

Ravesli [Ravesli](#)


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

## Урок №65. Оператор switch

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 1 Авг 2020 |

 106721

[↑](#)  50

На этом уроке мы рассмотрим еще один [оператор управления потоком выполнения программы](#) — оператор switch, а также то, зачем его использовать и как это делать эффективно.

Оглавление:

1. [Зачем использовать оператор switch?](#)
2. [Оператор switch](#)
3. [Лейблы case](#)
4. [Лейбл по умолчанию](#)
5. [switch и fall-through](#)
6. [switch и оператор break](#)
7. [Несколько стейтментов внутри блока switch](#)
8. [Объявление переменной и её инициализация внутри case](#)
9. [Тест](#)

## Зачем использовать оператор switch?

Хоть мы и можем использовать сразу несколько [операторов if/else](#) вместе — читается и смотрится это не очень. Например:

```
1 #include <iostream>
2
3 enum Colors
4 {
5     COLOR_GRAY,
```

```
6     COLOR_PINK,  
7     COLOR_BLUE,  
8     COLOR_PURPLE,  
9     COLOR_RED  
10 };  
11  
12 void printColor(Colors color)  
13 {  
14     if (color == COLOR_GRAY)  
15         std::cout << "Gray";  
16     else if (color == COLOR_PINK)  
17         std::cout << "Pink";  
18     else if (color == COLOR_BLUE)  
19         std::cout << "Blue";  
20     else if (color == COLOR_PURPLE)  
21         std::cout << "Purple";  
22     else if (color == COLOR_RED)  
23         std::cout << "Red";  
24     else  
25         std::cout << "Unknown";  
26 }  
27  
28 int main()  
29 {  
30     printColor(COLOR_BLUE);  
31  
32     return 0;  
33 }
```

Использование ветвления `if/else` для проверки значения одной переменной — практика распространенная, но язык C++ предоставляет альтернативный и более эффективный **условный оператор ветвления `switch`**. Вот вышеприведенная программа, но уже с использованием оператора `switch`:

```
1 #include <iostream>  
2  
3 enum Colors {  
4     COLOR_GRAY,  
5     COLOR_PINK,  
6     COLOR_BLUE,  
7     COLOR_PURPLE,  
8     COLOR_RED  
9 };  
10  
11 void printColor(Colors color)  
12 {  
13     switch (color)  
14     {  
15         case COLOR_GRAY:
```

```
16         std::cout << "Gray";
17         break;
18     case COLOR_PINK:
19         std::cout << "Pink";
20         break;
21     case COLOR_BLUE:
22         std::cout << "Blue";
23         break;
24     case COLOR_PURPLE:
25         std::cout << "Purple";
26         break;
27     case COLOR_RED:
28         std::cout << "Red";
29         break;
30     default:
31         std::cout << "Unknown";
32         break;
33     }
34 }
35
36 int main()
37 {
38     printColor(COLOR_BLUE);
39     return 0;
40 }
```

Общая идея операторов switch проста: выражение оператора switch (например, `switch(color)`) должно производить значение, а каждый **кейс** (англ. «*case*») проверяет это значение на соответствие. Если кейс совпадает с выражением switch, то выполняются инструкции под соответствующим кейсом. Если ни один кейс не соответствует выражению switch, то выполняются инструкции после кейса default (если он вообще указан).

Из-за своей реализации, операторы switch обычно более эффективны, чем цепочки if/else. Давайте рассмотрим это более подробно.

## Оператор switch

Сначала пишем **ключевое слово switch** за которым следует выражение, с которым мы хотим работать. Обычно это выражение представляет собой только одну переменную, но это может быть и нечто более сложное, например,  $nX + 2$  или  $nX - nY$ . Единственное ограничение к этому выражению — оно должно быть интегрального типа данных (т.е. типа char, short, int, long, long long или enum). Переменные [типа с плавающей точкой](#) или неинтегральные типы использоваться не могут.

После выражения switch мы объявляем блок. Внутри блока мы используем **лейблы** (англ. «*labels*») для определения всех значений, которые мы хотим проверять на соответствие выражению. Существуют два типа лейблов.

## Лейблы case

Первый вид лейбла — это **case** (или просто «кейс»), который объявляется с использованием **ключевого слова case** и имеет константное выражение. Константное выражение — это то, которое генерирует константное значение, другими словами: либо литерал (например, 5), либо перечисление (например, COLOR\_RED), либо константу (например, переменную x, которая была объявлена с ключевым словом const).

Константное выражение, находящееся после ключевого слова case, проверяется на равенство с выражением, находящимся возле ключевого слова switch. Если они совпадают, то тогда выполняется код под соответствующим кейсом.

Стоит отметить, что все выражения case должны производить уникальные значения. То есть вы не сможете сделать следующее:

```
1 switch (z)
2 {
3     case 3:
4     case 3: // нельзя, значение 3 уже используется!
5     case COLOR_PURPLE: // нельзя, COLOR_PURPLE вычисляется как 3!
6 };
```

Можно использовать сразу несколько кейсов для одного выражения. Следующая функция использует несколько кейсов для проверки, соответствует ли параметр p числу из [ASCII-таблицы](#):

```
1 bool isDigit(char p)
2 {
3     switch (p)
4     {
5         case '0': // если p = 0
6         case '1': // если p = 1
7         case '2': // если p = 2
8         case '3': // если p = 3
9         case '4': // если p = 4
10        case '5': // если p = 5
11        case '6': // если p = 6
12        case '7': // если p = 7
13        case '8': // если p = 8
14        case '9': // если p = 9
15            return true; // возвращаем true
16        default: // в противном случае, возвращаем false
17            return false;
18    }
19 }
```

В случае, если p является числом из ASCII-таблицы, то выполнится первый кейс — return true;.

## Лейбл по умолчанию

Второй тип лейбла — это **лейбл по умолчанию** (так называемый *«default case»*), который объявляется с использованием **ключевого слова default**. Код под этим лейблом выполняется, если ни один из кейсов не соответствует выражению switch. Лейбл по умолчанию является необязательным. В одном switch может быть только один default. Обычно его объявляют самым последним в блоке switch.

В вышеприведенном примере, если `p` не является числом из ASCII-таблицы, то тогда выполняется лейбл по умолчанию и возвращается `false`.

## switch и fall-through

Одна из самых каверзных вещей в switch — это последовательность выполнения кода. Когда кейс совпал (или выполняется default), то выполнение начинается с первого кейса, который находится после соответствующего кейса и продолжается до тех пор, пока не будет выполнено одно из следующих условий завершения:

- Достигнут конец блока switch.
- Выполняется оператор `return`.
- Выполняется оператор `goto`.
- Выполняется оператор `break`.

Обратите внимание, если ни одного из этих условий завершения не будет, то выполняться будут все кейсы после того кейса, который совпал с выражением switch. Например:

```
1 switch (2)
2 {
3     case 1: // Не совпадает!
4         std::cout << 1 << '\n'; // пропускается
5     case 2: // Совпало!
6         std::cout << 2 << '\n'; // выполнение кода начинается здесь
7     case 3:
8         std::cout << 3 << '\n'; // это также выполнится
9     case 4:
10        std::cout << 4 << '\n'; // и это
11    default:
12        std::cout << 5 << '\n'; // и это
13 }
```

Результат:

```
2
3
```

4  
5

А это точно не то, что нам нужно! Когда выполнение переходит из одного кейса в следующий, то это называется **fall-through**. Программисты почти никогда не используют fall-through, поэтому в редких случаях, когда это все-таки используется — программист оставляет [комментарий](#), в котором сообщает, что fall-through является преднамеренным.

## switch и оператор break

**Оператор break** (объявленный с использованием **ключевого слова break**) сообщает компилятору, что мы уже сделали всё, что хотели с определенным switch-ом (или циклом while, do while или for) и больше не намерены с ним работать. Когда компилятор встречает оператор break, то выполнение кода переходит из switch на следующую строку после блока switch. Рассмотрим вышеприведенный пример, но уже с корректно вставленными операторами break:

```
1 switch (2)
2 {
3     case 1: // не совпадает - пропускается
4         std::cout << 1 << '\n';
5         break;
6     case 2: // совпало! Выполнение начинается со следующего стейтмента
7         std::cout << 2 << '\n'; // выполнение начинается здесь
8         break; // оператор break завершает выполнение switch-a
9     case 3:
10        std::cout << 3 << '\n';
11        break;
12    case 4:
13        std::cout << 4 << '\n';
14        break;
15    default:
16        std::cout << 5 << '\n';
17        break;
18 }
19 // Выполнение продолжается здесь
```

Поскольку второй кейс соответствует выражению switch, то выводится 2, и оператор break завершает выполнение блока switch. Остальные кейсы пропускаются.

**Предупреждение:** Не забывайте использовать оператор break в конце каждого кейса. Его отсутствие — одна из наиболее распространенных ошибок новичков!

## Несколько стейтментов внутри блока switch

Еще одна странность в switch заключается в том, что вы можете использовать несколько стейтментов под каждым кейсом, не определяя новый блок:

```
1 switch (3)
2 {
3     case 3:
4         std::cout << 3;
5         boo();
6         std::cout << 4;
7         break;
8     default:
9         std::cout << "default case\n";
10        break;
11 }
```

## Объявление переменной и её инициализация внутри case

Вы можете объявлять, но не инициализировать переменные внутри блока case:

```
1 switch (x)
2 {
3     case 1:
4         int z; // ок, объявление разрешено
5         z = 5; // ок, операция присваивания разрешена
6         break;
7
8     case 2:
9         z = 6; // ок, переменная z была объявлена выше, поэтому мы можем использовать
10        break;
11
12    case 3:
13        int c = 4; // нельзя, вы не можете инициализировать переменные внутри case
14        break;
15
16    default:
17        std::cout << "default case" << std::endl;
18        break;
19 }
```

Обратите внимание, что, хотя переменная `z` была определена в первом кейсе, она также используется и во втором кейсе. Все кейсы считаются частью одной и той же **области видимости**, поэтому, объявив переменную в одном кейсе, мы можем спокойно использовать её без объявления и в других кейсах.

Это может показаться немного нелогичным, поэтому давайте рассмотрим это детально. Когда мы определяем **локальную переменную**, например, `int y;`, то переменная не создается в этой точке — она фактически создается в начале блока, в котором объявлена. Однако, она не видна в программе до точки объявления. Само объявление не выполняется, оно просто сообщает компилятору, что переменная уже может использоваться в коде. Поэтому переменная, объявленная в одном кейсе, может использоваться в другом кейсе, даже если кейс, объявляющий переменную, никогда не выполняется.

Однако инициализация переменных непосредственно в кейсах запрещена и вызовет ошибку компиляции. Это связано с тем, что инициализация переменной *требует выполнения*, а кейс, содержащий инициализацию, может никогда не выполниться!

Если в кейсе нужно объявить и/или инициализировать новую переменную, то это лучше всего сделать, используя **блок стейтментов** внутри кейса:

```
1  switch (1)
2  {
3      case 1:
4          { // обратите внимание, здесь указан блок
5              int z = 5; // хорошо, переменные можно инициализировать внутри блока, который
6                  std::cout << z;
7                  break;
8          }
9      default:
10         std::cout << "default case" << std::endl;
11         break;
12 }
```

**Правило:** Если нужно инициализировать и/или объявить переменные внутри кейса — используйте блоки стейтментов.

## Тест

### Задание №1

Напишите функцию `calculate()`, которая принимает две переменные типа `int` и одну переменную типа `char`, которая, в свою очередь, представляет одну из следующих математических операций: `+`, `-`, `*`, `/` или `%` (остаток от числа). Используйте `switch` для выполнения соответствующей математической операции над целыми числами, а результат возвращайте обратно в `main()`. Если в функцию передается недействительный математический оператор, то функция должна выводить ошибку. С оператором деления выполняйте целочисленное деление.

### Ответ №1

```
1  #include <iostream>
2
3  int calculate(int x, int y, char op)
4  {
5      switch (op)
6      {
7          case '+':
8              return x + y;
9          case '-':
10             return x - y;
11          case '*':
12             return x * y;
13          case '/':
```



```
14         return x / y;
15     case '%':
16         return x % y;
17     default:
18         std::cout << "calculate(): Unhandled case\n";
19         return 0;
20     }
21 }
22
23 int main()
24 {
25     std::cout << "Enter an integer: ";
26     int x;
27     std::cin >> x;
28
29     std::cout << "Enter another integer: ";
30     int y;
31     std::cin >> y;
32
33     std::cout << "Enter a mathematical operator (+, -, *, /, or %): ";
34     char op;
35     std::cin >> op;
36
37     std::cout << x << " " << op << " " << y << " is " << calculate(x, y, op) << "\n";
38
39     return 0;
40 }
```

## Задание №2

Определите перечисление (или класс enum) `Animal`, которое содержит следующих животных: `pig`, `chicken`, `goat`, `cat`, `dog` и `ostrich`. Напишите функцию `getAnimalName()`, которая принимает параметр `Animal` и использует `switch` для возврата типа животного в качестве строки. Напишите еще одну функцию — `printNumberOfLegs()`, которая использует `switch` для вывода количества лап соответствующего типа животного. Убедитесь, что обе функции имеют кейс `default`, который выводит сообщение об ошибке. Вызовите `printNumberOfLegs()` в `main()`, используя в качестве параметров `cat` и `chicken`.

Пример результата выполнения вашей программы:

```
A cat has 4 legs.
A chicken has 2 legs.
```

## Ответ №2

```
1 #include <iostream>
2 #include <string>
3
4 enum Animal
5 {
```

```
6     ANIMAL_PIG,
7     ANIMAL_CHICKEN,
8     ANIMAL_GOAT,
9     ANIMAL_CAT,
10    ANIMAL_DOG,
11    ANIMAL_OSTRICH
12 };
13
14 std::string getAnimalName(Animal animal)
15 {
16     switch (animal)
17     {
18         case ANIMAL_CHICKEN:
19             return "chicken";
20         case ANIMAL_OSTRICH:
21             return "ostrich";
22         case ANIMAL_PIG:
23             return "pig";
24         case ANIMAL_GOAT:
25             return "goat";
26         case ANIMAL_CAT:
27             return "cat";
28         case ANIMAL_DOG:
29             return "dog";
30
31         default:
32             return "getAnimalName(): Unhandled enumerator";
33     }
34 }
35
36 void printNumberOfLegs(Animal animal)
37 {
38     std::cout << "A " << getAnimalName(animal) << " has ";
39
40     switch (animal)
41     {
42         case ANIMAL_CHICKEN:
43         case ANIMAL_OSTRICH:
44             std::cout << "2";
45             break;
46
47         case ANIMAL_PIG:
48         case ANIMAL_GOAT:
49         case ANIMAL_CAT:
50         case ANIMAL_DOG:
51             std::cout << "4";
52             break;
53
54         default:
```

```
55         std::cout << "printNumberOfLegs(): Unhandled enumerator";
56         break;
57     }
58
59     std::cout << " legs.\n";
60 }
61
62 int main()
63 {
64     printNumberOfLegs(ANIMAL_CAT);
65     printNumberOfLegs(ANIMAL_CHICKEN);
66
67     return 0;
68 }
```

Оценить статью:

★★★★★ (273 оценок, среднее: 4,87 из 5)



[← Урок №64. Операторы условного ветвления if/else](#)

[Урок №66. Оператор goto](#)



**Комментариев: 50**



1. **Юрий:**  
[25 ноября 2020 в 21:55](#)

2 задача

```
1  #include <iostream>
2  enum class Animals
3  {
4      PIG,
5      CHICKEN,
6      CAT,
7      DOG,
8      OSTRICH,
9  };
10 std:: string getAnimalName(Animals animal)
11 {
12     switch (animal)
```

```
13     {
14     case Animals::PIG:
15         return "Свинья";
16         break;
17     case Animals::CHICKEN:
18         return "Курица";
19         break;
20     case Animals::CAT:
21         return "Кот";
22         break;
23     case Animals::DOG:
24         return "Пёс";
25         break;
26     case Animals::OSTRICH:
27         return "Страус";
28         break;
29     default:
30         return "Ошибка";
31         break;
32     }
33 }
34 std::string printNumberOfLegs(Animals animal)
35 {
36     switch (animal)
37     {
38     case Animals::PIG:
39         return "Четыре лапы.";
40         break;
41     case Animals::CHICKEN:
42         return "Две лапы.";
43         break;
44     case Animals::CAT:
45         return "Четыре лапы.";
46         break;
47     case Animals::DOG:
48         return "Четыре лапы.";
49         break;
50     case Animals::OSTRICH:
51         return "Две лапы.";
52         break;
53     default:
54         return "Ошибка.";
55         break;
56     }
57 }
58 void result(Animals animal)
59 {
60     std::cout << getAnimalName(animal) << " имеет " << printNumberOfLegs(animal) << "\n";
61 }
```

```
62 int main()
63 {
64     setlocale(LC_ALL, "rus");
65     result(Animals::PIG);
66     result(Animals::CHICKEN);
67     result(Animals::CAT);
68     result(Animals::DOG);
69     result(Animals::OSTRICH);
70 }
```

[Ответить](#)



2. *Инкогнито:*

[29 октября 2020 в 19:29](#)

```
1  #include <iostream>
2  #include <string>
3
4  enum class Animals {
5      pig,
6      chicken,
7      goat,
8      cat,
9      dog,
10     ostrich
11 };
12
13 std::string getAnimal(Animals animal) {
14     std::string name;
15     switch (animal)
16     {
17     case Animals::pig:
18         return name = " pig ";
19         break;
20     case Animals::chicken:
21         return name = " chicken ";
22         break;
23     case Animals::goat:
24         return name = " goat ";
25         break;
26     case Animals::cat:
27         return name = " cat ";
28         break;
29     case Animals::dog:
30         return name = " dog ";
31         break;
32     case Animals::ostrich:
33         return name = " ostrich ";
```

```
34         break;
35     default: {
36         std::cout << "Error";
37         return 0;
38     }
39     break;
40 }
41 }
42 }
43 void printNumberOfLegs(std::string name) {
44
45     int legs;
46     switch (name)
47     {
48     case " pig ":
49         legs = 4;
50         break;
51     case " chicken ":
52         legs = 2;
53         break;
54     case " goat" :
55         legs = 4;
56         break;
57     case " cat ":
58         legs = 4;
59         break;
60     case " dog ":
61         legs = 4;
62         break;
63     case " ostrich ":
64         legs = 2;
65         break;
66     default: {
67         std::cout << "Error";
68         break;
69     }
70         std::cout << name << "has" << legs << "legs.";
71     }
72 }
73 int main() {
74
75     printNumberOfLegs(getAnimal(Animals::cat));
76     return 0;
77 }
```

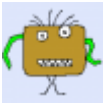
Знаю, что структура не самая лучшая, но не понимаю из-за чего выдает ошибку:

46 строка: switch (name)

name — expression must have integral or enum type;

И к каждому кейсу этой же функции — this constant expression has type "const char \*" instead of the required "std::string" type

[Ответить](#)



1. *Inkogito:*

[29 ноября 2020 в 11:32](#)

В конструкции switch в case вариантах можно использовать только численные значения или перечисления, перевод ошибки компилятора)

[Ответить](#)



3. *Ruslan:*

[7 сентября 2020 в 14:12](#)

Задание №2

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <string>
4
5  enum class Animal
6  {
7      pig,
8      chicken,
9      goat,
10     cat,
11     dog,
12     ostrich
13 };
14
15 std::string get_animal_name (Animal animal)
16 {
17     switch (animal)
18     {
19         case (Animal::pig):
20             return "A pig " ;
21         case (Animal::chicken):
22             return "A chicken ";
23         case (Animal::goat):
24             return "A goat ";
25         case (Animal::cat):
26             return "A cat ";
27         case (Animal::dog):
28             return "A dog ";
29         case (Animal::ostrich):
30             return "A ostrich ";
```

```
31         default:
32             std::cout << "Error: wrong animal!\n";
33             std::exit (0);
34     }
35 }
36
37 void print_number_of_legs (Animal animal)
38 {
39     switch (animal)
40     {
41         case (Animal::pig):
42             std::cout << get_animal_name (Animal::pig);
43             std::cout << "has 4 legs.\n";
44             break;
45         case (Animal::chicken):
46             std::cout << get_animal_name (Animal::chicken);
47             std::cout << "has 2 legs.\n";
48             break;
49         case (Animal::goat):
50             std::cout << get_animal_name (Animal::goat);
51             std::cout << "has 4 legs.\n";
52             break;
53         case (Animal::cat):
54             std::cout << get_animal_name (Animal::cat);
55             std::cout << "has 4 legs.\n";
56             break;
57         case (Animal::dog):
58             std::cout << get_animal_name (Animal::dog);
59             std::cout << "has 4 legs.\n";
60             break;
61         case (Animal::ostrich):
62             std::cout << get_animal_name (Animal::ostrich);
63             std::cout << "has 2 legs.\n";
64             break;
65         default:
66             std::cout << "Errorr: wrong animal!\n";
67             std::exit (0);
68     }
69 }
70
71 Animal input_name_animal ()
72 {
73     std::cout << "Enter animal: pig, chicken, goat, cat, dog, ostrich)\n";
74     std::string name {};
75     std::cin >> name;
76     std::cout << '\a';
77
78     if (name == "pig")
79         return Animal::pig;
```



```
80     else if (name == "chicken")
81         return Animal::chicken;
82     else if (name == "goat")
83         return Animal::goat;
84     else if (name == "cat")
85         return Animal::cat;
86     else if (name == "dog")
87         return Animal::dog;
88     else if (name == "ostrich")
89         return Animal::ostrich;
90     else
91     {
92         std::cout << "Error: wrong animal!\n";
93         std::cout << "Try again...\n";
94         return (input_name_animal ());
95     }
96 }
97
98
99 int main ()
100 {
101     Animal animal {};
102     animal = input_name_animal ();
103
104     print_number_of_legs (animal);
105
106     return 0;
107 }
```

#### [Ответить](#)



4. *Ruslan:*

[7 сентября 2020 в 14:10](#)

#### Задание №1

```
1  #include <iostream>
2  #include <cstdlib>
3
4  int calculate (int first_var, int last_var, char math_op)
5  {
6      switch (math_op)
7      {
8          case ('+'):
9              return (first_var + last_var);
10         case ('-'):
11             return (first_var - last_var);
12         case ('*'):
```

```
13         return (first_var * last_var);
14     case ('/'):
15     {
16         if (last_var != 0)
17             return (first_var / last_var);
18         else
19         {
20             std::cout << "Errorr: wrong number! Division by zero!\n";
21             std::exit (0);
22         }
23     }
24
25
26     case ('%'):
27     {
28         if (last_var != 0)
29             return (first_var % last_var);
30         else
31         {
32             std::cout << "Errorr: wrong number! Division by zero!\n";
33             std::exit (0);
34         }
35     }
36     default:
37         std::cout << "Errorr: wrong math operator!\n";
38         std::exit (0);
39 }
40
41
42 int main ()
43 {
44     std::cout << "Enter integer last number\n";
45     int a {};
46     std::cin >> a;
47
48     std::cout << "Enter integer first number\n";
49     int b {};
50     std::cin >> b;
51
52     std::cout << "Enter math operator: +, -, *, /, or % \n";
53     char op {};
54     std::cin >> op;
55
56     std::cout << "Your result:\n";
57     std::cout << a << ' ' << op << ' ' << b << ' ' << '=' << ' ' ;
58     std::cout << calculate (a, b, op) << std::endl;
59
60     return 0;
61 }
```

62 | }

[Ответить](#)5. *Rock:*[29 июля 2020 в 15:53](#)

```
1  #include<iostream>
2
3  enum  Animals
4  {
5      CAT,
6      PIG,
7      CHICKEN,
8      DOG,
9      COW,
10     HORSE,
11     BAG,
12 };
13
14 std::string getAnimalName(Animals animal)
15 {
16     switch (animal)
17     {
18         case CAT :
19         {
20             return "Кошка ";
21         }
22         case PIG :
23         {
24             return "Свинья ";
25         }
26         case CHICKEN:
27         {
28             return "Курица ";
29         }
30         case DOG:
31         {
32             return "Собака ";
33         }
34         case COW:
35         {
36             return "Корова ";
37         }
38         case HORSE:
39         {
40             return "Лошадь ";
41         }
```

```
42         default:
43         {
44             std::cout << "Животное которое вы ввели не найдено" << std::endl;
45             exit(1);
46         }
47
48     }
49
50 }
51
52 std::string printNumberOfLegs(Animals animal)
53 {
54     switch (animal)
55     {
56     case CAT:
57     {
58         return "имеет 4 лапы." ;
59     }
60     case PIG:
61     {
62         return "имеет 4 лапы.";
63     }
64     case CHICKEN:
65     {
66         return "имеет 2 лапы.";
67     }
68     case DOG:
69     {
70         return "имеет 4 лапы.";
71     }
72     case COW:
73     {
74         return "имеет 4 лапы.";
75     }
76     case HORSE:
77     {
78         return "имеет 4 лапы.";
79     }
80
81     }
82 }
83
84
85 void infoOfAnimals(Animals animal)
86 {
87     std::cout << getAnimalName(animal);
88     std::cout << printNumberOfLegs(animal)<<std::endl;
89 }
90
```

```
91 int main()
92 {
93     setlocale(0, "");
94
95     infoOfAnimals(CAT);
96     infoOfAnimals(CHICKEN);
97
98     return 0;
99 }
```

[Ответить](#)6. *Viktor:*[9 июля 2020 в 20:42](#)

Так и не понятно, можно инициализировать переменную в "case" или нет!? Автор пишет что нет и даёт объяснение, что "case" может не сработать. Каким образом тогда, в следующем абзаце идёт речь об инициализации внутри блока "case"? Как блок может снимать запрет!?

[Ответить](#)1. *Александр:*[14 августа 2020 в 14:13](#)

У блока локальная область видимости

[Ответить](#)7. *Onium:*[22 июня 2020 в 20:24](#)

Написал второе задание через структуры

```
1  #include <iostream>
2
3  enum class Animal
4  {
5      pig,
6      chicken,
7      goat,
8      cat,
9      dog,
10     ostrich,
11 };
12
13 struct Animals
14 {
15     Animal name;
```

```
16     int legs;
17 };
18
19
20 std::string getAnimalName(Animals name)
21 {
22     switch(name.name)
23     {
24     case Animal::pig:
25     {
26         return "pig" ;
27         break;
28     }
29     case Animal::chicken:
30     {
31         return "chicken";
32         break;
33     }
34     case Animal::goat:
35     {
36         return "goat";
37         break;
38     }
39     case Animal::cat:
40     {
41         return "cat";
42         break;
43     }
44     case Animal::dog:
45     {
46         return "dog";
47         break;
48     }
49     case Animal::ostrich:
50     {
51         return "ostrich";
52         break;
53     }
54     default:
55         return "Error";
56     }
57 }
58
59 void printNumberOfLegs(Animals pet)
60 {
61     std::cout << "A " << getAnimalName(pet) << " has " << pet.legs << " legs" <<
62 }
63
64
```

```
65 int main()
66 {
67     Animals cat{ Animal::cat, 4 };
68     Animals chicken{ Animal::chicken, 2 };
69
70     printNumberOfLegs(cat);
71     printNumberOfLegs(chicken);
72
73
74     return 0;
75 }
```

### Ответить



8. Яна:

[12 июня 2020 в 12:48](#)

Имеет ли жизнь такой вариант? Вроде как сразу видно что выводится. Или функцию main() лучше не засорять тем, что можно сделать в функциях об назв. животных и их лап?

```
1  #include <iostream>
2  #include <string>
3
4  using std::cout;
5  using std::string;
6
7  enum Animal
8  {
9      ANIMAL_PIG,
10     ANIMAL_CHICKEN,
11     ANIMAL_GOAT,
12     ANIMAL_CAT,
13     ANIMAL_DOG
14 };
15
16 string printNumberOfLegs(Animal leg)
17 {
18     switch (leg)
19     {
20     case Animal::ANIMAL_PIG:
21     case Animal::ANIMAL_CAT:
22     case Animal::ANIMAL_DOG:
23         return ("4");
24         break;
25     case Animal::ANIMAL_CHICKEN:
26     case Animal::ANIMAL_GOAT:
27         return ("2");
28         break;
```

```
29
30     default:
31         cout << "Произошла ошибка!";
32         break;
33     }
34     return 0;
35 }
36
37 string getAnimalName (Animal animal)
38 {
39
40     switch (animal)
41     {
42     case ANIMAL_PIG:
43         return ("Свинка");
44         break;
45     case ANIMAL_CHICKEN:
46         return ("Курица");
47         break;
48     case ANIMAL_GOAT:
49         return ("Коза");
50         break;
51     case ANIMAL_CAT:
52         return ("Кот");
53         break;
54     case ANIMAL_DOG:
55         return ("Собака");
56         break;
57     default:
58         cout << "Произошла ошибка!";
59         break;
60     }
61     return 0;
62 }
63
64 int main()
65 {
66     setlocale(0, "");
67
68     cout << getAnimalName(ANIMAL_CAT) << " имеет " << printNumberOfLegs(ANIMAL_CAT);
69     cout << getAnimalName(ANIMAL_GOAT) << " имеет " << printNumberOfLegs(ANIMAL_GOAT);
70 }
```

[Ответить](#)



9. Борис:

[30 апреля 2020 в 15:41](#)



Первый пример в статье: как switch может быть эффективнее, если с ним аж на 7 строк больше? 😊  
Притом что читабельность совсем не улучшилась (а может, даже ухудшилась из-за постоянных break). Очень редко юзаю switch.

[Ответить](#)



10. *mloborev:*

[19 марта 2020 в 15:24](#)

```
1  #include <iostream>
2
3  int calculate(int a, char op, int b)
4  {
5      switch (op)
6      {
7          case '+':
8              return a + b; break;
9          case '-':
10             return a - b; break;
11          case '*':
12             return a * b; break;
13          case '/':
14             return a / b; break;
15          case '%':
16             return a % b; break;
17          default:
18             std::cout << "Wrong symbol. Operation can't be completed.\n";
19             exit(0);
20      }
21 }
22
23 int main()
24 {
25     int a, b;
26     char op;
27     std::cout << "Print down two integers and operation symbol between them: ";
28     std::cin >> a >> op >> b;
29
30     std::cout << "Result: " << calculate(a, op, b) << std::endl;
31     return 0;
32 }
```

[Ответить](#)



11. *Inviser666:*

[18 января 2020 в 13:42](#)

№1. по-моему, неплохо)

```
1 #include <iostream>
2 #include <cmath>
3 #include <cstdlib>
4
5 double calculate(int x, char op, int y);
6 int enternum();
7 char enterop();
8
9 int main() {
10     std::cout << "it's calculator" << std::endl;
11     std::cout << std::endl;
12
13     int x = enternum();
14     char op = enterop();
15     int y = enternum();
16
17     double calc = calculate(x,op,y);
18     std::cout << "result is " << calc << std::endl;
19
20     system("pause");
21 }
22 double calculate(int x, char op, int y) {
23     double j = x;
24     switch (op) {
25         case '+': {
26             return x + y;
27         }
28         case '-': {
29             return x - y;
30         }
31         case '/': {
32             return j / y;
33         }
34         case '*': {
35             return x * y;
36         }
37         case '%': {
38             return x % y;
39         }
40         default:
41             std::cout << "Error";
42             break;
43     }
44 }
45 int enternum() {
46     std::cout << "enter a number: ";
47     int x;
48     std::cin >> x;
```

```

49     return x;
50 }
51 char enterop() {
52     std::cout << "enter an operation (+,-,*,/, %): ";
53     char x;
54     std::cin >> x;
55     return x;
56 }

```

Ответить

1. *Константин:*  
[26 января 2020 в 11:47](#)

ключевое слово break -то где, а?

Ответить

1. *Inviser666:*  
[26 января 2020 в 12:33](#)

а зачем писать break после return?

Ответить

1. *Константин:*  
[29 января 2020 в 18:41](#)

хочешь по лучше спрятать вещь — поставь её на самое видное место — ретурна я то и не просёк:-)



2. *foo:*  
[8 сентября 2020 в 19:24](#)

не бойсь, компилятор выкинуть должен ненужное, и бинарный код будет таким же, как и без break.



12. *armus1:*  
[23 ноября 2019 в 22:02](#)

```

1 #include <iostream>
2 #include <string>
3
4 enum class Animal //определяем класс Animal с животными
5 {

```

```
6     PIG,
7     CHICKEN,
8     GOAT,
9     CAT,
10    DOG,
11    OSTRICH
12 };
13
14 std::string getAnimalName(Animal animals) //пишем функцию для возврата типа животного
15 {
16     using namespace std;
17
18     switch (animals)
19     {
20     case Animal::PIG:
21         return "pig";
22         break;
23     case Animal::CHICKEN:
24         return "chicken";
25         break;
26     case Animal::GOAT:
27         return "goat";
28         break;
29     case Animal::CAT:
30         return "cat";
31         break;
32     case Animal::DOG:
33         return "dog";
34         break;
35     case Animal::OSTRICH:
36         return "ostrich";
37         break;
38     default:
39         return "ERROR";
40         break;
41     }
42 }
43
44 void printNumberOfLegs(Animal animals) //пишем функцию для вывода количества лап
45 {
46     using namespace std;
47
48     switch (animals)
49     {
50     case Animal::PIG:
51         cout << "A" << getAnimalName(animals) << "has 4 legs" << endl;
52         break;
53     case Animal::CHICKEN:
```

```

55     cout << "A" << getAnimalName(animals) << "has 2 legs" << endl;
56     break;
57     case Animal::GOAT:
58         cout << "A" << getAnimalName(animals) << "has 4 legs" << endl;
59         break;
60     case Animal::CAT:
61         cout << "A" << getAnimalName(animals) << "has 4 legs" << endl;
62         break;
63     case Animal::DOG:
64         cout << "A" << getAnimalName(animals) << "has 4 legs" << endl;
65         break;
66     case Animal::OSTRICH:
67         cout << "A" << getAnimalName(animals) << "has 2 legs" << endl;
68         break;
69     default:
70         cout << "ERROR" << endl;
71         break;
72     }
73 }
74
75 int main()
76 {
77     using namespace std;
78
79     printNumberOfLegs(Animal::CAT);
80     printNumberOfLegs(Animal::CHICKEN);
81
82     return 0;
83 }

```

### [Ответить](#)



13. *armus1:*

[21 ноября 2019 в 19:59](#)

```

1  #include <iostream>
2
3  int Calculate()//пишем функцию, запрашивающую два целых числа у пользователя и ма
4  {
5      using namespace std;
6
7      cout << "Enter the first integer:" << endl;
8      int a;
9      cin >> a;
10
11     cout << "Enter the second integer:" << endl;
12     int b;
13     cin >> b;

```

```
14
15     cout << "Select one of the following mathematical operations: +, -, *, / or %"
16     char ch;
17     cin >> ch;
18
19     switch (ch)//определяем ветвление switch для выполнения математической операц
20     {
21         case '+':
22             cout << a + b;
23             return a + b;
24             break;
25         case '-':
26             cout << a - b;
27             return a - b;
28             break;
29         case '*':
30             cout << a * b;
31             return a * b;
32             break;
33         case '/':
34             cout << a / b;
35             return a / b;
36             break;
37         case '%':
38             cout << a % b;
39             return a % b;
40             break;
41         default:
42             cout << "ERROR: this mathematical operator is incorrect";
43             break;
44     }
45 }
46
47 int main()
48 {
49     using namespace std;
50     Calculate();
51
52     return 0;
53 }
```

[Отвечить](#)



14. Алексей:

[3 ноября 2019 в 19:23](#)

Добрый день Юрий.

Подобное пишу на bash. Правда можно было гораздо оптимальнее, но не будет об этом думать, учимся.

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <windows.h>
4
5  int input(int &x)
6  {
7      system("CLS");
8      std::cout << "Enter value: " << std::endl;
9      std::cin >> x;
10     system("CLS");
11     std::cout << "Value added!" << std::endl;
12     Sleep(1000);
13
14     return x;
15 }
16
17 char oper(char &op)
18 {
19     system("CLS");
20     std::cout << "Enter operator - \"+\", \"-\", \"*\", \"/\", \"%\": " << std::endl;
21     std::cin >> op;
22     system("CLS");
23     std::cout << "Operator added!" << std::endl;
24     Sleep(1000);
25
26     return op;
27 }
28
29 void mathOperation(const int &x, const char &op, const int &y)
30 {
31     switch (op)
32     {
33     case '+': std::cout << "Your result is " << x + y << std::endl; break;
34     case '-': std::cout << "Your result is " << x - y << std::endl; break;
35     case '*': std::cout << "Your result is " << x * y << std::endl; break;
36     case '/': std::cout << "Your result is " << x / y << std::endl; break;
37     case '%': std::cout << "Your result is " << x / y << std::endl; break;
38     }
39 }
40
41
42 int main()
43 {
44     int x = input(x);
45     int y = input(y);
46     char op = oper(op);
```

```
47
48     system("CLS");
49     mathOperation(x, op, y);
50
51     system("pause");
52     return 0;
53 }
```

### [Ответить](#)



15. *Игорь:*

[19 сентября 2019 в 13:06](#)

```
1  //2 задание!
2  //хотелось сделать его немного посложнее для себя(добавил пользовательский ввод ж
3
4  #include "pch.h"
5  #include <iostream>
6  #include <string>
7  using namespace std;
8
9  enum class Animals
10 {
11     PIG,
12     CHIKEN,
13     GOAT,
14     CAT,
15     DOG,
16     OSTRICH,
17 };
18
19 string getAnimalName(Animals animals)//функция возвращает тип животного в качестве
20 {
21     switch (animals)
22     {
23     case Animals::CAT:
24         return "Cat";
25     case Animals::CHIKEN:
26         return "Chicken";
27     case Animals::DOG:
28         return "Dog";
29     case Animals::GOAT:
30         return "Goat";
31     case Animals::OSTRICH:
32         return "Ostrich";
33     case Animals::PIG:
34         return "Pig";
35     default://тут это не обязательно
```



```
36     return "Error:Incorrect enum animals!";
37 }
38 }
39
40 void printNumberOfLegs(Animals legs)//функция выводит количества лапок соответств
41 {
42     cout << "A " << getAnimalName(legs) << " has ";
43     switch (legs)
44     {
45     case Animals::CHIKEN:
46     case Animals::OSTRICH:
47         cout << "2 ";
48         break;
49     case Animals::CAT:
50     case Animals::DOG:
51     case Animals::GOAT:
52     case Animals::PIG:
53         cout << "4 ";
54         break;
55     default://тут это не обязательно
56         cout << "Error: incorrect enum animals!";
57     }
58     cout << "legs.\n";
59 }
60
61 int main()
62 {
63     string x; //в эту переменную запишем пользовательский ввод
64     //выводим список животных(я думаю можно более простым способом вывести, но я
65     cout << getAnimalName(Animals::CAT)<<"\n";
66     cout << getAnimalName(Animals::CHIKEN)<<"\n";
67     cout << getAnimalName(Animals::DOG)<<"\n";
68     cout << getAnimalName(Animals::GOAT)<<"\n";
69     cout << getAnimalName(Animals::OSTRICH)<<"\n";
70     cout << getAnimalName(Animals::PIG)<<"\n";
71     cout << "\nEnter an animal from the list: ";
72     cin >> x; //пользователь вводит животное с клавиатуры
73     //проверяем, если совпадений по животным нет, то выводим ошибку
74     if (x != getAnimalName(Animals::CAT) && x != getAnimalName(Animals::CHIKEN) &&
75         cout << "Error: incorrect enum animals!\n";
76     else
77     {
78         //если же есть совпадение, то выводим инфу по соответствующему животному
79         if (x == getAnimalName(Animals::CAT))
80             printNumberOfLegs(Animals::CAT);
81         if (x == getAnimalName(Animals::CHIKEN))
82             printNumberOfLegs(Animals::CHIKEN);
83         if (x == getAnimalName(Animals::DOG))
84             printNumberOfLegs(Animals::DOG);
```

```

85     printNumberOfLegs(Animals::DOG);
86     if (x == getAnimalName(Animals::GOAT))
87         printNumberOfLegs(Animals::GOAT);
88     if (x == getAnimalName(Animals::OSTRICH))
89         printNumberOfLegs(Animals::OSTRICH);
90     if (x == getAnimalName(Animals::PIG))
91         printNumberOfLegs(Animals::PIG);
92
93 }

```

[Ответить](#)

1. Павел:

[4 мая 2020 в 21:20](#)

```

1 | if (x != getAnimalName(Animals::CAT) && x != getAnimalName(Animals::CHICKEN) &&

```

Очень длинная строка, а нельзя просто отправить проверить в списке enum не перечисляя каждое животное?

[Ответить](#)

16. Игорь:

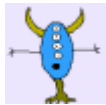
[18 сентября 2019 в 16:46](#)

```

1 //первое задание
2 #include "pch.h"
3 #include <iostream>
4
5
6 int calculate()
7 {
8     using namespace std;
9     setlocale(LC_CTYPE, "rus");
10    int a;
11    int b;
12    char op;
13    cout << "Введите два числа и операцию.\n\n" << "1-е число: ";
14    cin >> a;
15    cout << "2-е число: ";
16    cin >> b;
17    cout << "Введите операцию(+, -, *, /, %): ";
18    cin >> op;
19    if (op != '+' && op != '-' && op != '*' && op != '/' && op != '%')//если введе
20    {
21        return false;

```

```
22     }
23     else
24     {
25         switch (op)//перечисление выбора оператора, если он корректный
26         {
27             case '+':
28                 return a + b;
29                 break;
30             case '-':
31                 return a - b;
32                 break;
33             case '*':
34                 return a * b;
35                 break;
36             case '/':
37                 return a / b;
38                 break;
39             case '%':
40                 return a % b;
41         }
42     }
43 }
44
45 int main()
46 {
47     setlocale(LC_CTYPE, "rus");
48     using namespace std;
49     int x;
50     x = calculate();//присваиваем значение функции calculate переменной x
51     if (x == false)//если функция calculate вернула false, то известит о неко
52         cout << "\nНе корректная операция!";
53     else
54         cout << "\nОтвет: " << x;
55 }
```

[Ответить](#)

17. zashiki:

[28 августа 2019 в 15:44](#)

т.е. если в кейсах есть оператор возврата return, то оператор break не нужен?

[Ответить](#)

18. Максим:

[3 августа 2019 в 23:13](#)

## Оцените пожалуйста

## Задание 1:

```
1  #include <iostream>
2
3  int calculate(int x, int y, char oper)
4  {
5      switch (oper)
6      {
7          case '+':
8              return x + y;
9              break;
10         case '-':
11             return x - y;
12             break;
13         case '*':
14             return x * y;
15             break;
16         case '/':
17             return x / y;
18             break;
19         case '%':
20             return x % y;
21             break;
22         default:
23             std::cout << "Error! Unknown operator";
24             return 0;
25             break;
26     }
27 }
28
29 int getNumber()
30 {
31     std::cout << "Enter an integer: ";
32     int num;
33     std::cin >> num;
34     return num;
35 }
36
37 char getOperator()
38 {
39     std::cout << "Enter the operator: ";
40     char oper;
41     std::cin >> oper;
42     return oper;
43 }
44
45 int main()
```

```
46 {
47     int x = getNumber();
48     int y = getNumber();
49
50     char oper = getOperator();
51
52     std::cout << "Take your answer: " << x << " " << oper << " " << y << " = " <<
53
54     return 0;
55 }
```

Задание 2 (я усовершенствовал до ввода пользователем данных)

```
1  #include <iostream>
2  #include <string>
3
4  enum class Animal
5  {
6      PIG = 1,
7      CHICKEN,
8      GOAT,
9      CAT,
10     DOG,
11     OSTRICH,
12 };
13
14 void startInfo()
15 {
16     std::cout << "Welcome!!!\n\n";
17     std::cout << "1 - Pig\n";
18     std::cout << "2 - Chicken\n";
19     std::cout << "3 - Goat\n";
20     std::cout << "4 - Cat\n";
21     std::cout << "5 - Dog\n";
22     std::cout << "6 - Ostrich\n\n";
23 }
24
25 short getAnimalCode(std::string order)
26 {
27     std::cout << "Choose the " << order << " animal (enter the code): ";
28     short code;
29     std::cin >> code;
30     std::cout << std::endl;
31     return code;
32 }
33
34 std::string getAnimalName(Animal animal)
35 {
36     switch (animal)
```

```
37     {
38     case Animal::PIG:
39         return "pig";
40         break;
41     case Animal::CHICKEN:
42         return "chicken";
43         break;
44     case Animal::GOAT:
45         return "goat";
46         break;
47     case Animal::CAT:
48         return "cat";
49         break;
50     case Animal::DOG:
51         return "dog";
52         break;
53     case Animal::OSTRICH:
54         return "ostrich";
55         break;
56     default:
57         return "getAnimalName(): Unhandled enumerator!";
58         break;
59     }
60 }
61
62 short getNumberOfLegs(Animal animal)
63 {
64     short fourLegs(4);
65     short twoLegs(2);
66
67     switch (animal)
68     {
69         case Animal::PIG:
70         case Animal::GOAT:
71         case Animal::CAT:
72         case Animal::DOG:
73             return fourLegs;
74
75             break;
76         case Animal::CHICKEN:
77         case Animal::OSTRICH:
78             return twoLegs;
79             break;
80
81         default:
82             std::cout << "getNumberOfLegs(): Unhandled enumerator!";
83     }
84 }
85
```

```
86 void printResult(std::string animalName, short legs)
87 {
88     std::cout << "A " << animalName << " has " << legs << " legs.\n";
89 }
90
91 void printNumberOfLegs(short code)
92 {
93     Animal animal = static_cast<Animal>(code);
94     std::string animalName = getAnimalName(animal);
95     short legs = getNumberOfLegs(animal);
96     printResult(animalName, legs);
97 }
98
99 int main()
100 {
101     startInfo();
102
103     short animal1 = getAnimalCode("first");
104     short animal2 = getAnimalCode("second");
105
106     printNumberOfLegs(animal1);
107     printNumberOfLegs(animal2);
108
109     return 0;
110 }
```

P.S. Я в курсе, что так не желательно делать, но ситуация требовала

```
1 | Animal animal = static_cast<Animal>(code);
```

code типа short

[Ответить](#)



19. Алексей:

[17 июля 2019 в 15:17](#)

Да, это нечто.

Колупал не слабо 2е задание с этим перечислителями и enum.  
Разобрался все же.

Нашел интересную особенность. Если enum с классом, то код весь верный, хотя "enum class" нету, только "enum". Все компилирует.

Господи, простейшие программы для теста. Ото бы не тупил, все элементарно.

[Ответить](#)



20. *Алексей:*

[17 июля 2019 в 14:35](#)

Намного проще, чем думал и писал.

Правда почему здесь нету в конце "break;"?

```
1      case ANIMAL_CHICKEN: return "chicken";
2      case ANIMAL_OSTRICH: return "ostrich";
3      case ANIMAL_PIG:      return "pig";
4      case ANIMAL_GOAT:     return "goat";
5      case ANIMAL_CAT:      return "cat";
6      case ANIMAL_DOG:      return "dog";
```

Это же ошибка.

[Ответить](#)



1. *Борис:*

[30 апреля 2020 в 15:33](#)

Какая ошибка? Там return'ы стоят. А они завершают конструкцию switch аналогично break'у. Поэтому break и не нужны там.

[Ответить](#)



21. *Алексей:*

[17 июля 2019 в 13:57](#)

Немного доработал, ибо выполняло в любом случаи.

```
1  #include <iostream>
2
3  int calculate(int x, char op, int y)
4  {
5      switch(op)
6      {
7          case '+': return x + y; break;
8          case '-': return x - y; break;
9          case '*': return x * y; break;
10         case '/': return x / y; break;
11         case '%': return x % y; break;
12         default:
13             std::cout << "calculate(): Unhandled case\n";
14             return 0;
15     }
16 }
```



```
17
18 int main()
19 {
20     std::cout << "Enter variable:";
21     int x;
22     std::cin >> x;
23
24     std::cout << "Enter variable2:";
25     int y;
26     std::cin >> y;
27
28     std::cout << "Enter operator:";
29     char op;
30     std::cin >> op;
31
32     if (op == '+' || op == '-' || op == '/' || op == '*' || op == '%') std::cout << "I
33     else std::cout << "Error!\n";
34 }
```

Единственный вопрос — как в if это оптимизировать. Сделать массив из значений для op.

#### [Ответить](#)



22. *Михаил:*

[29 мая 2019 в 05:14](#)

```
1 #include <iostream>
2 #include <string>
3
4 enum class Animal {
5     pig, chicken, goat, cat, dog, ostrich
6 };
7
8 std::string getAnimalName(Animal animal) {
9     switch (animal) {
10     case Animal::pig:
11         return "свинья";
12         break;
13     case Animal::chicken:
14         return "курица";
15         break;
16     case Animal::goat:
17         return "коза";
18         break;
19     case Animal::cat:
20         return "кошка";
21         break;
22     case Animal::dog:
```

```
23     return "собака";
24     break;
25 case Animal::ostrich:
26     return "страус";
27     break;
28 default:
29     std::cout << "Неведомая зверушка!";
30     exit(0);
31 }
32 }
33
34 int printNumberOfLegs(Animal animal) {
35     int num;
36     switch (animal) {
37         case Animal::pig:
38         case Animal::goat:
39         case Animal::cat:
40         case Animal::dog:
41             num = 4;
42             break;
43         case Animal::chicken:
44         case Animal::ostrich:
45             num = 2;
46             break;
47         default:
48             std::cout << "Неведомая зверушка!";
49             exit(0);
50     }
51     std::cout << "Это " << getAnimalName(animal) << " на " << num << " ногах.\n"
52 }
53
54 int main() {
55     setlocale(LC_ALL, "rus");
56     printNumberOfLegs(Animal::cat);
57     printNumberOfLegs(Animal::chicken);
58     return 0;
59 }
```

### [Ответить](#)



1. *Nikita:*

[9 июня 2019 в 21:36](#)

printNumberOfLegs получается void

[Ответить](#)



1. *Михаил:*  
[6 августа 2019 в 12:46](#)

Точно, пропустил этот момент. Но программа работает и так.

[Ответить](#)



23. *Анна:*  
[19 мая 2019 в 16:59](#)

Зачем в switch расписывать default, если с неправильным параметром функция просто не запустится? Выходит, что default не к чему.

[Ответить](#)



1. *Юрий:*  
[6 июня 2019 в 15:06](#)

В default вы можете просто вывести сообщение пользователю, что он ввёл некорректные значения, чтобы пользователь получил информативный ответ, почему что-то не работает/что он сделал не так и вообще, что ему нужно делать.

[Ответить](#)



24. *Владимир:*  
[16 мая 2019 в 11:33](#)

Задание №2

```
1 #include <iostream>
2 #include <string>
3
4 enum class A
5 {
6     PIG, CHICKEN, GOAT, CAT, DOG, OSTRICH
7 };
8
9 std::string get_a(A x)
10 {
11     switch (x)
12     {
13         case A::PIG:         return "pig";
14         case A::CHICKEN:     return "chicken";
15         case A::GOAT:        return "goat";
16         case A::CAT:         return "cat";
17         case A::DOG:         return "dog";
```

```
18     case A::OSTRICH:    return "ostrich";
19
20     default:           return "default";
21 }
22 }
23
24 void print_legs(A y)
25 {
26     using namespace std;
27
28     switch (y)
29     {
30     case A::CHICKEN:
31     case A::OSTRICH:
32         cout << "A " << get_a(y) << " has\t- 2 legs\n";
33         break;
34
35     case A::PIG:
36     case A::CAT:
37     case A::GOAT:
38     case A::DOG:
39         cout << "A " << get_a(y) << " has\t- 4 legs\n";
40         break;
41
42     default:
43         cout << "A " << get_a(y) << " has\t- ? legs\n";
44         break;
45     }
46 }
47
48 int main()
49 {
50     //print_legs(A::PIG);
51     print_legs(A::CHICKEN);
52     //print_legs(A::GOAT);
53     print_legs(A::CAT);
54     //print_legs(A::OSTRICH);
55
56     return 0;
57 }
```

### [Ответить](#)



25. **Вячеслав:**

[9 марта 2019 в 20:31](#)

ВОТ ТАК Я ВЫПОЛНИЛ ВТОРОЕ ЗАДАНИЕ:

```

1 #include "pch.h"
2 #include <iostream>
3 #include <string> //подключаем библиотеку типа string для функции
4
5 enum Animal //объявляем перечисления животных enum
6 {
7     ANIMAL_PIG,
8     ANIMAL_CHICKEN,
9     ANIMAL_GOAT,
10    ANIMAL_CAT,
11    ANIMAL_DOG,
12    ANIMAL_OSTRICH,
13 };
14 //объявляем функцию типа string для вывода имен животных
15 std::string getAnimalName(Animal animal)
16 {
17     switch (animal) //объявляем оператор switch с параметром перечислений enum
18     { //объявляем условия:
19         case ANIMAL_PIG: //если при вызове функции в main() в качестве параметров испо
20             return "Pig"; //то выведет это
21         case ANIMAL_CHICKEN: //если использовать это
22             return "Chicken"; //то выведет это
23         case ANIMAL_GOAT: //если использовать это
24             return "Goat"; //то выведет это
25         case ANIMAL_CAT: //если использовать это
26             return "Cat"; //то выведет это
27         case ANIMAL_DOG: //если использовать это
28             return "Dog"; //то выведет это
29         case ANIMAL_OSTRICH: //если использовать это
30             return "Ostrich"; //то выведет это
31         default: //если указать недействительное имя
32             return "getAnimalName(): Unhandled enumerator"; //то получим необработанное
33     }
34 }
35
36 //объявляем функцию с невозвратным типом для вывода числа лап у животных
37 void printNumberOfLegs(Animal animal)
38 { //объявляем локально пространство имен
39     using namespace std;
40     //делаем вывод в консоль с вызовом функции имен животных
41     cout << "A " << getAnimalName(animal) << " has ";
42
43     switch (animal) //подключаем оператор switch с параметрами перечисления животн
44     { //создаем условия:
45         case ANIMAL_CHICKEN: //если ввести кур
46         case ANIMAL_OSTRICH: //и страусов
47             cout << " 2 "; //то в консоли выведет, что у них по 2 лапы
48             break; //окончание кейсов с двумя лапами
49

```

```

50
51     case ANIMAL_PIG://если ввести свинью
52     case ANIMAL_GOAT://козу
53     case ANIMAL_CAT://кошку
54     case ANIMAL_DOG://или собаку
55         cout << " 4 ";//то выведет, что у них по 4 лапы
56         break;//окончание кейсов с 4 лапами
57
58     default://если указать несуществующих в этой программе животных
59         cout << "printNumberOfLegs(): Unhandled enumerator";//то получим необрабо
60         break;//окончание default
61     }
62
63     cout << " legs\n";//вывод надписи
64 }
65
66 int main()
67 {
68     //вызываем функции с количеством лап животных
69     printNumberOfLegs(ANIMAL_CAT);
70     printNumberOfLegs(ANIMAL_CHICKEN);
71
72     return 0;
73 }

```

### Ответить



26. Вячеслав:

9 марта 2019 в 18:58

вот так получилось первое задание :

```

1  #include "pch.h"
2  #include <iostream>
3  //объявляем целочисленную функцию calculate() с двумя целочисленными параметрами
4  int calculate(int a, int b, char op)
5  {
6      //объявляем локальное пространство имен
7      using namespace std;
8      switch (op)//объявляем оператор switch с параметром op типа char
9      {
10         //объявляем условия:
11         case '+'://если ввести оператор +
12             return a + b;//то проводится операция сложения
13         case '-'://если ввести оператор -
14             return a - b;//то проводится вычитание
15         case '*'://если ввести оператор *
16             return a * b;//то проводится умножение
17         case '/'://если ввести знак /
18             return a / b;//то проводится целочисленное деление
19     }
20 }

```

```
17     case '%': //если ввести оператор %
18         return a % b; //то происходит деление с остатком
19     default: //если указать недействительный оператор
20         cout << "calculate(): Unhandled case\n"; //получаем необработанное значение
21         return 0;
22     }
23 }
24
25 int main()
26 {
27     using namespace std;
28     cout << "Enter a number:";
29     int a;
30     cin >> a;
31
32     cout << "Enter another number:";
33     int b;
34     cin >> b;
35
36     cout << "Enter a mathematical operator (+, -, *, /, % ):";
37     char op;
38     cin >> op;
39
40     cout << a << " " << op << " " << b << "=" << calculate(a, b, op) << endl;
41
42     return 0;
```

### Ответить



27. *Andrey:*

1 октября 2018 в 13:21

У меня QT ругается, если в функцию с возвратом стринг, засунуть только свитч:  
switch control reaches end of non-void function [-Wreturn-type]

Вот сам кусочек кода.

```
1 std::string takeName(Animal animal)
2 {
3     switch (animal)
4     {
5     case Animal::PIG:
6         return "Pig";
7     case Animal::CHICKEN:
8         return "Chicken";
9     case Animal::GOAT:
10        return "Goat";
11     case Animal::CAT:
```

```
12         return "Cat";
13     case Animal::DOG:
14         return "Dog";
15     case Animal::OSTRICH:
16         return "Ostrich";
17     }
18 }
```

То есть он не видит возврата внутри switch?

А еще у меня тут нет default: потому что на него тоже фреймворк ругается, говорит что ты учел все позиции из enum class Animal и default тебе не нужен, он не будет использован.

[Ответить](#)



1. Евгений:

[2 августа 2019 в 15:40](#)

Думаю, что компилятор справедливо считает, что в switch может не выполниться ни один case и тогда никакой return не выполнится, что будет ошибкой, т.к. функция обязательно должна что-то вернуть

[Ответить](#)



28. Роман:

[26 июля 2018 в 18:01](#)

я один не понимаю смысл использования перечислений?

[Ответить](#)



1. Борис:

[30 апреля 2020 в 15:37](#)

Я тоже их недолюбиваю, и никогда не использовал. Хотя, не отрицаю, что иногда с ними удобнее.

[Ответить](#)



29. smigles:

[8 июля 2018 в 16:29](#)

Зачем использовать break после default, если switch заканчивается?

[Ответить](#)



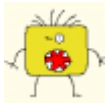
30. smigles:

[7 июля 2018 в 21:17](#)



Зачем использовать break в default, если после него switch заканчивается?

[Ответить](#)



1. Евгений:

[2 августа 2019 в 15:41](#)

Обычно break после default и не ставят

[Ответить](#)



31. Алибек:

[16 мая 2018 в 13:17](#)

Не лучше ли инициализировать и/или объявлять переменные до лейбла case?

[Ответить](#)



1. Юрий:

[16 мая 2018 в 21:31](#)

В статье рассматривается как можно делать. Как делать конкретно вам — дело ваше.

[Ответить](#)



32. Юрий:

[19 марта 2018 в 11:23](#)

```
1 #include <iostream>
2 #include <string>
3
4 int calculate(char znak, int a, int b)
5 {
6     switch (znak)
7     {
8         case '*':
9             return a * b;
10            break;
11         case '/':
12            return a / b;
13            break;
14         case '+':
15            return a + b;
16            break;
17         case '-':
18            return a - b;
19            break;
```

```
20
21     default:
22         std::cout << "ERROR!!!!" << std::endl;
23
24
25     }
26     return 0;
27 }
28
29 int main()
30 {
31     setlocale(LC_ALL, "Russian");
32
33     int a,b;
34     char znak;
35
36     std::cout << "Введите выражение : ";
37     std::cin >> a;
38     std::cin >> znak;
39     std::cin >> b;
40
41     std::cout << a << " " << znak << " " << b << " = " << calculate(znak, a, b) << "\n";
42
43
44
45     system("pause");
46     return 0;
47 }
```

Работает тоже, но вопрос другой не совсем пойму как работает `std::cin`, если ввести "10 +50" или "10+50" или 10 + 50" или любое количество пробелов, ответ будет правильным. `Std::cin` отбрасывает все пробелы? пробел это не `char`?

#### [Ответить](#)

**R**

1. Юрий:

[29 марта 2018 в 18:40](#)

У вас в программе 3 `cin`-а подряд. Если вы ввели первое значение, а затем пробелы, то для первого `cin` эти пробелы означают окончание потока входных данных. Затем у вас сразу же после пробелов и появления второго значения срабатывает второй `cin` (в который записывается только второе значение, без пробелов), затем третий.

Если вы для одного `cin` введете число 3, а затем 4 пробела, а затем снова 6 и только тогда нажмете Enter, то в переменную сохраниться только 3. Всё, что идёт за пробелами, после первого значения, в входном потоке `cin` — игнорируется. Попробуйте сами.

#### [Ответить](#)



33. Александр:

[22 января 2018 в 09:37](#)

А если вот мне просто не нравится switch априори могу же я продолжать использовать if else?

[Ответить](#)

R

1. Юрий:

[22 января 2018 в 13:54](#)

В этих уроках рассказывается, что есть в C++ и как это использовать. А уже использовать ли это вам — это дело ваше. Если вам лучше использовать if else — используйте if else.

[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.

[TELEGRAM](#)  [КАНАЛ](#)

[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 «SameGame» на C++/MFC](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

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