Warunek wielokrotnego wyboru switch ... case

Działanie instrukcji **switch** jest zupełnie inne niż w przypadku instrukcji **if** o czym będziesz mógł się przekonać w niniejszym rozdziale.

Różnice pomiędzy instrukcjami wyboru

Za pomocą instrukcji warunkowej **if** możemy określić dokładnie co ma się wydarzyć w zależności od stanu jednej lub kilku zmiennych. Instrukcja **if** daje nam pełną kontrolę nad przebiegiem programu. W języku C++ jest jednak dostępna również instrukcja wielokrotnego wyboru **switch**. W przypadku niej możemy wykonywać decyzje tylko i wyłącznie na podstawie wartości jednej zmiennej. Możliwości instrukcji **switch** są nieporównywalnie mniejsze, jednak używanie jej w niektórych przypadkach jest znacznie korzystniejsze dla szybkości działania programu i estetyki kodu niż użycie instrukcji **if**.

Składnia switch ... case

Zanim przejdziemy do omawiania poszczególnych słów kluczowych oraz tego jak działa warunek wielokrotnego wyboru (zwany również przełącznikiem), zapoznajmy się ze składnią tejże instrukcji:

```
switch( zmienna )
{
case wartosc_1:
    //jakiś kod
    break;

case wartosc_2:
    //jakiś kod
    break;

    //...
case wartosc_n:
    //jakiś kod
    break;

default:
    //jakiś kod
    break;
}
```

Słowo kluczowe switch

Instrukcja **switch(...)** służy do podejmowania decyzji wyłącznie na podstawie wartości jednej zmiennej. Zmienna ta **musi być** typem podstawowym i jednocześnie typem całkowitym. Oznacza to w konsekwencji, że wybór będziemy mogli dokonywać tylko i wyłącznie na podstawie wartości liczby całkowitej czy też kodu znaku ASCII (który również jest liczbą). Zmienną, na której chcemy pracować podajemy w nawiasach zaokrąglonych, które muszą znaleźć się zaraz po wystąpieniu słowa kluczowego **switch**.

```
#include <iostream>
using namespace std;
int main()
{
    char zmienna;
    switch( zmienna )
    {
        //...
    }
    return 0;
}
```

Dozwolone typy danych

Jedynymi dozwolonymi typami danych w instrukcji switch są liczby całkowite. Oznacza to, że możemy użyć tylko i wyłącznie zmiennych, które są typów takich jak: **char, short, int, long, long long**. Do tego dochodzi również typ wyliczeniowy **enum**, który również jest w rzeczywistości liczbą całkowitą.

Błędy kompilacji

Gdy uczymy się programować jesteśmy zmuszeni do przyswajania ogromnej ilości wiedzy. Niektóre ważne rzeczy podczas procesu nauki potrafią jednak umknąć naszej uwadze, co w przypadku programowania bardzo często kończy się wyrzuceniem błędu przez kompilator. W przypadku instrukcji switch możemy się spotkać np. z następującym błędem:

```
error: switch quantity not an integer
```

Błąd ten otrzymamy wtedy, gdy zmienna przekazana do instrukcji switch będzie innego typu niż liczba całkowita czyli przykładowo będzie typu **float**. Czytając komunikaty kompilatora **ze zrozumieniem** w 90% przypadków nie będziesz musiał udawać się do internetu w poszukiwaniu przyczyny wystąpienia błędu, ponieważ kompilator dokładnie napisze Ci co i gdzie poszło źle.

Słowo kluczowe case

Skoro już wiemy jak wybrać zmienną tą, która nas interesuje to teraz powinniśmy dowiedzieć się jak w zależności od wartości zmiennej wykonać jakiś kawałek kodu. Jak można się łatwo domyślić po składni już przedstawionej w niniejszym rozdziale do tego celu służy słowo kluczowe **case**. Ogólny zapis wygląda tak:

```
case...:
```

Zapis ten oznacza: jeśli wartość zmiennej (występującej po słowie kluczowym **switch**) będzie równa wartości umieszczonej po słowie kluczowym **case**, to wykonaj kod (znajdujący się po znaku dwukropka). Zdaję sobie sprawę, że niezbyt przyjaźnie jest napisane powyższe zdanie, jednak spróbuj się w nie dobrze wczytać - możesz spróbować też je przeczytać z pominięciem tego co jest w nawiasach. Jeśli to nie pomoże skup się dobrze na poniższym przykładzie i przeanalizuj jego działanie w praktyce - z pewnością stanie się wtedy wszystko jasne:

```
#include <iostream>
using namespace std;
int main()
{
    int liczba;
    cout << "Podaj liczbe: ";
    cin >> liczba;
    switch( liczba )
    {
    case 2:
        cout << "dwa" << endl;
        break;
    case 1:
        cout << "jeden" << endl;
        break;
    case 3:
        cout << "trzy" << endl;
        break;
    }
    return 0;
}
```

Słowo kluczowe default

Instrukcja sterująca switch oferuje nam jeszcze jedną instrukcję i jest nią **default:**. Wejście w kod znajdujący się po słowie kluczowym **default** nastąpi tylko wtedy, gdy żaden z case'ów nie zostanie spełniony. Korzystając więc z faktu, że nic więcej nie mam do powiedzenia na temat tego słowa kluczowego przejdziemy teraz do przykładu:

```
#include <iostream>
using namespace std;
int main()
{
    int liczba;
    cout << "Podaj liczbe: ";
    cin >> liczba;
    switch( liczba )
    {
    case 2:
        cout << "dwa" << endl;
        break;
    case 1:
        cout << "jeden" << endl;
        break;
    case 3:
        cout << "trzy" << endl;
        break;
    default:
        cout << "ani jeden, ani dwa, ani trzy" << endl;
        break;
    }
    return 0;
}
```

Przykładowy wynik programu:

```
Podaj liczbe: 5
ani jeden, ani dwa, ani trzy
```

Słowo kluczowe break

Słowo kluczowe break oznacza: przerwij wykonywanie obecnej **instrukcji sterującej**. W tym przypadku instrukcją sterującą jest switch. Użycie słowa kluczowego **break** ma bardzo duże znaczenie w instrukcji sterującej **switch ... case**. Informuje ona bowiem kompilator, że dany **case** się wykonał i program ma już opuścić instrukcję sterującą **switch**. Jeśli zabraknie słowa kluczowego **break** to program wykona wszystkie instrukcje od momentu spełnienia warunku aż do samego końca switch'a. Bardzo dobrym przykładem demonstrującym konsekwencje braku słowa kluczowego break jest przerobienie programu, który już mieliśmy. Po przeróbce będzie on wyglądał tak:

```
#include <iostream>
using namespace std;
int main()
{
    int liczba;
    cout << "Podaj liczbe: ";
    cin >> liczba;
    switch( liczba )
    {
    case 2:
        cout << "dwa" << endl;
    case 1:
        cout << "jeden" << endl;
    case 3:
        cout << "trzy" << endl;
    }
    return 0;
}
```

Jeśli teraz w programie wpiszemy np. wartość **1** to na ekranie ujrzymy następujący rezultat:

```
Podaj liczbe: 1
jeden
trzy
```

Zgodnie z opisem, który już wcześniej został przedstawiony: od momentu gdy został spełniony warunek wykonają się wszystkie instrukcje aż do napotkania słowa kluczowego break lub bloku kończącego instrukcję sterującą switch.

Tworzenie zmiennych wewnątrz switch'a

Podczas tworzenia swojego programu może się zdarzyć tak, że będziesz chciał utworzyć zmienną wewnątrz instrukcji sterującej **switch**. Kompilator niestety nie obejdzie się z Tobą delikatnie i zacznie wykrzykiwać następujące hasło:

```
error: jump to case label
error:   crosses initialization of 'char mojaZmienna'
```

Błąd ten uzyskaliśmy poprzez próbę utworzenia zmiennej:

```
char mojaZmienna = 'x';
```

Problemem dla kompilatora nie jest tutaj fakt, że zmienna istnieje, tylko że trzeba ją zainicjalizować. Możliwości rozwiązania tego problemu mamy trzy.

- Pierwszym z nich jest nieinicjalizowanie zmiennej w miejscu jej tworzenia. Przykład:

```
char mojaZmienna;
mojaZmienna = 'a';
```

Powyższe rozwiązanie pomimo, iż poprawne to nie zda egzaminu gdy zaczniesz programować obiektowo lub zaczniesz używać np. typu **string**.

- Innym rozwiązaniem, które jest uniwersalne i skuteczne zawsze niezależnie od typu danych to tworzenie zmiennych przed instrukcją wielokrotnego wyboru **switch**. Rozwiązanie to będzie poprawne, jednak nasze ego zapewne nie zostanie zaspokojone. W końcu lubimy stawiać na swoim, czyli zmienna ma być w środku.
- Trickiem jaki należy więc zastosować aby zmienna była w środku to opakowanie kodu w klamry **{ }.** Przykład demonstrujący trzy rozwiązania:

```
#include <iostream>
using namespace std;
int main()
{
    int liczba;
    cout << "Podaj liczbe: ";
    cin >> liczba;
    char innaZmienna = 'B';
    switch( liczba )
    {
    case 1:
        char mojaZmienna;
        mojaZmienna = 'A';
        cout << "Znak " << mojaZmienna << endl;
        break;
    case 2:
        cout << "Znak " << innaZmienna << endl;
        break;
```

```
    case 3:
        {
            char ostatniaZmienna = 'C';
            cout << "Znak " << ostatniaZmienna << endl;
            break;
        }
        default:
        cout << "ani jeden, ani dwa, ani trzy" << endl;
        break;
    }
    return 0;
}
```

Polecam stosowanie ostatniego rozwiązania, jeśli potrzebujesz zmiennej tylko i wyłącznie wewnątrz **case'a**. Odradzam jednocześnie stosowania pierwszego rozwiązania gdyż prędzej czy później będziesz zmuszony do korzystania z drugiego lub trzeciego wariantu.

Zadanie

Napisz prosty kalkulator dla dwóch liczb, obsługujący cztery działania matematycznie: **+**, **-**, ***** i **/**.