

Ravesli [Ravesli](#)


- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExр](#)
- [Ассемблер](#)
- [Купить .PDF](#)

Урок №41. Условный тернарный оператор, оператор sizeof и Запятая

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 7 Сен 2020 |

 40732

[1](#)  5

На этом уроке мы рассмотрим условный тернарный оператор, оператор Запятую и вспомним оператор sizeof в языке C++.

Оглавление:

1. [Оператор sizeof](#)
2. [Оператор Запятая](#)
3. [Условный тернарный оператор](#)
4. [Условный тернарный оператор вычисляется как выражение](#)

Оператор sizeof

Мы уже рассматривали оператор sizeof на [уроке №30](#).

Оператор	Символ	Пример	Операция
sizeof	sizeof	sizeof(type) sizeof(variable)	Возвращает размер типа данных или переменной в байтах

Тогда мы использовали его для определения размера конкретных типов данных. Но также sizeof можно использовать и с переменными:

```
1 #include <iostream>
2
3 int main()
4 {
5     double t = 7.0;
```

```
6 | std::cout << sizeof(t); // выводим размер переменной t в байтах
7 | }
```

Оператор Запятая

Оператор Запятая (или «*оператор Comma*») позволяет обрабатывать несколько выражений (в то время, когда, обычно, позволяет только одно).

Оператор Символ Пример Операция

Запятая , x, y Вычисляется x, затем вычисляется y, а затем возвращается значение y

Выражение, в котором находится этот оператор, будет иметь значение правого операнда. Например:

```
1 | int x = 0;
2 | int y = 2;
3 | int z = (++x, ++y); // инкремент переменных x и y
```

Переменной z будет присвоен результат вычисления ++y (правого операнда), что равно 3.

Почти в каждом случае, стейтмент, в котором есть оператор Запятая, лучше записывать в виде отдельных инструкций. Вышеприведенный код корректнее будет записать следующим образом:

```
1 | int x = 0;
2 | int y = 2;
3 | ++x;
4 | ++y;
5 | int z = y;
```

Обратите внимание, оператор Запятая имеет самый низкий **приоритет** из всех операторов (даже ниже, чем в оператора присваивания), поэтому следующие две строки кода делают не одно и то же:

```
1 | z = (a, b); // сначала вычисляется выражение (a, b), которое равняется значению b, а
2 | z = a, b; // вычисляется как "(z = a), b", поэтому переменной z присваивается значение a
```

Большинство программистов не используют оператор Comma вообще (разве что только в **циклах for**).

Обратите внимание, запятая, которая используется в вызовах функций, не является оператором Comma:

```
1 | int sum = add(x, y); // эта запятая не является оператором Comma
```

Аналогично, при объявлении нескольких переменных в одной строке, запятая используется как разделитель, а не как оператор:

```
1 | int x(3), y(5); // эта запятая не является оператором Comma
```

Правило: Избегайте использования оператора Comma (исключением являются циклы for).

Условный тернарный оператор

Условный (тернарный) оператор (обозначается как `? :`) является единственным тернарным оператором в языке C++, который работает с 3-мя операндами. Из-за этого его часто называют просто *«тернарный оператор»*.

Оператор Символ Пример Операция

Условный <code>?:</code>	<code>c ? x : y</code>	Если <code>c</code> — ненулевое значение (<code>true</code>), то вычисляется <code>x</code> , в противном случае — <code>y</code>
--------------------------	------------------------	---

Оператор `?:` предоставляет сокращенный способ (альтернативу) ветвления `if/else`.

Стейтменты `if/else`:

```
if (условие)
    выражение;
else
    другое_выражение;
```

Можно записать как:

`(условие) ? выражение : другое_выражение;`

Обратите внимание, операнды условного оператора должны быть выражениями (а не стейтментами).

Например, ветвление `if/else`, которое выглядит следующим образом:

```
if (условие)
    x = значение1;
else
    x = значение2;
```

Можно записать как:

`x = (условие) ? значение1 : значение2;`

Большинство программистов предпочитают последний вариант, так как он читабельнее.

Давайте рассмотрим еще один пример. Чтобы определить, какое значение поместить в переменную `larger`, мы можем сделать так:

```
1 if (x > y)
2     larger = x;
3 else
4     larger = y;
```

Или вот так:

```
1 larger = (x > y) ? x : y;
```

Обычно, часть с условием помещают внутри скобок, чтобы убедиться, что приоритет операций корректно сохранен и так удобнее читать.

Помните, что оператор `?:` имеет очень низкий приоритет, из-за этого его следует записывать в круглых скобках.

Например, для вывода x или y , мы можем сделать следующее:

```
1 if (x > y)
2     std::cout << x;
3 else
4     std::cout << y;
```

Или с помощью тернарного оператора:

```
1 std::cout << ((x > y) ? x : y);
```

Давайте рассмотрим, что произойдет, если мы не заключим в скобки весь условный оператор в вышеприведенном случае. Поскольку оператор $<<$ имеет более высокий приоритет, чем оператор $?:$, то следующий стейтмент (где мы не заключили весь тернарный оператор в круглые скобки, а только лишь условие):

```
1 std::cout << (x > y) ? x : y;
```

Будет обрабатываться как:

```
1 (std::cout << (x > y)) ? x : y;
```

Таким образом, в консольном окне мы увидим 1 (true), если $x > y$, в противном случае — выведется 0 (false).

Совет: Всегда заключайте в скобки условную часть тернарного оператора, а лучше весь тернарный оператор.

Условный тернарный оператор — это удобное упрощение ветвления `if/else`, особенно при присваивании результата переменной или возврате определенного значения. Но его не следует использовать вместо сложных ветвлений `if/else`, так как в таких случаях читабельность кода резко ухудшается и вероятность возникновения ошибок только растет.

Правило: Используйте условный тернарный оператор только в тривиальных случаях.

Условный тернарный оператор вычисляется как выражение

Стоит отметить, что условный оператор вычисляется как выражение, в то время как ветвление `if/else` обрабатывается как набор стейтментов. Это означает, что тернарный оператор $?:$ может быть использован там, где `if/else` применить невозможно, например, при инициализации константы:

```
1 bool inBigClassroom = false;
2 const int classSize = inBigClassroom ? 30 : 20;
```

Здесь нельзя использовать `if/else`, так как константы должны быть инициализированы при объявлении, а стейтмент не может быть значением для инициализации.

Оценить статью:



(329 оценок, среднее: 4,94 из 5)



← [Урок №40. Инкремент, декремент и побочные эффекты](#)



[Урок №42. Операторы сравнения](#) →

Комментариев: 5



1. *Андрей:*

[22 мая 2019 в 08:58](#)

Всем доброго времени суток!

при программировании Ардуино для передачи данных в коде используется `sizeof(data)` для передачи данных в массив.

Не могу понять как это работает. ведь `sizeof(data)` передает только размер а не значение.

```

1  #include <SPI.h>                                // Подключаем библиот
2  #include <nRF24L01.h>                            // Подключаем файл на
3  #include <RF24.h>                                // Подключаем библиот
4  #include <iarduino_4LED.h>                        // Подключаем библиот
5  #include <Servo.h>                                // Подключаем библиот
6  RF24      radio(9, 10);                          // Создаём объект rac
7  iarduino_4LED dispLED(2,3);                      // Создаём объект dis
8  Servo      myservo;                              // Создаём объект mys
9  int        data[2];                              // Создаём массив для
10 void setup(){
11     delay(1000);
12     myservo.attach(4);                            // Подключаем объект
13     dispLED.begin();                              // Иницилируем работу
14     radio.begin();                                // Иницилируем работу
15     radio.setChannel(5);                          // Указываем канал пр
16     radio.setDataRate (RF24_1MBPS);               // Указываем скорости
17     radio.setPALevel  (RF24_PA_HIGH);              // Указываем мощности
18     radio.openReadingPipe (1, 0x1234567890LL);     // Открываем 1 трубу
19     radio.startListening ();                      // Включаем приёмник
20 // radio.stopListening ();                        // Выключаем приёмник
21 }
22 void loop(){
23     if(radio.available()){                        // Если в буфере име
24         radio.read(&data, sizeof(data));          // Читаем данные в ма
25         dispLED.print(data[0]);                   // Выводим показания
26         myservo.write(map(data[1],0,1023,0,180)); // Поворачиваем серво
27     }
28 }
```

[Ответить](#)1. *Степан:*[18 октября 2019 в 01:01](#)

Но сначала передается адрес массива, то есть функция уже имеет доступ к данному массиву. А второй параметр `sizeof(data)`, по моему мнению нужен чтоб узнать количество элементов массива путем деление общего объема памяти массива на объем памяти одного его элемента:

```
1 | int arraySize = sizeof(data) / sizeof(int);
```

[Ответить](#)2. *Константин:*[24 марта 2019 в 14:30](#)

Однозначно это только компилятор скажет:-)

[Ответить](#)3. *александр:*[16 ноября 2018 в 18:35](#)

Скажите, пожалуйста, код(ниже) с использованием "condition operator" грамотно написан ? Спасибо.

```
1 | //Output even and odd numbers.
2 | for (int i{}, j{}; i < 51; i += ((i % 2 == 0) ? 2 : 1), j += ((j % 2 == 0) ? 1
3 |     cout << setw(2) << i << " ..... " << j << endl;
4 | }
```

[Ответить](#)1. *Илья:*[3 ноября 2019 в 08:19](#)

С точки зрения синтаксиса — нормально. Но это нечитаемый ужас...

[Ответить](#)

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

- ☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию
- ☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.






Отправить комментарий

[TELEGRAM](#)  [КАНАЛ](#)

Электронная почта

[ПАБЛИК](#) 

ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020