Instrukcje, instrukcje warunkowe, pętle

1

Instrukcja

Instrukcja jest to definicja obliczenia i określenie sposobu wykonania tego obliczenia.

Program jest ciągiem instrukcji wykonywanych kolejno od pierwszej do ostatniej.

Instrukcje są tłumaczone na język wewnętrzny realizujący obliczenia zdefiniowane tą instrukcją.

2

Instrukcja prosta

```
int   m,  n = 1;   // deklaracja zmiennych

;                  // instrukcja pusta

m = n * n - 1;     // zmiana wartości m

n++;               // zmiana wartości n

m + n;             // bez rezultatu
```

3

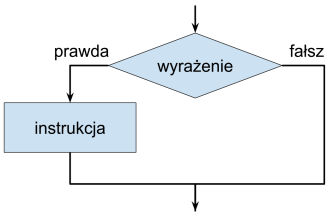
Instrukcja złożona

```
                { .... }

float p, q;
{
   p = 3.5;
   q = 7.1 + p++ ;
}
{p = q; q = 1;} // średnik przed klamrą }
                // jest wymagany
```

4

Instrukcja warunkowa

```
if   ( wyrażenie )   instrukcja
```

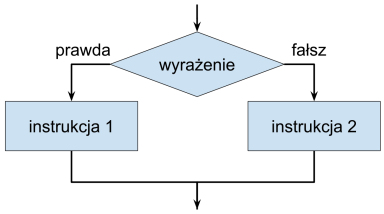


5

```
long   k, m;
char  flaga;

if (k > m)   flaga = 0;

if (k < m)
{
   flaga = 1;
   k = m - k;
}

if (m == 1)           // podwójne ==
{
   if ( k )           // k != 0
      flaga = 2;
   if ( !k )          // k == 0
      flaga = 3;
}
```

6

```
if   ( wyrażenie )
   instrukcja_1
else
   instrukcja_2
```



7

```
int  i,  f;
if (i > 5)  f = 3;  else  --f ;
/* średnik jest wymagany przed else */


double   ma,  winien,  saldo,  debet;
if (ma > winien)
{
   saldo = ma - winien;
   debet = -1;
}
else
{
   saldo = -1;
   debet = winien - ma;
}
```

8

9

```
if   (a)   if   (b)   c;   else   d;
   /* jest równoważne*/
if   (a) { if   (b)   c;   else   d; }
```

```
if (a) if (b) c; else d; else if (e) f; else g;
   /* jest równoważne*/
if (a)
{
   if (b) c;
   else d;
}
else
{
   if (e) f;
   else g;
}
```

10

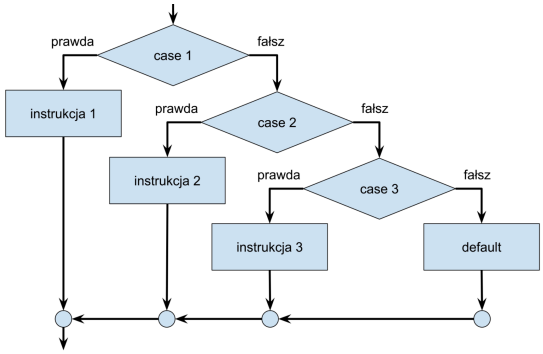
Instrukcja wyboru

```
switch  ( wyrażenie )
{
   case  stała_1: instrukcja_1;
                  break;
   case  stała_2: instrukcja_2;
                  break;
   . . .

   case  stała_n: instrukcja_n;
                  break;
   default:       instrukcja_domyślna;
}
```

Instrukcja może być instrukcją prostą, instrukcją złożoną lub ciągiem instrukcji prostych.

break, default - opcjonalne

11



12

```
int   ile_a,  ile_b,  ile_xy,  nieznany;
char  zn;

switch (zn)
{
   case 'a' :  ++ile_a;   break;

   case 'b' :  ++ile_b;   break;

   case 'x' :

   case 'y' :  ++ile_xy;  break;

   default  :  ++nieznany;
}
```

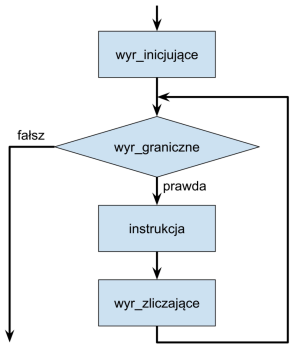
Pętla for

```
for (
      wyr_inicjujące;
      wyr_graniczne;
      wyr_zliczające
    )
{
   instrukcja;
}
```

Instrukcja może być instrukcją prostą, instrukcją złożoną lub ciągiem instrukcji prostych.

Zakończenie wykonywania pętli następuje gdy wyrażenie graniczne przyjmie wartość false.

13



14

```
int  s = 0;
int  i;

for  ( i = 0;  i <= 9;  ++i )  s += i;
```

```
int  i, k = 1525 ;
long m ;

for  ( i = k, m = 0;  i > 0; i -= 3 )
{
   if (i % 2)  ++m ;
}

/* wyrażenie inicjujące może mieć kilka przypisań */
```

15

```
bool dalej = true; // typ bool - standard C99, kompilator
                   // uruchamiamy z parametrem -std=c99
int gdzie;
int i;

for ( i = 0; i < N && dalej; ++i )
{
   ........

   if ( .... )
   {
      dalej = false;
   }
   else
   {
      ........
   }

   ........

}

gdzie = i;
```

16

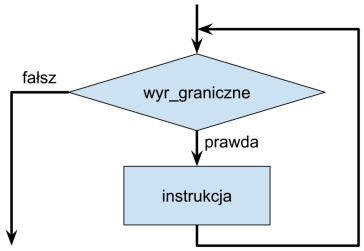
17

```
bool dalej = true;
int i,  gdzie = -1;
const int N = 10, szukany = 333;
int tab[N] = {0, 1, 333};

for (i = 0; i < N && dalej; ++i)
{
   if ( tab[i] == szukany )
   {
      dalej = false;
   }
}

gdzie = i;   // jaka jest wartość 'gdzie'?
```

18

Pętla while

```
while ( wyr_graniczne )
{
   instrukcja;
}
```



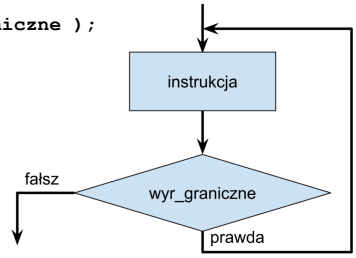
19

```
float   suma = 1573.821,   skladnik = 3.51;
int   licznik = 0;


while (suma > 1E-10)
{
   suma -= skladnik;

   skladnik *= skladnik;

   ++licznik;
}
```

20

Pętla do-while

```
do
{
   instrukcja;
}while ( wyr_graniczne );
```



21

```
long   ab = 3,  cd = 2;

do
{
       ab *= ab;

       cd += cd;

} while  (ab < cd);

/* ab == 9  cd == 4 */
```

22

Pytania

```
int s = 0, i, n;
for ( n = 0; n < 10; ++i )
{
     s += i;   // jaką wartość przyjmie s?
}


float  A = 3.485e2,  eps = 1.38534e-2;
long  k;
while (A != 0)
{
    A -= eps;
    ++k;
}   // ???????


unsigned char  k = 5;
do
{
   k -= 2;
} while (k != 0);   // ???????
```