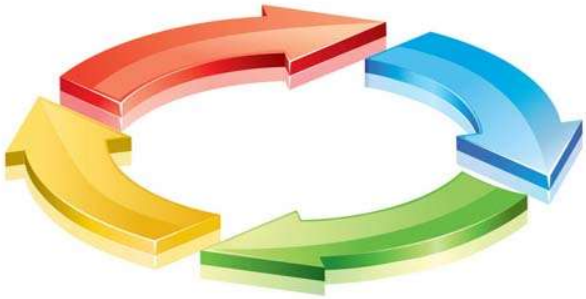
ЦИКЛЫ



Начнем с того что с циклами вы уже по-любому сталкивались: это конечно же основной цикл любой программы loop().

Цикл это грубо говоря рамки, код внутри которых выполняется сверху вниз и повторяется с начала, когда достигает конца. Продолжается это дело до тех пор, пока выполняется какое то условие. Есть два основных цикла, с которыми мы будем работать, это for и while.

Цикл for

Цикл for, в простонародии **счётчик**, в различных видах этот цикл есть и в других языках программирования, но на C++ он имеет очень гибкую настройку. При создании цикл принимает три “значения” (настройки): ***инициализация, условие и изменение***. Цикл for обычно содержит переменную, которая изменяется на протяжении работы цикла, мы можем пользоваться её меняющимся значением в своих целях. Переменная является **локальной для цикла**, если она создаётся при инициализации.

- **Инициализация** – здесь обычно присваивают начальное значение переменной цикла. Например:
  `int i = 0;`
- **Условие** – здесь задаётся условие, при котором выполняется цикл. Как только условие нарушается, цикл завершает работу. Например:
  `i < 100;`
- **Изменение** – здесь указывается изменение переменной цикла на каждой итерации. Например:
  `i++;` (то же что `i=i+1`)

Объединим три предыдущих пункта в пример:

```
for (int i = 0; i < 100; i++) {

  // тело цикла

}
```

В теле цикла мы можем пользоваться значением переменной

i

, которая примет значения **от 0 до 99** на протяжении работы цикла, после этого цикл завершается. Как это использовать?

Например:

Создадим простой цикл для движения драгстера вперед по условию. «Пока количество оборотов колеса драгстера меньше 20 – ехать вперед».

//kol – переменная, в которой будет храниться значение кол-ва оборотов колеса драгстера

```
for (int kol = 0; kol < 20; kol++) {

  // драгстер едет впередтело цикла

}
```

Что касается особенностей использования for в языке C++: любая его настройка является **необязательной**, то есть её можно не указывать для каких-то особенных алгоритмов. Например, вы не хотите создавать переменную цикла, а использовать для этого другую имеющуюся переменную. Пожалуйста! Но не забывайте, что разделители настроек (точка с запятой) **обязательно должны присутствовать на своих местах**, даже если настроек нет!

```
// есть у нас какая-то переменная
int kol = 0;//где то в программе уже используется переменная kol, в которую записывается кол-во оборотов колеса драгстера
…
…
//
for (; kol < 20; kol++) {
// драгстер едет вперед
// В данном примере используется уже не локальная (не видимая для другой части программы) переменная kol, а глобальная, к которой можно обратиться из любой части программы.
}
```

В цикле for можно сделать несколько счётчиков, несколько условий и несколько инкрементов, разделяя их при помощи оператора, смотрите пример:

```
// объявить i и j
// прибавляті+1 и j+2 (+=2)
for (int i = 0, j = 0; i < 10; i++, j += 2) {
  // тут i меняется от 0 до 9
  // и j меняется от 0 до 18
}
```

Также в цикле **может вообще не быть настроек**, и такой цикл можно считать вечным, замкнутым:

```
for (;;) {
  // выполняется вечно...
```

```
}
```

Использование замкнутых циклов не очень приветствуется, но иногда является очень удобным способом поймать какое-то значение, или дать программе “зависнуть” при наступлении ошибки. Но, как мы знаем, нет ничего вечного, поэтому из цикла таки можно выйти при помощи оператора

break.

Оператор break

Оператор break (англ. “ломать”) позволяет досрочно покинуть цикл, использовать его можно как по условию, так и как-угодно-удобно. Например давайте досрочно выйдем из цикла при достижении какого-то значения:

```
for (int i = 0; i < 100; i++) {
  // покинуть цикл при достижении 50
  if (i == 50) break;
}
```

Или вот такой абстрактный пример, покинем “вечный” цикл при нажатии на кнопку:

```
for (;;) {
  if (кнопка нажата) break;
}
```

Выход из цикла является не единственным интересным инструментом, ещё есть оператор пропуска – continue

Оператор continue

Оператор continue (англ. “продолжить”) досрочно завершает **текущую итерацию** цикла и переходит к следующей. Например, *«При нажатии на кнопку – загорается светодиод, но не при каждом нажатии, начиная со второго, первый раз при нажатии на кнопку ничего не произойдет»*

```
// если кнопка нажата, то:
for (int i = 0 ;; i++) {
  // если в цикле нет условия завершения, будет выполняться всегда
  if (i == 0) continue;
```

светодиод загорается // при первом нажатии, когда i еще равна 0, светодиод не загорится, цикл начнет выполняться заново

}

Цикл while

Цикл while (англ. “пока”), он же называется цикл с предусловием, выполняется до тех пор, пока верно указанное условие. Если условие сразу неверно, цикл даже не начнёт свою работу и будет полностью пропущен. Объявляется очень просто: ключевое слово

While, далее **условие** в скобках, и вот уже тело цикла:

```
int i = 0;

while (i < 10) {

  i++;

}
```

Как видно из примера - это полный аналог цикла for с настройками

(int i = 0; i < 10; i++) . Единственное отличие в том, что на последней итерации

i примет значение 10, так как на значении 9 цикл разрешит выполнение.

Цикл while тоже удобно использовать как вечный цикл, например, ожидая наступления какого-либо события (нажатие кнопки):

```
// выполняется, пока не нажата кнопка

while (кнопка нажата);
```

Помимо цикла с предусловием есть ещё цикл с постусловием, так называемый

do while

Цикл do while

do while – “делать пока”, работа этого цикла полностью аналогична циклу while за тем исключением, что здесь условие задаётся после цикла, т.е. **цикл выполнится один раз**, затем проверит условие, а не наоборот.

```
do {

  //выполняется какой то цикл
```

```
} while (условие);
```

Пример:

```
int i = 0;
do {
        //выполняется какой-то цикл
        i++;
} while (условие);


while (i < 10) {
  i++;
}
```

Где применять? Да где угодно, по ходу написания собственных программ вы почувствуете, где удобнее использовать тот или иной цикл.

Видеоурок на тему циклов в Ардуино:
https://www.youtube.com/watch?v=gIVcKbSeqFo