Sterowanie warunkowe

Dobrze napisany program powinien być przygotowany na każdą ewentualność i nietypową sytuację, jaka może się przydarzyć w czasie jego działania. W niektórych przypadkach nawet proste czynności mogą potencjalnie kończyć się niepowodzeniem, zaś porządna aplikacja musi radzić sobie z takimi drobnymi (lub całkiem sporymi) kryzysami.

Oczywiście program nie uczyni nic, czego nie przewidziałby jego twórca. Dlatego też ważnym zadaniem programisty jest opracowanie kodu reagującego odpowiednio na nietypowe sytuacje, w rodzaju błędnych danych wprowadzonych przez użytkownika lub braku pliku potrzebnego aplikacji do działania.

Możliwe są też przypadki, w których dla kilku całkowicie poprawnych sytuacji, danych itp. trzeba wykonać zupełnie inne operacje. Ważne jest wtedy rozróżnienie tych wszystkich wariantów i skierowanie działania programu na właściwe tory, gdy ma miejsce któryś z nich.

Do wszystkich tych zadań stworzono w C++ (i w każdym języku programowania) zestaw odpowiednich narzędzi, zwanych instrukcjami warunkowymi. Ich przeznaczeniem jest właśnie dokonywanie różnorakich wyborów, zależnych od ustalonych warunków.

Instrukcja warunkowa if

Instrukcja `if` (‘jeżeli’) pozwala wykonać jakiś kod tylko wtedy, gdy spełniony jest określony warunek. Jej działanie sprowadza się więc do sprawdzenia tegoż warunku i, jeśli zostanie stwierdzona jego prawdziwość, wykonania wskazanego bloku kodu.

Zadanie 1.
Stworzyć plik `pierwszy.cpp` i wpisać w nim następujący kod:

```
#include<iostream>
using namespace std;
int main()
{
  int liczba;
  cout << "Podaj liczbe wieksza od 10: ";
  cin >> liczba;
  if (liczba > 10)
  {
     cout << "Dziekuje." << endl;
     cout << "Podana liczba jest wieksza od 10." << endl;
  }
  system("pause");
  return 0;
}
```

następnie skompilować plik oraz uruchomić program.

Prosta wersja instrukcji `if` nie zawsze jest wystarczająca – nieeleganckie zachowanie naszego przykładowego programu jest dobrym tego uzasadnieniem. Powinien on wszakże pokazać stosowny komunikat również wtedy, gdy użytkownik nie wykaże się chęcią współpracy i nie wprowadzi żądanej liczby. Musi więc uwzględnić przypadek, w którym warunek badany przez instrukcję `if` (u nas `liczba > 10`) nie jest prawdziwy i zareagować nań w odpowiedni sposób.
Naturalnie, można by umieścić stosowny kod po konstrukcji `if`, ale jednocześnie należałoby zadbać, aby nie był on

wykonywany w razie prawdziwości warunku.
Dlatego też C++, jako pretendent do miana nowoczesnego języka programowania, posiada bardziej sensowny i logiczny sposób rozwiązania tego problemu. Jest nim mianowicie fraza `else` ('w przeciwnym wypadku') – część instrukcji warunkowej `if`.
Korzystająca z niej, ulepszona wersja poprzedniej aplikacji przykładowej, może zatem wyglądać chociażby tak:

Zadanie 2.
Zmodyfikuj program z Zadania 1 w następujący sposób:

```
#include<iostream>
using namespace std;
int main()
{
  int liczba;
  cout << "Podaj liczbe wieksza od 10: ";
  cin >> liczba;
  if (liczba > 10)
  {
     cout << "Dziekuje." << endl;
     cout << "Podana liczba jest wieksza od 10." << endl;
  }
  else

    cout << "Podana liczba jest mniejsza badz rowna 10." << endl;

  system("pause");
  return 0;
}
```

Występujący tu blok `else` jest uzupełnieniem instrukcji `if` – kod w nim zawarty zostanie wykonany tylko wtedy, gdy określony w `if` warunek nie będzie spełniony. Dzięki temu możemy odpowiednio zareagować na każdą ewentualność, a zatem nasz program zachowuje się porządnie w obu możliwych przypadkach.

A zatem składnia pełnej wersji instrukcji `if`, uwzględniającej także blok alternatywny `else` wygląda następująco:

```
if (warunek)
{
      instrukcje_1;
}
else
{
      instrukcje_2;
}
```

Przy okazji zadania 2 warto jeszcze zwrócić uwagę na sposób wypisywania instrukcji w obrębie `if` oraz `else`. Instrukcja (wypisanie tekstu: „`Podana liczba jest mniejsza bądź rowna 10`") nie jest (a właściwie nie musi być) umieszczona w klamerkach. Dzieje się tak dlatego, że jest to instrukcja jednowierszowa, a w takich przypadkach klamerki możemy opuścić. W przeciwnym wypadku klamerki są już konieczne.

W powyższych przykładach zauważyliśmy z pewnością także, że warunek wewnątrz instrukcji `if` zawiera jeden z operatorów relacyjnych, a mianowicie `>`, który poznaliśmy na jednych z poprzednich zajęć. Nie jest to odosobniony przypadek, gdyż w zasadzie wszystkie warunki w instrukcji `if` zawierają operatory relacyjne. Co więcej, bardziej

złożone warunki zawierają również operatory logiczne. Obrazuje to poniższy program.

```
#include<iostream>
using namespace std;
int main()
{
  int liczba;
  cout << "Podaj liczbe z przedzialu (-3,4]: ";
  cin >> liczba;
  if ((liczba > -3)&&(liczba <= 4))
     cout << "Dziekuje. To jest dobra liczba." << endl;

  else

    cout << "Podana liczba nie nalezy do danego przedzialu." << endl;

  system("pause");
  return 0;
}
```

Często w programach istnieje konieczność sprawdzania więcej niż dwóch warunków. Wówczas należy zastosować następującą rozbudowaną postać instrukcji `if`:

```
if (warunek_1)
{
      instrukcje_1;
}
else if (warunek_2)
{
      instrukcje_2;
}

.
.
.

else
{
      instrukcje_...
}
```

Zadanie 3.
Napisać program, który pobierze od użytkownika liczbę całkowitą, sprawdzi, czy jest ona parzysta czy nie oraz wypisze odpowiednią informację na ekranie monitora.

Zadanie 4.
Napisać program wczytujący z klawiatury długości trzech odcinków i sprawdzający, czy da się z nich zbudować trójkąt. Jeśli tak, to obliczyć jego pole ze wzoru Herona oraz obwód i wypisać obliczone wielkości na ekranie monitora. Jeśli nie, to wypisać stosowną informację na ekranie.

Wykorzystując operatory logiczne rozwiązać następujące zadania:

Zadanie 5.
Napisać program, który pobierze od użytkownika współrzędne punktu A=(x,y), następnie sprawdzi, w której ćwiartce układu współrzędnych ten punkt leży oraz wypisze odpowiednią informację na ekranie monitora.

Zadanie 6.
Napisać program, który rozwiąże równanie postaci a*x+b=0, tzn:

a) pobierze od użytkownika liczby rzeczywiste a i b,
b) wypisze na ekranie monitora informacje o liczbie rozwiązań powyższego równania (jeśli równanie będzie posiadać jedno rozwiązanie, to program również wypisze je na ekranie).

Zadanie 7.
Napisać program, który rozwiąże (w zbiorze liczb rzeczywistych) równanie kwadratowe postaci a*x^2+b*x+c=0, tzn:

c) pobierze od użytkownika liczby rzeczywiste a, b, c,
d) wypisze na ekranie monitora informacje o liczbie rozwiązań powyższego równania (jeśli równanie będzie posiadać jedno lub dwa rozwiązania, to program również wypisze je na ekranie).

Instrukcja wyboru switch...case

Instrukcja `switch` ('przełącz') jest w pewien sposób podobna do `if`: jej przeznaczeniem jest także wybór jednego z wariantów kodu podczas działania programu. Pomiędzy obiema konstrukcjami istnieją jednak dość znaczne różnice. O ile `if` podejmuje decyzję na podstawie prawdziwości lub fałszywości jakiegoś warunku, o tyle `switch` bierze pod uwagę wartość podanego wyrażenia. Z tego też powodu może dokonywać wyboru spośród większej liczby możliwości niż tylko dwóch (prawdy lub fałszu). Składnię tej instrukcji możemy łatwo wywnioskować z poniższego kodu.

Zadanie 8.
Stworzyć plik `switch.cpp` i wpisać w nim następujący kod:

```
#include<iostream>
using namespace std;
int main()
{
  int opcja;
  float liczba1, liczba2;
  cout << "Podaj pierwsza liczbe rzeczywista: ";
  cin >> liczba1;
  cout << "Podaj druga liczbe rzeczywista: ";
  cin >> liczba2;
  cout << "Wybierz dzialanie:" << endl;
  cout << "[1] Dodawanie" << endl;
  cout << "[2] Odejmowanie" << endl;
  cout << "[3] Mnozenie" << endl;
  cout << "[4] Dzielenie" << endl;
  cout << "[0] Wyjscie" << endl;
  cout << "Twoj wybor: ";
  cin >> opcja;
  switch(opcja)
  {
    case 1:
        cout << liczba1 << " + " << liczba2 << " = " << liczba1 + liczba2 << endl;
        break;
```

```
    case 2:
        cout << liczba1 << " - " << liczba2 << " = " << liczba1 - liczba2 << endl;
        break;
    case 3:
        cout << liczba1 << " * " << liczba2 << " = " << liczba1 * liczba2 << endl;
        break;
    case 4:
        if(liczba2 == 0)
        {
           cout << "Nie dziel ... nigdy przez zero !!!" << endl;
        }
        else
        {
           cout << liczba1 << " / " << liczba2 << " = " << liczba1 / liczba2 << endl;
        }
        break;
    case 0:
        cout << "Koniec dzialania programu." << endl;
        break;
    default: cout << "Nieznana opcja!";
  }
  system("pause");
  return 0;
}
```

następnie skompilować plik oraz uruchomić program.