Operatory arytmetyczne i logiczne
Instrukcje warunkowe

Piotr Zierhoffer

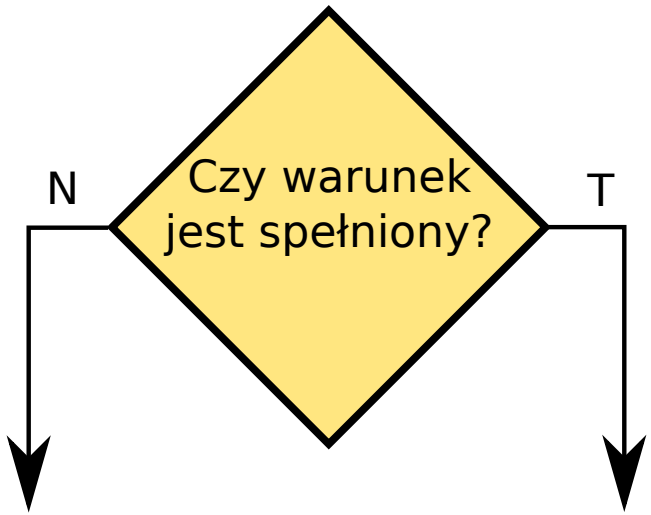
Institute of Computer Science
Poznań University of Technology

7 października 2012

O czym dzisiaj?

1. Podstawowe operatory to nie wszystko!
   - operatory logiczne
   - operatory bitowe
   - zaawansowane operatory arytmetyczne

2. Instrukcja warunkowa



Rozgrzewka!

Zadanie

Napisz program wczytujący trzy liczby i wypisujący na ekran ich różnicę!

Operatory

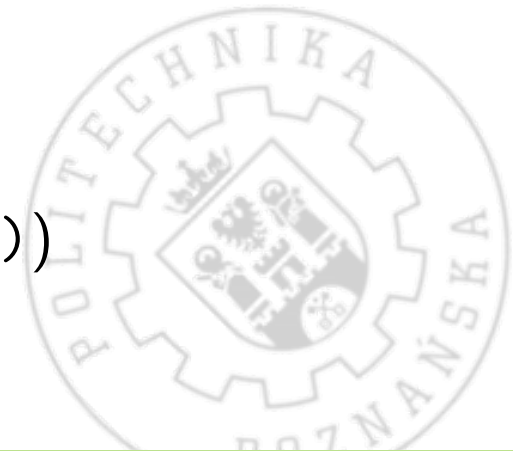
Wszystkie operatory mają określony priorytet.

W C jest 15 priorytetów dla 48(?) operatorów!

Jeżeli nie mamy pewności, używamy nawiasów!

Wyróżniamy operatory:

- unarne (`+`, `-`, `++`, `--`, `*`, `&`, `sizeof`, `...`)
- binarne (większość :) `+`, `-`, `*`, `/`, `&&`, `...`)
- operator trójargumentowy (`?:`)
- wywołania funkcji (`fun(arg1, arg2, ..., arg`$_n$`)`)



Operatory arytmetyczne

Inkrementacja i dekrementacja

`+, -, ++, --`

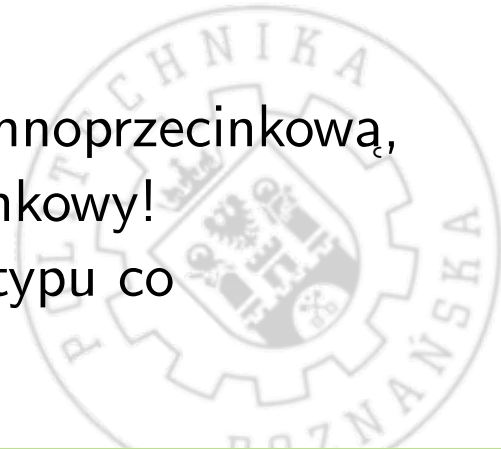
```
int a = 5;    //a == 5
int b = a++; //b == 5, a == 6
```

```
int a = 5;    //a == 5
int b = ++a; //b == 6, a == 6
```

Mnożenie, dzielenie, modulo

`*, /, %`

Uwaga! Aby operacja dzielenia zwróciła wartość zmiennoprzecinkową, przynajmniej jeden argument musi być zmiennoprzecinkowy!
Uwaga2! Wynik operacji nie musi mieć tego samego typu co argumenty (*promocja typów*)!



Operatory logiczne i porównania

W C nie występuje pojęcie fałszu i prawy. Jest `zero` i `nie-zero`. Każda wartość różna od zera uznawana jest za "prawdę".

Wyróżniamy następujące operatory logiczne:

- negacja `!` `( !5 == 0; !0 == 1; )`
- koniunkcja `&&` `( 5 && -7 == 1; 0 && 4 == 0; )`
- alternatywa `||` `( 5 || 0 == 1; 0 || 0 == 0 )`

Operatory `&&` i `||` są mądre!

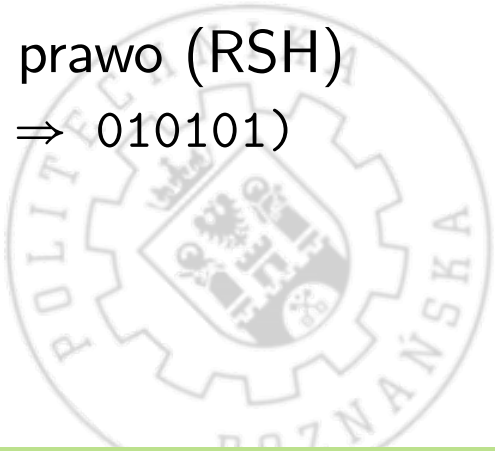
Operatory porównania

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

Operatory bitowe

Operatory bitowe służą do wykonywania operacji na poszczególnych bitach zmiennych.

- negacja (NOT)
  `(~010010 ⇒ 101101)`
- koniunkcja (AND)
  `(110010 & 101010 ⇒ 100010)`
- alternatywa (OR)
  `(110010 | 101010 ⇒ 111010)`
- alt. wykluczająca (XOR)
  `(110010^101011 ⇒ 011001)`
- przesunięcie w lewo (LSH)
  `(101010 << 1 ⇒ [1]010100)`
- przesunięcie w prawo (RSH)
  `(101010 >> 1 ⇒ 010101)`



Operatory złożone

Pozwalają na skrócenie zapisu.
`-=, +=, *=, /=, %=, &=, |=, ^=, <<=, >>=`

Przykład

$a \mathrel{+}= b \Leftrightarrow a = a + b$

$a \mathrel{<<}= 2 \Leftrightarrow a = a << 2$

Pozostałe operatory

Operatory wskaźnikowe

Operator pobrania adresu `&`, operator wyłuskania `*`.

Operator sizeof

Pozwala na sprawdzenie **rozmiaru** zmiennej — niezależnie od jej wartości!

```
sizeof(char) == 1; sizeof(int) == 4;
```

Przecinek

Rozdziela instrukcje, wyrażenie przyjmuje wartość ostatniej instrukcji.

```
int a = 3, 4, 5, 6; //a == 6
```
ten zapis jest naciągany...

Zadania

Zadanie 1

Jaki będzie zawartość zmiennych po wykonaniu kodu?

```
int a=2, b=3
int c, d;
c = (a++) + (b++);
d = (++a) + (++b);
```

Uwaga!

```
i = i++;
```

Wynik jest **niezdefiniowany**!

`http://en.wikipedia.org/wiki/Sequence_point`

Zadania

Zadanie 2

Wczytaj cztery liczby zmiennoprzecinkowe `a`, `b`, `c` i `d`.
Oblicz następujące wyrażenia:

$$\frac{ab}{b+a} \qquad \frac{a+b}{\frac{a}{d}-d}$$

$$\frac{a-b}{\frac{b}{c}+d}+\frac{1}{a} \qquad \frac{a+\frac{c}{d}}{\frac{b}{c}+d}-\frac{d+1}{1-\frac{a}{b}}\times(-c)$$

Zadanie domowe

Zadanie domowe 1

Napisz program, który na podstawie wysokości i średnicy podstawy stożka wyliczy jego objętość oraz pole powierzchni **całkowitej** (przyjmij stałą M_PI jako wartość liczby π. Należy dołączyć plik nagłówkowy `math.h`).

Zadanie domowe 2

Napisz program wypisujący tablice logiczne poniższych wyrażeń logicznych (jeżeli wyrażenie jest prawdziwe to napisz 1, jeżeli jest fałszywe to wypisz 0):

$$p \vee q \qquad p \wedge q$$
$$p \Rightarrow q \qquad p \Leftrightarrow q$$
$$\sim (p \wedge q) \vee (r \Rightarrow p)$$

Instrukcje warunkowe

Pozwalają na wykonanie kodu **pod warunkiem**.

Mamy dwa rodzaje instrukcji warunkowych:

- `if... else...`
- `switch...`

Instrukcja if

```
if( warunek )
  instrukcja1;
else
  instrukcja2;

if( a > 3 )
{
  b = 6;
  printf("Liczba␣jest␣wieksza␣od␣3!\n");
}
else
  b = -4;
```

Instrukcja switch

```
if(a==2) ...
else if(a==3) ...
else if(a ...
```

Można inaczej!

```
switch(a)
{
  case 1:
    printf("Jedyneczka\n");
  case 3:
    printf("Trójeczka\n");
  case 2:
    printf("Dwójeczka\n");
  default:
    printf("Inny numer!\n");
}
```

Instrukcja switch

```
switch(a)
{
  case 1:
    printf("Jedyneczka\n");
    break;
  case 3:
    printf("Trójeczka\n");
    break;
  case 2:
    printf("Dwójeczka\n");
    break;
  default:
    printf("Inny␣numer!\n");
    break;
}
```

Bonus: operator trójargumentowy

```
scanf("%d", &a);
char * b = a % 2 ? "nieparzysta" : "parzysta";
printf("Liczba␣%s\n", b);
```

albo jeszcze lepiej

```
scanf("%d", &a);
char * b = ((a % 2) ? "nieparzysta" : "parzysta");
printf("Liczba␣%s\n", b);
```

Co robi ten kod?

```
int a;
a = a > 0 ?  a :  -a;
```

Zadania

Zadanie

Wczytaj dwie liczby i wypisz je w kolejności niemalejącej.

Zadanie

Sprawdź, czy wczytana liczba jest parzysta za pomocą konstrukcji `if`, `switch` i `?:`.

Zadanie domowe

Zadanie domowe 3

Napisz program wyliczający rozwiązanie równania kwadratowego $ax^2 + bx + c = 0$. Pamiętaj o uwzględnieniu wypadków brzegowych!

Uwaga!

Zadania domowe muszą zawierać elementy interfejsu użytkownika, tj. użytkownik musi zostać poproszony o wprowadzenie danych (i musi wiedzieć np. którą zmienną właśnie podaje) a wynik musi być opatrzony opisem.