Temat: Pętle for w języku C

Operatory przypisania:

- Proste (=)
  - a = b;

- Złożone (+=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=):
  - a += b; //a = a + b;
  - a -= b; //a = a - b;
  - a *= b; //a = a * b;
  - a /= b; //a = a / b;
  - a %= b; //a = a % b;

Inkrementacja oraz dekrementacja:

Operatory inkrementacji oraz dekrementacji upraszczają pisane kodu – staje się on krótszy oraz bardziej czytelny. Zamiast stosować **i = i+1;** lub **i = i-1;** stosuje się (szczególnie w pętlach) jedne z poniższych operatorów:

- Przedrostkowe
  - --a predekrementacja
  - ++a preinkrementacja
- Przyrostkowe
  - a-- postdekrementacja
  - a++ postinkrementacja

Pętla for:

Konstrukcja:

- `for (wyrażenie1; warunek; wyrażenie2) instrukcja;`

Jeżeli po **for** jest tylko jedna instrukcja, to można pominąć **{ }**

-
```
for (wyrażenie1; warunek; wyrażenie2) {
  instrukcja;
  instrukcja;
  …
  instrukcja;
}
```

- **Wyrażenie1** w tym miejscu nadaje się początkową wartość licznikowi pętli np. **i=1**. Instrukcja ta wykonuje się tylko raz, przed wykonaniem pierwszego cyklu pętli.
- **Warunek** kończący pętlę for. Pętla wykonuje się dopóki warunek zwraca wartość TRUE. Warunek jest sprawdzany na początku każdego cyklu pętli.
- **Wyrażenie2** w tym miejscu zwiększamy lub zmniejszamy licznik pętli np.: **i=i+1** lub **i++**. Instrukcja wykonuje się na końcu każdego cyklu pętli
- Instrukcje wewnątrz pętli wykonywane są dopóki warunek zwraca wartość TRUE,

Przykład 1:

Spośród liczb od 1 do 1000 wypisz na ekranie wszystkie liczby podzielne przez 13. Użyj pętli for.

```
1  #include <stdio.h>
2
3  int main() {
4    int i;
5    for(i=1;i<=1000;i++) {
6      if(i%13==0) printf("%d, ",i);
7    }
8    return 0;
9  }
```

```
13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 143, 156, 169, 182, 195, 208, 221, 23
4, 247, 260, 273, 286, 299, 312, 325, 338, 351, 364, 377, 390, 403, 416, 429, 44
2, 455, 468, 481, 494, 507, 520, 533, 546, 559, 572, 585, 598, 611, 624, 637, 65
0, 663, 676, 689, 702, 715, 728, 741, 754, 767, 780, 793, 806, 819, 832, 845, 85
8, 871, 884, 897, 910, 923, 936, 949, 962, 975, 988,
```

Oto bardziej wydajne rozwiązanie zadania:

```c
#include <stdio.h>

int main() {
   int i;
   for(i=13;i<=1000;i+=13) {
     printf("%d, ",i);
   }
   return 0;
}
```

Przykład 2:

Zauważmy, że deklarując zmienną i przed pętlą for, zmienna pozostaje globalna i można korzystać z jej wartości po wykonaniu pętli.

```c
#include <stdio.h>

int main()
{
    int i;
    for(i=1; i<=3; i++)
    {
        printf("%d\n",i);
    }
    printf("Po wykonaniu petli i=%d",i);
    return 0;
}
```

```
1
2
3
Po wykonaniu petli i=4
```

Przykład 3:

Zdefiniowanie zmiennej w polu *wyrażenie1* pętli for tworzy zmienną lokalną, która występuje wyłącznie w ramach pętli for.

```c
    for(int i=1; i<=3; i++)
    {
        printf("\n%d",i);
    }
    printf("\n%d",i);
```

```
7       error: 'i' undeclared (first use in this function)
```

Przykład 4:

Wczytaj liczbę naturalną *n* (*n*>1). Narysuj na ekranie z gwiazdek trójkąt prostokątny. Użyj pętli for. Przykładowo dla *n*=3 narysowany trójkąt powinien wyglądać:

```
*
**
***
```

```c
#include <stdio.h>

int main() {
  int i, j, n;
  printf("Podaj n="); scanf("%d",&n);
  for(i=1;i<=n;++i) {
    for(j=1;j<=i;++j) printf("*");
    printf("\n");
  }
  return 0;
}
```

```
Podaj n=10
*
**
***
****
*****
******
*******
********
*********
**********
```

Przykład 5:

Wczytaj liczbę naturalną *n* (*n*>1). Narysuj na ekranie z gwiazdek trójkąt równoramienny, który nie jest trójkątem prostokątnym. Użyj pętli for. Przykładowo dla *n*=3 narysowany trójkąt powinien wyglądać:

```
  *
 ***
*****
```

```c
#include <stdio.h>

int main() {
  int i, j, n;
  printf("Podaj n="); scanf("%d",&n);
  for(i=1;i<=n;++i) {
    for(j=1;j<=n-i;++j)    printf(" ");
    for(j=1;j<=2*i-1;++j) printf("*");
    printf("\n");
  }
  return 0;
}
```

```
Podaj n=5
    *
   ***
  *****
 *******
*********
```

Własność pętli for:

UWAGA! Nie ma konieczności umieszczania instrukcji w miejscach: **wyrażenie1**, **warunek**, **wyrażenie2**. Każdą z nich możemy pominąć pozostawiając puste miejsca.

Zwróćmy uwagę na to, kiedy i ile razy w pętli for uruchamiane są instrukcje **wyrażenie1**, **warunek** oraz **wyrażenie2**.

Wyrażenie1 wykonuje się tylko raz, przed wykonaniem pierwszego cyklu pętli.

Wyrażenie2 wykonuje się na końcu każdego cyklu pętli.

- for (wyrażenie1; warunek; wyrażenie2) { }

Warunek jest sprawdzany na początku każdego cyklu pętli.

Przykład 6:

Poniższy program zadziała tak samo, jak pętla w Przykładzie 2, mimo braku instrukcji w miejscach wyrażenie1, warunek, wyrażenie2.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i=1;
6      for (;;)
7      {
8          if(i<=3)
9          {
10             printf("%d\n",i);
11             i++;
12         }
13         else
14         {
15             break;
16         }
17     }
18     printf("Po wykonaniu petli i=%d",i);
19     return 0;
20 }
```

- **Wyrażenie1** zostało zastąpione przez int i=1;
- **Warunek** został zastąpiony przez instrukcję warunkową oraz break;
- **Wyrażenie2** zostało zastąpione przez i++ wewnątrz for

Zadania do samodzielnej realizacji:

1. (1p) Napisz program drukujący na ekranie kolejne liczby naturalne podzielne przez 7. Ilość liczb wczytaj z klawiatury. Użyj pętli for. Oto wydruk dla *ile*=6:
   `0, 7, 14, 21, 28, 35`
   Zwróć uwagę, że na końcu wypisanych liczb nie ma przecinka.

2. (2p) Pobierz od użytkownika liczbę naturalną i. Wypisz tabliczkę mnożenia do i^2 w formie tabelki, np. dla i=10:

```
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
 4  8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

3. (2p) Wczytaj liczbę naturalną *n* (n>4). Narysuj na ekranie ze znaków *X* kontur trójkąta prostokątnego jak na poniższym przykładzie. Użyj pętli for.
   Dla *n*=5 powinien zostać narysowany trójkąt:

```
    X
   XX
  X X
 X  X
XXXXX
```

4. (3p) Napisz program znajdujący wszystkie trójki całkowitych liczb pitagorejskich z zadanego przedziału. Znalezione trójki liczb nie mogą się powtarzać, powinny być wypisane rosnąco ze względu na pierwszą liczbę.
   Np.: dla przedziału [10,30] mamy:

```
Trojki pitagorejskie w przedziale [a,b]
Prosze podac wartosc a =10
Prosze podac wartosc b =30
 10  24  26
 12  16  20
 15  20  25
 18  24  30
 20  21  29
```