KURS C++

Instrukcje warunkowe

Przygotował: Kamil Feliszewski

Spis treści

1 Instrukcje warunkowe **2**
1.1 Instrukcje warunkowe if 2
1.2 Praktyczne zastosowanie if 3
1.3 Instrukcja warunkowa switch 4

2 Praktyczne zastosowanie instrukcji warunkowych **4**

3 Estetyka i klamrowanie w instrukcjach warunkowych **5**

4 Zaawansowane instrukcje warunkowe **6**

1 Instrukcje warunkowe

> "Programowanie prowadzi do myśli, myśli prowadzą do uczuć, uczucia prowadzą do działań, działania prowadzą do wyników."

T. Harv Eker, Bogaty albo biedny. Po prostu różni mentalnie

1.1 Instrukcje warunkowe if

Dotychczasowe programy potrafiły tylko wykonywać instrukcje, które były umieszczone w funkcji "main". Obecnie nauczymy nasze programy podejmować decyzje. Do tego celu służą instrukcje warunkowe. Sprawdzają one czy są spełnione określone warunki, jeśli są to wykonują jakieś działania. Jeśli jednak warunki nie są spełnione program może wykonać inne czynności.
Instrukcja warunkowa if zbudowana jest w następujący sposób:

```
if(warunek) {
  instrukcja;
}
```

W powyższym przykładzie warunek może być to zmienna typu bool lub jakiekolwiek inne wyrażenie, którego wynikiem jest wartość logiczna. Jeżeli wartość tego wyrażenia jest różna od 0 (zero), instrukcja się wykona. W przeciwnym razie program pominie pierwszą instrukcję po warunku i przejdzie do następnej instrukcji bądź do następnej linii programu.
Złożona instrukcja warunkowa może wyglądać następująco:

```
if(warunek) {
  instrukcja1;
} else {
  instrukcja2;
}
```

If oznacza jeśli, jeżeli. Else oznacza w przeciwnym razie, w przeciwnym wypadku. Gdy warunek będzie spełniony wykona się instrukcja1, w przeciwnym wypadku wykona się instrukcja2.
Przykład:

```
#include<iostream>
using namespace std;
int wiek;
int main () {
  cout<<"Podaj ile masz lat:<<endl;
  cin>>wiek;
  if(wiek<18)
    cout<<"Jestes niepelnoletni";
  else
    cout<<"Jestes pelnoletni";
  getchar();
  return 0;
}
```

Istnieje możliwość wypisania więcej niż jednej instrukcji. Jeśli po spełnieniu lub po niespełnieniu warunku ma się wykonać więcej niż jedna instrukcja należy taki ciąg instrukcji zamknąć w klamrach {}.
Przykład:

```
if(warunek) {
  instrukcja1;
  instrukcja2;
  instrukcja3;
} else
  instrukcja4;
```

Kilka instrukcji zamkniętych w nawiasy klamrowe traktowane będzie przez kompilator jako jedna instrukcja.

1.2 Praktyczne zastosowanie if

Przykład:

```
#include<iostream>
using namespace std;
int main() {
  int a, b;
  a = 5;
  b = 3;
  if(a>b)
    cout<<"a jest wieksze od b";
  else
    cout<<"b jest wieksze od a";
    getchar();
    return 0;
}
```

Przykład złożonej instrukcji warunkowej:

```
#include<iostream>
using namespace std;
int main() {
        int a, b;
        a = 5;
        b = 3;
        if(a>b)
          cout<<"a jest wieksze od b";
        else
          if(a<b)
            cout<<"a jest mniejsze od b";
          else
            cout<<"a jest równe b";
        getch();
        return 0
}
```

1.3 Instrukcja warunkowa switch

Instrukcja warunkowa switch ma za zadanie przełączyć (przekierować) na odpowiedni wynik danej zmiennej.
`case` określa położenie danej zmiennej i jej wartość
`break` ma za zadanie zatrzymać wykonywanego `case`.
Przykład:

```
int a = 50;
switch(a) {
  case 5:
    //instrukcja1;
    break;
  case 50:
    //instrukcja2;
    break;
  default:
    //instrukcja3;
    break;
}
```

`default` ma za zadanie wykonać jakąś operacje po zakończeniu sprawdzenia "`case`". Jeśli żadna opcja z `case` nie jest spełniona wykona się wtedy `default`.

2 Praktyczne zastosowanie instrukcji warunkowych

Jeśli będziemy chcieli dokonać wyboru więcej niż dwóch możliwości i napisać menu wyboru nie musimy stosować jedynie instrukcji warunkowej `if`. Możemy zastosować instrukcję `switch` lub ich połączenie.

```
switch (warunek) {
  case 1:
    //instrukcja1;
    break;
  case 2:
    //instrukcja2;
    break;
  case 3:
    //instrukcja3;
    break;
  default:
    //instrukcja4;
}
```

Gdy zmienna warunek będzie miała wartość 1 wykonają się wszystkie instrukcje od ":" do najbliższej instrukcji "`break`". Gdy zmienna warunek będzie miała wartość 2 wykona się instrukcja2. Podobnie w przypadku gdy wartość zmiennej warunek będzie wynosić 3, wtedy wykona się instrukcja3. Jeśli wartość zmiennej warunek nie będzie równa żadnej z wymienionych wartości wykonają się instrukcje umieszczone po słowie kluczowym "`default`".

UWAGA:
Jeżeli nie wpiszemy słowa kluczowego `break` program będzie wykonywał wszystkie instrukcje po kolei dopóki nie napotka instrukcji `break` lub nie skończy się dana instrukcja warunkowa. Sytuacja taka bez słowa `break` jest w programowaniu niepożądana.

Przykład:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main() {
  float x,y;
  int odp;
  cout<<"Kalkulator\nPodaj pierwsza liczbe: ";
  cin>>x;
  cout<<"\nPodaj druga liczbe: ";
  cin>>y;
  cout<<"\n\nWybierz dzialanie:\n";
  cout<<"\n\n1 - dodawanie\n2 - odejowanie\n3 - mnozenie\n4 - dzielenie";
  cin>>odp;
  switch (odp) {
    case 1:
      cout<<x+y;
      break;
    case 2:
      cout<<x-y;
      break;
    case 3:
      cout<<x*y;
      break;
    case 4:
      if (y != 0)
        cout<<x/y;
      else
        cout<<"Nigdy nie dziel przez zero!";
        break;
     defautlt:
       cout<<"Bledny wybor. Wybierz pomiedzy 1 - 4.";
  }
  getch();
  return 0;
}
```

3 Estetyka i klamrowanie w instrukcjach warunkowych

Styl wcięć pokazuje jeden ze stylów "wcinania" instrukcji warunkowej if, jaki jest najlepszy styl wyrównywania nawiasów klamrowych. Choć dostępne są tuziny ich odmian, najczęściej stosowane są trzy z nich:

- umieszczenie otwierającego nawiasu klamrowego po warunku i wyrównanie nawiasu klamrowego zamykającego blok zawierający początek instrukcji warunkowej if:

```
if (warunek) {
  //instrukcje
}
```

- wyrównanie nawiasów klamrowych z instrukcją if i wcięcie jedynie bloku instrukcji:

```
if (warunek)
{
  //instrukcje
}
```

- wcięcie zarówno instrukcji, jak i nawiasów klamrowych:

```
if (warunek)
  {
    //instrukcje
  }
```

4 Zaawansowane instrukcje warunkowe

Warto zauważyć, że w klauzuli if lub else może być zastosowana dowolna instrukcja, nawet inna instrukcja if. Z tego powodu możemy natrafić na złożone instrukcje if, przyjmujące postać:

```
if (warunek1) {
  if (warunek2) {
    //instrukcja1;
  } else {
    if (warunek3) {
      //instrukcja2;
    } else {
      //instrukcja3;
    }
  }
} else {
  //instrukcja4;
}
```

Ta rozbudowana instrukcja warunkowa if działa następująco: jeżeli warunek1 ma wartość TRUE i warunek2 ma wartość TRUE, wykonaj instrukcja1. Jeśli warunek1 ma wartość TRUE, lecz warunek2 ma wartość FALSE, wtedy jeśli warunek3 ma wartość TRUE, wykonaj instrukcja2. Jeśli warunek1 ma wartość TRUE, lecz warunek2 i warunek3 mają wartość FALSE, wtedy wykonaj instrukcja3. Jeśli warunek1 ma wartość FALSE, wykonaj instrukcja4. Jak łatwo zauważyć, tak złożone instrukcje warunkowe if mogą wprawiać w zakłopotanie.
Przykład:

```
#include<iostream>
using namespace std;
int main() {
  inf firstNumber, secondNumber;
  cout<<"Wpisz 2 liczby.\nPierwsza: ";
  cin>>firstNumber;
  cout<<"\nDruga: ";
```

```
  cin>>secondNumber;
  cout<<"\n\n";
  if (firstNumber >= secondNumber) {
    if (firstNumber % secondNumber == 0) {
      if (fistNumber == secondNumber) {
        cout<<"Liczby sa takie same";
      } else {
        cout<<"Dziela sie bez reszty";
      }
    } else {
      cout<<"Nie dziela sie bez reszty";
    }
  } else {
    cout<<"Druga liczba jest wieksza";
  }
  return 0;
}
```

Program prosi o wpisanie dwóch liczb, jednej po drugiej. Następnie są one porównywane. Pierwsza instrukcja warunkowa if, sprawdza czy pierwsza liczba jest większa lub równa drugiej liczbie. Jeśli nie jest, wykonywana jest klauzula else. Jeśli pierwszej instrukcji warunkowej warunek jest spełniony (TRUE), wykonywany jest blok kodu. W tym przypadku sprawdzamy, czy reszta z dzielenia pierwszej liczby przez drugą liczbę wynosi 0 (zero), tj. czy obie liczby są podzielne przez siebie. Jeśli tak, liczby te mogą być takie same lub mogą być tylko podzielne przez siebie. Instrukcja warunkowa if sprawdza, czy te liczby są równe i w obu przypadkach wyświetli odpowiedni komunikat.

Choć dozwolone jest pomijanie nawiasów klamrowych w instrukcji warunkowej if zawierających tylko pojedynczą instrukcję oraz dozwolone jest zagnieżdżanie if:

```
if (x > y)
  if (x < z)
    x=y;
```

może to powodować zbyt dużo problemów ze zrozumieniem struktury kody w przypadku, gdy piszemy duże zagnieżdżone instrukcje.

PAMIĘTAJ

Białe spacje i wcięcia są ułatwieniem dla programisty, lecz nie stanowią żadnej różnicy dla kompilatora.

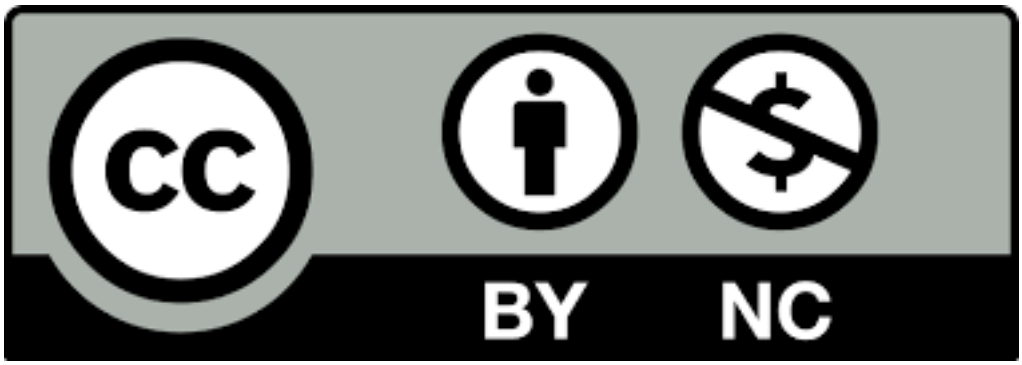
Łatwo jest się pomylić i błędnie wstawić instrukcję klauzulę else do niewłaściwej instrukcji warunkowej if:

```
#include<iostream>
using namespace std;
int main () {
  int x;
  cout<<"Wpisz liczbe mniejsza od 10 lub wieksza od 100";
  cin>>x;
  if (x >= 10)
    if ( x > 100)
      cout<<"Liczba jest wieksza od 100";
    else
```

```
      cout<<"Mniejsza niz 10";
  return 0;
}
```

W powyższym przykładzie programista miał zamiar poprosić o liczbę mniejszą niż 10 lub większą niż 100, następnie sprawdzić czy wartość jest poprawna, po czym wypisać odpowiedni komunikat. Jeśli warunek w pierwszej instrukcji warunkowej if jest spełniony (TRUE) zostaje wykonana następna instrukcja. Warunek jest spełniony, gdy wprowadzona liczba jest większa od 100. Jeśli liczba jest większa niż 100 wtedy wykonywana jest kolejna instrukcja. Jeśli wprowadzona liczba jest mniejsza od 10, wtedy warunek jest niespełniony (FALSE) i program przechodzi do kolejnej instrukcji po instrukcji warunkowej if. Klauzula else miała być dołączona dołączona do pierwszej instrukcji warunkowej if. Jednak w rzeczywistości ta klauzula else jest dołączona do drugiej instrukcji warunkowej if, co powoduje, że w programie występuje subtelny błąd. Błąd jest subtelny, gdyż kompilator go nie zauważy i nie zgłosi. Jest to w pełni poprawny program w języku C++, lecz nie wykonuje tego, do czego został napisany. Na dodatek w sporej liczbie testów przeprowadzonych przez programistę, będzie działał poprawnie. Dopóki wprowadzane będą liczby większe do 100, program będzie działał poprawnie.

```
#include<iostream>
using namespace std;
int main () {
  int x;
  cout<<"Wpisz liczbe mniejsza od 10 lub wieksza od 100";
  cin>>x;
  if (x >= 10)
    if ( x > 100)
      cout<<"Liczba jest wieksza od 100";
  else
    cout<<"Mniejsza niz 10";
  return 0;
}
```


BY
NC