Управляющие конструкции: циклы. Виды циклов. Бесконечный цикл. Вложенные циклы. Примеры

Управляющие конструкции: циклы
Цикл - управляющая конструкция, которая заставляет какой-то блок кода выполняться несколько раз.

Виды циклов
В Си есть 3 основных типов циклов:
Цикл `for`:

- `for` зачастую используется для повторения какого либо действия заданное количество раз.
- Синтаксис:

```
for (int i = 0; i < 10; i++)
{
// Код
}
```

1. `int i = 0` - переменная, которую можно задать как начальное число.
2. `i < 10` - условие выполнения кода.
3. `i++` - то действие, которое будет выполнятся после каждой итерации (выполнения кода внутри) цикла.
   Цикл `while`:

- `while` используется тогда, когда нужно выполнять код *пока* условие истинно. Самый простой цикл.
- Синтаксис:

```
int n = 10;
while (n > 0)
{
```

```
// Код
}
```

1. `n > 0` - условие, при котором выполняется цикл.
   Цикл `do-while`

- `do-while` это цикл, похожий на `while`, но его отличие от своего собрата, как и от `for`, тело цикла выполняется до проверки условия в скобках.
- Синтаксис:

```
int n = 10;
do
{
//Код
} while (n > 0);
```

1. Тело цикла описывается в блоке `do`.
2. Условие цикла описывается внутри `while` в скобках.

Бесконечный цикл
Бесконечный цикл, что логично, выполняется бесконечно. То есть, условие цикла всегда будет истинным.
Реализация в разных циклах:

- `for`:

```
for (;;) // <- можем опустить части цикла внутри скобок
{
// Код
}
```

или, к примеру:

```
for (int i = 0; i < 10; i--)
{
// Код
}
```

Есть множество способов сделать бесконечный цикл `for`

- `while`:

```
while(1)
{
// Код
}
```

- `do-while`:

```
do
{
// Код
} while (1);
```

Для завершения бесконечных циклов используется оператор перехода `break`.

Операторы перехода
Операторы перехода - в циклах: специальные операторы, предназначенные для операций внутри циклов.

- `break` - завершает выполнение цикла полностью.
- `continue` - переходит на следующую итерацию цикла, не выполняя оставшийся код.

Вложенные циклы
Вложенным циклом является цикл, который написан **внутри** другого цикла. В 99.9% случаев вложенные циклы делаются из циклов `for`.

```
for (int i = 0; i < 5; i++)
{
        for (int j = 0; j < 5; j++)
        {
                //Код, можно изменить переменную i
        }
}
```

При этом, код внутри вложенного цикла может изменить переменную, описанную в первом цикле.

Так как вложенный цикл выполняется столько раз, сколько будет итераций в первом цикле, несложно догадаться, что код внутри вложенного цикла будет выполнятся много раз, а именно: `количество_итераций_первого * количество_итераций_вложенного` раз.

Примеры

1. Цикл `for`: Вывод чисел от 1 до 10

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

2. Цикл `while`: Сумма чисел от 1 до 10

```
int i = 1;
int sum = 0;

while (i <= 10)
{
    sum += i;
    i++;
}

printf("Сумма чисел от 1 до 10: %d\n", sum);
```

3. Цикл `do-while`: Игра "Угадай число"

```
int secret = 7;
int guess;

do
{
    printf("Введите число: ");
    scanf("%d", &guess);

    if (guess < secret) {
        printf("Слишком мало!\n");
    } else if (guess > secret) {
        printf("Слишком много!\n");
    }
```

```
} while (guess != secret);

printf("Вы угадали!\n");
```

4. Вложенные циклы: Таблица умножения

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
        printf("%d x %d = %d\t", i, j, i * j);
    }
    printf("\n");
}
```

5. Прерывание цикла с `break`

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) {
        printf("Прерываем цикл на i = %d\n", i);
        break;
    }
    printf("%d\n", i);
}
```

6. Пропуск итерации с `continue`

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        continue; // Пропускаем четные числа
    }
    printf("%d\n", i);
}
```