Programowanie 1

Sławomir Pluciński (splucinski@pjwstk.edu.pl)

Operatory logiczne

Operator	Działanie	Przykład
!	negacja	! a
&&	koniunkcja (iloczyn logiczny)	a && b
||	alternatywa (suma logiczna)	a || b

- wyrażenia połączone operatorami && i || zawsze są wartościowane od strony lewej do prawej;
- kompilator oblicza wartość wyrażenia dotąd, dopóki na pewno nie wie jaki będzie wynik.

```
(a == 0)&&(m == 5)&&(x > 23)
```

Operatory bitowe

Operator	Działanie	Przykład
&	bitowa koniunkcja	a = b & c;
|	bitowa alternatywa	a = b | c;
^	bitowa różnica symetryczna	a = b ^ c;
<<	przesunięcie w lewo	a = b << c;
>>	przesunięcie w prawo	a = b >> c;
~	bitowa negacja	a = ~b

Operatory bitowe

Koniunkcja bitowa

```
liczba = 179 & 38;

10110011 (179)
00100110 (38)
--------
00100010 (34)
```

Alternatywa bitowa

```
liczba = 34 | 65;

00100010 (34)
01000001 (65)
--------
01100011 (99)
```

Różnica symetryczna

```
liczba = 34 ^ 118;

00100010 (34)
01110110 (118)
--------
01010100 (84)
```

Przesuniecie bitowe w lewo

```
liczba = 84 << 1;

01010100 (84)
    << 1
10101000 (168)
```

Przesuniecie bitowe w prawo

```
liczba = 84 >> 1;

01010100 (84)
    >> 1
00101010 (42)
```

Inkrementacja i dekrementacja

- Inkrementacja to dodanie 1
- Dekrementacja to odjęcie 1

Przykład:

```
i = i + 1;          // i ++

j = j - 1;          // j - -
```

- operatory inkrementacji i dekrementacji mogą występować w dwóch odmianach: przedrostkowej i przyrostkowej; i++ ;i-- ; ++i ; --i ;
- operator przedrostkowy jest obliczany przed przypisaniem;
- operator przyrostkowy jest obliczany po przypisaniu.

Przykład

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

int main(){
int i=10, j=10;
 cout << "i = " << i << endl
      << "j = " << j << endl;
 i++; ++j;
 cout << "i = " << i << endl
      << "j = " << j << endl;
 cout << "i = " << i++ << endl
      << "j = " << ++j << endl;
 cout << "i = " << i << endl
      << "j = " << j << endl;

}
```

```
i = 10
j = 10
i = 11
j = 11
i = 11
j = 12
i = 12
j = 12
```

Pozostałe operatory przypisania

Operator	Zapis skrócony	Zapis rozwinięty
+=	a += b;	a = a + b;
- =	a - = b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;
<<=	a <<= b;	a = a << b;
>>=	a >>= b;	a = a >> b;
&=	a &= b;	a = a & b;
|=	a |= b;	a = a | b;
^=	a ^= b;	a = a ^ b;

Operator sizeof

Operator, który pozwala sprawdzić ile pamięci zabiera dany typ lub obiekt.

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

int main(){
    cout << sizeof(int) << endl;
    cout << sizeof(char) << endl;
    cout << sizeof(long);
}
```

Operator rzutowania

(nazwa_typu) obiekt
lub
nazwa_typu (obiekt)

Przykład:

```
int a = 85;
char b;
b = (char) a;          // b = 'U'
```

Priorytety i łączność operatorów

Operator	Priorytet	Łączność	Działanie
::	17	L	zasięg globalny
::		P	zasięg klasy
.->	16	L	dostęp do składowej klasy
[ ]		L	indeksowanie
( )		L	wywołanie funkcji
sizeof ( )		L	rozmiar typu/obiektu
++	15	P	inkrementacja
- -		P	dekrementacja
~		P	negacja bitowa
!		P	negacja logiczna
- +		P	minus/plus jednoargumentowy
&		P	adres argumentu/referencja
*		P	dostęp pośredni
new		P	alokacja pamięci

Priorytety i łączność operatorów

Operator	Priorytet	Łączność	Działanie
delete	15	P	zwalnianie pamięci
delete[ ]	15	P	usuwanie tablicy dynamicznej
( )	15	P	konwersja typu (rzutowanie)
.*	14	L	dostęp do składowej
->*	14	L	dostęp do składowej
* / %	13	L	mnożenie, dzielenie, modulo
- +	12	L	odejmowanie, dodawanie
<< >>	11	L	przesuwanie w lewo/ w prawo
< <=	10	L	mniejsze/ nie większe
> >=	10	L	większe/nie mniejsze
== !=	9	L	równe/nie równe
&	8	L	koniunkcja bitowa
^	7	L	bitowa różnica symetryczna
|	6	L	alternatywa bitowa

Priorytety i łączność operatorów

Operator	Priorytet	Łączność	Działanie
&&	5	L	koniunkcja logiczna
||	4	L	alternatywa logiczna
? :	3	L	wyrażenie warunkowe
=	2	P	przypisanie
*=	2	P	mnożenie i przypisanie
/=	2	P	dzielenie i przypisanie
%=	2	P	modulo i przypisanie
+=	2	P	dodanie i przypisanie
-=	2	P	odjęcie i przypisanie
<<=	2	P	przesunięcie w lewo i przypisanie
>>=	2	P	przesunięcie w prawo i przypisanie
&=	2	P	koniunkcja bitowa i przypisanie
^=	2	P	różnica symetryczna i przypisanie
|=	2	P	alternatywa bitowa i przypisanie
,	1	L	ustalenie kolejności

Priorytety i łączność operatorów

- Priorytet oznacza kolejność przykładania operatorów do ich argumentów.
- Jeżeli wyrażenie zawiera różne operatory, to wartościowanie wyrażenia przebiega w kolejności, którą określa priorytet operatorów.
- Można wymusić inną kolejność wartościowania, zamykając wyrażenie lub część wyrażenia w nawiasach okrągłych.
- Operatory języka C++ są prawo- lub lewostronnie łączne.
- Oznacza to, że jeżeli w wyrażeniu są dwa operatory o jednakowym priorytecie, to najpierw jest wykonywany operator prawy (lewy).

Algorytm sortowania przez wstawianie

6 5 3 1 8 7 2 4

1. Utwórz zbiór elementów posortowanych i przenieś do niego dowolny element ze zbioru nieposortowanego.
2. Weź dowolny element ze zbioru nieposortowanego.
3. Wyciągnięty element porównuj z kolejnymi elementami zbioru posortowanego, póki nie napotkasz elementu równego lub elementu większego (jeśli chcemy otrzymać ciąg niemalejący) lub nie znajdziemy się na początku/końcu zbioru uporządkowanego.
4. Wyciągnięty element wstaw w miejsce, gdzie skończyłeś porównywać.
5. Jeśli zbiór elementów nieuporządkowanych jest niepusty, wróć do punktu 2.

Algorytm sortowania bąbelkowego

Polega na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeżeli zaburza ona porządek, w jakim się sortuje tablicę.

Sortowanie kończy się, gdy podczas kolejnego przejścia nie dokonano żadnej zmiany.

Napisy

Przetwarzanie tekstów

- Ciągi znaków stanowią drugi, po liczbach, rodzaj informacji przetwarzanej przez komputery.
- Zaleta ciągów – są bardziej zrozumiałe dla człowieka
- Wada – zajmują więcej miejsca w pamięci
- Wada – w przeciwieństwie do typów liczbowych, mają zmienny rozmiar

Przetwarzanie tekstów

- Język C w ogóle nie miał odrębnego typu zmiennych do przechowywania napisów.
- Do przetwarzania tekstów trzeba było używać mało poręcznych tablic znakowych.
- Język C++ jest wyposażony w mechanizmy łatwej manipulacji tekstem.
- Rozwiązania te są częścią Biblioteki Standardowej C++
- Wystarczy dołączyć plik nagłówkowy:

```
#include <string>
```

Deklaracja zmiennej tekstowej

```
string napis;
```

- Zmienna napis jest na początku pusta, nie zawierająca żadnego znaku

```
string napis = ”Ala ma kota”;

lub

string napis_1 (”Ala ma kota”);
```

Deklaracja zmiennej tekstowej

```
string napis_2 = (”Ala ma kota”, 3);
napis_2 = "Ala"
```

- Drugi parametr (np. 3) określa liczbę znaków fragmentu tekstu, licząc od początku.
- Indeksowanie znaków napisu zaczyna się od 0.

Łączenie napisów

```
string str1 = ”gra”;
string str2 = ”ty”;

string str3 = str1 + str2;
str3 = "graty";
```

Operator +=

```
string str = ”abc”;
str += ”def”;

str = ”abcdef”;
```

Operatory dla zmiennych typu string

Operator	Działanie
+	Złożenie dwóch argumentów typu string
!=	Test – czy dwa argumenty typu string nie są równe
==	Test – czy dwa argumenty typu string są równe
<	Test – czy argument z lewej strony operatora jest mniejszy od argumentu stojącego z prawej strony operatora
<=	Test – czy argument z lewej strony operatora jest mniejszy lub równy argumentowi stojącemu z prawej strony operatora

Operatory dla zmiennych typu string

Operator	Działanie
<<	Operator wstawiania argumentu typu string do strumienia wyjściowego
>	Test – czy argument z lewej strony operatora jest większy od argumentu stojącego z prawej strony operatora
>=	Test – czy argument z lewej strony operatora jest większy lub równy argumentowi stojącemu z prawej strony operatora
>>	Operator wyjmowania argumentu typu string ze strumienia wejściowego

Operatory dla zmiennych typu string

Funkcja	Działanie
stoi	Konwertuje ciąg znaków na liczbę typu int
stol	Konwertuje ciąg znaków na liczbę typu long
stoul	Konwertuje ciąg znaków na liczbę typu unsigned long
stoll	Konwertuje ciąg znaków na liczbę typu long long
stoull	Konwertuje ciąg znaków na liczbę typu unsigned long long
stof	Konwertuje ciąg znaków na liczbę typu float
stod	Konwertuje ciąg znaków na liczbę typu double
stold	Konwertuje ciąg znaków na liczbę typu long doulbe
to_string	Konwertuje liczby typu long long, unsigned long long oraz long double na ciąg znaków
swap	Zamienia miejscami dwa argumenty typu string
getline	Pobiera ze strumienia wejściowego pojedynczą linię znaków (do ogranicznika)