KURS C++

Zmienne i operatory

Przygotował: Kamil Feliszewski

Spis treści

1 Zmienne

> "Maszyna analityczna nie ma niczego tworzyć. Może zrobić wszystko, czego wykonywanie potrafimy określić. Może przeprowadzać analizy, ale nie ma mocy, aby przewidywać relacje analityczne i prawa. Jej zadaniem jest pomoc w udostępnieniu nam tego, co jest nam już znajome."

Ada Augusta, Hrabina Lovelace (1815 - 1852)

1.1 Zmienne w C++

W C++ aby zadeklarować daną zmienną należy na wstępnie określić jej typ, a następnie podać jej nazwę. Istnieją różne typy zmiennych, a różnica między nimi bezpośrednio dotyczy ich zakresów i sposobów reprezentacji. Najczęściej stosowane zmienne to:

- zmienne sterujące: i, j, k, a, b, c, x, y, z - są to zmienne jednoznakowe, które przechowują informację o rodzaju zmiennej bądź działaniu na nich wykonywanych
- zmienne wieloczłonowe np. ala_ma_kota - charakteryzuje się je tym iż poszczególne elementy są łączone podkreślnikiem (podłogą)
- zmienne skrótów nazw własnych - np. licz, tab

Zmienne mogą przyjmować różne formy, lecz powinny one być zrozumiałe zarówno dla programisty jak i dla użytkownika.

Nie należy w deklaracji zmiennych stosować:

- przerw pomiędzy słowami w deklaracji, np. ala ma kota
- używać polskich znaków typu: ą, ć, ę, ł, ń, ó, ś, ż, ź
- zaczynać deklaracji od liczby np. 999pogotowie
- słów kluczowych w deklaracji zmiennych

W C++ zmienne posiadają różne typy:

- int - z ang. integer, co oznacza zakres liczb całkowitych, np.: -356, -4000, -567, -893, -4089, 1, 6, 2000, 3567, 5053
- float - liczby zmiennoprzecinkowe (w C++ pisane po kropce), np.: 2,14, 3,14, -8,02
- double - liczby zmiennoprzecinkowe, nazywane również zmiennymi podwójnej precyzji, różnią się one od float tym, że po kropce liczb wyświetlanych jest o wiele więcej, nawet do 15 cyfr
- string - ciąg znaków, np. ala, liczba pierwsza. W tym typie zmiennych konieczne jest dołączenie dyrektywy "#include<string>"
- char - typ znakowy np. $, %, a, z. Zakres ten dotyczy zakresu kodu ASCII.
- bool - typ TRUE FALSE, określany jako typ logiczny zwracający prawdę bądź fałsz. W przypadku fałszu jest to wartość 0 (zero), a w przypadku prawdy równe czemukolwiek innemu różnemu od zera

Istnieje również deklaracja stałej: `CONST ala="kot";`

2 Operatory

Aby przeprowadzić odpowiednie operacje na zmiennych należy użyć odpowiednich operatorów, których zadaniem będzie coś policzy, wyświetlić, porównać, przypisać itd.

2.1 Operatory arytmetyczne

Operatory arytmetyczne służą do wykonywania wszelkiego rodzaju działań na zmiennych liczbowych.

Operator	**Składnia**	**Opis**
+	a + b	operator dodawania
-	a - b	operator odejmowania
*	a * b	operator mnożenia
/	a / b	operator dzielenia
%	a % b	operator reszty z dzielenia dwóch liczb całkowitych

2.2 Operatory logiczne

Operatory logiczne mają zastosowanie w miejscach, gdzie występują różnego rodzaju warunki, np. instrukcje warunkowe, pętle.

Operator	**Składnia**	**Opis**
&&	a && b	i (and) logiczne
||	a || b	lub (or) logiczne
!	!a	zaprzeczenie (negacja)

2.3 Operatory relacyjne

Operatory relacyjne służą do porównania dwóch elementów, najczęściej wykorzystywane w instrukcjach warunkowych i pętlach.

Operator	**Składnia**	**Opis**
<	a < b	mniejszy
>	a > b	większy
<=	a <= b	mniejszy równy
>=	a >= b	większy równy
==	a == b	równy
!=	a != b	różny

2.4 Operatory przypisania

Operatory przypisania służą do nadania wartości zmiennej znajdującej się po lewej stronie, wartości zmiennej znajdującej się po prawej stronie.

Operator	Składnia	Opis
/=	a /= b, a = a / b	podzielenie wartości zmiennej a przez wartość zmiennej b i przypisanie wyniku do zmiennej a
*=	a *= b, a = a * b	pomnożenie wartości zmiennej a przez wartość zmiennej b i przypisanie wyniku do zmiennej a
+=	a += b, a = a + b	dodanie do wartości zmiennej a wartość zmiennej b i przypisanie wyniku do zmiennej a
-=	a -= b, a = a -b	odjęcie od wartości zmiennej a wartości zmiennej b i przypisanie wyniku do zmiennej a
%=	a %= b, a = a % b	obliczenie reszty z dzielenie wartości zmiennej a przez wartość zmiennej b i przypisanie wyniku do zmiennej a

2.5 Operatory bitowe

Dzięki operatorom bitowym i operacją na nich można dokonywać tzw. przesunięć o daną wartość bitu (w prawo lub w lewo). Następstwem tych operacji jest zwiększenie bądź zmniejszenie wartości bitu. Co znajduje szeroki zakres w zastosowaniach praktycznych zarówno w świecie programistycznym jak i elektronice zaawansowanej.

Bity w świecie informatyki stanowią najmniejszą wartość. Ich reprezentacja określana jest przez zastosowanie praktyczne kodu binarnego, a mianowicie dwóch wartości (0,1). Aby stosować ten kod w praktyce należy zapoznać się z liczbami wielokrotności potęgi liczby 2.

Każda liczba podniesiona do potęgi 0 daje 1.

Przykład: przekonwertowanie liczby binarnej na dziesiętną $1011_{(2)}$:
$1^{2^3}0^{2^2}1^{2^1}1^{2^0}$ co daje nam 8+0+2+1 = 11

Zamiana liczby dziesiętnej na binarną przebiega w następujący sposób: dzielimy liczbę przez 2, jeżeli ma resztę z dzielenia piszemy 1, jeżeli nie ma piszemy 0:

Kolejna liczba	Reszta z dzielenia przez 2
11	1
5	1
2	0
1	1

Odczytujemy od dołu do góry i otrzymujemy: 1011.

Aby dokonać przesunięcia liczby binarnej o bit (w prawo) należy naszej liczby usunąć liczbę z prawej strony o określoną ilość bitów, a następnie uzupełnić ją o tyle zer z lewej strony co usunęliśmy z prawej:
1101
0110

Przesunięcie o daną ilość bitów w lewo odbywa się na tej samej zasadzie co w prawo, lecz kasowane są liczby z lewej strony, a zera dodawane z prawej:
1101
1010

2.6 Lista operatorów

W języku C++ istnieją również inne operatory:

Operator	Opis	Przykład
::	przestrzeń nazw	std::cout
->	wybór składowy	obiekt -> metoda()
[]	indeksowanie tablicy	tablica[1]
()	wywołanie funkcji	suma(a, b)
++	zwiększenie o jeden	a++
- -	zmniejszenie i jeden	a- -
typeid	identyfikacja typu	typeid(a)
dynamic_cast	dynamiczna konwersja typu	dynamic_cast(a)
static_cast	statyczna konwersja typu	static_cast(a)
reinterpret_cast	niesprawdzona konwersja typu	reinterpret_cast<double*>(a)
const_cast	konwersja z lub na CONST	const_cast<const int*>(&y)
~	negacja bitowa	~x
!	negacja logiczna	!a
+	plus jednoargumentowy	a + b
-	minus jednoargumentowy	a - b
&	pobranie adresu	&a
*	wydobycie wartości	int* a
new	przydział pamięci	new int[10]
delete	zwolnienie obiektu	delete dynOb
delete[]	zwolnienie tablicy	delete[] tablica
(typ)wyrażenie	konwersja typu w stylu C	(int) d
.* ->*	wybór składowej	obj.*displayP
*	mnożenie	a * b
/	dzielenie	a / b
%	modulo	a % b
«	bitowe przesunięcie w lewo	a « 2
»	bitowe przesunięcie w prawo	a » 2
<	mniejszy	a < b
<=	mniejszy równy	a <= b
>	większy	a > b
>=	większy równy	a >= b
==	równy	a == b
!=	nierówny	a != b
&&	logiczne i (and)	a && b
||	logiczne lub (or)	a || b
? :	wyrażenie warunkowe	result = (x > y) ? x : y
=	przypisanie wartości	a = b
*=	wykonaj operacje i przypisz wartość	a *= b
/=	wykonaj operacje i przypisz wartość	a /= b
+=	wykonaj operacje i przypisz wartość	a += b
-=	wykonaj operacje i przypisz wartość	a -= b
throw	zgłoszenie wyjątku	throw invalidt("wyjątek");
,	operator przecinkowy	result = (x++, y++, x + y)

3 Zastosowanie praktyczne operatorów w języku C++

Aby zastosować poprawnie dany operator w praktyce należy odpowiednio dobrać właściwe zmienne, a co się z tym wiąże odpowiednie typy tych zmiennych.

3.1 Operatory arytmetyczne

- + dodaje dwie wartości, które znajdują się po jego lewej i prawej stronie. Całe takie wyrażenie zwraca określony wynik. Jeżeli liczba znajduje się tylko po jego prawej stronie to traktowany jest on po prostu jako dodatni znak liczby.
  Przykłady:

```
int liczba = 2 + 2; \\wynik: 4
liczba = +7; \\wynik: 7
liczba = 3 + 7 + 5; \\wynik: 15
```

- - służy on do odejmowania dwóch liczb jeśli znajduje się liczba tylko po jego prawej stronie to jest on traktowany jako znak ujemny liczby.
  Przykłady:

```
int liczba = 5 - 3; \\wynik: 2
liczba = -8; \\wynik: -8
liczba = 7 - 2 - 3; \\wynik: 2
```

- * służy do wykonania mnożenia. Wymaga liczb po obu stronach.
  Przykłady:

```
int liczba =  2 * 2; \\wynik: 4
```

- / służy do wykonywania dzielenia. Wymaga liczb po obu stronach.
  Przykłady:

```
int liczba =  2 / 2; \\wynik: 1
```

- % służy do obliczenia reszty z dzielenia. Wymaga liczb po obu stronach.
  Przykład:

```
int liczba =  3 % 2; \\wynik: 1
```

Najpierw wykonuje się mnożenie ale w razie wątpliwości, który z operatorów działa pierwszy stosuje się nawiasy. Wtedy to co jest w nawiasach ma wyższy priorytet.

3.2 Operatory przypisania

- = przypisuje on wartość znajdującą się po jego prawej stronie do zmiennej, która znajduje się po jego lewej stronie.
  Przykłady:

```
int liczba =  2 ; \\wynik: 2
```

W językach programowania zapisuje się w postaci `liczba = liczba + 3;` **Jest to zwiększenie zmiennej liczba o wartość 3. W C++ można to zapisać o wiele prościej, a mianowicie** `liczba += 3;`

3.3 Operatory porównania

Operatory porównania służą do porównywania ze sobą dwóch wartości i zwracają wartości logiczne:

- == zwraca wartość TRUE tylko wtedy, gdy oba argumenty są ze sobą równe
- != zwraca wartość TRUE tylko wtedy, gdy argumenty są różne (nierówne)
- > zwraca wartość TRUE tylko wtedy, gdy lewy argument jest większy od prawego
- < zwraca wartość TRUE tylko wtedy, gdy prawy argument jest większy od lewego
- >= zwraca wartość TRUE tylko wtedy, gdy lewy argument jest większy lub równy argumentowi prawemu
- <= zwraca wartość TRUE tylko wtedy, gdy prawy argument jest większy lub równy argumentowi lewemu

Przykłady:

```
cout << 4 == 3 + 1;
cout << 2 != 5;
cout << 8 > 12;
cout << 2 < 3;
```

3.4 Operatory logiczne

Operatory logiczne służą do formułowania wyrażeń logicznych:

- && (i, AND) zwraca wartość logiczną 1 tylko wtedy, gdy oba argumenty mają wartość 1

p	q	AND
0	0	0
0	1	0
1	0	0
1	1	1

- || (lub, OR) zwraca wartość logiczną 1 w przypadku, gdy pierwszy bądź drugi argument ma wartość 1

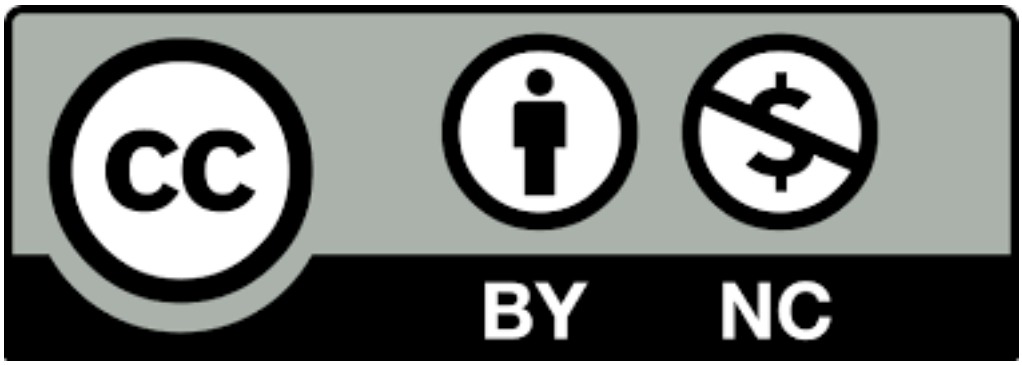
p	q	OR
0	0	0
0	1	1
1	0	1
1	1	1

- ! (negacja, NOT) zwraca 1 gdy argument posiada wartość logiczną 0 (zero)

p	NOT
0	1
1	0

Przykłady:

```
cout << 0 \&\& 1; \\ 0
cout << 1 || 0; \\ 1
cout << !0; \\ 1
```


BY
NC