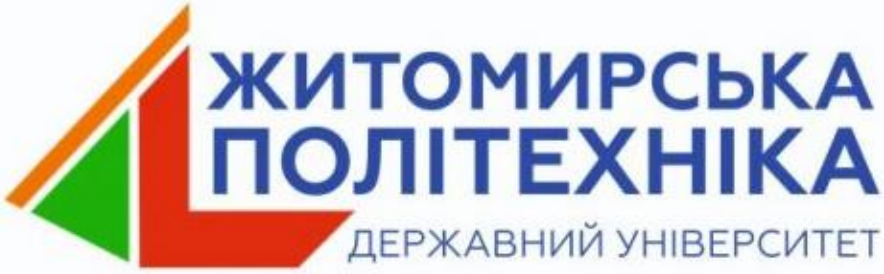


Лекція 6.
Цикл for.
Оператори break та continue

У мові Сі є два оператори, які реалізують цикл з **передумовою**:

- while
- for

Цикл **for** зручно використовувати, коли треба *перебрати інтервал* чисел або *«пробігтися»* по послідовності.

Цикл **for** називають ще:

- циклом з лічильником;
- циклом з параметром;

Синтаксис циклу **for**:

for (ініціалізація лічильника **;** умова **;** зміна лічильника **)**

оператор**;**

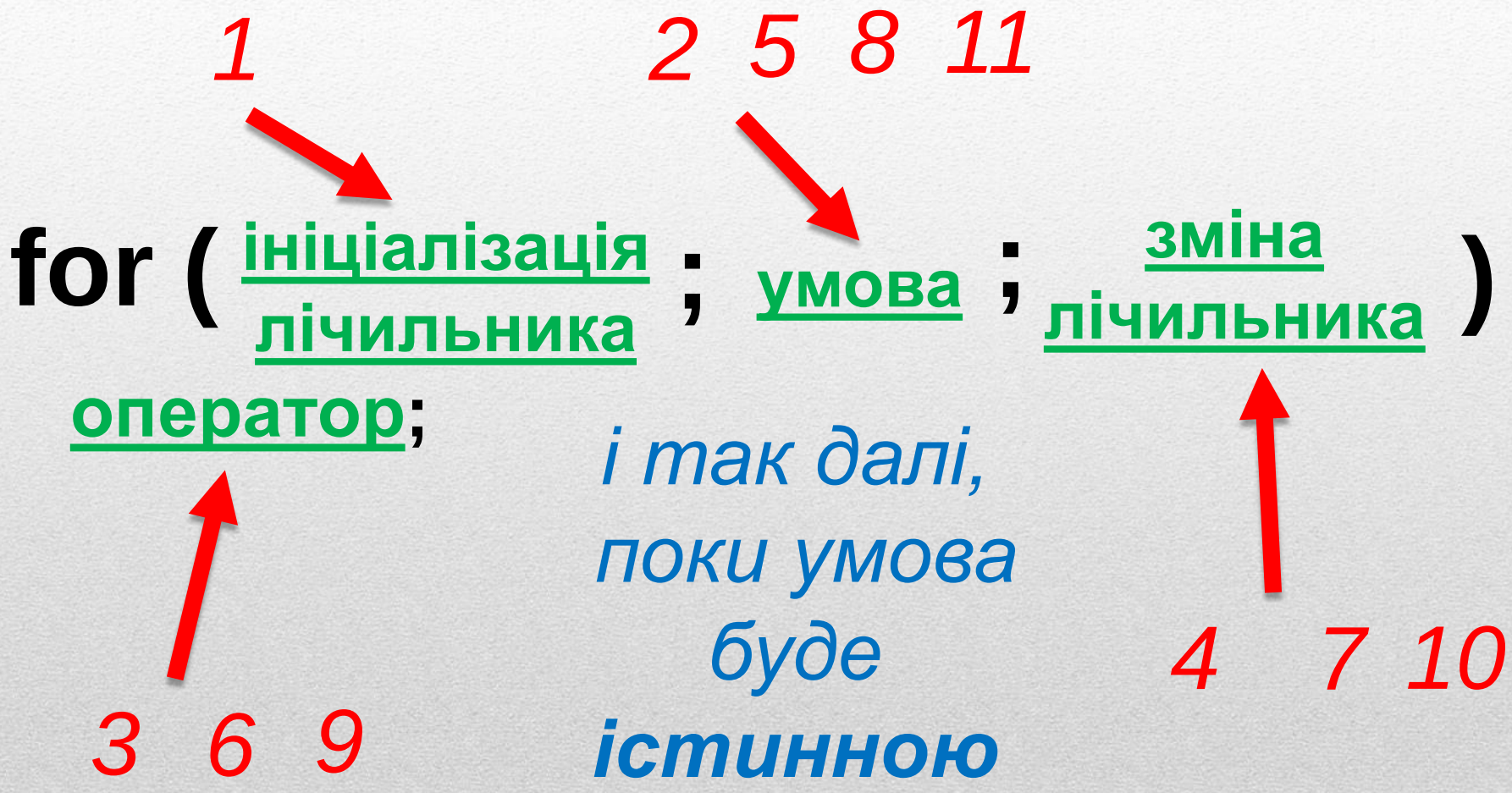
for (ініціалізація лічильника ; умова ; зміна лічильника)

оператор;

Схема роботи циклу **for**:

1. здійснюється **ініціалізація лічильника** циклу
2. перевіряється **істинність умови**
3. якщо умова істинна – **виконується тіло циклу** (оператор),
   якщо хибна – **вихід з циклу**
4. здійснюється **зміна лічильника** і перехід до пункту 2.

Отже, схема роботи циклу for виглядає так:



Отже, можна зробити такі висновки:

1) ініціалізація лічильника виконується один раз перед початком циклу

2) перевірка умови виконується перед початком кожної ітерації

3) зміна лічильника здійснюється наприкінці кожної ітерації

Цикл **for** використовують, коли потрібно перебрати певну послідовність чисел.

Завдання. Визначити кількість чисел від 1 до N, які діляться на 3

```
#include <stdio.h>
int main(){
    int n, result = 0;
    printf("n = "); scanf("%d", &n);
    for (int i = 1; i <= n; i++)
      if (i % 3 == 0)
        result++;
    printf("result = %d\n", result);
return 0;
}
```

Завдання. Визначити кількість чисел від 1 до N, які діляться на 3. Інший варіант розв'язання:

```
#include <stdio.h>
int main()
{
   int n, result = 0;
   printf("n = ");
   scanf("%d", &n);
   for (int i = 3; i <= n; i+=3)
      result++;
   printf("result = %d\n", result);
}
```

збільшення лічильника на 3

Цикл for завжди можна замінити циклом while і навпаки

```
for ( ініціалізація лічильника ; умова ; зміна лічильника )
    тіло циклу;
```

еквівалентні цикли

```
ініціалізація лічильника;
while ( умова )
{
    тіло циклу;
    зміна лічильника;
}
```

Приклад:

```
int n, i, result = 0;
printf("n = "); scanf("%d", &n);
for (i = 3; i <= n; i+=3)
    result++;
```

```
int n, i = 3, result = 0;
printf("n = "); scanf("%d", &n);
while (i <= n)
{
    result++;
    i+=3;
}
```

У циклі **for** може бути відсутньою будь-яка з трьох компонент:

for (ініціалізація лічильника ; умова ; зміна лічильника)

тіло циклу;

Якщо у циклі **for** відсутня умова, тоді вона вважається завжди істинною.

Ось такий цикл є синтаксично коректним, але він є нескінченним:

```
for (;;)
{
}
```

еквівалентні цикли

```
while (1)
{
}
```

Для того, щоб перервати цикл у мові Сі існує оператор **break**;

Оператор **break** дозволяє завершити будь-який цикл: **while**, **for** або **do … while**

За допомогою нього можна забезпечити завершення нескінченного циклу.

```
while (умова1)
{
  оператор1;
  if (умова2)
    break;
  оператор2;
}
```

*умова **продовження** циклу (якщо умова істинна, то тіло циклу виконується ще раз)*

*умова **завершення** циклу (якщо умова істинна, то цикл переривається)*

Отже, наступні цикли є еквівалентними:

```
int n, i = 3, result = 0;
printf("n = "); scanf("%d", &n);
```

```
while (i <= n)
{
  result++;
  i += 3;
}
```

еквівалентні цикли

```
while (1)
{
  if (i > n)
    break;
  result++;
  i += 3;
}
```

!(i <= n) *еквівалентні умови* i > n

У циклах можна використовувати також оператор **continue**;

Оператор **continue** завершує поточну ітерацію і передає управління:

- **у циклах while та do … while – на перевірку умови;**
- **- у циклі for – на зміну лічильника циклу**

Приклад:

```
double a = -5,
       b = 5,
       h = 0.5, x, res;
for (x = a; x <= b; x += h)
{
  if (x == 0)
    continue;
  res = 1 / x;
  printf("1 / %f == %f\n", x, res);
}
```

сюди

*якщо **x** дорівнює нулю, то одразу здійснюється перехід на зміну лічильника*

```
int i = 1, k = 10, sum = 0;
for (; i <= k; i++)
   sum = sum + i;
printf("sum = %d\n", sum);
```

```
int k = 10, sum = 0;
for (int i = 1; i <= k; ){
   sum = sum + i;
   i++;
}
printf("sum = %d\n", sum);
```

```
int k = 10;
for (int i = 1, j = 2; i <= k; i++, j += 2)
printf("i = %3d   j = %3d\n", i, j);
```

```
double sum = 0.0, a = 0.0;
double b = 1.0, h = 0.01, x;
for (x = a; x < b; x += h)
sum += x;
printf("%.3f\n", sum);
```

```
int i, j;
for (i = 2; i < 10; i++) {
  for (j = 2; j < 10; j++)
   printf("%d*%d=%d\n",i,j,
i*j);
printf("\n");
}
```

```c
int i = 1, j = 0;
while (i < 10) {
   while (j < i) {
    printf("%c", '*');
    j++;
   }
   printf("\n");
   j = 0; i++;
}
```

```c
for (int i = 1; i < 10; i++) {
   for(int j=0; j<i; j++)
      printf("%c", '*');
   printf("\n");
}
```

Приклади:

1	Підрахувати суму тільки непарних чисел з деякого діапазону

```
int result = 0;
for (int i = 0; i < 10; i++){
    if (i % 2 == 0) continue;
    result += i;
}
printf("result = %d", result);
```

2	Вивести x^2 тільки для додатних чисел

```
int x;
for (int i = 0; i < 10; ++i){
    scanf("%d", &x);
    if (x < 0) continue;
    printf("=%d\n", x*x);
}
```

?

Приклади:

1	Знайти суму чисел, кратних трьом, в діапазоні від 0 до 50.
2	Знайти кількість чисел, кратних 5, в діапазоні від 0 до 45.
3	Обчислити значення функції F (x)=$e^{4x}\sqrt{x} - 4x$ на відрізку [1,5] кроком 1.
4	Обчислити значення функції F(x)=1+e^x – $4x$ на відрізку [1,5.2] кроком 0,2.

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<windows.h>
#include <conio.h>
int main() {
SetConsoleCP(1251); SetConsoleOutputCP(1251);
int y = 1; char x;
while (y) {
    printf("Введіть букву\n");
    x = _getch();
    switch (x){
        case'а': printf("АКУЛА\n"); break;
        case'б': printf("БУРУНДУК\n"); break;
        case'ф': printf("ФАРБА\n"); break;
        default: printf("нема\n"); break;
    }
    printf("Продовжити так - 1 ні - 0\n"); scanf("%i", &y);
}
return 0;
}
```