Wyrażenia i operatory

Wyrażenia

Wyrażenia elementarne:

- liczby, znaki, zmienne, stałe, wywołania funkcji

Operatory:

- jedno- dwu- lub trójargumentowe
- o ustalonych priorytetach
- nawiasy okrągłe dla ustalenia kolejności obliczeń

operator przypisania	=
operator dostępu pośredniego	*
operator wyznaczania wskaźnika (referencja)	&

Operator przypisania

```
int    i,   j,   k;

i = 1;        //zmienna i przyjmuje wartość 1

j = i;        //zmienna j przyjmuje wartość zmiennej i

k = dni;      //zmienna k przyjmuje wartość stałej dni
```

Konwersja wartości

lewy typ (różny) prawy typ ➔ typ (lewy) == typ (prawy)

Konwersja rozszerzająca ➔ zwiększenie liczby bajtów
Konwersja zawężająca ➔ zmniejszenie liczby bajtów

```
int li32 = 21212345;
long long li64 = li32; // konwersja rozszerzająca
short li16 = li32;     // konwersja zawężająca
                       // strata danych li16 == -21319
```

Dla operatora oznacza to konwersję na typ lewej strony.
Możliwa jest strata informacji.

```
char c;
c = 258;               // c == ?
```

Operatory arytmetyczne

```
*   /   +   -   %
```

```
int  liczba,  nieparzystosc;
nieparzystosc = liczba % 2; // 0 - p, 1 - np
```

$$\frac{3x^2 + 5x - 1}{7x((2x+3)(1-x)+5)+3}$$

```
(3 * x * x + 5 * x - 1) /
                (7 * x * ((2 * x + 3) * (1 - x) + 5) + 3)
```

```
int  a = 1700000000,  b = 1900000000, suma;
suma = a + b;  // nadmiar stałopozycyjny

float  x = 0.5e35,  y = 0.2e5,  z;
z = x * y;     // nadmiar zmiennopozycyjny
```

Operatory zmniejszania i zwiększania

++ --

```
++alfa      --beta  // przed obliczeniem wyrażenia
alfa++      beta--  // po obliczeniu wyrażenia
```

```
float  x = 2.5,  y;

x ++ ;              // równoważne x = x + 1;
++ x ;              // równoważne x = x + 1;
x -- ;              // równoważne x = x - 1;
-- x ;              // równoważne x = x - 1;
y = ++(2 * x);      // błąd


int  i = 3,  j = 4,  s;

s = j++ + i;        // s == 7   j == 5
j = 4;
s = ++j + i;        // s == 8   j == 5
```

Operatory relacyjne

```
0       :  false
nie 0   :  true
```

```
<    <=   ==   !=   >=   >
```

```
bool  mniejsze,  rowne,  nierowne, wiekszerowne;
int   i = 5;
float  x = 12.3;

mniejsze     = i < x;        // true
rowne        = i == x;       // false
nierowne     = i != x;       // true
wiekszerowne = i >= x;       // false


double  x = 1.5E-5,  y = x;
double  *wsk_x = & x,  *wsk_y = & y;

rowne = wsk_x == wsk_y;    // false
wsk_x = & y;
rowne = wsk_x == wsk_y;    // true
```

Operatory logiczne

```
!   &&   ||
```

```
int a, b, c;
bool z;

z = a < b && b < c;              // a < b < c
```

```
int rok = 2000;
bool przestepny;

przestepny = !(rok % 4)   &&   rok % 100 || !(rok % 400);
```

```
( a > b ) && ( k <= f++ )        // optymalizacja
( k > 5 ) || (c < ( b = 7 ) )
```

Złożone operatory przypisania

```
*=    /=    %=    +=    -=    <<=    >>=    &=    ^=    |=
```

```
double  kurs,  zwyzka;
kurs += zwyzka;    // kurs = kurs + zwyzka;
```

```
int x = 10;
x -= 5 - 2 ;  // ?
```

```
int i, j, k;
i = (j = 5) + 1; // równ. j = 5; i = j + 1;
i = j = k = 0;   // równ. i = 0; j = 0; k = 0;
```

Operator warunkowy

```
wyrażenie_1 ? wyrażenie_2 : wyrażenie_3
```

```
float x, y, max;

max = x > y ? x : y;
```

Operator rozmiaru

```
long liczba_1;

dlugosc_1 = sizeof liczba_1; //   == 4

dlugosc_1 = sizeof long;     //   == 4
```

Priorytety i łączność operatorów

```
  a ∘ b ∘ c
((a ∘ b) ∘ c)  // lewostronnie łączne
 (a ∘ (b ∘ c)) // prawostronnie łączne
```

```
char a, b, c, d, e;

a + b - c - d + e;
/* lewostronnie łączne czyli (((a + b) - c) - d) + e; */

a ? b : c ? d : e;
/* prawostronnie łączne czyli a ? b : (c ? d : e); */
```