Język C++: instrukcje „break“, „switch“ i „continue“

Na dzisiejszych zajęciach uzupełnimy nieco znajomość języka C++, rozszerzając ją o instrukcje sterujące pętlami oraz instrukcję wyboru wielowariantowego. Na początek zajmijmy się instrukcjami sterującymi pętlami. Przeanalizujmy poniższy program (plik *brentinue.cpp*):

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

int main()
{
unsigned i;

   cout<<"Program demonstruje dzialanie instrukcji \"break\" oraz \"continue\""<<endl;
   cout<<"Instrukcja \"break\" konczy dzialanie petli (lub instrukcji switch)"<<endl;
   cout<<"Tutaj podczas 5 obiegu"<<endl<<endl;

   for(i=0; i<100; i++)
   {
         cout<<i<<" ";
         // Po wypisaniu pieciu liczb mamy dosyc...
         if( i>=5 )
         {
            cout<<endl<<endl;
            break;
         }
   }

   cout<<"Instrukcja \"continue\" konczy dany obieg petli"<<endl;
   cout<<"Dla i nieparzystych ( i%2 = 1 ) pominiemy wypisanie liczby"<<endl<<endl;
   for(i=0; i<10; i++)
         if( i%2 ) continue; else cout<<i<<" ";

   cout<<endl;
}
```

Po wykonaniu powyższego programu na ekranie pojawi się następujący tekst:

```
Program demonstruje dzialanie instrukcji "break" oraz "continue"
Instrukcja "break" konczy dzialanie petli (lub instrukcji switch)
Tutaj podczas 5 obiegu

0 1 2 3 4 5

Instrukcja "continue" konczy dany obieg petli
Dla i nieparzystych ( i%2 = 1 ) pominiemy wypisanie liczby

0 2 4 6 8
```

W pierwszej pętli *for()* użyto instrukcji *break*. Przeanalizujmy, jak zadziałała ta instrukcja. Pętla została zaprogramowana tak, aby wykonała się sto razy. W pętli zaprogramowane jest wypisywanie liczb – kolejnych wartości zmiennej *i*, jednak na ekranie pojawia się tylko sześć liczb, a nie sto. Odpowiedzialna jest za to instrukcja *break*. W instrukcji warunkowej *if()* sprawdzamy, czy wartość zmiennej *i* wynosi przynajmniej 5, i jeśli tak jest, wypisujemy na ekran dwa znaki przejścia do nowego wiersza i kończymy pętlę za pomocą instrukcji *break*.

Instrukcja *break* służy do zakończenia pętli *for()*, *while()* oraz *do ... while()*. Innym jej zastosowaniem jest zakończenie ciągu instrukcji wewnątrz instrukcji *switch()*, więcej szczegółów zostanie podane podczas omówienia następnego programu. Pamiętać należy o tym, że instrukcja *break* kończy wyłącznie jedną pętlę – tę, w której została umieszczona! Jeżeli w programie istnieją np. dwie pętle zagnieżdżone jedna wewnątrz drugiej, to umieszczenie instrukcji break w wewnętrznej pętli spowoduje zakończenie działania wyłącznie pętli wewnętrznej. Równoznaczne

będzie to w takiej sytuacji przejściu do kolejnego kroku pętli zewnętrznej.

Druga z pętli w przykładowym programie pokazuje zastosowanie instrukcji *continue*. Działanie tej instrukcji polega na zakończeniu wykonywania aktualnego kroku pętli (można ją stosować ze wszystkimi trzema rodzajami pętli dostępnymi w języku *C++*), poprzez przejście do instrukcji aktualizującej indeksy w pętli *for* (instrukcje zawarte po drugim średniku), zaś dla pozostałych pętli na sprawdzeniu warunku dla kolejnego kroku pętli. Jeśli pętla zawiera blok instrukcji do wykonania, to można to sobie wyobrazić jako skok do zamykającego nawiasu klamrowego tego bloku.

Ostatnią instrukcją, jaką się dzisiaj zajmiemy jest instrukcja *switch()*. Służy ona do realizacji wyboru wielowariantowego. Działanie jej omówię na przykładzie (plik *switch.cpp*):

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

int main()
{
int x;

   cout<<"Instrukcja switch sluzy do podejmowania wielowariantowych wyborow"<<endl;
   cout<<"Podaj liczbe (0, 1 lub 2): ";
   cin>>x;

   switch( x )
   {
         case 0:  cout<<"Podales 0"<<endl;
                  break;

         case 1: cout<<"Podales 1"<<endl;
                  // Tutaj nie damy break; i zobaczymy co wyjdzie...

         case 2: cout<<"Mogles podac rowniez 2..."<<endl;
                  break;

         default: cout<<"Oszukujesz, podales cos innego niz 0, 1 lub 2"<<endl;
   }
}
```

Instrukcja *switch* przyjmuje w nawiasach wyrażenie, którego wartość będzie sprawdzana. Wyrażenie to musi dawać wartości typu całkowitego *int* bądź typu, dla którego wykonalna jest konwersja na typ *int*. Użytkownik programuje pewne reakcje na wybrane wartości badanego wyrażenia, pisząc ciąg instrukcji wewnątrz nawiasów klamrowych, podobnie jak definiuje instrukcje w klamrach pisząc funkcje. Przed każdą z takich instrukcji może zostać wpisana następująca konstrukcja:

case *stała* :

Stałe w takich zapisach to wartości badanego wyrażenia, na które zareaguje nasz program – muszą być one różne od siebie w ramach jednej instrukcji *switch*. Jak działa instrukcja *switch()* ? Po kolei, rozpoczynając od góry, dla każdej podanej stałej zostaje sprawdzone, czy wartość wyrażenia jest z nią zgodna. Jeśli nie jest zgodna, przechodzi się do sprawdzania kolejnej stałej, w przeciwnym wypadku wykonywane są wszystkie instrukcje, aż do końca bloku. Jeśli podczas szukania zgodności zostanie wyczerpany zestaw stałych podanych przez użytkownika (żadna z nich nie jest zgodna z wartością badanego wyrażenia), wykonywane są instrukcje stojące za specjalną etykietą *default* (ang. domyślny), na takich samych zasadach, jak przy zgodności z etykietą zawierającą wartość podaną przez użytkownika. Pamiętać należy, że etykieta *default* nie musi być ostatnią etykietą w bloku, choć zazwyczaj jest umieszczana jako ostatnia.

Typowe jest kończenie listy instrukcji dla poszczególnych przypadków za pomocą instrukcji

break, jak to jest pokazane w przykładowym programie dla liczb 0 oraz 2. W takim wypadku oczywiście nie zostaną wykonane wszystkie instrukcje aż do końca bloku po znalezieniu zgodności wartości testowanego wyrażenia z jednym z założonych przypadków. Działanie instrukcji *break* polega tutaj na zakończeniu instrukcji *switch*.

Zadanie

Proszę sprawdzić działanie programu *switch.cpp* w zależności od wartości wprowadzonych przez użytkownika.

Następnie proszę napisać program, który obliczy wartość wielomianu w podanym przez użytkownika punkcie, przy czym stopień wielomianu ma wynosić 0, 1, 2 lub 3, zaś podanie innego stopnia należy potraktować jako błąd i wypisać stosowny komunikat na ekran. Zarówno wartość argumentu, jak i współczynniki wielomianu podaje użytkownik.