Instrukcja warunkowa if

*Instrukcje warunkowe*

Nasze życie składa się z nieustannych wyborów. Jeśli jest pora obiadu i jesteśmy bardzo głodni, to zjemy obfity posiłek, w przeciwnym wypadku zadowolimy się np. tylko kanapką. Chcąc przejść przez przejście dla pieszych z sygnalizacją bacznie zwracamy uwagę na światło. Jeśli jest zielone, to przechodzimy przez ulicę, jeśli czerwone, to stoimy na chodniku. W sklepie z obuwiem wybieramy tylko te, które nam odpowiadają pod każdym względem. Nasze poszukiwania odpowiednich butów o tyle były utrudnione, że w sklepie z reguły jest więcej butów niż tylko dwie pary. W każdym z podanych sytuacji nasze zachowanie było zdeterminowane pewnymi warunkami, np. czy światło czerwone, czy zielone?

Sprawdzenie warunków i przejście do wykonywania odpowiednich operacji umożliwiają tzw. instrukcje warunkowe if i switch.

**Instrukcje proste**

**Definicja**

Instrukcja prosta jest to instrukcja, które nie zawiera żadnych innych instrukcji.

**Instrukcja pusta**

Instrukcja pusta nie powoduje wykonania żadnych czynności ze strony programu. Instrukcję taką zapisuje się w postaci średnika nie poprzedzonego żadnym słowem kluczowym, ani wyrażeniem. Dobrym zwyczajem jest umieszczanie jej w osobnym wierszu programu.

Przykład 2.1.

Wywołanie instrukcji pustej

int main()

{

; // to jest instrukcja pusta

}

Mając wybór

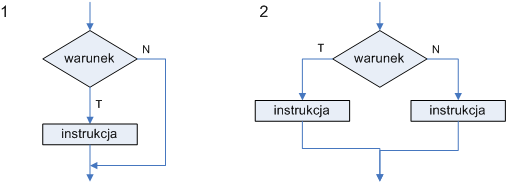
Ogólna postać instrukcji warunkowej **if** (jeżeli) może przyjmować dwie formy:

1. if (warunek)  
    instrukcja

lub

1. if (warunek)  
    instrukcja1  
   else  
    instrukcja2

Powyższym instrukcjom odpowiadają następujące fragmenty schematów blokowych.



*Warunek*, to dowolne wyrażenie składające się przede wszystkim z operatorów relacji, które przyjmuje dwa stany: prawdę (true) albo fałsz (false), inaczej dwie wartości: 1 (true) albo 0 (false). Ponadto warunek może być prosty lub złożony - połączenie wyrażeń warunkowych operatorami logicznymi && albo ||.

**Przykład 2.6.**  
*Przykłady wyrażeń warunkowych*

(n == 3)  
(a+2\*b <= c)  
((a>10) && (a<=0)) // poprawny również jest zapis (a>10 && a<=0)

Na schemacie blokowym zamieszczonym wyżej można również zauważyć *instrukcję*. Jest to dowolna jedna instrukcja prosta lub złożona.

**Definicja**  
**Instrukcją złożoną** nazywamy blok co najmniej dwóch instrukcji ujętych w nawiasy klamrowe.

**Przykład 2.7.** *Przykład wywołania instrukcji warunkowej z blokiem instrukcji.*

if (n>0)  
 cout << "Liczba większa od zera" << endl;  
else  
{  
 n = 4;  
 cout << "Teraz zmienna n przyjmuje wartosc rowna 4";  
 cout << endl;  
}

Znając już co ważniejsze sformułowania związane z instrukcją warunkową można opisać prezentowane schematy blokowe.

**Pierwsza forma (1)**: jeśli *warunek* jest prawdziwy (przyjmuje wartość jeden), to wykonywana jest *instrukcja*. Program następnie przechodzi do kolejnej instrukcji. W przypadku, gdy *warunek* okaże się fałszywy (przyjmuje wartość zero), to instrukcja zostanie pominięta przez program.

**Druga forma (2)**: jeśli *warunek* jest prawdziwy (przyjmuje wartość 1), to zostanie wykonana *instrukcja1*, w przeciwnym wypadku, tj. *warunek* będzie fałszywy (przyjmuje wartość 0) program wykona *instrukcja2*.

**Ćwiczenie 2.3**

*Wskaż niepoprawne wyrażenia warunkowe.*

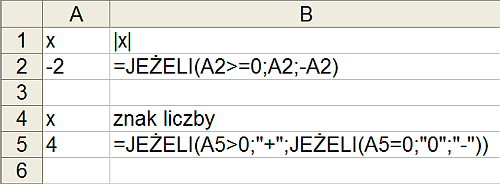
a) c >= 0  
b) a = b  
c) (b1 > 0) || (b1 < 0)  
d) a != 0  
e) (a = 0) && (a > 0)

**Uwaga**  
Instrukcją może być również kolejna instrukcja **if**. Taką sytuację określamy jako zagnieżdżenie instrukcji.

**Przykład 2.8.**  
*Dwa fragmenty programów z zagnieżdżeniem instrukcji jeżeli, które sprawdzają, jaki znak ma liczba k.*

1. if (k > 0)  
    cout << "Liczba dodatnia";  
   else  
    if (k == 0)  
    cout << "Liczba równa zero";  
    else  
    cout << "Liczba ujemna";
2. if (k >= 0)  
    if (k == 0)  
    cout << "Liczba równa zero";  
    else  
    cout << "Liczba dodatnia";  
   else  
    cout << "Liczba ujemna";

**Uwaga**  
Instrukcję **if** można również znaleźć w Arkuszach Kalkulacyjnych w popularnych programach Microsoft Excel i OpenOffice Calc.



**Ćwiczenia**

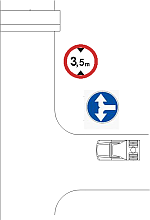
Ćwiczenie 2.4.

Uzupełnij brakujące części programu (brakujące fragmenty oznaczono symbolami [...]). Program powinien sprawdzać, czy podana liczba całkowita jest liczbą ujemną. W przypadku, gdy wprowadzona liczba jest liczbą ujemną, to program powinien wyświetlać napis "liczba ujemna", w przeciwnym razie "liczba nieujemna".  
Nie zapomnij program skompilować i wypróbować.

1. // Program sprawdza, czy podana liczba calkowita jest liczba ujemna,
2. // czy liczba nieujemna
3. // liczba\_ujemna.cpp
5. #include <iostream>
7. **using** **namespace** std;
9. **int** main()
10. {
11. **int** liczba;
13. cout << "Podaj liczbe calkowita";
14. cin >> liczba;
16. **if** ([...])
17. cout << "Liczba ujemna" << endl;
18. **else**
19. cout << "Liczba nieujemna" << endl;
21. **return** 0;
22. }

**Ćwiczenie 2.5.**

W ruchu ulicznym na kierowcę może czekać wiele niespodzianek, na które musi być przygotowanym. Może się na przykład zdarzyć taka sytuacja, że wysokość samochodu może powodować, że nie można przejechać pod mostem lub wjechać do tunelu, bo może się okazać, że ów most lub tunel są nieodpowiednich rozmiarów w stosunku do potrzeb samochodu. Z reguły przed tego typu przeszkodą stoi znak informujący jak wysoki samochód może wjechać.



Na powyższym rysunku widzimy, że samochód po skręcie w prawo kieruje się do przeszkody drogowej. By móc kontynuować jazdę kierowca musi wiedzieć, że jego pojazd spełnia wymogi co do wysokości pojazdu. W tym przypadku jest to 3.5 metra.  
  
Twoim zadaniem jest uzupełnienie brakujących fragmentów programu. Nie zapomnij program skompilować i wypróbować.

[view plaincopy to clipboardprint?](https://youngcoder.eu/index.php/node/show/8500)

// Program sprawdza, czy samochod przejedzie pod mostem

// most.cpp

*#include <iostream>*

*#define MOST [...]   // ograniczenie na wysokosc samochodu*

*using namespace std;*

*int main()*

*{*

*int wys;        // wysokosc samochodu*

*cout << "Podaj wysokosc samochodu";*

*cin >> wys;*

*[...]*

*cout << "Samochod nie przejedzie" << endl;*

*[...]*

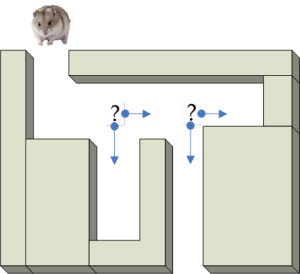
*cout << "Samochod" << endl;*

*return 0;*

*}*

*Ćwiczenie 2.6.*

Biedny chomik, rzec by można. Jego zadaniem jest przejść labirynt. Na szczęście nie taki straszny. Na jego drodze znajdują się dwie krzyżówki, które mogą spowodować, że chomik może trafić w ślepą uliczkę.



Z rysunku nie trudno odgadnąć, że prawidłową marszrutą chomika jest droga określona kierunkami: prosto, dół.  
Twoim zadaniem jest napisanie programu, który umożliwi chomikowi przejście przez labirynt. Poniżej, by ułatwić Tobie zadanie został umieszczony fragment kodu. Twoim zadaniem jest dopisanie brakujących fragmentów. Nie zapomnij program skompilować i wypróbować.

*// Program wyprowadza chomika z labiryntu*

*// chomlab.cpp*

*#include <iostream>*

*using namespace std;*

*int main()*

*{*

*string kierunek;*

*// mozliwe kierunki: dol, prosto*

*cout << "Wyprowadz chomika z labiryntu." << endl;*

*cout << "Mozliwe kierunki: prosto, dol" << endl << endl;*

*// pierwsza krzyzowka*

*cout << "Podaj kierunek: ";*

*cin >> kierunek;*

*if ([...])*

*cout << "Nie tedy droga" << endl;*

*if (kierunek == "prosto")*

*{*

*cout << "Chomik jest coraz blizej sera" << endl;*

*// druga krzyzowka*

*[...]*

*[...]*

*cout << "Gratulacje! Pomogles wyjsc chomikowi z labiryntu." << endl;*

*else*

*cout << "Chomik zabladzil" << endl;*

*}*

*return 0;*

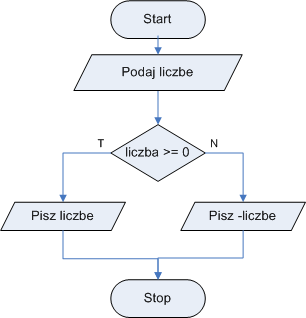
*}*

**Przykłady zastosowań**

**Zastosowanie instrukcji warunkowej if można znaleźć w poniższych przykładach.**

**Przykład 2.9.**  
*Program znajduje wartość bezwzględną z liczby całkowitej.*

Schemat blokowy algorytmu obliczającego wartość bezwzględną z podanej liczby może wyglądać następująco:



Przykładowy kod źródłowy programu:

// Wartosc bezwzgledna z liczby calkowitej

// wartosc\_bezwzgledna.cpp

#include <iostream>

using namespace std;

int main()

{

    int liczba;

    cout << "Podaj liczbe całkowitą: " << endl;

    cin >> liczba;

    if (liczba >= 0)

        cout << liczba;

    else

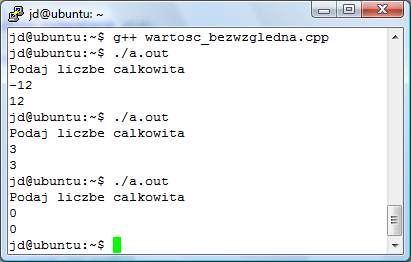
        cout << -liczba;

    cout << endl;

    return 0;

}

Efekt wywołania programu wartosc\_bezwzgledna.cpp:



Przykład 2.10.  
Wersja programu znajdującego wartość bezwzględną z liczby całkowitej bez użycia instrukcji warunkowej, a jedynie operatora warunkowego.

// Wartosc bezwzgledna z liczby calkowitej

// Wersja 2

#include <iostream>

using namespace std;

int main()

{

    int liczba, wynik;

    cout << "Podaj liczbe calkowita: " << endl;

    cin >> liczba;

    wynik = liczba>=0?liczba:-liczba;

    cout << wynik << endl;

    return 0;

}

Wersja jeszcze krótsza:

// Wartosc bezwzgledna z liczby calkowitej

// Wersja 3

#include <iostream>

using namespace std;

int main()

{

    int liczba;

    cout << "Podaj liczbe calkowita: " << endl;

    cin >> liczba;

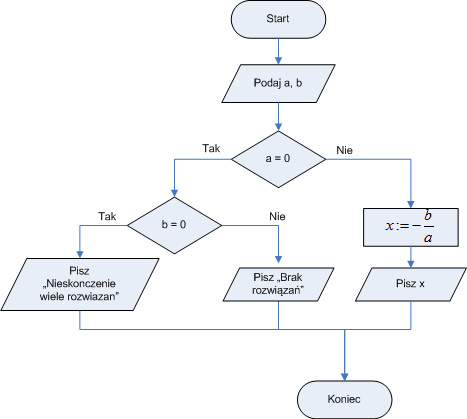
    cout << (liczba>=0?liczba:-liczba) << endl;

    return 0;

}

Przykład 2.11.  
Program rozwiązujący równanie liniowe ax+b=c.

Schemat blokowy algorytmu znajdujący rozwiązanie równania ax+b=c:



Przykładowy kod programu:

[view plaincopy to clipboardprint?](https://youngcoder.eu/index.php/node/show/8501)

// Program rozwiazuje rownanie liniowe postaci ax + b = c

// rownania\_liniowe.cpp

#include <iostream>

using namespace std;

int main()

{

    double a, b, c;

    double x;

    cout << "Podaj wspolczyniki rowniania: " << endl;

    cin >> a >> b >> c;

    if (a == 0.0)

        if (b == c)

            cout << "Nieskonczenie wiele rozwiazan";

        else

            cout << "Brak rozwiazania";

    else

    {

        x = (c-b)/a;

        cout << "Rozwiazaniem jest x=" << x;

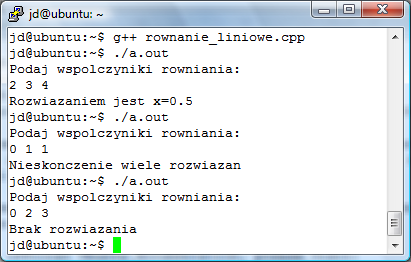
    }

    cout << endl;

    return 0;

}

Efekt wykonania programu rownanie\_liniowe.cpp:



**Zadania**

## Zadanie 1

## Wartość bezwzględna

Wartość bezwzględna (moduł liczby) – liczba rzeczywista „bez znaku” lub bardziej obrazowo odległość liczby od zera. Na przykład: Wartość bezwzględna z liczby -7 jest 7, a z liczby 8 jest 8.

Definicja wartości bezwzględnej:

**Uwaga:** W programie nie wolno używać funkcji matematycznej służącej do obliczenia wartości bezwzględnej.

## Wejście

W pojedynczym wierszu podana jest liczba rzeczywista n ϵ <-10000;10000>.

## Wyjście

W pojedynczym wierszu standardowego wyjścia wypisana jest liczba rzeczywista z dokładnością do drugiego miejsca po przecinku, będąca wynikiem rozwiązania wyrażenia |n|.

## Przykład 1

**Wejście:**

-4

**Wyjście:**

4.00

## Przykład 2

**Wejście:**

2.12

**Wyjście:**

2.12

Zadanie 2

Liczba parzysta

W Bajtocji pewien uczony Sigma odkrył dziwną właściwość liczby całkowitej. Otóż zauważył, że pewne liczby w wyniku dzielenia przez 2 dają resztę 0, a pewne liczby w wyniku wykonywania tego samego działania zwracają resztę 1. Jako, że nikt inny przed nimi tego nie odkrył, to owe liczby, które dzieliły się przez 2 nazwał liczbami parzystymi, a liczby, które nie dzieliły się przez 2 (dawały resztę 1), nazwał liczbami nieparzystymi. Sigma był z siebie dumny, ale jak każdy wielki uczony nie przepadał za żmudnymi rachunkami. Tak więc, zwrócił się do Ciebie o pomoc.

Twoim zadaniem jest napisać program, który określi, czy podana liczba całkowita jest to liczba parzysta, czy wręcz przeciwnie, liczba nieparzysta.

Wejście

W pojedynczym wierszu znajduje się liczba całkowita n, taka, że n ∈ <− 1000;1000>.

Wyjście

W przypadku liczby parzystej na standardowym wyjściu zostanie wypisana litera P, w przypadku określenia, że wprowadzona liczba, to liczba nieparzysta, na wyjściu zostanie wypisania litera N.

Przykład 1

**Wejście:**

12

**Wyjście:**

P

Przykład 2

**Wejście:**

-3

**Wyjście:**

N

Zadanie 3

Przeprowadzka

Rodzice Bajtka postanowili, że Bajtek w wieku 14 lat nie będzie już zajmował pokoju wspólnie ze swoim młodszym bratem Komodorkiem. Tata Bajtka zaproponował, co by zajął pokój na poddaszu. Niestety, zajęcie tego pokoju będzie wiązało się z przeniesieniem rzeczy Bajtka i zakupieniem części mebli do owego pokoju. Okazało się jednak, że blat okrągłego stołu może nie przejść przez drzwi.

Twoim zadaniem jest napisać program, który na podstawie podanych wymiarów drzwi i średnicy blatu okrągłego stołu określi, czy będzie można wnieść ów stół przez te drzwi.

Wejście

W pojedynczym wierszu znajdują się trzy liczby rzeczywiste dodatnie *a*, *b* i *s*, gdzie *a*, *b* ϵ (0;2], *s* ϵ (0;4].

Wyjście

W pojedynczym wierszu znajduje się napis **TAK**, gdy blat stołu można przenieść przez drzwi, **NIE**, gdy blat stołu nie da się przenieść przez drzwi.

Przykład 1

**Wejście:**

1 2 2.3

**Wyjście:**

NIE

Przykład 2

**Wejście:**

1.12 1.9 0.14

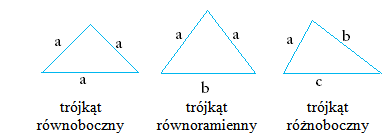
**Wyjście:**

TAK

Zadanie 4

Podział trójkątów

Oj biedny jest nasz Bajtek. Zamiast grać w piłkę, to musi rozwiązywać zadanie z matematyki. Mimo, że zadanie domowe nie jest trudne, to jednak mu się je wykonać, a na dworze ładna pogoda. Dziś na lekcji matematyki Bajtek nauczył się rozpoznawać trójkąty. Poznał, co to jest trójkąt równoboczny, równoramienny i różnoboczny.



Twoim zadaniem jest napisać program, który na podstawie trzech liczb naturalnych będącymi bokami trójkąta określi, czy jest to trójkąt równoboczny, trójkąt równoramienny, trójkąt różnoboczny. Zakładamy, że z podanych wartości można zbudować trójkąt.

Wejście

W pojedynczym wierszu znajdują się trzy liczby naturalne *a*, *b*, *c* z przedziału (0;231-1).

Wyjście

W pojedynczym wierszu znajduje się napis **RWB**, **RWR** lub **RZB**. Skróty te oznaczają:

RWB trójkąt równoboczny  
RWR trójkąt równoramienny  
RZB trójkąt różnoboczny

Przykład 1

**Wejście:**

2 2 2

**Wyjście:**

RWB

Przykład 2

**Wejście:**

3 3 5

**Wyjście:**

RWR

Przykład 3

**Wejście:**

3 4 5

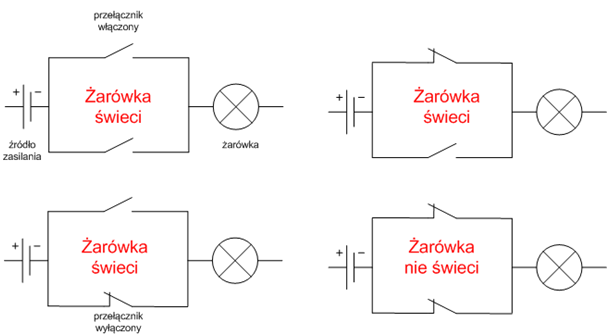
**Wyjście:**

RZB

**Zadanie 5**

**Żarówka**

**Tata Bajtka kupił mu zestaw małego elektryka. Chłopiec w instrukcji dodanej do tego zestawu zobaczył schematy obwodu elektrycznego. Postanowił sprawdzić, czy faktycznie żarówka będzie się świecić lub nie, jeśli połączymy elementy elektryczne zgodnie z poniższymi schematami.**

****

**Twoim zadaniem jest napisać program, który na podstawie podanych dwóch liczb całkowitych *p* i *q* sprawdzi, czy zachodzi między nimi alternatywa.**

**Wejście**

**W pojedynczym wierszu znajdują się dwie liczby naturalne *p* i *q*, gdzie *p*, *q* ϵ {0, 1}. Cyfra 0 oznacza, że przełącznik jest wyłączony, a 1 oznacz, że przełącznik jest włączony.**

**Wyjście**

**W pojedynczym wierszu znajduje się liczba całkowita *s* ϵ {0, 1} , gdzie 0 oznacza, że żarówka się nie świeci, a 1 natomiast, że żarówka się świeci.**

**Przykład 1**

**Wejście:**

**1 0**

**Wyjście:**

**1**

**Przykład 2**

**Wejście:**

**0 0**

**Wyjście:**

**0**

**Zadanie 6**

**Analizator płci**

**Pewien zwariowany naukowiec chce stworzyć robota, który będzie miał rozbudowany ośrodek rozpoznawania mowy. Jednym z elementów oprogramowania tego robota ma być program odpowiedzialny za rozpoznawanie płci na podstawie imienia podanego przez rozmówcę. Naukowiec jest tak pochłonięty pracą, że to zadanie zlecił Tobie.**

**Twoim zadaniem jest napisać program, który na podstawie podanego imienia poda płeć rozmówcy.**

**Wejście**

**W pojedynczym wierszu znajduje się niezerowy ciąg liter (bez polskich znaków), ale nie większy niż 50 znaków.**

**Wyjście**

**W przypadku ciągu liter kończącego się na literę**a**, program wypisuje tekst**dziewczyna!**, w innym przypadku tekstem wyświetlanym będzie**chlopak?**.**

**Przykład 1**

Wejście:

**Bartosz**

Wyjście:

**chlopak?**

**Przykład 2**

Wejście:

**Monika**

Wyjście:

**dziewczyna!**