

# Цикли



*Лекція №9*

*Дисципліна «Програмування»*



# Основні поняття

**Цикл** – це різновид керівної конструкції у мові **C**, яка призначена для організації багаторазового виконання набору інструкцій (команд).

Також циклом може називатися будь-яка послідовність команд, яка багатократно виконується та організована будь-яким чином (наприклад, за допомогою умовного переходу).

**Тіло циклу** – послідовність інструкцій, яка призначена для багаторазового виконання.

**Ітерація** – одноразове виконання тіла циклу.

**Умова виходу з циклу** або **умова завершення циклу** – вираз, який визначає чергове виконання ітерації або завершення циклу.



# Основні поняття

**Лічильник циклу** – змінна, в якій зберігається номер поточної ітерації.

Частинами виконання будь-якого циклу є:

- початкова ініціалізація змінних циклу;
- перевірка умови виходу з циклу;
- виконання тіла циклу;
- оновлення змінної циклу на кожній ітерації.

Крім того, мова **C** надає засоби для керування ходом виконання циклу:

- оператори виходу з циклу незалежно від істинності умови виходу (інструкція **break**);
- оператори пропущення ітерації (інструкція **continue**).



# Цикл з передумовою **while**

**Цикл з передумовою** виконується до тих пір, поки умова, що позначена в круглих дужках після ключового слова **while**, є істинною.

Ця умова перевіряється до початку виконання тіла циклу, тому тіло може не виконатися жодного разу (якщо умова одразу є хибною).

```
while (умова)
{
    // тіло циклу
}
```

Зверніть увагу на те, що наприкінці першого рядка **while** (умова) крапка з комою не ставиться. Це найпоширеніша помилка, що призводить до зациклювання програми.



# Цикл з передумовою while

```
#include <stdio.h>
#include <windows.h>

int main(void)
{
    int speed;           // Початкова швидкість автомобіля
    int count;           // Кількість ітерацій циклу

    SetConsoleOutputCP(1251);

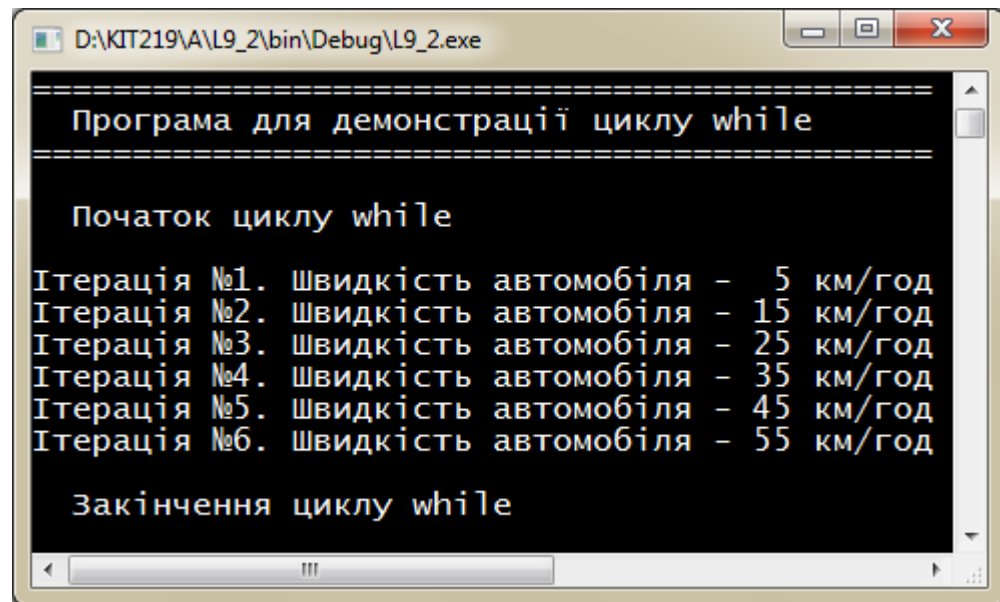
    printf("=====\n");
    printf("  Програма для демонстрації циклу while      \n");
    printf("=====\n");

    count = 1;
    speed = 5;

    printf("\n  Початок циклу while\n\n");
```

# Цикл з передумовою while

```
while ( speed < 60 )
{
    printf("Ітерація №%d. Швидкість автомобіля - %2d км/год\n",
           count, speed);
    speed += 10;    // додаємо швидкість
    count++;        // збільшуємо кількість ітерацій на 1
}
printf("\n  Закінчення циклу while\n\n");
return 0;
}
```



```
D:\KIT219\A\L9_2\bin\Debug\L9_2.exe

=====
Програма для демонстрації циклу while
=====

Початок циклу while

Ітерація №1. Швидкість автомобіля -  5 км/год
Ітерація №2. Швидкість автомобіля - 15 км/год
Ітерація №3. Швидкість автомобіля - 25 км/год
Ітерація №4. Швидкість автомобіля - 35 км/год
Ітерація №5. Швидкість автомобіля - 45 км/год
Ітерація №6. Швидкість автомобіля - 55 км/год

Закінчення циклу while
```



# Цикл з постумовою **do...while**

**Цикл з постумовою** – це цикл, в якому умова перевіряється після виконання тіла циклу. Тобто тіло циклу завжди виконується хоча б один раз. У мові **C** роль циклу з постумовою відіграє цикл **do...while**.

```
do
{
    // тіло циклу
}
while (умова) ;
```

Для циклу з постумовою істинна умова в круглих дужках (після **while**) трактується як умова продовження (цикл завершується, коли умова хибна).

Зверніть увагу на те, що наприкінці останнього рядка **while** (умова) ; крапка з комою ставиться.



# Цикл з постумовою do...while

```
#include <stdio.h>
#include <windows.h>
#include <time.h>

int main(void)
{
    int balance;           // поточний баланс
    int removal;           // зняття грошей з рахунку

    SetConsoleOutputCP(1251);
    printf("//=====\\n");
    printf("// Програма для демонстрації циклу do...while \\n");
    printf("//=====\\n");
    printf("Поточний час:      %s\\n\\n", __TIME__);
    printf("Поповнити рахунок: ");
    scanf("%d", &balance);
    printf("Рахунок поповнено!\\n");
    printf("=====\\n");
    printf("Ваш поточний рахунок складає:  %2d грн\\n", balance);
    printf("=====\\n");
    printf("\\nПочаток циклу do...while\\n");
```





# Цикл з постумовою do...while

```
do                                // початок циклу do...while
{
    printf("\n===== \n");
    printf("Поточний баланс:    %d грн\n", balance);
    printf("===== \n");
    printf("Зняти з рахунку:    ");
    scanf("%d", &removal);
    printf("===== \n");

    balance -= removal;    // знімаємо з рахунку

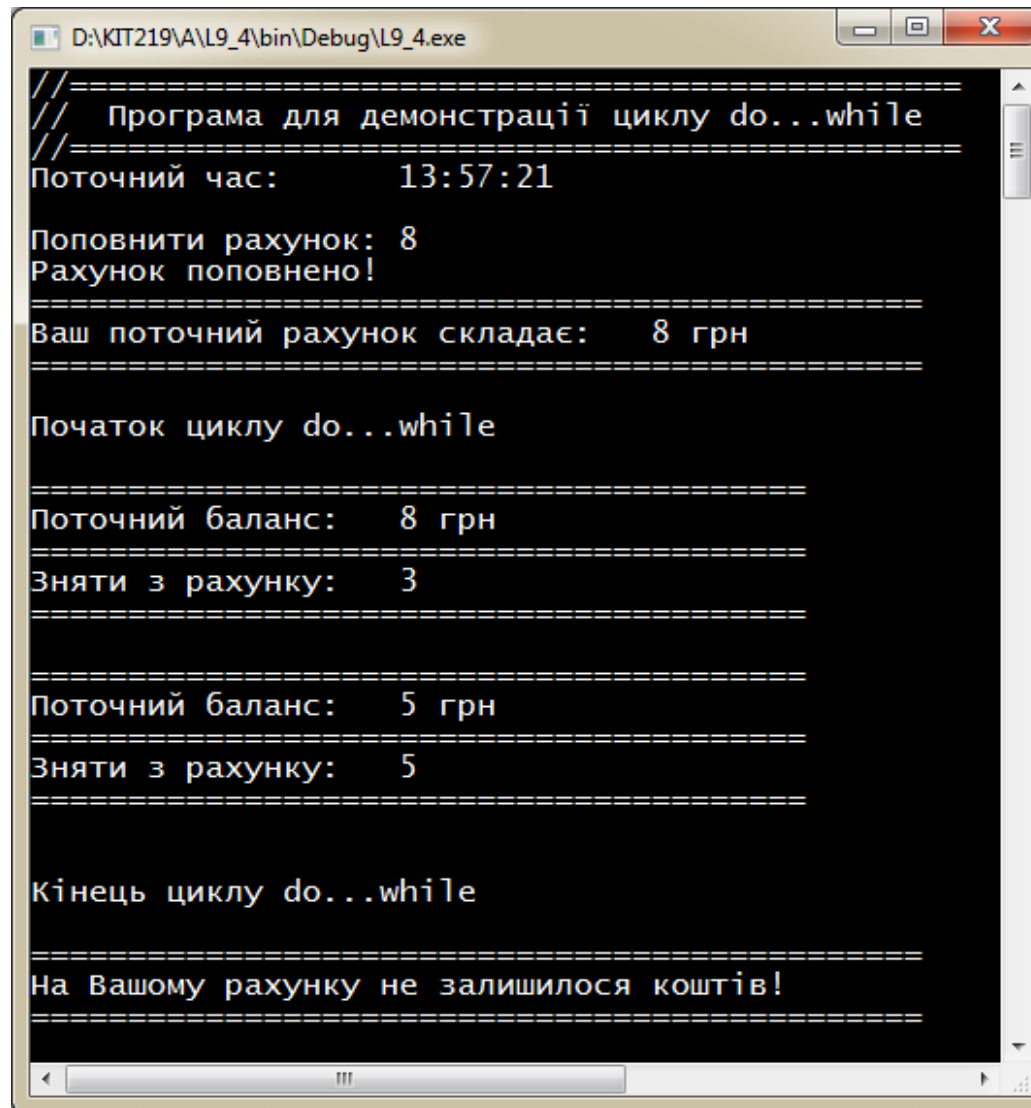
} while( balance > 0 );    // кінець циклу do...while

printf("\n\nКінець циклу do...while\n\n");
printf("===== \n");
printf("На Вашому рахунку не залишилося коштів!\n");
printf("===== \n");

return 0;

}
```

# Цикл з постумовою do...while



```
D:\KIT219\A\L9_4\bin\Debug\L9_4.exe

//=====
//  Програма для демонстрації циклу do...while
//=====
Поточний час:      13:57:21

Поповнити рахунок: 8
Рахунок поповнено!

=====
Ваш поточний рахунок складає:   8 грн
=====

Початок циклу do...while

=====
Поточний баланс:   8 грн
=====
Зняти з рахунку:   3
=====

=====
Поточний баланс:   5 грн
=====
Зняти з рахунку:   5
=====

Кінець циклу do...while

=====
На Вашому рахунку не залишилося коштів!
=====
```



# Цикл з лічильником for

**Цикл з лічильником** – це цикл, в якому змінна циклу модифікує своє значення від заданого початкового значення до кінцевого значення з певним кроком, і для кожного значення цієї змінної тіло циклу виконується один раз. В мові **C** цей тип циклу реалізується оператором **for**.

```
for (вираз_1; вираз_2; вираз_3)
{
    // тіло циклу
}
```

вираз\_1 – це присвоювання початкового значення;  
вираз\_2 – умова виходу з циклу (умовний вираз);  
вираз\_3 – збільшення або зменшення кроку циклу.

Будь-який з цих виразів може бути відсутнім, але **крапку з комою необхідно залишати.**



# Цикл з лічильником for

В мові С цикл **for**, незважаючи на синтаксичну форму, цикл з лічильником насправді є циклом з передумовою. Тобто в **С** конструкція циклу:

```
for (i = 0; i < 10; i++)  
{  
    // тіло циклу  
}
```

фактично являє собою інший варіант запису конструкції:

```
i = 0;  
while (i < 10)  
{  
    // тіло циклу  
    i++;  
}
```



# Цикл з лічильником for

```
#include <stdio.h>
#include <windows.h>

int main(void)
{
    int i;

    SetConsoleOutputCP(1251);

    printf("//=====\\n");
    printf("//  Програма для демонстрації циклу for  \\n");
    printf("//=====\\n");
    printf("\\nPочаток циклу for\\n\\n");
    for(i = 0; i < 10; i++)
    {
        printf("%3d\\n", i);
    }
    printf("\\nЗакінчення циклу for\\n\\n");
    return 0;
}
```

```
D:\KIT219\A\L9_5\bin\Debug\L9_5.exe
//=====
//  Програма для демонстрації циклу for  \\n
//=====
\\nPочаток циклу for\\n\\n
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
\\nЗакінчення циклу for\\n\\n
```



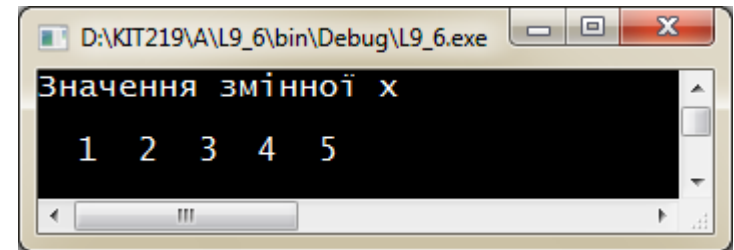
# Цикл з лічильником for

Стандарт **C99** дозволяє оголошувати змінну у виразі ініціалізації інструкції **for**. Область видимості такої змінної буде обмежена інструкцією **for**.

```
#include <stdio.h>
#include <windows.h>

int main(void)
{
    SetConsoleOutputCP(1251);

    printf("Значення змінної x\n\n");
    for(int x = 1; x <= 5; ++x)
    {
        printf("%3d", x);
    }
    printf("\n\n");
    return 0;
}
```





# Цикл з виходом з середини

**Цикл з виходом з середини** – це найзагальніший тип умовного циклу.

Синтаксично такий цикл оформлюється за допомогою трьох інструкцій:

- початок циклу;
- кінець циклу;
- інструкція (команда) виходу з циклу.

Інструкція початку позначає точку програми, з якої починається тіло циклу, інструкція кінця – точку, де тіло закінчується. Всередині тіла має бути присутня команда виходу з циклу, при виконанні якої цикл завершується, і керування передається на оператор, який йде після інструкції кінця циклу. Команда виходу має викликатися при виконанні умови виходу з циклу.



# Цикл з виходом з середини

Команда дострокового виходу застосовується, коли необхідно перервати виконання циклу, в якому умови виходу ще не досягнуто. Таке буває, наприклад, коли під час виконання циклу зустрічається помилка, після якої подальше виконання циклу не має сенсу.

У мові **C** цикл з виходом з середини може бути побудований за допомогою будь-якого умовного циклу та інструкції дострокового виходу з циклу (такої, як **break** або **exit**) або інструкції безумовного переходу **goto**.

Команда безумовного переходу **goto** здійснює перехід на команду, яка розміщується безпосередньо за межами циклу, всередині якого вона знаходиться.





# Цикл з виходом з середини

Так у мові **C** два наступних цикли працюють цілком однаково:

```
// Застосування оператора break
```

```
while (умова)
```

```
{
```

```
    // оператори
```

```
    if (помилка) break;
```

```
    // оператори
```

```
}
```

```
// продовження програми
```

```
    // Аналогічний фрагмент без оператора break
```

```
while (умова)
```

```
{
```

```
    // оператори
```

```
    if (помилка) goto label;
```

```
    // оператори
```

```
}
```

```
label: // продовження програми
```



# Нескінченний цикл

Іноді в програмах використовуються цикли, вихід з яких не передбачений логікою програми. Такі цикли називаються **безумовними** або **нескінченними**.

Вихід з таких циклів частіше за все відбувається завдяки використанню будь-якої умови, істинне значення якої дозволяє застосувати інструкцію **break** або **exit**.

```
for( ; ; )  
{  
    ch = getchar();                // введення символу  
    if(ch == '#')  
    {  
        printf("Ітерація не завершена\n");  
        break;                    // вихід з циклу  
    }  
    ch = getchar();                // зчитуємо <Enter>  
    i++;  
    printf("Ітерація завершена\n");  
}
```



# Нескінченний цикл

```
while (1)
{
    ch = getchar();                // введення символу

    switch (ch)
    {
        case '1': printf("Натиснута клавіша '1'\n");
                    break;
        case '2': printf("Натиснута клавіша '2'\n");
                    break;
        case '#': printf("Програма завершена\n");
                    exit(0);
        default: printf("Натиснута не та клавіша\n");
                  break;
    }
    ch = getchar();                // зчитуємо <Enter>
}
```



# Пропуск ітерації `continue`

У мові **C** в якості команди пропуску ітерації використовується інструкція **`continue`** в конструкції циклу.

Даний оператор застосовується, коли в поточній ітерації циклу необхідно пропустити всі команди до кінця тіла циклу.

При цьому сам цикл не переривається, умови продовження або виходу обчислюються звичайним чином.

Дія цього оператора аналогічна безумовному переходу на рядок всередині тіла циклу, наступний за останньою його командою.

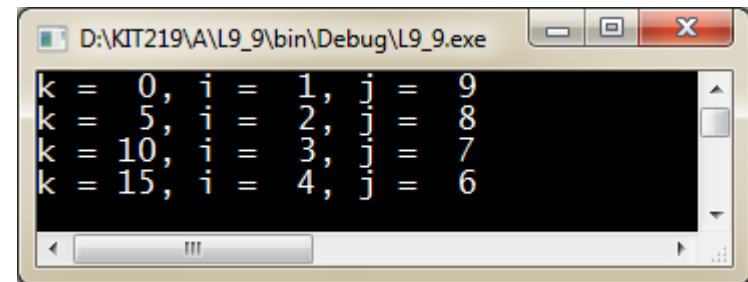
# Пропуск ітерації continue

```
#include <stdio.h>
#include <windows.h>

int main(void)
{
    int i = 0, j = 10;

    SetConsoleOutputCP(1251);

    for(int k = 0; k < 20; k++)
    {
        if(k % 5 != 0) continue;
        i++;
        j--;
        printf("k = %2d, i = %2d, j = %2d\n", k, i, j);
    }
    printf("\n");
    return 0;
}
```



```
D:\KIT219\A\L9_9\bin\Debug\L9_9.exe
k =  0, i =  1, j =  9
k =  5, i =  2, j =  8
k = 10, i =  3, j =  7
k = 15, i =  4, j =  6
```



# Вкладені цикли

В **с** існує можливість утворювати цикл всередині тіла іншого циклу. Такий цикл має назву **вкладеного циклу**.

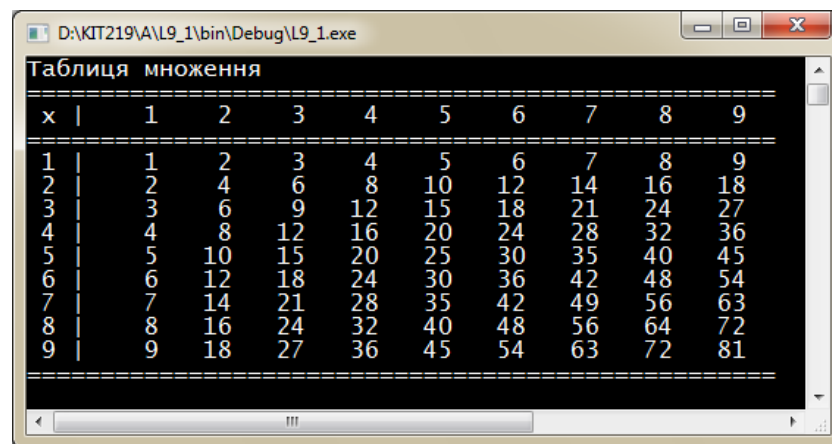
Вкладений цикл по відношенню до циклу, в тіло якого він вкладений, буде йменуватися **внутрішнім циклом**, і навпаки цикл, в тілі якого існує вкладений цикл, буде мати назву **зовнішнього циклу**.

Всередині вкладеного циклу може бути наступний вкладений цикл, утворюючи наступний рівень вкладеності і так далі.

Кількість рівнів вкладеності, як правило, не обмежується.

# Вкладені цикли

```
#include <stdio.h>
#include <windows.h>
int main(void)
{
    int i, j;
    SetConsoleOutputCP(1251);
    printf("Таблиця множення\n");
    printf("=====\n");
    printf(" x |   1   2   3   4   5   6   7   8   9   \n");
    printf("=====\n");
    for(i = 1; i < 10; i++)
    {
        printf("%2d |", i);
        for(j = 1; j < 10; j++)
            printf("%5d", i * j);
        printf("\n");
    }
    printf("=====\n");
    return 0;
}
```



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