Temat C4. Stosowanie instrukcji warunkowych w językach C++ i Python

Warto powtórzyć

1. Wyjaśnij pojęcie *słowo kluczowe*.
2. Na czym polega deklarowanie zmiennych w języku C++?
3. Czy w języku Python deklaruje się zmienne? Uzasadnij odpowiedź.
4. W jaki sposób możemy nadać wartości zmiennym w językach C++ i Python?
5. Do czego stosuje się instrukcję przypisania?
6. Jak wyróżnić blok instrukcji (blok kodu) w językach C++ i Python?

1. Sytuacje warunkowe

Sytuacja warunkowa występuje wtedy, gdy wynik lub dalsze działanie zależą od spełnienia (lub niespełnienia) pewnego warunku. Algorytm zawierający sytuacje warunkowe nazywamy **algorytmem z warunkami** (inaczej: **algorytmem z rozgałęzieniami**).

Sytuacje warunkowe występują w matematyce i fizyce, gdy wykonanie obliczeń zależy od warunku, który muszą spełniać dane, np. powinny być różne od zera lub dodatnie.

Ćwiczenie 1. Podajemy przykłady sytuacji warunkowych

1. Podaj przykład dwóch zadań, w których występują sytuacje warunkowe.
2. Określ, czy w zadaniu „Oblicz liczbę znaków różnych od spacji w dowolnym tekście.” (temat C1, ćwiczenie 2., punkt 3.) występuje sytuacja warunkowa. Uzasadnij odpowiedź.

2. Algorytmy z warunkami w językach C++ i Python

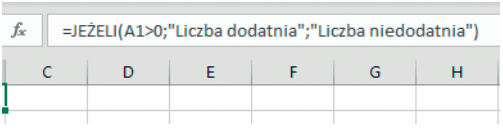
Chcemy napisać program sprawdzający, czy wprowadzona z klawiatury liczba jest dodatnia czy niedodatnia (ujemna albo równa zero). Jak zrealizować ten algorytm w arkuszu kalkulacyjnym? W jaki sposób napisać program realizujący sytuację warunkową w wybranym języku programowania?

2.1. Proste warunki logiczne

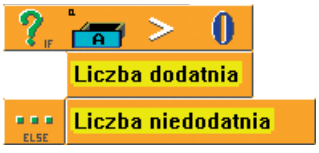
Aby w wybranym języku programowania napisać program realizujący algorytm z warunkami, korzystamy z **instrukcji warunkowej**.

Instrukcja warunkowa ma postać i działanie zbliżone do funkcji JEŻELI arkusza kalkulacyjnego:
=JEŻELI(*test_logiczny*;*wartość_jeżeli_prawda*;*wartość_jeżeli_fałsz*).

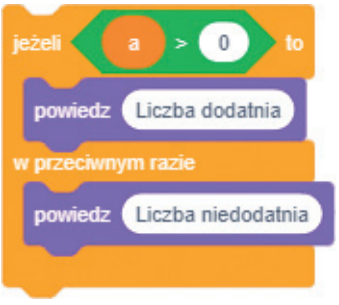
Arkusz kalkulacyjny Excel



Baltie



Scratch



C++

```
if(a > 0)
    cout << "Liczba dodatnia";
else
    cout << "Liczba niedodatnia";
```

Python

```
if a > 0:
    print("Liczba dodatnia")
else:
    print("Liczba niedodatnia")
```

Rys. 1. *Zastosowanie funkcji JEŻELI w arkuszu kalkulacyjnym i instrukcji warunkowej w dydaktycznych środowiskach programowania (Baltie, Scratch) oraz w językach programowania (C++, Python) do realizacji tego samego algorytmu z warunkiem*

Ćwiczenie 2. Analizujemy realizację sytuacji warunkowej w różnych narzędziach

Porównaj zapis sytuacji warunkowej w różnych narzędziach przedstawiony na rysunku 1. Zwróć uwagę na postać instrukcji warunkowej. Wskaż podobieństwa i różnice.

Działanie instrukcji warunkowej jest w większości języków programowania podobne. Sprawdzany jest warunek logiczny (po słowie kluczowym **`if`**) – jeśli jest prawdziwy, wykonywana jest instrukcja (instrukcje) umieszczona (umieszczone) po warunku (*lista_instrukcji1*); jeśli fałszywy – wykonywana jest instrukcja (instrukcje) po słowie **`else`** (*lista_instrukcji2*), a jeśli brak tej części instrukcji – wykonywana jest od razu *kolejna_instrukcja* zapisana po instrukcji warunkowej.

Instrukcja warunkowa	**`if`**(*warunek*)
	  lista_instrukcji1;
	`else`
	  lista_instrukcji2;
	kolejna_instrukcja;
		C++

	`if` *warunek*:
	  lista_instrukcji1
	`else`:
	  lista_instrukcji2
	kolejna_instrukcja
		Python

Uproszczona postać instrukcji warunkowej	**`if`**(*warunek*) *lista_instrukcji1*; *kolejna_instrukcja*;	C++
	`if` *warunek*: *lista_instrukcji1* *kolejna_instrukcja*	Python

Warunek po słowie kluczowym **`if`** może być:

- **prostym warunkiem logicznym**, np.

  `x > 0, a == 9, liczba <= -5, a != 9` C++ i Python

lub

- **złożonym warunkiem logicznym**, np.:

  `x > -5 && x <= 0, x < -5 || x > 0` C++

  `x > -5 and x <= 0, x < -5 or x > 0` Python

Jako *lista_instrukcji1* i *lista_instrukcji2* mogą wystąpić również kolejne instrukcje warunkowe lub **blok instrukcji** (**blok kodu**).

C++ W języku C++ blok instrukcji wyznaczają nawiasy klamrowe `{}`.

Python W języku Python ważne są wcięcia w programie – *listę_instrukcji1* i *listę_instrukcji2* należy przesunąć w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji). Jeśli *lista_instrukcji1* i *lista_instrukcji2* obejmują więcej instrukcji, wszystkie należy tak samo przesunąć. Jest to sposób wyróżnienia **bloku kodu**. Jeśli nie zastosujemy wcięcia, komputer potraktuje *listę_instrukcji1* i *listę_instrukcji2* jako *kolejną_instrukcję* i wykona ją niezależnie od spełnienia warunku.

Język	Operator	Określenie	Przykład warunku	Interpretacja wyrażenia
C++ i Python	==	równy	`a == b`	*a* równe *b*
	!=	różny	`a != b`	*a* różne od *b*
	<	mniejszy	`a < b`	*a* mniejsze od *b*
	>	większy	`a > b`	*a* większe od *b*
	>=	większy lub równy	`a >= b`	*a* większe lub równe *b*
	<=	mniejszy lub równy	`a <= b`	*a* mniejsze lub równe *b*

Tabela 1. *Podstawowe operatory porównania w językach C++ i Python*

Uwaga: Do porównywania wartości używamy operatora porównania ==. Nie stosujemy w tym celu operatora przypisania =. W języku C++ zapisanie warunku za pomocą operatora przypisania, np. `if(x = 0)` nie spowoduje błędu kompilacji, ale program będzie działał niepoprawnie.

Przykład 1. Zapisywanie algorytmu z warunkiem logicznym prostym w językach C++ i Python

Chcemy zrealizować w językach C++ i Python algorytm obliczania wartości bezwzględnej liczby całkowitej na podstawie specyfikacji zadania 1. podanej w przykładzie 3. z tematu C1.

C++

[*] Wartosc_bezw.cpp

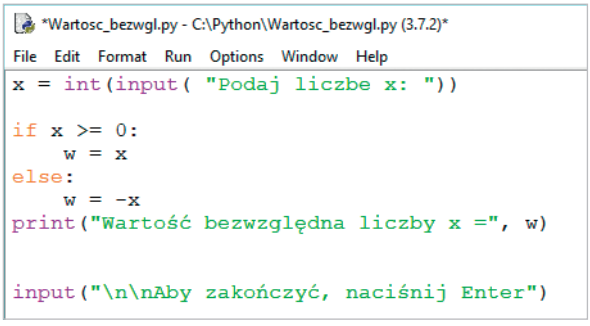
```
#include <iostream>
using namespace std;
int main()
{
    int x, w;
    cout << "Podaj liczbe x: ";
    cin >> x;
    if (x >= 0)
        w = x;
    else
        w = -x;
    cout << "Wartosc bezwzgledna liczby x = " << w;

    return 0;
}
```

C++ Uwagi:

- Programiści używający języka C++ lubią korzystać z konstrukcji języka skracających zapis. Instrukcję: `if (x >= 0) w = x; else w = -x;` można zastąpić instrukcją: `w = x >= 0 ? x : -x;`
- Wartość równą 0 uważa się za logiczny fałsz, a wartość różną od zera – za logiczną prawdę.

Python



Python Uwagi:

- Programiści używający języka Python lubią korzystać z konstrukcji języka skracających zapis. Instrukcję warunkową z powyższego programu można zastąpić instrukcją: `w = x if x >= 0 else -x`
- Wartość równą 0 uważa się za logiczny fałsz, a wartość różną od zera – za logiczną prawdę (tak jak w języku C++).

Ćwiczenie 3. Realizujemy algorytm obliczania wartości bezwzględnej liczby w wybranym języku programowania

1. Przepisz wybrany program podany w przykładzie 1. (zależnie od używanego języka programowania). Sprawdź, czy program jest zgodny ze specyfikacją zadania 1. podaną w przykładzie 3. z tematu C1.
2. Zapisz program w pliku pod nazwą *Wartosc_bezwzgl* i wykonaj program dla kilku różnych wartości *x*.

Uwaga

W języku C++ musimy skompilować program przed uruchomieniem. Nie będziemy o tym przypominać w każdym ćwiczeniu i zadaniu.

Ćwiczenie 4. Realizujemy algorytm z warunkiem logicznym prostym w wybranym języku programowania

1. Napisz specyfikację zadania i program w wybranym języku programowania realizujący algorytm sprawdzania, czy wprowadzona liczba jest dodatnia czy niedodatnia. Wykorzystaj zapis instrukcji warunkowej z rysunku 1.
2. Zapisz program w pliku pod nazwą *Liczby* i wykonaj program dla kilku różnych wartości zmiennych.

2.2. Złożone warunki logiczne

Wszystkie klasy twojej szkoły rywalizują w tzw. lidze klas. Zdobycie największej liczby punktów daje wygraną. Liczba punktów zależy m.in. od frekwencji (np. większa od 94%) i średniej ocen (np. większa od 4,0 lub równa 4,0) w ostatnim półroczu. W jaki sposób napisać program, w którym będziemy mogli obliczać punkty zależnie od spełnienia obydwu warunków?

Jest to przykład sytuacji warunkowej z warunkiem logicznym złożonym. Jeśli frekwencja i średnia ocen spełniają podane warunki, to klasa uzyskuje punkty. Mamy tu do czynienia z logiczną koniunkcją. W wybranym języku programowania zastosujemy zatem instrukcję warunkową z warunkiem złożonym. W instrukcjach warunkowych warunki złożone zapisujemy, stosując odpowiednie operatory logiczne (tabela 2.).

Język	Operator	Określenie	Przykład warunku logicznego	Interpretacja wyrażenia
C++	`||`	alternatywa logiczna (lub)	`a < -5 || a > 5`	*a* mniejsze od –5 lub *a* większe od 5
Python	**`or`**		`a < -5 or a > 5`	
C++	`&&`	koniunkcja logiczna (i)	`a > 0 && a < 10`	*a* większe od 0 i *a* mniejsze od 10
Python	**`and`**		`a > 0 and a < 10`	

Tabela 2. *Operatory logiczne w językach C++ i Python*

Przykład 2. Zapisywanie algorytmu z warunkiem logicznym złożonym w językach C++ i Python

Zadanie: Pewna klasa uczestniczy w lidze klas. Napisz program obliczający, ile wynosi liczba punktów (*pkt*) danej klasy zależnie od uzyskanej frekwencji (*f*) i średniej ocen (*so*) w ostatnim półroczu. Klasa otrzymuje dodatkowe 20 punktów, jeśli frekwencja jest powyżej 94% ($f > 94\%$) i średnia ocen nie jest mniejsza niż 4,0 ($so \geq 4{,}0$). Zdobytą wcześniej liczbę punktów i frekwencję wprowadzaj z klawiatury, a aktualną liczbę punktów wyświetlaj na ekranie.

Dane: liczba całkowita $pkt > 0$ oznaczająca punkty wcześniej zdobyte przez klasę, liczba rzeczywista $f > 0$ oznaczająca procent frekwencji, liczba rzeczywista `so` oznaczająca średnią ocen.

Wyniki: liczba całkowita *pkt* oznaczająca aktualną liczbę punktów danej klasy.

Opis rozwiązania: W rozwiązaniu zastosujemy instrukcję warunkową (w wersji uproszczonej) z warunkiem złożonym zawierającym operator koniunkcji (`&&` lub **`and`**). Zastosujemy wersję uproszczoną, ponieważ niezależnie od spełnienia czy niespełnienia warunku, na ekranie ma być wyświetlana aktualna liczba punktów.

Uwaga: Jeśli podczas pisania programu zwrócimy uwagę na sposób formatowania poszczególnych elementów kodu programu, możemy uniknąć niepotrzebnych błędów. Warto zauważyć, że tekst kodu programu ma różne kolory w zależności od pełnionej roli.

C++ Na przykład: słowa kluczowe i nazwy zmiennych są czarne, operatory, nawiasy – czerwone, napisy – niebieskie.

Python Na przykład: nazwy funkcji `print()`, `input()` są fioletowe, napisy – zielone, słowa kluczowe – pomarańczowe, a nazwy zmiennych – czarne.
Dzięki temu już na etapie pisania programu możemy zauważać błędy, ponieważ nazwa danej funkcji zostanie zaznaczona kolorem fioletowym dopiero wtedy, gdy napiszemy ją w całości poprawnie. Gdy zrobimy błąd, nazwa pozostanie czarna.

C++

[*] Liga.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    int pkt;
    float f, so;

    cout << "Podaj liczbe punktow zdobytych przez klase: ";
    cin >> pkt;
    cout << "Podaj frekwencje klasy: ";
    cin >> f;
    cout << "Podaj srednia ocen klasy: ";
    cin >> so;
    if(f > 94 && so >= 4.0)
        pkt = pkt + 20;
    cout << "Aktualna liczba punktow wynosi: " << pkt;

    return 0;
}
```

Python

Liga.py - C:\Python\Liga.py (3.7.2)

File Edit Format Run Options Window Help

```python
pkt = int(input("Podaj liczbę punktów zdobytych przez klasę: "))
f = float(input("Podaj frekwencję klasy: "))
so = float(input("Podaj średnią ocen klasy: "))

if f > 94 and so >= 4.0:
    pkt = pkt + 20
print("Aktualna liczba punktów wynosi: " , pkt)


input("\n\nAby zakończyć, naciśnij Enter")
```

Ćwiczenie 5. Realizujemy algorytm z warunkiem logicznym złożonym w wybranym języku programowania

1. Utwórz nowy plik źródłowy wybranego języka programowania. Przepisz kod programu podany w przykładzie 2.
2. Zapisz program w pliku pod nazwą *Liga* i wykonaj program dla kilku różnych wartości zmiennych. Wyjaśnij, co się dzieje w poszczególnych wierszach programu. Dlaczego zastosowano instrukcję warunkową w wersji uproszczonej (bez części z **`else`**)?

Ćwiczenie 6. Realizujemy algorytm z warunkiem logicznym złożonym w wybranym języku programowania

1. Klasa uczestniczy w lidze klas. Jeśli klasa w ostatnim semestrze uzyskała frekwencję powyżej 93% lub średnią ocen powyżej 4,2, to w nagrodę pojedzie na wycieczkę. Napisz program sprawdzający, czy klasa zakwalifikuje się na wycieczkę. Frekwencję i średnią ocen wprowadzaj z klawiatury, a komunikat „nagroda” lub „brak nagrody” wyświetlaj na ekranie.
2. Zapisz program w pliku pod nazwą *Wycieczka*.

W językach C++ i Python stosuje się **operatory przypisania**, np. instrukcję przypisania: `pkt = pkt + 20` można zapisać krócej: `pkt += 20` (tabela 3.).

Język	Przykład zastosowania	Odpowiednik	Znaczenie
C++ i Python	`s += a`	`s = s + a`	przypisanie wartości zmiennej `s` poprzedniej wartości tej zmiennej, powiększonej o `a`
	`s -= a`	`s = s - a`	przypisanie wartości zmiennej `s` poprzedniej wartości tej zmiennej, pomniejszonej o `a`
	`s *= a`	`s = s * a`	przypisanie wartości zmiennej `s` poprzedniej wartości tej zmiennej, powiększonej `a` razy
	`s /= a`	`s = s / a`	przypisanie wartości zmiennej `s` poprzedniej wartości tej zmiennej, pomniejszonej `a` razy

Tabela 3. *Operatory przypisania w językach C++ i Python*

3. Sprawdzanie poprawności danych

Chcemy napisać program obliczający pierwiastek kwadratowy danej liczby. Pierwiastki kwadratowe w zbiorze liczb rzeczywistych można jednak obliczyć tylko dla liczb nieujemnych. Jak sprawdzić w programie, czy wprowadzona przez użytkownika liczba jest nieujemna?

Podczas pisania programów ważne jest sprawdzanie poprawności wprowadzanych danych. Dzięki temu jesteśmy zabezpieczeni przed niespodziewanym przerwaniem pracy programu oraz zapewniamy zgodność danych wejściowych ze specyfikacją zadania.

Poprawny program powinien spełniać kilka kryteriów:

- rozwiązywać problem, dla którego został utworzony;
- rozwiązywać problem dla wszystkich danych określonych w specyfikacji i odpowiednio reagować na wprowadzanie niepoprawnych danych;
- rozwiązywać problem w sposób jak najbardziej efektywny, optymalnie wykorzystując zasoby komputera.

Z punktu widzenia wygody obsługi programu wskazane byłoby, aby użytkownik był natychmiast powiadamiany o wprowadzeniu niepoprawnych danych i mógł je od razu poprawić.

Przykład 3. Sprawdzanie poprawności danych

M Moduł

Plik zawierający kod przeznaczony do wykorzystania w innych programach.

Zadanie: Napisz program obliczający pierwiastek kwadratowy *p* z liczby rzeczywistej dodatniej *x*, wprowadzanej z klawiatury. Program ma sprawdzać, czy wprowadzona przez użytkownika liczba jest poprawna (nieujemna) i wyprowadzać na ekran wynik obliczenia lub komunikat o błędnych danych.

Dane: liczba rzeczywista *x*.

Wyniki: pierwiastek kwadratowy *p* z liczby *x* lub komunikat „*x* mniejsze od zera".

Opis rozwiązania: Aby sprawdzić, czy wprowadzona przez użytkownika liczba spełnia określone w zadaniu warunki, zastosujemy instrukcję warunkową. Pierwiastki kwadratowe w zbiorze liczb rzeczywistych można obliczyć tylko dla liczb nieujemnych. W programie sprawdzamy, czy wprowadzona liczba *x* jest ujemna. Jeśli tak, wyprowadzamy komunikat o błędnych danych, w przeciwnym wypadku wykonujemy obliczenia i wyprowadzamy wynik na ekran.
Wykorzystamy standardową funkcję matematyczną `sqrt()`, służącą do obliczania pierwiastka kwadratowego.

C++

[*] Pierwiastek.cpp

```
1  #include <iostream>
2  #include <cmath>          /* sqrt */
3  using namespace std;
4  int main()
5  {
6      float x, p;
7
8      cout << "Podaj x: ";
9      cin >> x;
10     if (x < 0)
11         cout << "x mniejsze od zera";
12     else
13     {
14         p = sqrt(x);
15         cout << "Pierwiastek z " << x << " = " << p;
16     }
17     return 0;
18 }
```

dołączenie biblioteki standardowej `cmath` zawierającej funkcje matematyczne, m.in. funkcję `sqrt()` zwracającą pierwiastek kwadratowy z nieujemnej liczby rzeczywistej

wywołanie funkcji `sqrt()`

Python

Pierwiastek.py - C:\Python\Pierwiastek.py (3.7.2)

File Edit Format Run Options Window Help

```
from math import sqrt                    #sqrt

x = float(input("Podaj x: "))

if x < 0:
    print("x mniejsze od zera")
else:
    p = sqrt(x)
    print( "Pierwiastek z ", x,  " = ", p)

input("\n\nAby zakończyć, naciśnij Enter")
```

importowanie z **modułu** `math` (zawierajacego funkcje matematyczne) funkcji `sqrt()` zwracającej pierwiastek kwadratowy z nieujemnej liczby rzeczywistej

wywołanie funkcji `sqrt()`

Uwaga

W przykładach i ćwiczeniach dotyczących obliczeń nie uwzględniamy przypadków wprowadzenia danych innych niż liczba. Takie przypadki omówimy na informatyce realizowanej w zakresie rozszerzonym.

Ćwiczenie 7. Sprawdzamy poprawność danych liczbowych

1. Utwórz nowy plik źródłowy wybranego języka programowania. Przepisz odpowiedni kod programu podany w przykładzie 3.
2. Zapisz program w pliku pod nazwą *Pierwiastek* i wykonaj dla kilku różnych wartości zmiennych. Wyjaśnij, co się dzieje w poszczególnych wierszach programu. Dlaczego zastosowano instrukcję warunkową w pełnej wersji (z `else`)?

Ćwiczenie 8. Stosujemy instrukcję warunkową do sprawdzania poprawności wprowadzanych danych liczbowych

1. Napisz program obliczający pole trójkąta. W programie uwzględnij sprawdzanie poprawności wprowadzanych danych (podstawy i wysokości trójkąta).
2. Zapisz program w pliku pod nazwą *Trojkat*.

4. Algorytmy z warunkami zagnieżdżonymi

Chcemy napisać program realizujący algorytm wyboru najmniejszej z trzech różnych liczb całkowitych wprowadzanych z klawiatury. W jaki sposób po sprawdzeniu, czy pierwsza liczba jest mniejsza od drugiej, sprawdzać, czy pierwsza jest też mniejsza od trzeciej oraz druga od trzeciej?

Instrukcje warunkowe, podobnie jak funkcja JEŻELI w arkuszu kalkulacyjnym, mogą się **zagnieżdżać**, czyli po warunku oraz po słowie `else` może wystąpić kolejna instrukcja warunkowa.

Ćwiczenie 9. Piszemy specyfikację problemu

Sformułuj zadanie i napisz specyfikację zadania do sytuacji problemowej podanej w nawiasach klamrowych w punkcie 4. tego tematu. Przyjmij następujące nazwy zmiennych *a*, *b*, *c* (dla sprawdzanych liczb), *najmniejsza* (dla zapamiętania najmniejszej liczby).

Przykład 4. Zapisywanie algorytmu z warunkami zagnieżdżonymi w językach C++ i Python

Napiszemy program realizujący algorytm wyboru najmniejszej z trzech różnych liczb całkowitych. W rozwiązaniu zastosujemy zagnieżdżone instrukcje warunkowe.

C++

Najmniejsza_z_trzech.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a, b, c, najmniesza;

    cout << "Podaj trzy liczby: ";
    cin >> a >> b >> c;
    if (a < b)
        if (a < c)
            najmniesza = a;
        else
            najmniesza = c;
    else
        if (b < c)
            najmniesza = b;
        else
            najmniesza = c;
    cout << "Najmniesza liczba wynosi: " << najmniesza << endl;
    return 0;
}
```

C++ Uwagi:

- W przypadku instrukcji zagnieżdżonych zaleca się pisać słowo kluczowe **`else`** pod odpowiadającym mu słowem **`if`** (dla zachowania czytelności).
- Wszystkie instrukcje **`if`** można zastąpić zapisem:
  `najmniejsza = a < b ? (a < c ? a : c) : (b < c ? b : c);`

Python

```python
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
c = int(input("Podaj trzecią liczbę: "))

if a < b:
    if a < c:
        najmniesza = a
    else:
        najmniesza = c
else:
    if b < c:
        najmniesza = b
    else:
        najmniesza = c

print("Najmniesza liczba wynosi:", najmniesza)

input("\n\nAby zakończyć, naciśnij Enter")
```

Ćwiczenie 10. Stosujemy zagnieżdżone instrukcje warunkowe

1. Przepisz program podany w przykładzie 4. (zależnie od używanego języka programowania).
2. Zapisz program w pliku pod nazwą *Najmniejsza_z_trzech*. Zwróć uwagę na zapis zagnieżdżonych instrukcji warunkowych.
3. Uruchom program i przetestuj dla następujących trójek liczb (wartości zmiennych *a*, *b*, *c*): (3; -56; 0), (76; 123; -2), (44; 44; 44), (-70; -40; -40), (56; 68; 88), (100; 0; 2).
4. Wyjaśnij, co dzieje się w poszczególnych wierszach programu, zwracając uwagę na działanie instrukcji warunkowej. W zeszycie przedmiotowym dokończ opis: *Sprawdzamy, czy pierwsza liczba jest mniejsza od drugiej. Jeśli tak, to…*
5. Odpowiedz na pytanie: *Kiedy będzie wykonane przypisanie* `najmniejsza = b`*?*

Ćwiczenie 11. Zapisujemy problem sprzedawcy komputera (z tematu C1) w języku programowania

1. Przypomnij sobie problem sprzedawcy komputerów (przykład 1. z tematu C1).
2. Korzystając ze specyfikacji zadania (przykład 4. z tematu C1) i zaprojektowanego rozwiązania (przykład 5. z tematu C1), napisz program realizujący przedstawione rozwiązanie. Zadbaj o sprawdzanie poprawności wprowadzanych danych (zgodnie ze specyfikacją).
3. Zapisz program w pliku pod nazwą *Cena_komputera*, uruchom i przetestuj dla różnych wartości zmiennych (*w*, *cenak*, *cenap*).

Warto zapamiętać

- W algorytmie z warunkami występują sytuacje warunkowe – wynik lub dalsze działanie algorytmu zależą od spełnienia lub niespełnienia warunku.
- W językach programowania do zapisania algorytmów z warunkami stosujemy instrukcje warunkowe. Działanie tych instrukcji jest podobne w większości języków programowania.
- Warunek w instrukcji warunkowej może być prosty lub złożony. W warunkach złożonych stosujemy operatory alternatywy i koniunkcji logicznej: `||`, `&&` (w języku C++) i **`or`**, **`and`** (w języku Python).
- Program powinien efektywnie rozwiązywać problem dla wszystkich danych określonych w specyfikacji.
- Należy dbać o zgodność danych wejściowych ze specyfikacją i sprawdzać w programie, czy użytkownik wprowadził właściwe dane.
- W programach komputerowych można stosować zagnieżdżone instrukcje warunkowe.

Pytania i polecenia

1. Wyjaśnij, na czym polega sytuacja warunkowa. Podaj przykład sytuacji warunkowej.
2. W jaki sposób zapisuje się w językach programowania algorytmy zawierające sytuacje warunkowe?
3. Omów działanie instrukcji warunkowej w wybranym języku programowania. Porównaj działanie tej instrukcji do działania funkcji JEŻELI w arkuszu kalkulacyjnym.
4. Podaj dwa przykłady zapisania złożonych warunków logicznych.
5. Jakie kryteria powinien spełniać poprawny program?
6. Podaj przykład sytuacji, w której należy sprawdzić poprawność danych.
7. Omów sposób zapisywania zagnieżdżonej instrukcji warunkowej.

Zadania

Uwagi:

- Staraj się dodawać do programów odpowiednie komunikaty dla użytkownika.
- Pisz przejrzyste programy oraz dodawaj w odpowiednich miejscach komentarze.
- Każdy program uruchom i przetestuj dla kilku różnych danych, nawet jeśli w zadaniu nie ma takiego polecenia (w przypadku języka C++ najpierw skompiluj program).

1. Kiedy zostanie wykonana instrukcja `print(x - y)` (zapisana w języku Python), a kiedy instrukcja `cout << x - y` (w języku C++) w programach z podpunktów a i b? Rozważ przypadki: `x > y`, `x = y`, `x < y`. Czym różni się efekt działania instrukcji **`if`** w punktach a i b?

   a.

   C++

```
cin >> x >> y;
if (x > y)
    x = x - y;
cout << x - y;
```

   Python

```
x = int(input("Podaj pierwszą liczbę: "))
y = int(input("Podaj drugą liczbę: "))
if x > y:
    x = x - y
print(x - y)
```

   b.

   C++

```
cin >> x >> y;
if (x > y)
    x = x - y;
else
    cout << x - y;
```

   Python

```
x = int(input("Podaj pierwszą liczbę: "))
y = int(input("Podaj drugą liczbę: "))
if x > y:
    x = x - y
else:
    print(x - y)
```

2. Napisz specyfikację zadania i program realizujący algorytm sprawdzania, czy wprowadzona z klawiatury liczba jest parzysta czy nieparzysta. Zapisz plik pod nazwą *Parzyste_nieparzyste*.
   Wskazówka: Oblicz resztę z dzielenia wprowadzonej liczby przez 2.
3. Napisz specyfikację zadania i program realizujący algorytm sprawdzania, czy liczba jest podzielna przez 3. Zapisz plik pod nazwą *Podzielna_przez_3*.
   Wskazówka: Oblicz resztę z dzielenia wprowadzonej liczby przez 3.
4. Napisz specyfikację zadania i program realizujący algorytm sprawdzania, która z dwóch liczb rzeczywistych jest większa. Uwzględniaj przypadek, gdy liczby są równe. Zapisz plik pod nazwą *Wieksza_z_dwoch*.
5. Napisz specyfikację zadania: *Wprowadź dowolną liczbę rzeczywistą. Gdy liczba jest dodatnia, obliczaj pole koła o promieniu równym tej liczbie. Dla liczby ujemnej podawaj jej wartość bezwzględną, a dla równej zero – wyprowadź napis „równa zero”*. Napisz program na podstawie zapisanej specyfikacji. Zapisz plik pod nazwą *Zadanie5*.
6. Napisz specyfikację zadania i program obliczający prędkość samochodu *v*, gdy dane są droga *s* i czas *t*. Jeśli prędkość samochodu przekracza 90 km/h, wyświetlaj komunikat „za szybko”. Jeśli prędkość będzie równa 90/km, wyświetlaj komunikat „w sam raz”, a jeśli mniejsza, wyświetlaj komunikat „za wolno”. Program napisz na podstawie przygotowanej wcześniej specyfikacji. Zapisz plik pod nazwą *Predkosc*.
7. Klasa ma być podzielona na dwie grupy, w zależności od wyniku testu kompetencji językowych z wybranego języka obcego lub oceny na świadectwie ukończenia szkoły podstawowej. Jeśli dany uczeń uzyskał z testu wynik powyżej 90% punktów lub jeśli na świadectwie ukończenia szkoły podstawowej miał z danego języka ocenę nie niższą niż 5, to kwalifikuje się do grupy zaawansowanej. W przeciwnym wypadku kwalifikuje się do grupy podstawowej. Napisz program sprawdzający, do jakiej grupy zakwalifikuje się dany uczeń. Liczbę punktów uzyskanych z testu i ocenę ze świadectwa wprowadzaj z klawiatury, a komunikat: „grupa zaawansowana” lub „grupa podstawowa” wyświetlaj na ekranie. Zapisz plik pod nazwą *Grupy*.

8. Napisz specyfikację zadania i program realizujący algorytm obliczania objętości walca o wysokości *h* i promieniu podstawy *r*, gdzie *h* i *r* są dodatnimi liczbami rzeczywistymi. Zadbaj o poprawność wprowadzanych danych, zgodnie z zapisaną wcześniej specyfikacją. Dla liczb niedodatnich powinien być wyprowadzany napis błędne dane. Zapisz plik pod nazwą *Objetosc_walca*.
9. Otwórz plik *Przepis* zapisany w ćwiczeniu 9. z tematu C2 lub 10. z tematu C3. Dodaj do programu sprawdzanie poprawności wprowadzanych danych, zgodnie ze specyfikacją. Zapisz plik pod tą samą nazwą.

Dla zainteresowanych

10. Napisz specyfikację zadania i program sprawdzający, czy z odcinków o długościach wprowadzonych przez użytkownika z klawiatury można zbudować trójkąt (warunek istnienia trójkąta przypomnij sobie z lekcji matematyki). Program ma sprawdzać dziesięć trójek liczb i zależnie od rezultatu wyprowadzać komunikaty: „można zbudować trójkąt” lub „nie można zbudować trójkąta”. Dodaj do programu sprawdzanie poprawności wprowadzanych danych. Zapisz plik pod nazwą *Trojkat*.
11. Napisz specyfikację zadania i program sprawdzający, czy wprowadzone trzy liczby naturalne są bokami trójkąta prostokątnego. Zależnie od rezultatu program ma wyprowadzać komunikaty „tak” lub „nie”. Zapisz plik pod nazwą *Trojkat_prostokat*.
12. Otwórz plik *Najmniejsza_z_trzech* zapisany w ćwiczeniu 10. Zmodyfikuj program tak, aby był uwzględniony przypadek, gdy liczby są równe. Zapisz plik pod tą samą nazwą.
13. Zadanie dla dwóch osób: Każda z osób projektuje zadanie według własnego pomysłu (można wzorować się na zadaniach z tego tematu) – formułuje treść zadania, pisze specyfikację i przekazuje zadanie drugiej osobie, aby napisała program. Potem program wraca do autora zadania, który testuje go dla różnych danych.