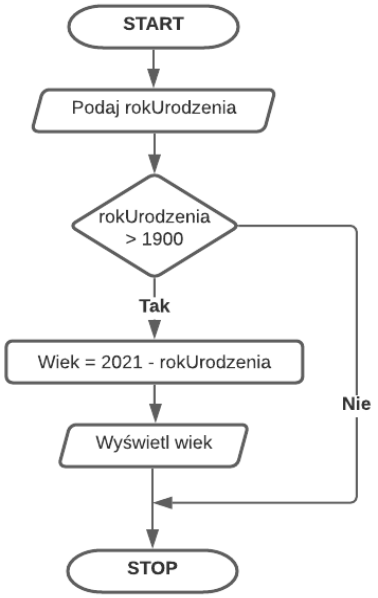
W języku programowania przedstawione powyżej kroki zapisaliśmy np. w sposób podany poniżej.

```
Scanner sc = new Scanner(System.in);

System.out.println("Podaj swój rok urodzenia");

int rokUrodzenia = sc.nextInt();

int wiek = 2021 - rokUrodzenia;

System.out.println("Masz " + wiek + " lat");
```

W powyższym przypadku użytkownik na prośbę systemu podaje swój rok urodzenia, następnie obliczany jest wiek, który ostatecznie wraz z tekstem wiodącym jest wyświetlany na ekranie. Przebieg programu jest zawsze dokładnie taki, nawet gdy użytkownik poda nieprawidłowe dane — np. rok 1900.

if, czyli jeżeli

Jednak zwykle wchodząc na jakąś stronę internetową, czy uruchamiając jakąś aplikację chcemy mieć wpływ na to, co zostanie wyświetlone na ekranie, jak będzie działał program. Aby tak się stało podejmujemy pewne decyzje za pomocą udostępnionym nam narzędzi interfejsu pozwalających na skierowanie działania programu w interesującym nas kierunku.



Podejmując określoną decyzję rozgałęziamy przebieg programu. Od tej chwili kolejne instrukcje nie są już wykonywane od góry do dołu, jedna po drugiej i od początku do końca tylko zgodnie z określonym — w informatyce używa się pojęcia przepływu sterowania — który kierujemy w kierunku realizacji pożądanej funkcjonalności. Podstawowym narzędziem rozdzielenia przepływu sterowania jest instrukcja warunkowa if.

Słowo if oznacza po angielsku jeżeli i jego składnia jako deklaracji warunkowej pokazana jest poniżej.

```
if (warunek) działanie;
```

Tak więc po słowie kluczowym if zamieszczamy parę nawiasów okrągłych, a w nich warunek który zwraca jedną z dwóch wartości: true albo false, czyli prawdę albo fałsz.

W przypadku gdy warunek zwraca prawdę wykonywane jest działanie. Przykładowo jeżeli wartość zmiennej jest dodatnia wyświetlany jest stosowny komunikat.

```
if (x > 0) System.out.println("Wartość x > 0");
```

Natomiast w przypadku gdy warunek zwraca fałsz działanie jest pomijane.

Dotyczy to także przypadku w którym działanie ma postać bloku składającego się z wielu, np. trzech poleceń. Gdy wprowadzona przez użytkownika wartość zmiennej x jest ujemna cały boki poleceń jest pomijany.

```
if (x > 0)

{
     System.out.println("Wartość x > 0");
     System.out.println("Uruchom program ponownie");
     System.out.println("i podaj poprawną wartość");
}
```

Poniższy przykład obrazuje modyfikację przedstawionego kodu źródłowego aplikacji obliczającej wiek użytkownika na podstawie podanego roku urodzenia, zmodyfikowanego przez użycie instrukcji warunkowej. W tym przypadku wiek obliczany jest jedynie w przypadku gdy użytkownik podał rok urodzenia większy niż 1900, natomiast w przeciwnym przypadku obliczenia i wyświetlanie ich wyników są pomijane.

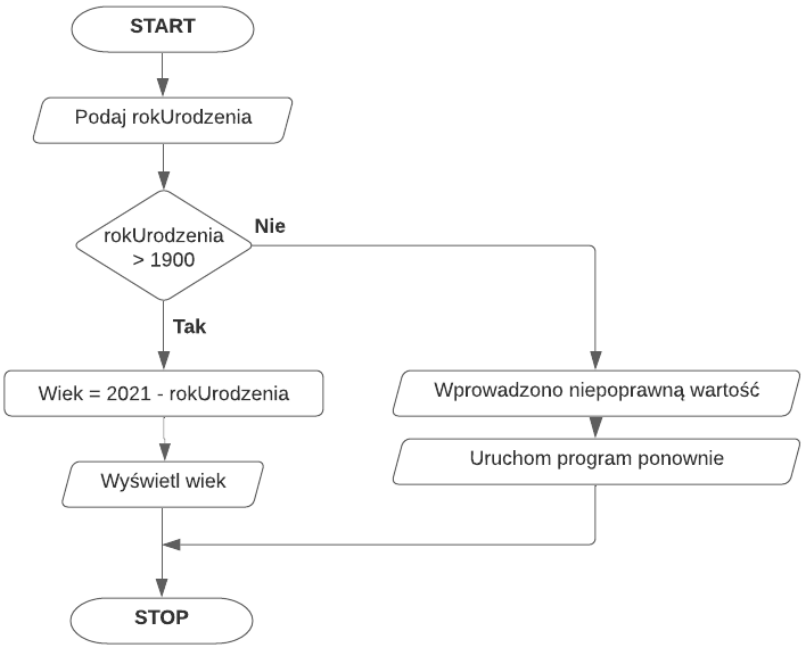
```
Scanner sc = new Scanner(System.in);

System.out.println("Podaj swój rok urodzenia");

int rokUrodzenia = sc.nextInt();

if(rokUrodzenia > 1900)

{

int wiek = 2021 - rokUrodzenia;

System.out.println("Masz " + wiek + " lat");

}
```

else, czyli w przeciwnym przypadku

Deklaracja else jest uzupełnieniem funkcjonalności funkcji warunkowej o możliwość deklarowania działania lub działań w przypadku gdy rezultat weryfikacji warunku jest fałszywy.

W takiej sytuacji w przypadku gdy jest prawdziwy realizowano jest jedno działanie lub zbiór działań, natomiast gdy nie jest spełniony inne działanie lub zbiór działań.

Najlepiej obrazuje to kolejny schemat blokowy, tym razem obrazujący rozgałęzienie przepływu sterowania oraz realizacje alternatywnych zestawów poleceń. W przypadku gdy podany przez użytkownika rok urodzenia jest większy niż 1900, wtedy obliczany jest wiek i wyświetlany na ekranie komputera, natomiast gdy użytkownik poda mniejszą wartość lub równą 1900 wyświetlany jest komunikat o tym, że wprowadzono niepoprawną wartość oraz konieczności ponownego uruchomienia programu[2].



Tak więc w tym przypadku możemy uzupełnić podany wcześniej kod źródłowy programu

```
Scanner sc = new Scanner(System.in);

System.out.println("Podaj swój rok urodzenia");

int rokUrodzenia = sc.nextInt();

if(rokUrodzenia > 1900)

{

     int wiek = 2021 - rokUrodzenia;

     System.out.println("Masz " + wiek + " lat");

}

else
```

[2] Możliwość wielokrotnego powtarzania wprowadzania wartości do czasu wprowadzenia poprawnej wartości przez użytkownika jest przedstawione w instrukcji dotyczącej pętli.

```
{
     System.out.println("Podano niepoprawną wartość");
     System.out.println("Uruchom program ponownie");
}
```

if, else if, else

Ostatnim przypadkiem użycia instrukcji warunkowej jest sytuacja w której sprawdzane jest po kolei wiele warunków i pierwszy z nich który zwrócić prawdę zostanie wykonany, a jeżeli żaden nie zwróci prawdy to wykonany zostanie polecenie lub blok poleceń zadeklarowanych w części else. Tym razem prezentację rozpoczniemy od kodu źródłowego.

```
Scanner sc = new Scanner(System.in);

System.out.println("Podaj swój rok urodzenia");

int rokUrodzenia = sc.nextInt();

if(rokUrodzenia > 1900)

{

     int wiek = 2021 - rokUrodzenia;

     System.out.println("Masz " + wiek + " lat");

}

else if (rokUrodzenia < 1900)

{
     System.out.println("Podano zbyt małą wartość");
     System.out.println("Jest mało prawdopodobne,"
   +   "że masz ponad 100 lat!");
}

else if (rokUrodzenia > 2021)

{
     System.out.println("Wprowadziłeś zbyt dużą wartość");
     System.out.println("Jest mało prawdopodobne,"
   +   "że masz ponad 100 lat!");
}

else

{
    System.out.println("Bez obliczeń wiem, że masz 121 lat!");
    System.out.println("Gratuluję wspaniałego zdrowia!");
}
```

Sekwencję if else można powtarzać wielokrotnie, czego jednak się unika stosując w zamian raczej polecenie switch.

switch

Polecenie switch jest wygodną alternatywą dla if / else if / else w przypadku gdy dostępnych przypadków w instrukcji warunkowej jest wiele.

Ogólna forma tego polecenia podana jest poniżej, gdzie wyrażenie nie jest wyrażeniem logicznym, tylko zmienną o określonej wartości sprawdzanej na samym początku wykonywania polecenia switch, natomiast x, y są konkretnymi wartościami które może przyjmować wyrażenie.

```
switch (wyrażenie)

{

case x:
            // polecenia
            break;


case y:
            // polecenia
            break;

default:
            // polecenia
}
```

Wyrażenie case x: oznacza przypadek gdy wyrażenie ma wartość x , natomiast polecenie break powoduje wyjście z wyrażenia switch . W przypadku pominięcia polecenia break wykonywane są kolejne polecenia aż do napotkania kolejnego polecenia break , a gdy takie nie wystąpi to wykonywane są także polecenia z przypadku domyślnego (default).

Wspomniana zmienna może być typu podstawowego takiego jak int czy char , może także być typu String , ale nie może być np. typu double .

Poniższy przykład ilustruje użycie polecenia switch dla najbardziej oczywistego przypadku, czyli zmiennej typu int .

Indywidualne karty w talii kart zawierającej 52 karty można reprezentować za pomocą kolejnych liczb od 0 do 51, gdzie 0 może oznaczać dwójkę karo, 1 trójkę, a wartości 9, 10, 11 i 12 odpowiednio waleta, damę, króla i asa.

Natomiast kolejna liczba, czyli 13 przy tych założeniach reprezentuje już dwójkę w kolejnym kolorze czyli kier, 14 trójkę, itd.

Tak więc dysponując wartością reprezentującą pojedynczą kartę można łatwo ustalić jej rangę oraz kolor. Przykładowo wartość 35 reprezentuje waleta pik, co wynika z wyniku dzielenia wartości karty przez ilość kart w określonym kolorze, których jest 13.

```
35 / 13 = 2

35 % 13 = 9
```

Wynik pierwszego dzielenia, 2, oznacza trzeci kolor licząc od zera, czyli pik. Natomiast wartość 9 — otrzymana przez obliczenie reszty z dzielenia 35 przez — zgodnie z tym co napisano wcześniej odpowiada waletowi.

Za pomocą dwóch instrukcji switch możemy łatwo zaprogramować wyświetlenie odpowiedniej rangi karty oraz jej koloru.

```
int karta = sc.nextInt();

int ranga - karta % 13;
int kolor = karta / 13;

switch (ranga)

{
     case : 0
          System.out.print("2 ");
          break;

     case : 1
          System.out.print("3 ");
          break;

     case : 2
          System.out.print("4 ");
          break;

     case : 3
          System.out.print("5 ");
          break;

     case : 4
          System.out.print("6 ");
          break;

     case : 5
          System.out.print("7 ");
          break;
```

```
     case : 6
          System.out.print("8 ");
          break;

     case : 7
          System.out.print("9 ");
          break;

     case : 8
          System.out.print("10 ");
          break;

     case : 9
          System.out.print("Walet ");
          break;

     case : 10
          System.out.print("Dama ");
          break;

     case : 11
          System.out.print("Król ");
          break;

     case : 12
          System.out.print("As ");
}

switch (kolor)

{
     case : 0
          System.out.println("♦");
          break;

     case : 1
          System.out.println("♥");
          break;

     case : 2
          System.out.println("♠");
          break;


     case : 3
          System.out.println("♣");
          break;

     default:
          System.out.println("Nie ma takiego koloru");
}
```

Warto zwrócić uwagę na ważny fakt, że zadeklarowanie bloku poleceń wewnątrz pojedynczej opcji switch nie wymaga użycia klamr, które muszą obejmują wszystkie dostępne opcje.

Co więcej deklarowanie przypadka default nie jest obligatoryjne. Co więcej w pierwszym switch-u dodanie go nie miałoby sensu. W związku z tym, że nie ma też potrzeby użycia polecenia break w przypadku ostatniej opcji. Natomiast w przypadku drugiej deklaracji switch wystąpienie opcji default jest zasadne i dlatego też została zadeklarowana.

Polecenie switch może być użyte w połączeniu ze zmienną typu char , przeznaczoną do przechowywania pojedynczych znaków, co może być przydatne podczas programowania gier w których użytkownikowi daje się możliwość sterowania przebiegiem gry za pomocą klawiszy na klawiaturze.

```
char akcja = sc.nextChar();

switch (akcja)

{
     case 'w':

          break;

     case 'a':

          break;

     case 's':

          break;

     case: 'z':

          break;

default:

     System.out.println("Wybrano inną opcję");
}
```

Ostatni przykład ilustruje użycie polecenia switch ze zmienną typu String przeznaczoną do przechowywania ciągów znaków na przykładzie "aplikacji" wyświetlającej wróżby dla określonych znaków zodiaku.

```
System.out.print("Podaj swój znak zodiaku: ");

znak = sc.next();
```

```
switch (znak)

{
     case: "Baran":

     System.out.println("Czeka Ciebie trudny rok, pełen wyrzeczeń");
     break;

     case: "Wodnik":

     System.out.println("W tym roku zarobisz dużo pieniędzy");
     break;

     System.out.println("Niestety, pojawią się problemy ze zdrowiem");

     case: "Rak":

     System.out.println("Spotkasz miłość swojego życia");
     break;

     case: "Koziorożec":

    System.out.println("Osiągniesz wielki sukces zawodowy");
    break;

    case: "Bliźnięta":

    System.out.println("Udasz się w podróż o której marzyłeś");
    break;

    case: "Lew":

    System.out.println("Pojawi się nowy członek rodziny");
    break;

    case: "Waga":

    System.out.println("Czas zacząć bardziej dbać o zdrowie");
    break;

    case: "Ryby":

    System.out.println("Wkrótce zmienisz pracę na mniej płatną");
    break;

    case: "Strzelec":

    System.out.println("Twój partner ma dla ciebie niespodziankę");
    break;

    case: "Skorpion":
```

```
    System.out.println("Nauczysz się kolejnego języka");
    break;

    case: "Byk":

    System.out.println("Zadbaj o ubezpieczenie mieszkania");
    break;

    case: "Panna":

    System.out.println("Stracisz dużo pieniędzy na giełdzie");
}
```

Oczywiście w tym miejscu abstrahujemy od tego, czy aplikacja o takiej architekturze ma jakikolwiek sens :) .

Wyrażenie warunkowe

Omawiając instrukcję warunkową warto wspomnieć o bardzo wygodnym w niektórych przypadkach wyrażeniu warunkowym, którego ogólna forma jest następująca:

```
zmienna = warunek ? wartość_prawda : wartość_fałsz ;
```

Za pomocą wyrażenia warunkowego można na przykład uzyskać wartość bezwzględną liczby. Gdy warunek określony z prawej strony równania jest prawdą wtedy wyrażenie zwraca wartość a, natomiast gdy jest fałszywe wartość -a .

```
int a;
...

a = a > 0 ? a : -a
```

Zadania

Zadania wymagające wykorzystania instrukcji warunkowej if / else dostępne są w sekcje *Instrukcje warunkowe* zbioru zadań.

Przejdź do zbioru zadań: Instrukcja warunkowa

Dywagacje

Określenie "po kolei" wydaje się intuicyjnie proste, ale może być rozumiane na co najmniej trzy sposoby.

1. Wykonywanie poleceń następuje jedno po drugim. Tzn. po *wywołaniu* jednego polecenia można natychmiast *wywołać* następne. Ten sposób realizacji poleceń nazywa się asymetrycznym.

2. Czas wykonania poszczególnych poleceń może znacznie się różnić, ich realizacja po kolei może być rozumiana jako wykonywanie kolejnego polecenia dopiero po zakończeniu wykonywania poprzedniego, co zwykle wymaga większego lub mniejszego czasu oczekiwania. Ten sposób realizacji poleceń nazywa się symetrycznym i jest tradycyjnym sposobem realizacji poleceń programu komputerowego.

3. Trzeci sposób rozumienia określenia po kolei — który au rebours jest tematem tej instrukcji — określa liniowy ciąg poleceń wykonywanych w ten czy inny sposób od góry do dołu, po kolei i bez rozgałęzień. Sekwencja poleceń jest jak sekwencja kart do gry które po kolei, jedna po drugiej wykłada się na stół, a gdy wyłoży się ostatnią gra się kończy.

Dowiedz się więcej

---

1. **Polecenie switch**
   https://www.w3schools.com/java/java_switch.asp
   https://www.programiz.com/java-programming/switch-statement