Wykład III - zmienne, operatory

niedziela, 18 marca 2018 09:15

Inicjalizacja zmiennych

Nazwa_typu identyfikator = wartość_początkowa;

Nazwa_typu ident1 = wart_poczatk1, ident2 = wart_poczatk2

Przykład:
int a=2, b=0 <-> int a=2; int b=0

Miejsce deklarowania zmiennej

Zalecenie: na początku programu, o ile to możliwe (na początku bloku instrukcji).

Zakres ważności zmiennej

/* Zakres waznosci zmiennych */

```
#include <stdio.h>
int Integer;
char aCharacter;
unsigned int LiczbaSynow;

int main ()
{
        unsigned short Wiek;
        float Aliczba, Inny;
        printf("Wprowadz swój wiek: ");
        scanf("%u", &Wiek);
        return 0;
}
```

zmienne globalne

zmienne lokalne

instrukcje

Stałe

1. **Literał -** stała, której postać jest ściśle określona (stała dosłowna). Nie jest definiowany w programie.
   Podział literałów:
   - liczby całkowite
   - liczby rzeczywiste
2. **Liczby całkowite**
   a. dopuszczalne zapisy:
      i. dziesiętny
      ii. ósemkowy (oktalny)
      iii. szesnastkowy (heXadecymalny)
   b. przykłady:
      i. 75 /*dziesietny*/
      ii. 0113 /*ósemkowy*/
         □ cyfra zero (0)
      iii. 0x4b /*szesnastkowy*/
         □ znaki zero i x (0x)
3. **Liczby rzeczywiste**
   a. zapis z kropką dziesiętną:
      ▪ 3.14159
      ▪ 3.0
   b. zapis w notacji wykładniczej
      ▪ 6.02e23
      ▪ 1.6e-19 (1.6 x 10^(-19))
4. **Przyrostki wymuszania typu stałej**

Typ	Przyrostek	Przykład

kolos

unsigned int	u	65u
long	l, L	65L
unsigned long	ul, UL	65UL
long long	ll, LL	65LL
unsigned long long	ull, ULL	65ULL

5. **Znak**
   a. zapis znaku: 'a'
   b. Kody escape

\n	newline
\r	carriage return
\t	tabulation
\v	vertical tabulation
\b	backspace
\f	page feed
\a	alert (beep)
\'	pojedynczy apostrof (')
\"	podwójny apostrof (")
\?	znak zapytania (?)
\\	ukośnik pochylony w lewo (\)

   c. przykłady:
      - '\n'
      - '\t'
      - '\"'
      - '\\'
6. **Łańcuchy znaków**
   a. Zapis łańcucha znaków:
      - "Halo Ziemianie!"
      - "Left \t Right"
      - "one\ntwo\nthree"
      - "lancuch zapisany w \
        dwu liniach"
   b. **Uwaga:** kilka łańcuchów znaków jest łączonych ze sobą, o ile dzielą je tylko białe znaki:
      "tworzymy" " jeden " "lancuch " "znakow" <-> "tworzymy jeden lancuch znakow"

Deklarowanie stałych

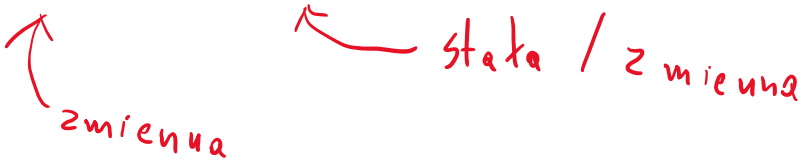
Modyfikator const:
const *nazwa_typu* identyfikator = wartość
const int width = 100;
const char tab = '\t';
const zip = 12440;

Uwaga: jeżeli w deklaracji nie zostanie podany typ, to kompilator przyjmie, że jest to typ int.

Operatory

Operator przypisania (=)

Identyfikator = wyrażenie



a = b;

int a, b;	/* a:?	b:? */
a = 10;	/* a:10	b:? */
b = 4;	/* a:10	b:4 */
a = b;	/* a:4	b:4 */
b = 7;	/* a:4	b:7 */

a = 2 + (b = 5)	b=5 a=2 + b;

a = b = c = 5;	a = 5;
	b = 5;
	c = 5;

Przykładowy program:

```
/* przypisywanie wartosci */
#include <stdio.h>

in
t main ()
{
        int y;
        int a, b, c, x = 6;

        printf(" Wartosc zmiennej a: %i \n", a);
        printf(" Wartosc zmiennej b: %i \n", b);
        printf(" Wartosc zmiennej c: %i \n", c);
        printf(" Wartosc zmiennej x: %i \n\n", x);

        a= 2 + (b = 5);
        printf(" Wartosc zmiennej a: %i \n", a);
        printf(" Wartosc zmiennej b: %i \n\n", b);

        a = b = c = 9;
        printf(" Wartosc zmiennej a: %i \n", a);
        printf(" Wartosc zmiennej b: %i \n", b);
        printf(" Wartosc zmiennej c: %i \n\n", c);

        printf("Wartosc wyrazenia przypisania: %i \n", y = a);

        return 0;
}
```

Obraz na ekranie:

```
        Wartosc zmiennej a: 4206592
        Wartosc zmiennej b: 4198592
        Wartosc zmiennej c: 7601688
        Wartosc zmiennej x: 6

Wartosc zmiennej a: 7
Wartosc zmiennej b: 5
```

Wartosc zmiennej a: 9
Wartosc zmiennej b: 9
Wartosc zmiennej c: 9

Wartosc wyrazenia przypisania: 9

Komentarze:

Pierwsze ciągi liczb (wartości zmiennych) są skutkiem wykorzystania danych, które zostały zapisane w pamięci wcześniej (w programie zdefiniowano na początku tylko zmienną x, reszta jest wczytywana z pamięci).

Operatory arytmetyczne

%	modulo (reszta z dzielenia)
/	dzielenie
*	mnożenie
-	odejmowanie
+	dodawanie

Złożone operatory przypisania:

%=	a %=b;	a = a % b;
/=	a /= b+1;	a = a/(b+1);
*=	a *= b;	a = a*b;
-=	a -= b;	a = a - b;
+=	a += b;	a = a + b;

Operator inkrementacji

++	operator jednoargumentowy

Forma końcówkowa (postfix):

a ++;	a = a + 1;
a ++	a += 1;

```
b=3;
a = b++;
/* a jest 3, b jest 4*/
```

Forma przedrostkowa (prefix):

++a;	a = a + 1;
++a;	a += 1;

```
b = 3;
a = ++b;
/* a jest 4, b jest 4 */
```

Operator dekrementacji

--	operator jednoargumentowy

Forma końcówkowa (postfix):

a --;	a = a - 1;

a --;	a -= 1;

Forma przedrostkowa (prefix):

--a;	a = a - 1;
--a;	a -= 1;

Operatory relacji

<=	mniejszy lub równy
>=	większy lub równy
<	mniejszy niż
>	większy niż
!=	różny od
==	równy

Przykłady

Wyrażenie	**Wartość**
(7 == 5)	false
(5>4)	true
(3!=2)	true
(6>=6)	true
(5<5)	false

Niech a=2, b=3, c=6

Wyrażenie	**Wartość**	**Uzasadnienie**
(a == 5)	false	
(a*b >= c)	true	(2*3>=6)
(b+4 > a*c)	false	(3+4 > 2*6)
((b=2)==a)	true	

Operatory logiczne

&&	iloczyn logiczny (AND)
||	suma logiczna (OR)
!	negacja (NOT)

Przykłady

Wyrażenie	**Wartość**
!(5==5)	false
!(6<=4)	true
!true	false
!false	true

Uwaga!

Argument 1 a	Argument 2 b	a||b	a&&b

Kolejność wykonywania operacji logicznych

- Wyrażenia logiczne są wykonywane w kierunku od lewej do prawej.
- Obliczenia wyrażeń logicznych są przerywane, jeżeli w ich trakcie okaże się, że wyrażenie będzie fałszywe.

Przykład:

x != 0 && 20/x < 5;	20/x jest obliczane tylko dla x różnego od 0.

Operator warunkowy "?"

Wyrażenie warunkowe
warunek ? wartość1 : wartość2

Przykłady

Wyrażenie	**Wartość**
7 == 5 ? 4 : 3	3
7 == 5 + 2 ? 4 : 3	4
5 > 3 ? a : b	a
a > b ? a : b	większa wartość z a,b

Przykład wykorzystania:
c = (x>y) ? 5 : 12;

Operatory bitowe

Przykład:
int m = 0x0f0f;
int k = 0x0ff0;

int a, b, c, d;

~ negacja bitowa

a	~a
0	1
1	0

Przykład:
d = ~m;
m 0000 1111 0000 1111
d 1111 0000 1111 0000

^ bitowa różnica symetryczna

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

Przykład:
c = m ^ k;
m 0000 1111 0000 1111
k 0000 1111 1111 0000
c 0000 0000 1111 1111

& iloczyn bitowy

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

Przykład:
c = m & k;
m 0000 1111 0000 1111
k 0000 1111 1111 0000
c 0000 1111 0000 0000

| suma bitowa

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

Przykład
m 0000 1111 0000 1111
k 0000 1111 1111 0000
c 0000 1111 1111 1111

<< przesunięcie w lewo
zmienna << ile_miejsc

Przykład:
int z = 0x30f1;
int w;

w = z << 2;

z 0011 0000 1111 0001
w 1100 0011 1100 0100

>> przesunięcie w prawo
zmienna >> ile_miejsc

Przykład:
int z = 0x30f1;
int w;

w - z >> 2;

z 0011 0000 1111 0001
w 0000 1100 0011 1100

Przykład 2:
signed int z = 0xf300;
signed int w;
w = z >> 2;
z 1111 0011 0000 0000
w 0011 1100 1100 0000
w' 1111 1100 1100 0000

Operatory bitowe: złożone operatory przypisania

<<=	a <<= b;	a = a << b;
>>=	a >>= b;	a = a >> b;
^=	a ^=	

Operator sizeof ()

Zapis:

sizeof (*nazwa typu*)	a = sizeof(char); /* a: 1 */
sizeof(*nazwa obiektu*)	short zmienna; a = sizeof(zmienna); /* a: 2 */

Wartością otrzymywaną w wyniku zastosowania operatora sizeof() jest stała, która jest określana przed rozpoczęciem programu.

Operator rzutowania

(operator jawnego przekształcenia typu)

Operator rzutowania służy do zmiany typu danej na inny.

(*nazwa typu*) *obiekt*	int i; float f = 3.14; i = (int) f; /* i: 3 */
nazwa typu (obiekt)	int a = 0x10ff; char b; b = char (a); /* b: 0xff */

Priorytet operatorów

Priorytet	Operator	Opis	Łączność
1	::	zakres	L
2	() [] -> . sizeof		L
3	++ --	inkrementacja / dekrementacja	R
3	~		

Łączność operatorów

Łączność operatora jednoargumentowego określa stronę, po której znajduje się argument operacji wskazanej przez operator.

Łączność operatora dwuargumentowego określa kolejność wykonywania obliczeń w wyrażeniu, w którym występują operatory o tym samym priorytecie.