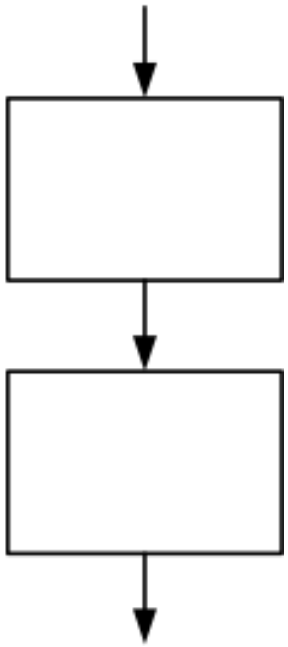
Основные алгоритмические структуры

Следование (линейный алгоритм)

Следование представляет собой последовательное выполнение операций и представляется алгоритмически последовательностью блоков «Процесс»:



Реализация следования в C++ имеет вид:

```
void foo()
{
  операция;
  операция;
  операция;
}
```

Реализация следования в Arduino имеет вид:

```
void foo()
{
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  digitalWrite(11, LOW);
  digitalWrite(12, HIGH);
}
```

Развилка (условие).

Развилка, в свою очередь, делится на
• неполную развилку;
• полную развилку;
• ветвление.

Развилка представляет собой блок выбора (проверка условия). Неполная развилка выполняет последовательность операций только по одной из веток.

Реализация неполной развилки в C++ имеет вид:

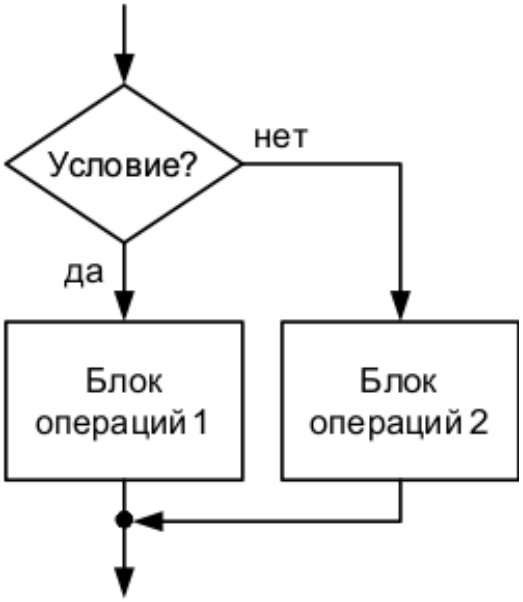
```
if (условие)
{
  операция;
}
```

Реализация неполной развилки в Arduino имеет вид:

```
void foo()
{
  if (state == true) // Проверка условия. Если false — пропускаем digitalWrite(8, HIGH);
  {
    digitalWrite(8, HIGH);
  }
}
```

Полная развилка выполняет последовательность операций по каждой из двух веток (при выполнении или невыполнении условия):

Реализация полной развилки в C++ имеет вид:



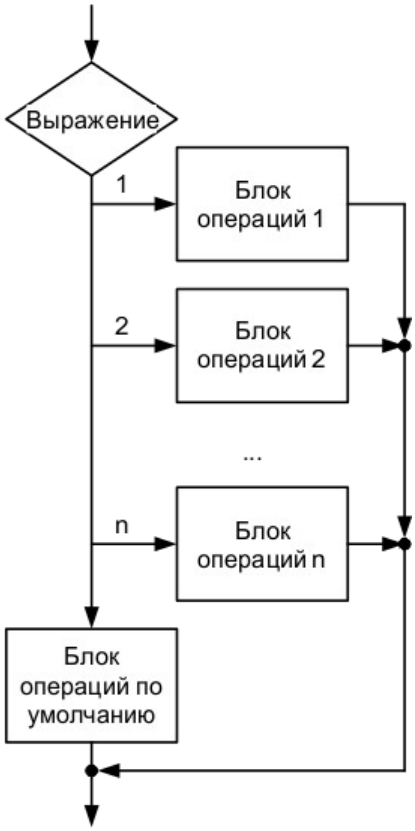

```
if (условие)
{
  операция блока 1;
}
else
{
  операция блока 2;
}
```

Реализация полной развилки в Arduino имеет вид:

```
void foo()
{
  if (state == true) // Проверка условия. Если false — пропускаем digitalWrite(8, HIGH);
  {
    digitalWrite(8, HIGH);
  }
  else
  {
    digitalWrite(8, LOW);
  }
}
```

Ветвление

Ветвление представляет собой операцию множественного выбора, при которой проверка условия может иметь более двух возможных вариантов:



Реализация ветвления в C++ имеет вид:

```
switch (выражение)
{
 case 1:
  блок операций 1;
 break;
 case 2:
  блок операций 2;
 break;
…
 case n:
  блок операций n;
 break;
 default:
  блок операций по умолчанию;
}
```

Реализация ветвления в Arduino имеет вид:

```
switch(Flag) // Сравниваем переменную с несколькими значениями
 {
  case 0:
   led8(); // Включаем 8 светодиод
  break;
  case 1:
   led9(); // Включаем 9 светодиод
  break;
  case 2:
   led10(); // Включаем 10 светодиод
  break;
  default:
   ledsOFF(); // Все светодиоды выключены
  break;  }
```

Цикл

Существует 3 основных вида циклов:
• цикл с предусловием;
• цикл с постусловием;
• параметрический цикл.

Цикл с предусловием осуществляет проверку условия перед началом своего выполнения. В случае если условие не выполняется, происходит выход из цикла. Цикл с предусловием может не выполниться ни одного раза.

Реализация цикла с предусловием на C++:

```
while (условие)
  {
   операции;
  }
```

Реализация цикла с предусловием на Arduino:

```
while(i < 5) // Пока переменная i меньше 5, код выполняется.
   {
      ++i;
      digitalWrite(8, HIGH);
      delay(500);
      digitalWrite(8, LOW);
      delay(500);
   }
```

Цикл с постусловием всегда выполняется хотя бы один раз, поскольку проверка условия осуществляется после выполнения операций цикла.

Реализация на C++ цикла с постусловием:

```
do {
  операция;
}
while (условие);
```

Реализация на Arduino цикла с постусловием:

```
do {
   if(data == true) {
   digitalWrite(9, HIGH);
  }
}
while((data = digitalRead(button)) != 0);
```

Параметрический цикл — это цикл с заданным числом повторений.

Реализация параметрического цикла с пояснением:

```
for (П = НЗ; П <= КЗ; П += Ш)
 {
   операция;
 }
```

где П — параметр, НЗ — начальное значение, КЗ — конечное значение (в общем случае — условие продолжения цикла, Ш — шаг):

Реализация параметрического цикла на Arduino:

```
for (int i = 0; i < 15; i = i + 5)
  {
   digitalWrite(13, i);
   delay(100);
  }
```