Wykład 3: Podstawowe operatory.

dr inż. Andrzej Stafiniak

Wrocław 2024

Podstawowe informacje o operatorach

- W celu przeprowadzania jakichkolwiek działań na różnego typu danych wymagane są **operatory.**
- **Operatory** podlegają kolejność wykonywanych działań, za co odpowiada ich wartości **priorytetu** (ang. operator precedence).
- W przypadku dwóch **operatorów** o tym samy priorytecie o kolejności działań będzie decydował parametr łączności (ang. associativity), opisujący który z operatorów wykonany zostanie jako pierwszy: stojący z lewej strony zmiennej (łączność lewostronna) lub z prawej strony (łączność prawostronna).
- Jeżeli nie pamiętamy, który z dwóch operatorów ma większy priorytet, możemy posłużyć się dodatkowymi nawiasami okrągłymi ().

Operatory - priorytet i łączność (C++)

Priorytet	Operator	Opis	Łączność
1	:: [C++]	rozdzielczość zakresu	lewostronna
2	x**++** x**––** **type**(x) **type**{x} x **()** x **[]** **.** / **–>**	postinkrementacja postdekrementacja rzutowanie typu rzutowanie typu [od C++11] wywołanie funkcji indeks dostep do składowych klasy (struktury)	lewostronna
3	**++**x **––**x **+**x **–**x **!** **~** **(type)**x *****x **&**x **sizeof** **new**/**new[]** **delete**/**delete[]**	preinkrementacja predekrementacja promocja liczby negacja liczby logiczna negacja (NOT) bitowa negacja (NOT) rzutowanie typu (w stylu C) dereferencja (wyłuskanie) adres rozmiar (w bajtach) dynamiczna alokacja pamięci [C++] dynamiczne zwolnienie pamięci [C++]	prawostronna

Operatory - priorytet i łączność (C++)

Priorytet	Operator	Opis	Łączność
4	.* / –>*	dostęp do składowych klasy (struktury) przez wskaźnik	lewostronna
5	x * y x / y x % y	mnożenie dzielenie dzielenie z reszta	lewostronna
6	x + y x – y	dodawanie odejmowanie	lewostronna
7	<< >>	przesuniecie bitowe w lewo przesuniecie bitowe w prawo	lewostronna
8	<=>	trójwynikowe porównanie [od C++20]	lewostronna
9	< <= > >=	mocna relacja mniejszości słaba relacja mniejszości silna relacja większości słaba relacja większości	lewostronna
10	== !=	jest równe jest różne	lewostronna
11	&	iloczyn bitowy (AND)	lewostronna
12	^	bitowa alternatywa wykluczajaca (XOR)	lewostronna
13	|	suma bitowa (OR)	lewostronna

Operatory - priorytet i łączność (C++)

Priorytet	Operator	Opis	Łaczność
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna
16	x ? y : z	warunek trójskładnikowy	prawostronna
	throw	rzucenie wyjatku [C++]	
	=	przypisanie	
	+=	przypisanie i dodawanie	
	-=	przypisanie i odejmowanie	
	*=	przypisanie i mnozenie	
	/=	przypisanie i dzielenie	
	%=	przypisanie i dzielenie z reszta	
	<<=	przypisanie i przesuniecie bitowe w lewo	
	>>=	przypisanie i przesuniecie bitowe w prawo	
	&=	przypisanie i bitowa koniunkcja	
	^=	przypisanie i bitowa alternatywa wykluczajaca	
	|=	przypisanie i bitowa alternatywa	
17	,	przecinek	lewostronna

Operator przypisdania

Priorytet	Operator	Opis	Łączność
16	=	przypisanie	prawostronna

```
int x = 7;
```

W sensie matematycznym operator przypisanie nie oznacza „równa się”.

```
x = x + 7;
```

Możliwość przypisywania wielokrotnego.

```
int x, y, z;
x = y = z = 7;
```

Przebiega działania od prawej strony do lewej.

Operatory arytmetyczne

Priorytet	Operator	Opis	Łączność
5	x * y x / y x % y	mnożenie dzielenie dzielenie z reszta	lewostronna
6	x + y x – y	dodawanie odejmowanie	lewostronna

Istotna kolejność działań zgodnie z priorytetem operatorów i zasadami arytmetyki.

```
int x, y, z;
z = x / y;
```

Uwaga! Rzutowanie niejawne, dzielenie wartości całkowitych mimo reszty z dzielenia daje wynik całkowity.

Operator rzutowania – promocja typu

Priorytet	Operator	Opis	Łączność
2	type(x) type{x}	rzutowanie typu [w stylu C++] rzutowanie typu [od C++11]	lewostronna
3	(type) x	rzutowanie typu (w stylu C)	prawostronna

Niejawne rzutowanie typów

```
#include <stdio.h>

int main(){
    int x = 1;
    int y = 3;
    float z = x / y;

    printf("z = %f", z);

    return 0;
}
```

Wynik `z = 0.000000`

Niejawne rzutowanie typów - czyli dokonywane przez kompilator w sposób automatyczny.

Dwa argumenty operatora / – `int`
kompilator – wynik – `int`

Jak to naprawić ?

Operator rzutowania – promocja typu

Priorytet	Operator	Opis	Łączność
2	**type**(x) **type**{x}	rzutowanie typu [w stylu C++] rzutowanie typu [od C++11]	lewostronna
3	**(type)** x	rzutowanie typu (w stylu C)	prawostronna

Niejawne rzutowanie typów

```
#include <stdio.h>

int main(){
    int x = 1;
    int y = 3;
    float z = x / y;

    printf("z = %f", z);

    return 0;
}
```

Wynik `z = 0.000000`

Niejawne rzutowanie typów - czyli dokonywane przez kompilator w sposób automatyczny.

Dwa argumenty operatora / – `int`
kompilator – wynik – `int`

Jak to naprawić ?

Zastosować **mechanizm promocji typów.**

Operator rzutowania – promocja typu

Priorytet	Operator	Opis	Łączność
2	type(x) type{x}	rzutowanie typu [w stylu C++] rzutowanie typu [od C++11]	lewostronna
3	(type) x	rzutowanie typu (w stylu C)	prawostronna

Jawne rzutowanie typów

```
#include <stdio.h>

int main(){
    int x = 1;
    int y = 3;
    float z = (float) x / y;

    printf("z = %f", z);

    return 0;
}
```

Wynik `z = 0.333333`

Algorytm promocji typów:

`char` -> `short` -> `int` -> `long`

liczby całkowite -> `float` -> `double`

```
#include <stdio.h>

int main(){
    int x = 1;
    float y = 3;
    float z = x / y;

    printf("z = %f", z);

    return 0;
}
```

```
#include <stdio.h>

int main(){
    int x = 1;

    float z = x / 3.0;

    printf("z = %f", z);

    return 0;
}
```

Operator rzutowania – promocja typu

Priorytet	Operator	Opis	Łączność
2	type(x) type{x}	rzutowanie typu [w stylu C++] rzutowanie typu [od C++11]	lewostronna
3	(type) x	rzutowanie typu (w stylu C)	prawostronna

Jawne rzutowanie typów

```
#include <stdio.h>

int main(){
    int x = 1;
    int y = 3;
    float z = (float) x / y;

    printf("z = %f", z);

    return 0;
}
```

Wynik `z = 0.333333`

Jawne rzutowanie typów ładniej w stylu C++

```
#include <iostream>

int main(){
    int x = 1;
    int y = 3;
    float z =  static_cast<float> (x) / y;
    printf("z = %f", z);
    return 0;
}
```

Wynik `z = 0.333333`

Operatory inkrementacji i dekrementacji

Priorytet	Operator	Opis	Łączność
2	x++ x−−	postinkrementacja postdekrementacja	lewostronna
3	++x −−x	preinkrementacja predekrementacja	prawostronna

Różnica między trybem przedrostkowym (++x) a przyrostkowym (x++).

++x, --x;

```
#include <stdio.h>

int main()
{
    int x = 1;
    int y = ++x;
    // Najpierw zostanie zwiększona wartość zmiennej x o jeden, a następnie zostanie użyta wartość 2

    printf( "x = %d; y = %d\n", x, y ); // Wyświetli x = 2; y = 2
    return 0;
}
```

Operatory inkrementacji i dekrementacji

Priorytet	Operator	Opis	Łączność
2	x++ x−−	postinkrementacja postdekrementacja	lewostronna
3	++x −−x	preinkrementacja predekrementacja	prawostronna

Różnica między trybem przedrostkowym (++x) a przyrostkowym (x++).

x++, x--;

```
#include <stdio.h>

int main()
{
    int x = 1;
    int y = x++;
    // Najpierw zostanie użyta wartość 1, a następnie zmienna x zostanie zwiększona o jeden

    printf( "x = %d; y = %d\n", x, y ); // Wyświetli x = 2; y = 1
    return 0;
}
```

Operatory relacyjne

Priorytet	Operator	Opis	Łączność
9	< <= > >=	mocna relacja mniejszości słaba relacja mniejszości silna relacja większości słaba relacja większości	lewostronna
10	== !=	jest równe jest różne	lewostronna

- Operatory relacyjne mają łączność od lewej do prawej.
- Zwracają one wartości typu **`bool`**.

```c
#include <stdio.h>

int main(){
    int x = 1<3<2;
    printf("x = %i", x);
    return 0;
}
```

```
x = 1
```

```c
#include <stdio.h>

int main(){
    int x = 1>3>2;
    printf("x = %i", x);
    return 0;
}
```

```
x = 0
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

! - zaprzeczenie logiczne

a	!a
0	1
1	0

- Operand jest niejawnie konwertowany na typ **bool**.
- Wynik jest typu **bool**.

```
#include <stdio.h>

int main()
{
    int x = 0;
    int y = !x;
    int x1 = 3;
    int y1 = !x1;

    printf( "y = %d; y1 = %d",y,y1);
    //Wyświetli y = 1; y1 = 0

    return 0;
}
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

|| - suma logiczne

a	b	a || b
0	0	0
0	1	1
1	0	1
1	1	1

&& - iloczyn logiczne

a	b	a && b
0	0	0
0	1	0
1	0	0
1	1	1

- Operand jest niejawnie konwertowany na typ `bool`.
- Wynik jest typu `bool`.

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia logicznego:

```
2 > 20 && 3 == 3
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia logicznego:

```
2 > 20 || 3 == 3
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia logicznego:

```
!(2 > 20 || 3 == 3)
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia:

```
2 > 0 + 3 == 0
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia:

```
3 >= 2 + 2 == 0
```

```
3 >= 2 + 2 == 2
```

```
3 >= 2 + 2 == 4
```

Operatory logiczne

Priorytet	Operator	Opis	Łączność
3	!	logiczna negacja (NOT)	prawostronna
14	&&	iloczyn logiczny (AND)	lewostronna
15	||	suma logiczna (OR)	lewostronna

Podaj wynik wyrażenia:

```
3 >= 2 + 2 == 0
```

```c
#include <stdio.h>

int main(){
    int x = 3 >= 2 + 2 == 0;
    printf("x = %i", x);
    return 0;
}
```

```
x = 1
```

```
3 >= 2 + 2 == 2
```

```c
#include <stdio.h>

int main(){
    int x = 3 >= 2 + 2 == 2;
    printf("x = %i", x);
    return 0;
}
```

```
x = 0
```

```
3 >= 2 + 2 == 4
```

```c
#include <stdio.h>

int main(){
    int x = 3 >= 2 + 2 == 4;
    printf("x = %i", x);
    return 0;
}
```

```
x = 0
```

Operatory bitowe

Priorytet	Operator	Opis	Łączność
3	~	bitowa negacja (NOT)	prawostronna
11	&	iloczyn bitowy (AND)	lewostronna
12	^	bitowa alternatywa wykluczajaca (XOR)	lewostronna
13	|	suma bitowa (OR)	lewostronna

```
# include <stdio.h>

int main () {
    char a = 0b00001010;          //a=42
    char b = 0b00001110;          //b=58

    printf("a = %d \t\t0b00001010\n",a);
    printf("b = %d \t\t0b00001110\n",b);
    printf("----------------------------\n");
    printf("~a = %d \t0b11110101\n", ~a);
    printf("a & b = %d \t0b00001010\n", a&b);
    printf("a | b = %d \t0b00001110\n", a|b);
    printf("a ^ b = %d \t0b00000100\n", a^b);
    return 0;
}
```

```
a = 10          0b00001010
b = 14          0b00001110
----------------------------
~a = -11        0b11110101
a & b = 10      0b00001010
a | b = 14      0b00001110
a ^ b = 4       0b00000100
```

Operator warunkowy

Priorytet	Operator	Opis	Łączność
16	x ? y : z	warunek trójskładnikowy	prawostronna

? : - jest to rodzaj instrukcji warunkowej typu **`if else`** w formie operatora.

```
x ? y : z
```

Jeśli **x** jest prawdziwe, to całe wyrażenie warunkowe ma wartość taką samą jak **y**.

Jeśli **x** jest fałszywe, to całe wyrażenie warunkowe otrzymuje wartość **z**.

- Operand **x** jest niejawnie konwertowany na typ **`bool`**.

Operator warunkowy

Priorytet	Operator	Opis	Łączność
16	x ? y : z	warunek trójskładnikowy	prawostronna

```c
#include <stdio.h>

int main()
{
    int x =30;

    x%2 ? printf( "Liczba nieparzysta x=%d", x) :  printf( "Liczba parzysta x=%d", x) ;
    //Wyświetli - Licbza parzysta x = 30

    int a = 4;
    int b = 3;
    int wieksza = a>b ? a : b;
    printf( "\nLiczba wieksza to %d", wieksza);
    //Wyświetli - Licbza wieksza to 4

    printf( "\n %d", 0?1:2);
    //Wyświetli - 2

    return 0;
}
```

Operator sizeof

Priorytet	Operator	Opis	Łączność
3	sizeof	rozmiar (w bajtach)	prawostronna

sizeof – operator zwracający rozmiar zmiennej lub typu w bajtach (**size_t**).

```
sizeof(nazwaZmiennej) ;
sizeof nazwaZmiennej ;

sizeof(typZmiennej) ;
```

Operator sizeof

Priorytet	Operator	Opis	Łączność
3	sizeof	rozmiar (w bajtach)	prawostronna

sizeof – operator zwracający rozmiar zmiennej lub typu w bajtach (**size_t**).

```
int main () {
    int x = 10;
    char c = 'a';
    printf("Size of x: %d \n", sizeof(x));
    printf("Size of c: %d \n", sizeof c);
    printf("Size of long double: %d \n", sizeof(long double));
    return 0;
}
```

```
Size of x: 4
Size of c: 1
Size of long double: 12
```

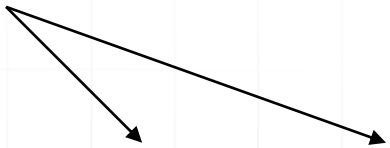
Wyrażenie vs. instrukcja

Wyrażenie to kombinacja różnych operatorów i operandów (argumentów) dająca wartość, którą można gdzieś przypisać, podstawić do funkcji, porównać itp...

```
x = 15
x + 10
z = x / y
(x>y)&&(x==y)
```

Instrukcja jest to polecenie przekazane do wykonania, może nią być m. in. każde wyrażenie zakończone średnikiem.

```
x = 15;
y = x + 10;
z = x / y;
```



```
if (.....)
```

```
..... ? x : y
```

Ale: instrukcja warunkowa wyrażenie warunkowe