# Wyrażenia

**Wyrażenie jest ciągiem operatorów i argumentów** (np. stałych lub zmiennych), który określa sekwencję obliczeń prowadzącą do uzyskania wartości wynikowej. W wyrażeniu mogą występować pary nawiasów ( ) wskazujące kolejność wykonywania działań przy obliczaniu wartości tego wyrażenia.

**Stałe wyrażenie** jest wyrażeniem, którego wartość jest znana na etapie kompilacji.

Wartość wyrażenia wyliczana jest poprzez wykonanie operacji określonych przez występujące w wyrażeniu operatory. Istotną rolę odgrywa przy tym **priorytet** i **wiązanie** operatora.

**Priorytet** operatorów wyznacza kolejność, w jakiej wykonywane są poszczególne operacje:

- *Operacja o wyższym priorytecie wykonywana jest przed operacją o niższym priorytecie.*
- *W przypadku operacji o jednakowych priorytetach o kolejności ich wykonania decyduje wiązanie operatora.*

**Wiązanie** operatora określa kolejność wykonania operacji w przypadku jednakowych priorytetów operacji w wyrażeniu:

- *Wiązanie **lewostronne** - operacje wykonywane są w wyrażeniu w kierunku od strony lewej do prawej.*
- *Wiązanie **prawostronne** - operacje wykonywane są w wyrażeniu w kierunku od strony prawej do lewej.*

**Nawiasy ( )** wymuszają inną kolejność wykonywania oparacji:

*Operacje zawarte w parze nawiasów wykonywane są przed operacjami występującymi poza nawiasami.*

# Operatory języka C

## *Operatory arytmetyczne:*

**+** dodawanie i jednoargumentowy plus

**-** odejmowanie i jednoargumentowa zmiana znaku

***** mnożenie

**/** dzielenie (dla typów całkowitych obcięcie części ułamkowej)

**%** modulo

**Priorytety i wiązania operatorów arytmetycznych:**

| Priorytet | Operator | Wiązanie | Nazwa operatora |
|---|---|---|---|
| **2** | **+** | Prawostronne | Jednoargumentowy plus |
| | **-** | Prawostronne | Jednoargumentowy minus |
| **3** | ***** | Lewostronne | Mnożenie |
| | / | Lewostronne | Dzielenie |
| | **%** | Lewostronne | Modulo |
| **4** | + | Lewostronne | Dodawanie |
| | - | Lewostronne | Odejmowanie |

**Uwaga: 2 to priorytet wyższy niż 3; 3 to priorytet wyższy niż 4.**

## L-wartość

Pojęcie l-wartości dotyczy wyrażenia reprezentującego obszar w pamięci, do którego można wpisać pewną wartość. Typowym przykładem l-wartości jest lewa strona operatora przypisania.

```
int x=3,k;
k=x*(x-1);          //l-wartosc: zmienna k
```

## *Operatory inkrementacji (zwiększania) i dekrementacji (zmniejszania):*

**++** inkrementacja

**--** dekrementacja

Argumentem operatora inkrementacji i dekrementacji może być wyłącznie wyrażenie stanowiące **l-wartość** typu arytmetycznego lub wskaźnikowego. W efekcie wykonania operacji inkrementacji następuje zwiększenie o 1 wartości argumentu, a wykonanie operacji dekrementacji powoduje zmniejszenie wartości argumentu o 1.

Operatory inkrementacji i dekrementacji występują w formie ***prefiksowej*** **++k, --k** lub ***postfiksowej*** **k++, k--**. Operator **++** oraz **--** wraz ze swoim argumentem przedstawia wyrażenie, którego wartość równa jest:

- wartości wyrażenia **przed modyfikacją** dla operacji postfiksowej,
- wartości wyrażenia **po modyfikacji** dla operacji prefiksowej,

czyli:

- dla formy postfiksowej (**k++, k--**) **najpierw** pierwotna wartość k jest użyta w kolejnej operacji, w której argumentem jest (**k++),lub (k--)**, a potem wartość k jest zwiększana (zmniejszana) o 1,
- dla formy prefiksowej (++**k, --k**) **najpierw** wartość k jest zwiększana (zmniejszana) o 1, a potem tak zmodyfikowana wartość k jest użyta w kolejnej operacji, w której argumentem jest (**k++),lub (k--).**

Przykład:

Jeżeli **k** równa się 3 to

**x = k++;** nadaje zmiennej **x** wartość 3

**x = ++k;** nadaje zmiennej **x** wartość 4

a w obu powyższych przypadkach wartością **k** stanie się 4.

**Uwaga:** Operacje inkrementacji i dekrementacji wykonywane na wskaźnikach podlegają zasadom ***arytmetyki wskaźników***.

**Priorytety i wiązania operatorów:**

| Priorytet | Operator | Wiązanie | Nazwa operatora |
|---|---|---|---|
| **2** | **++** | Prawostronne | Inkrementacja |
| | **--** | Prawostronne | Dekrementacja |

**Zinterpretuj elementy kodu źródłowego następującego programu:**

```
#include <stdio.h>
int main()
{
  int a=7;
  int b=7;
  int c=7;
  int d=7;
  printf("\n  a=%d,   b=%d, c  =%d, d  =%d",a,b,c,d);
  printf("\n++a=%d, --b=%d, c++=%d, d--=%d",++a,--b,c++,d--);
  printf("\n  a=%d,   b=%d, c  =%d, d  =%d",a,b,c,d);
  getch(); return 0;
}
```

```
  a=7,   b=7, c  =7, d  =7
++a=8, --b=6, c++=7, d--=7
  a=8,   b=6, c  =8, d  =6
```

## *Operatory przypisania*

Lewym argumentem operatora przypisania **=** musi być l-wartość, a prawym argumentem może być wyrażenie odpowiedniego typu, którego wartość po obliczeniu przypisywana jest lewemu argumentowi.

**Uwaga:**

> **Przypisanie nie jest instrukcją lecz wyrażeniem. Wartością tego wyrażenia jest wartość przypisywana lewemu argumentowi operatora przypisania.**

**Zadanie:** uruchom następujący program, zinterpretuj elementy kodu źródłowego:

```
#include <stdio.h>
int main()
{
    int a,b,c,d;
    a=(b=5+(c=(d=8)+3))*2;
    printf("\na=%d, b=%d, c=%d, d=%d",a,b,c,d);
    getch();
}
```

```
a=32, b=16, c=11, d=8
```

Oprócz standardowego operatora przypisania występują następujące złożone operatory przypisania:

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

Ogólnie wyrażenie z operatorem przypisania:

**x opr= y**

jest równoważne wyrażeniu:

**x = x opr y**

gdzie **opr** oznacza jeden z wymienionych wcześniej złożonych operatorów przypisania.

```
k+=x;          //rownowazne k=k+x;
n/=y;          //rownowazne n=n/y;
f*=f*=f;       //rownowazne f=f*f*f*f;
a+=b+=c;       //rownowazne a=a+b+c; b=b+c;
```

**Priorytety i wiązania operatorów przypisania:**

| Priorytet | Operator | Wiązanie | Nazwa operatora |
|---|---|---|---|
| **14** | = | Prawostronne | Przypisania |
| | += | Prawostronne | Złożony przypisania |
| | -= | Prawostronne | Złożony przypisania |
| | *= | Prawostronne | Złożony przypisania |
| | /= | Prawostronne | Złożony przypisania |
| | %= | Prawostronne | Złożony przypisania |
| | &= | Prawostronne | Złożony przypisania |
| | \|= | Prawostronne | Złożony przypisania |
| | <<= | Prawostronne | Złożony przypisania |
| | >>= | Prawostronne | Złożony przypisania |
| | ^= | Prawostronne | Złożony przypisania |

**Zadanie:** uruchom następujący program, zinterpretuj elementy kodu źródłowego:

```
#include <stdio.h>
int main()
{
  int a=5,b=5;
  printf("\n%3d%3d",a,b);
  a-=3;
  b=-3;
  printf("\n%3d%3d\n",a,b);
  getch();
}
```

```
  5   5

  2  -3
```

## *Operatory relacji*

Argumentami operatorów relacji mogą być wyrażenia arytmetyczne lub wskaźnikowe. Do grupy operatorów relacji należą cztery operatory:

**<** **mniejszy**

**<=** **mniejszy lub równy**

**>** **większy**

**>=** **większy lub równy**

Operatory relacji badają, czy spełniona jest odpowiednia relacja pomiędzy argumentami. **W języku C nie występują pojęcia "prawda", "fałsz".**

Wartość wyrażenia relacyjnego:

***wyr1 rel wyr2***

gdzie ***wyr1*** i ***wyr2*** są wyrażeniami i ***rel*** jest operatorem relacji, **jest typu `int`** i równa się:

**1** jeżeli relacja jest spełniona,

**0** jeżeli relacja nie jest spełniona.

**Uwaga:** Wyrażenie relacyjne daje wynik liczbowy, który można wykorzystać w dalszych obliczeniach.

```
(5>3)+2*(7<=4)-(6>=6)*3       //wartosc -2

4<6<8                         //wartosc  1

8>6>4                         //wartosc  0
```

## *Operatory równości*

Argumentami operatorów równości mogą być wyrażenia arytmetyczne lub wskaźnikowe. Grupę operatorów równości tworzą operatory:

**==** **równy**

**!=** **różny**

Dla dwóch wyrażeń ***wyr1*** i ***wyr2*** , wyrażenie ***wyr1 == wyr2*** przyjmuje wartości liczbowe typu **`int:`**

**1** jeżeli ***wyr1*** i ***wyr2*** mają **jednakowe** wartości,

**0** w przeciwnym przypadku.

Dla dwóch wyrażeń ***wyr1*** i ***wyr2*** , wyrażenie ***wyr1*** **!=** ***wyr2*** przyjmuje wartości liczbowe typu **`int:`**

**1** jeżeli ***wyr1*** i ***wyr2*** mają **różne** wartości,

**0** w przeciwnym przypadku.

## *Operatory logiczne*

Do budowania złożonych wyrażeń relacyjnych wykorzystywane są operatory logiczne:

**!** **negacji**

**&&** **koniunkcji**

**||** **alternatywy**

Argumentami operatorów logicznych mogą być wyrażenia arytmetyczne.

Wartość wyrażenia **!a** jest równa:

**1** jeżeli **a** równa się **0**,

**0** w pozostałych przypadkach.

Wartość wyrażenia **a && b** jest równa:

**1** jeżeli **a** jest różne od **0** i **b** jest różne od **0**,

**0** w pozostałych przypadkach.

Wartość wyrażenia **a || b** jest równa:

**0** jeżeli **a** jest równe **0** i **b** jest równe **0**,

**1** w pozostałych przypadkach.

**Uwaga:** Obliczanie wartości wyrażenia logicznego może zakończyć się przed jego pełnym przeglądnięciem. W momencie, gdy można jednoznacznie określić wartość wyrażenia, przerywane są dalsze obliczenia:

```
int a,b=4;
a = b>6 && b++<12;  //Wartosc wyrazenia b>6 jest rowna 0, czyli
                    //wartosc calego wyrazenia logicznego jest
                    //rowna 0. Dalsza czesc wyrazenia logiczneg
                    //nie jest juz obliczana. Operacja b++ nie
                    //jest wykonywana. Wartosc b wynosi 4.


int a,b=8;
a = b>6 && b++<12;  //Wartosc wyrazenia b>6 jest rowna 1, czyli
                    //nie mozna jeszcze ustalic wartosci calego
                    //wyrazenia logicznego. Musi byc obliczona
                    //pozostala czesc wyrazenia logicznego.
                    //Jest wykonywana operacja b++. Wartosc b
                         //wynosi 9.
```

## *Operator warunku*

Jedyny występujący w języku C operator trójargumentowy

**? :**

może zostać użyty w następującej formie:

*wyr1* **?** *wyr2* **:** *wyr3*

gdzie *wyr1* musi być wyrażeniem typu arytmetycznego lub wskaźnikowego, a *wyr2* i *wyr3* mogą być wyrażeniami dowolnego, ale jednakowego typu. Operator warunku jako wynik daje:

- Wartość wyrażenia ***wyr2***, jeżeli wyrażenie *wyr1* ma wartość różną od **0**. W tym przypadku nie jest obliczana wartość wyrażenia *wyr3*.
- Wartość wyrażenia ***wyr3***, jeżeli wyrażenie *wyr1* ma wartość równą **0**. W tym przypadku nie jest obliczana wartość wyrażenia *wyr2*.

**Uwaga:** Każdorazowo obliczane są wartości jedynie dwóch z trzech wyrażeń stanowiących argumenty operatora warunku.

Operator warunku w wygodny sposób może zastąpić instrukcję warunkową **if**.

**Zadanie.** uruchom program, zinterpretuj elementy kodu źródłowego:

```
#include <stdio.h>
int main()
{
    int x,y;
    printf("podaj dwie rozne liczby calkowite: ");
    scanf("%d %d",&x,&y);
    printf("wieksza jest %d, a mniejsza %d",x>y?x:y,x<y?x:y);
    getch(); return 0;
}
```

**Priorytety operatorów:**

| Priorytet | Operator | Wiązanie | Nazwa operatora |
|---|---|---|---|
| **2** | **+** | Prawostronne | Jednoargumentowy plus |
| | **-** | Prawostronne | Jednoargumentowy minus |
| | **!** | Prawostronne | Logiczna negacja |
| | **++** | Prawostronne | Inkrementacja |
| | **- -** | Prawostronne | Dekrementacja |
| **3** | ***** | Lewostronne | Mnożenie |
| | / | Lewostronne | Dzielenie |
| | **%** | Lewostronne | Modulo |
| **4** | + | Lewostronne | Dodawanie |
| | - | Lewostronne | Odejmowanie |
| **6** | < | Lewostronne | Relacja mniejszy |
| | <= | Lewostronne | Relacja mniejszy lub równy |
| | > | Lewostronne | Relacja większy |
| | >= | Lewostronne | Relacja większy lub równy |
| **7** | == | Lewostronne | Równy |
| | != | Lewostronne | Różny |
| **11** | && | Lewostronne | Logiczna koniunkcja |
| **12** | \| \| | Lewostronne | Logiczna alternatywa |
| **13** | ? **:** | Prawostronne | Warunek |
| **14** | = | Prawostronne | Przypisania |
| | += | Prawostronne | Złożony przypisania |
| | -= | Prawostronne | Złożony przypisania |
| | *= | Prawostronne | Złożony przypisania |
| | /= | Prawostronne | Złożony przypisania |
| | %= | Prawostronne | Złożony przypisania |
| | &= | Prawostronne | Złożony przypisania |
| | \|= | Prawostronne | Złożony przypisania |
| | <<= | Prawostronne | Złożony przypisania |
| | >>= | Prawostronne | Złożony przypisania |
| | ^= | Prawostronne | Złożony przypisania |

**Uwaga:** Wiązanie prawostronne mają jedynie operatory: jednoargumentowe, warunku i przypisania. Pozostałe operatory mają wiązanie lewostronne.

*Pozostałe rodzaje operatorów:*
*- bitowe,*
*- sizeof,*
*- adresu,*
*- wyłuskania,*
*- składowej,*
*- indeksowania,*
*- wywołania funkcji*
*- rzutowania,*
*- połączenia*
*zostaną omówione w dalszej części kursu.*

# Instrukcje

Moduł źródłowy zawiera **instrukcje** pogrupowane w bloki funkcyjne. W bloku funkcyjnym instrukcje wykonywane są sekwencyjnie w kolejności ich zapisania w module źródłowym.

**Etykieta** jest identyfikatorem, po którym następuje dwukropek. Każda instrukcja w programie może być opatrzona etykietą umieszczoną bezpośrednio przed tą instrukcją. Etykiety służą do wyróżniania pewnych instrukcji w celu wskazania ich w innych instrukcjach programu (**`switch`**, **`goto`**).

## *Instrukcja wyrażeniowa*

**wyrażenie ;**

Dowolne wyrażenie zakończone średnikiem staje się instrukcją. Jej wykonanie polega na wyznaczeniu wartości wyrażenia. Uzyskana wartość jest pomijana.

**Uwaga:** W przypadku instrukcji wyrażeniowej główną korzyścią z jej wykonania są jej efekty uboczne.

Zadania:

1. Określ wartość wyrażeń:

   3+14%5*2-7/2

   3+14%5*2-7/+2

   3+14%-5*-2-7/2

   3+14%(5*(2-7))/2

   3+14%5*2-7./2

   3+14.%5*2-7/2

   3+14%5*2-7/2e0

   3+14%5*2.0-7/2e-3

2. Określ wartość zmiennej całkowitej **a** po wykonaniu sekwencji instrukcji, uzasadnij odpowiedź:

   a=2;

   a+=a*=a+=++a;

3. Dla jakiej wartości zmiennej całkowitej a wyrażenie a%=++a;

   wygeneruje błąd? Jaka będzie wartość wyrażenia w pozostałych przypadkach?

4. **Zadanie C03**. Napisz program obliczający i wyświetlający wartość *y* (*a* jest liczbą rzeczywistą podawaną z klawiatury przez użytkownika):

$$y = \cfrac{1}{a^2 + \cfrac{1}{a^2 + \cfrac{1}{a^2 + 1}}}$$

Tekst źródłowy programu, oraz (poniżej tekstu) wklejone okno wyników wykonania programu (dla wybranej wartości a) umieść w pliku C03.doc (na jednej stronie) i prześlij przez Moodle w wyznaczonym terminie (wymagany format programu Word).