

Ravesli [Ravesli](#)


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


Урок №44. Конвертация чисел из двоичной системы в десятичную и наоборот

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновлено: 11 Сен 2020 |

 71397

[| 55](#)

Чтобы научиться конвертировать числа из двоичной (бинарной) системы счисления в десятичную и наоборот, прежде всего необходимо понять, как целые числа представлены в двоичной системе. Мы уже немного говорили об этом на [уроке №31](#).

Оглавление:

1. [Представление чисел в двоичной системе](#)
2. [Конвертация чисел из двоичной системы в десятичную](#)
3. [Способ №1: Конвертация чисел из десятичной системы в двоичную](#)
4. [Способ №2: Конвертация чисел из десятичной системы в двоичную](#)
5. [Еще один пример](#)
6. [Сложение двоичных чисел](#)
7. [Числа signed и метод «two's complement»](#)
8. [Почему так важен тип данных?](#)
9. [Тест](#)
10. [Ответы](#)

Представление чисел в двоичной системе

Рассмотрим обычное десятичное число, например, число 5623. Интуитивно понятно, что означают все эти цифры: $(5 * 1000) + (6 * 100) + (2 * 10) + (3 * 1)$. Так как в десятичной системе счисления всего 10 цифр, то каждое значение умножается на множитель 10 в степени ⁿ.

Выражение, приведенное выше, можно записать следующим образом: $(5 * 10^3) + (6 * 10^2) + (2 * 10^1) + (3 * 1)$.

Двоичные числа работают по аналогичной схеме, за исключением того, что в системе всего 2 числа (0 и 1) и множитель не 10, а 2. Так же как запятые (или пробелы) используются для улучшения читабельности больших десятичных чисел (например, 1, 427, 435), двоичные числа пишутся группами — в каждой по 4 цифры (например, 1101 0101).

Десятичное значение Двоичное значение

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Конвертация чисел из двоичной системы в десятичную

В примерах, приведенных ниже, предполагается, что мы работаем с целочисленными значениями `unsigned`. Рассмотрим 8-битное (1-байтовое) двоичное число: 0101 1110. Оно означает $(0 * 128) + (1 * 64) + (0 * 32) + (1 * 16) + (1 * 8) + (1 * 4) + (1 * 2) + (0 * 1)$. Если суммировать, то получим десятичное $64 + 16 + 8 + 4 + 2 = 94$.

Вот тот же процесс, но в таблице. Мы умножаем каждую двоичную цифру на её значение, которое определяется её положением. Выполним конвертацию двоичного числа 0101 1110 в десятичную систему:

Двоичный символ	0	1	0	1	1	1	1	0
* Значение символа	128	64	32	16	8	4	2	1
= Результат (94)	0	64	0	16	8	4	2	0

А теперь конвертируем двоичное 1001 0111 в десятичную систему:

Двоичный символ	1	0	0	1	0	1	1	1
* Значение символа	128	64	32	16	8	4	2	1
= Результат (151)	128	0	0	16	0	4	2	1

Получается:

1001 0111 (двоичное) = 151 (десятичное)

Таким способом можно легко конвертировать и 16-битные, и 32-битные двоичные числа, просто добавляя столбцы. Обратите внимание, проще всего начинать отсчет справа налево, умножая на 2 каждое последующее значение.

Способ №1: Конвертация чисел из десятичной системы в двоичную

Первый способ конвертации чисел из десятичной системы счисления в двоичную заключается в непрерывном делении числа на 2 и записывании остатков. Если остаток («r» от англ. «remainder») есть, то пишем 1, если нет — пишем 0. Затем, читая остатки снизу вверх, мы получим готовое двоичное число.

Например, конвертация десятичного числа 148 в двоичную систему счисления:

```
148 / 2 = 74 r0
74 / 2 = 37 r0
37 / 2 = 18 r1
18 / 2 = 9 r0
9 / 2 = 4 r1
4 / 2 = 2 r0
2 / 2 = 1 r0
1 / 2 = 0 r1
```

Записываем остатки снизу вверх: 1001 0100.

148 (десятичное) = 1001 0100 (двоичное)

Вы можете проверить этот ответ путем конвертации двоичного числа обратно в десятичную систему:

$(1 * 128) + (0 * 64) + (0 * 32) + (1 * 16) + (0 * 8) + (1 * 4) + (0 * 2) + (0 * 1) = 148$

Способ №2: Конвертация чисел из десятичной системы в двоичную

Этот способ хорошо подходит для небольших двоичных чисел. Рассмотрим десятичное число 148 еще раз. Какое наибольшее число, умноженное на 2 (из ряда 1, 2, 4, 8, 16, 32, 64, 128, 256 и т.д.), меньше 148? Ответ: 128.

148 >= 128? Да, поэтому 128-й бит равен 1. $148 - 128 = 20$
20 >= 64? Нет, поэтому 64-й бит равен 0.
20 >= 32? Нет, поэтому 32-й бит равен 0.

20 >= 16? Да, поэтому 16-й бит равен 1. $20 - 16 = 4$
4 >= 8? Нет, поэтому 8-й бит равен 0.
4 >= 4? Да, поэтому 4-й бит равен 1. $4 - 4 = 0$, что означает, что все остальные биты равны 0.

Примечание: Если ответом является «Да», то мы имеем true, что означает 1. Если ответом является «Нет», то мы имеем false, что означает 0. Детально об этом читайте в материалах [урока №34](#).

Результат:

$$148 = (1 * 128) + (0 * 64) + (0 * 32) + (1 * 16) + (0 * 8) + (1 * 4) + (0 * 2) + (0 * 1) = 1001\ 0100$$

То же самое, но в таблице:

Двоичный символ	1	0	0	1	0	1	0	0
* Значение символа	128	64	32	16	8	4	2	1
= Результат (148)	128	0	0	16	0	4	0	0

Еще один пример

Конвертируем десятичное число 117 в двоичную систему счисления, **используя способ №1**:

```
117 / 2 = 58 r1
58 / 2 = 29 r0
29 / 2 = 14 r1
14 / 2 = 7 r0
7 / 2 = 3 r1
3 / 2 = 1 r1
1 / 2 = 0 r1
```

Запишем число, состоящее из остатков (снизу вверх):

117 (десятичное) = 111 0101 (двоичное)

А теперь выполним ту же конвертацию, но с **использованием способа №2**:

Наибольшее число, умноженное на 2, но которое меньше 117 — это 64.

117 >= 64? Да, поэтому 64-й бит равен 1. 117 - 64 = 53.
 53 >= 32? Да, поэтому 32-й бит равен 1. 53 - 32 = 21.
 21 >= 16? Да, поэтому 16-й бит равен 1. 21 - 16 = 5.

5 >= 8? Нет, поэтому 8-й бит равен 0.
 5 >= 4? Да, поэтому 4-й бит равен 1. 5 - 4 = 1.
 1 >= 2? Нет, поэтому 2-й бит равен 0.
 1 >= 1? Да, поэтому 1-й бит равен 1.

Результат:

117 (десятичное) = 111 0101 (двоичное)

Сложение двоичных чисел

В некоторых случаях (один из них мы рассмотрим ниже) вам может понадобиться выполнить сложение двух двоичных чисел. Это на удивление легко (может быть даже проще, чем сложение десятичных чисел), хотя поначалу может показаться немного странным, но вы быстро к этому привыкните.

Рассмотрим сложение следующих двух небольших двоичных чисел:

0110 (6 в десятичной системе) +
0111 (7 в десятичной системе)

Во-первых, числа нужно записать в столбик (как показано выше). Затем справа налево и сверху вниз мы добавляем каждый столбец с цифрами, как будто это десятичные числа. Так как в бинарной системе есть только два числа: 0 и 1, то всего есть 4 возможных исхода:

$$\rightarrow 0 + 0 = 0$$

$$\rightarrow 0 + 1 = 1$$

$$\rightarrow 1 + 0 = 1$$

$$\rightarrow 1 + 1 = 0, \text{ и } 1 \text{ переносим в следующую колонку}$$

Начнем с первой колонки (столбца):

0110 (6 в десятичной системе) +
0111 (7 в десятичной системе)

1

$0 + 1 = 1$. Легко.

Вторая колонка:

1
0110 (6 в десятичной системе) +
0111 (7 в десятичной системе)

01

$1 + 1 = 0$, и 1 остается в памяти до следующей колонки.

Третья колонка:

11
0110 (6 в десятичной системе) +
0111 (7 в десятичной системе)

101

А вот здесь уже немного сложнее. Обычно $1 + 1 = 0$ и остается единица, которую мы переносим в следующую колонку. Тем не менее, у нас уже есть 1 из предыдущего столбца и нам нужно добавить еще 1. Что делать? А вот что: 1 остается, а еще 1 мы переносим дальше.

Последняя колонка:

```

11
0110 (6 в десятичной системе) +
0111 (7 в десятичной системе)
----
1101

```

$0 + 0 = 0$, но так как есть еще 1, то результат — 1101.

13 (десятичное) = 1101 (двоичное)

Вы спросите: «А как добавить десятичную единицу к любому другому двоичному числу (например, к 1011 0011)?». Точно так же, как мы это делали выше, только числом снизу является двоичная единица. Например:

```

      1 (переносим в следующую колонку)
1011 0011 (двоичное число)
0000 0001 (1 в двоичной системе)
-----
1011 0100

```

Числа signed и метод «two's complement»

В примерах, приведенных выше, мы работаем только с целыми числами `unsigned`, которые могут быть только положительными. Сейчас же мы рассмотрим то, как работать с числами `signed`, которые могут быть как положительными, так и отрицательными.

С целыми числами `signed` используется **метод «two's complement»**. Он означает, что самый левый (самый главный) бит используется в качестве **знакового бита**. Если значением знакового бита является 0, то число положительное, если 1 — число отрицательное.

Положительные числа `signed` хранятся так же, как и положительные числа `unsigned` (с 0 в качестве знакового бита). А вот отрицательные числа `signed` хранятся в виде обратных положительных чисел + 1. Например, выполним конвертацию -5 из десятичной системы счисления в двоичную, используя метод «two's complement»:

```

Сначала выясняем бинарное представление 5: 0000 0101
Затем инвертируем все биты (конвертируем в противоположные): 1111 1010
Затем добавляем к числу единицу: 1111 1011

```

Конвертация -76 из десятичной системы счисления в двоичную:

```

Представление положительного 76: 0100 1100
Инвертируем все биты: 1011 0011
Добавляем к числу единицу: 1011 0100

```

Почему мы добавляем единицу? Рассмотрим это на примере 0 (нуля). Если противоположностью отрицательного числа является его положительная форма, то 0 имеет два представления: 0000 0000 (положительный ноль) и 1111 1111 (отрицательный ноль). При добавлении единицы, в 1111 1111 произойдет **переполнение**, и значение изменится на 0000 0000. Добавление единицы позволяет избежать наличия двух представлений нуля и упрощает внутреннюю логику, необходимую для выполнения арифметических вычислений с отрицательными числами.

Перед тем, как конвертировать двоичное число (используя метод «two's complement») обратно в десятичную систему счисления, нужно сначала посмотреть на знаковый бит. Если это 0, то смело

используйте способы, приведенные выше, для целых чисел `unsigned`. Если же знаковым битом является 1, то тогда нужно инвертировать все биты, затем добавить единицу, затем конвертировать в десятичную систему, и уже после этого менять знак десятичного числа на отрицательный (потому что знаковый бит изначально был отрицательным).

Например, выполним конвертацию двоичного 1001 1110 (используя метод «two's complement») в десятичную систему счисления:

Имеем: 1001 1110

Инвертируем биты: 0110 0001

Добавляем единицу: 0110 0010

Конвертируем в десятичную систему счисления: $(0 * 128) + (1 * 64) + (1 * 32) + (0 * 16) + (0 * 8) + (0 * 4) + (1 * 2) + (0 * 1) = 64 + 32 + 2 = 98$

Так как исходный знаковый бит был отрицательным, то результатом является -98.

Почему так важен тип данных?

Рассмотрим двоичное число 1011 0100. Что это за число в десятичной системе счисления? Вы, наверное, подумаете, что это 180, и, если бы это было стандартное двоичное число `unsigned`, то вы были бы правы. Однако, если здесь используется метод «two's complement», то результат будет другой: -76. Также значение еще может быть другое, если оно закодировано каким-то третьим способом.

Так как же язык C++ понимает в какое число конвертировать 1011 0100: в 180 или в -76?

Еще на [уроке №28](#) мы говорили: «Когда вы указываете тип данных переменной, компилятор и процессор заботятся о деталях конвертации этого значения в соответствующую последовательность бит определенного типа данных. Когда вы просите ваше значение обратно, то оно «восстанавливается» из соответствующей последовательности бит в памяти».

Тип переменной используется для конвертации бинарного представления числа обратно в ожидаемую форму. Поэтому, если вы указали целочисленный тип данных `unsigned`, то компилятор знает, что 1011 0100 — это стандартное двоичное число, а его представление в десятичной системе счисления — 180. Если же типом переменной является целочисленный тип `signed`, то компилятор знает, что 1011 0100 закодирован с помощью метода «two's complement» и его представлением в десятичной системе счисления является число -76.

Тест

Задание №1

Конвертируйте двоичное число 0100 1101 в десятичную систему счисления.

Задание №2

Конвертируйте десятичное число 93 в 8-битное двоичное число `unsigned`.

Задание №3

Конвертируйте десятичное число -93 в 8-битное двоичное число signed (используя метод «two's complement»).

Задание №4

Конвертируйте двоичное число 1010 0010 в десятичное unsigned.

Задание №5

Конвертируйте двоичное число 1010 0010 в десятичное signed (используя метод «two's complement»).

Задание №6

Напишите программу, которая просит пользователя ввести число от 0 до 255. Выведите его как 8-битное двоичное число (в парах по 4 цифры). Не используйте побитовые операторы.

Подсказки:

- Воспользуйтесь способом конвертации №2. Предполагается, что наименьшим числом для сравнения является 128.
- Напишите функцию для проверки входных чисел: являются ли они больше чисел, умноженных на 2 (т.е. чисел 1, 2, 4, 8, 16, 32, 64 и 128). Если это так, то выводится 1, если нет — выводится 0.

Ответы

Ответ №1

Двоичный символ 0 1 0 0 1 1 0 1
* Значение символа 128 64 32 16 8 4 2 1
= Результат (77) 0 64 0 0 8 4 0 1

Ответ: 77.

Ответ №2

Используя способ №1:

93 / 2 = 46 r1
46 / 2 = 23 r0
23 / 2 = 11 r1
11 / 2 = 5 r1
5 / 2 = 2 r1
2 / 2 = 1 r0
1 / 2 = 0 r1

Остатки читаем снизу вверх и записываем в одну строку: 101 1101.

Ответ: 0101 1101.

Используя способ №2:

Наибольшее число, умноженное на 2, но которое меньше 93 — это 64.

93 >= 64? Да, 64-й бит равен 1. $93 - 64 = 29$.

29 >= 32? Нет, 32-й бит равен 0.

29 >= 16? Да, 16-й бит равен 1. $29 - 16 = 13$.

13 >= 8? Да, 8-й бит равен 1. $13 - 8 = 5$.

5 >= 4? Да, 4-й бит равен 1. $5 - 4 = 1$.

1 >= 2? Нет, 2-й бит равен 0.

1 >= 1? Да, 1-й бит равен 1.

Ответ: 0101 1101.

Ответ №3

Мы уже знаем из предыдущего примера, что 93 — это 0101 1101

Поэтому инвертируем биты: 1010 0010

И добавляем единицу: 1010 0011

Ответ: 1010 0011.

Ответ №4

Работая справа налево:

$$1010\ 0010 = (0 * 1) + (1 * 2) + (0 * 4) + (0 * 8) + (0 * 16) + (1 * 32) + (0 * 64) + (1 * 128) = 2 + 32 + 128 = 162$$

Ответ: 162.

Ответ №5

Имеем: 1010 0010

Инвертируем биты: 0101 1101

Добавляем единицу: 0101 1110

Конвертируем в десятичную систему счисления: $16 + 64 + 8 + 4 + 2 = 94$

Так как здесь используется метод «two's complement», а знаковый бит является отрицательным, то результат: -94

Ответ: -94.

Ответ №6

```
1 #include <iostream>
2
3 // x - это число, которое мы будем тестировать.
4 // pow - это множитель 2 (например, 128, 64, 32 и т.д.)
5 int printandDecrementBit(int x, int pow)
6 {
7     // Проверяем, является ли x больше определенного числа, умноженного на 2 и выводим
8     if (x >= pow)
9         std::cout << "1";
10    else
11        std::cout << "0";
12}
```

```
13 // Если x больше, чем число, умноженное на 2, то вычитаем его из значения
14 if (x >= pow)
15     return x - pow;
16 else
17     return x;
18 }
19
20 int main()
21 {
22     std::cout << "Enter an integer between 0 and 255: ";
23     int x;
24     std::cin >> x;
25
26     x = printandDecrementBit(x, 128);
27     x = printandDecrementBit(x, 64);
28     x = printandDecrementBit(x, 32);
29     x = printandDecrementBit(x, 16);
30
31     std::cout << " ";
32
33     x = printandDecrementBit(x, 8);
34     x = printandDecrementBit(x, 4);
35     x = printandDecrementBit(x, 2);
36     x = printandDecrementBit(x, 1);
37
38     return 0;
39 }
```

Оценить статью:

★★★★★ (204 оценок, среднее: 4,85 из 5)



← [Урок №43. Логические операторы: И, ИЛИ, НЕ](#)

[Урок №45. Побитовые операторы](#) →



Комментариев: 55



1. [DarkSavant:](#)
[23 августа 2020 в 16:44](#)

У меня так получилось. Проверил — для чисел от 0 до 256 работает нормально. Впрочем поменяв значение переменной d можно конвертировать числа в любом диапазоне.

```
1 #include <iostream>
```

```

2  #include <cmath>
3
4  int dig(int a, int b){
5  int temp;
6  temp=a-b;
7  if (temp<0) {  std::cout << "0";
8                return a;}
9  else {std::cout << "1";
10         return (a-b);}
11 }
12
13
14 int main (void){
15 int a(0), d(128);
16 std::cout << "Enter number between 0-255" << std::endl;
17 std::cin >> a;
18 do { a=dig(a,d);
19     d=d/2; } while (d>0);
20 return 0;}

```

[Ответить](#)



2. *Artur:*

[25 июля 2020 в 22:56](#)

Буду рад критике

```

1  #include <iostream>
2  #include <cmath>
3
4  using std::cout;
5  using std::cin;
6
7  int getDec() {
8      cout << "Enter a number (0-255) in decimal number system: ";
9      int x;
10     cin >> x;
11     return x;
12 }
13
14 void nsd(int x, int i) {
15
16     int y = pow(2, i);    // первое число 2^i меньше нашего x
17
18     while (i >= 0) {      // пока степень >= 0, то есть от 2^i до 1
19         cout << (x >= y);  // выводим 1 если наше число больше или равно 2^i,
20         x -= y;
21         --i;
22         y /= 2;
23     }
24 }
25

```

```

26
27 int exp(int x) {
28     int i = 8;           // так как у нас есть информация, что введенное число
29     int y = pow(2, i);    // y == 256
30
31     while (x < y) {       // пока наше число меньше y
32         --i;             // убавляем степень на 1
33         y /= 2;          // делим y на 2 (128, 64, 32, ... 2, 1)
34     }
35     return i;            // получаем степень первого y, который оказался меньше
36 }
37
38 int main()
39 {
40     int x = getDec();     // получаем число
41     int i = exp(x);       // получаем степень
42     nsd(x, i);            // выводим в двоичной системе
43
44     return 0;
45 }

```

Ответить



3. Александр:

14 июля 2020 в 15:08

```

1  #include <iostream>
2  #include <math.h>
3
4  using namespace std;
5
6  int getNumber()
7  {
8      cout << "Enter a number from 0 to 255" << "\n";
9      int number;
10     cin >> number;
11     return number;
12 }
13
14 //Функция переводит десятичное число в двоичное
15 void decToBinary(int decimal)
16 {
17     //i - степень двойки (от 7 до 0)
18     for (int i = 7; i >= 0; i--)
19     {
20         //Когда степень равен 3 (5 знак), то ставим пробел , чтобы разбить двоичное
21         if (i == 3) cout << " ";
22
23         //Если число больше, чем 2 в степени i, то ставим 1 и вычитаем из него
24         if (decimal >= pow(2, i))
25

```

```
26     {
27         cout << '1';
28         decimal -= pow(2, i);
29     }
30
31     //В другом случае ставим 0
32     else cout << '0';
33 }
34 cout << "\n";
35 }
36
37 int main()
38 {
39     int number = getNumber();
40     decToBinary(number);
41 }
```

[ОТВЕТИТЬ](#)



4. *Настя:*

[12 июля 2020 в 13:48](#)

Вот мой код. Как мне кажется, самый простой и короткий. Прошу указать на ошибки, если они есть, буду рада критике и коррективам.

```
1  #include <iostream>
2  #include <conio.h>
3  #include <cstdlib>
4  #include <cstdio>
5  #include <cmath>
6
7  using namespace std;
8
9  void PrintBits(unsigned int num)
10 {
11     unsigned int bit = 524288;
12     int j = 0;
13     for(unsigned int i = bit; i > 0; i /= 2)
14     {
15         if(num >= i)
16         {
17             putchar('1');
18             num -= i;
19         }
20         else putchar('0');
21         ++j;
22         if(!(j % 4)) putchar(' ');
23     }
24 }
25
26 int main()
```

```

27 {
28
29     PrintBits(7584);
30     getch();
31     return 0;
32 }

```

Ответить



5. Максим:

7 июля 2020 в 00:05

```

1  #include <iostream>
2
3  //проверка больше ли введённое число пользователем, чем степень двойки
4  bool checkValueIsBigger(int valueForConversion,int degree) {
5      if (valueForConversion >= degree){
6          return true;
7      }
8      else return false;
9  }
10
11 //функция выводит число в консоль в двоичной системе
12 void conversionInBinary(int valueForConversion) {
13     for (int degree = 128; degree >= 1; degree /= 2) {
14
15         if (checkValueIsBigger(valueForConversion, degree)) { //проверка бо
16             valueForConversion -= degree;
17             std::cout << "1";
18         }
19         else std::cout << "0";
20
21         if (degree == pow(2, 4)){
22             std::cout << " ";
23         }
24     }
25 }
26
27 //функция ввода числа для дальнейшей конвертации
28 int valueForConversion() {
29     std::cout << "Enter the number between 0 and 255: ";
30     unsigned int value;
31
32     for (;;) { // бесконечный цикл для корректно введённого числа
33         std::cin >> value;
34
35         if (value >= 256 || value < 0) { //условие проверки диапазона от 0 до
36             std::cout << "Please enter the number between 0 and 255: ";
37             std::cin >> value;
38         }
39         else break; //выход из цикла в случае если число лежит в диапазоне от

```

```

40     }
41     return value;
42 }
43
44 int main()
45 {
46     int value = valueForConversion(); //записываем значение введенное пользователем
47     conversionInBinary(value); //вывод числа на консоль в двоичной системе
48     return 0;
49 }

```

Ответить



6. Pavel:

29 июня 2020 в 23:20

Получилось как то так, возможность для масштабирования полная (сразу переделал для чисел, занимающих 2 полных байта):

```

1  #include <iostream>
2  #include <math.h>
3
4  bool CheckData(int data, int rank) {
5      int i = pow(2,rank);
6      if (data >= i) {
7          return 1;
8      }
9      else {
10         return 0;
11     }
12 }
13
14 int main()
15 {
16     int x = 0; // Число в десятичной системе
17     int i = 15; // Здесь указываем количество разрядов
18     bool output = 0; // Один разряд числа
19     std::cout << "Enter a number ( 0 - 65535) : " << std::endl;
20     std::cin >> x;
21     if (x >= 0 && x <= 65535) { // Меняем тут предел
22         std::cout << "BIN : " << std::endl;
23         while (i >= 0) {
24             output = CheckData(x, i);
25             if (output == 1) {
26                 x = (x - (pow(2, i)));
27             }
28             std::cout << output;
29             if ((i == 4) || (i == 8) || (i == 12) || (i == 16)) {
30                 std::cout << " ";
31             }
32             i = i - 1;

```

```

33     }
34     std::cout << std::endl;
35 }
36 else {
37     std::cout << "Incorrect data!!!" << std::endl;
38 }
39 }

```

Ответить



7. Руслан:

[23 июня 2020 в 20:52](#)

Здравствуйте! Спасибо за уроки) Наверное лучшее, что мне удалось найти для первоначального знакомства с C++.

При написании руководствовался исключительно пройденным материалам до этого урока (хотя ветвление `if` только вскользь упоминалось, думаю для написания этого задания без него не обойтись).

Буду очень благодарен конструктивной критике.

```

1  #include <iostream>
2
3  bool range_check(unsigned short x) // функция проверки диапазона введенного числа
4  {
5      return (x >= 0 && x <= 255);
6  }
7
8  unsigned short input(unsigned short x) // функция ввода числа
9  {
10     std::cout << "Enter a number from 0 to 255" << std::endl;
11     std::cin >> x;
12     std::cout << '\a';
13     if (range_check(x))
14         return x;
15     else
16         std::cout << "Enter another number" << std::endl;
17         input(x);
18 }
19
20 void print(unsigned short x, unsigned short b128, unsigned short b64,
21 unsigned short b32, unsigned short b16, unsigned short b8, unsigned short b4,
22 unsigned short b2, unsigned short b1) // функция вывода результата
23 {
24     std::cout << "Your decimal number:" << '\t' << x << '\n';
25     std::cout << "is equal to a binary number:" << '\t' << b128 << b64 << b32
26     std::cout << b8 << b4 << b2 << b1 << std::endl;
27 }
28 /* функции для проверки входных чисел: являются ли они больше чисел,
29 умноженных на 2
30 */
31 unsigned short check_b128(unsigned short x, unsigned short b128)

```



```
32 {
33     b128 = (x >= 128) ? 1 : 0;
34
35     return b128;
36 }
37
38 unsigned short check_b64(unsigned short x128, unsigned short b64)
39 {
40     b64 = (x128 >= 64) ? 1 : 0;
41
42     return b64;
43 }
44
45 unsigned short check_b32(unsigned short x64, unsigned short b32)
46 {
47     b32 = (x64 >= 32) ? 1 : 0;
48
49     return b32;
50 }
51
52 unsigned short check_b16(unsigned short x32, unsigned short b16)
53 {
54     b16 = (x32 >= 16) ? 1 : 0;
55
56     return b16;
57 }
58 unsigned short check_b8(unsigned short x16, unsigned short b8)
59 {
60     b8 = (x16 >= 8) ? 1 : 0;
61
62     return b8;
63 }
64
65 unsigned short check_b4(unsigned short x8, unsigned short b4)
66 {
67     b4 = (x8 >= 4) ? 1 : 0;
68
69     return b4;
70 }
71
72 unsigned short check_b2(unsigned short x4, unsigned short b2)
73 {
74     b2 = (x4 >= 2) ? 1 : 0;
75
76     return b2;
77 }
78
79 unsigned short check_b1(unsigned short x2, unsigned short b1)
80 {
81     b1 = (x2 >= 1) ? 1 : 0;
82
83     return b1;
```

```
84 }
85 /* функции проверки равен ли бит 1
86 и разности для подсчета последующих битов*/
87
88 unsigned short m_b128(unsigned short x, unsigned short b128)
89 {
90     unsigned short a = (b128 == 1) ? (x -= 128) : x;
91
92     return a;
93 }
94
95 unsigned short m_b64(unsigned short x128, unsigned short b64)
96 {
97     unsigned short a = (b64 == 1) ? (x128 -= 64) : x128;
98
99     return a;
100 }
101
102 unsigned short m_b32(unsigned short x64, unsigned short b32)
103 {
104     unsigned short a = (b32 == 1) ? (x64 -= 32) : x64;
105
106     return a;
107 }
108
109 unsigned short m_b16(unsigned short x32, unsigned short b16)
110 {
111     unsigned short a = (b16 == 1) ? (x32 -= 16) : x32;
112
113     return a;
114 }
115
116 unsigned short m_b8(unsigned short x16, unsigned short b8)
117 {
118     unsigned short a = (b8 == 1) ? (x16 -= 8) : x16;
119
120     return a;
121 }
122
123 unsigned short m_b4(unsigned short x8, unsigned short b4)
124 {
125     unsigned short a = (b4 == 1) ? (x8 -= 4) : x8;
126
127     return a;
128 }
129
130 unsigned short m_b2(unsigned short x4, unsigned short b2)
131 {
132     unsigned short a = (b2 == 1) ? (x4 -= 2) : x4;
133
134     return a;
135 }
```

```
136
137 unsigned short m_b1(unsigned short x2, unsigned short b1)
138 {
139     unsigned short a = (b1 == 1) ? (x2 == 1) : x2;
140
141     return a;
142 }
143
144 int main()
145 {
146     unsigned short x = input(x); // инициализация пользовательским вводом
147
148     unsigned short b128 = check_b128(x, b128); // инициализация и расчет бита
149     unsigned short x128 = m_b128(x, b128); // инициализация и расчет временно
150
151     unsigned short b64 = check_b64(x128, b64);
152     unsigned short x64 = m_b64(x128, b64);
153
154     unsigned short b32 = check_b32(x64, b32);
155     unsigned short x32 = m_b32(x64, b32);
156
157     unsigned short b16 = check_b16(x32, b16);
158     unsigned short x16 = m_b16(x32, b16);
159
160     unsigned short b8 = check_b8(x16, b8);
161     unsigned short x8 = m_b8(x16, b8);
162
163     unsigned short b4 = check_b4(x8, b4);
164     unsigned short x4 = m_b4(x8, b4);
165
166     unsigned short b2 = check_b2(x4, b2);
167     unsigned short x2 = m_b2(x4, b2);
168
169     unsigned short b1 = check_b1(x2, b1);
170     unsigned short x1 = m_b1(x2, b1);
171
172     print(x, b128, b64, b32, b16, b8, b4, b2, b1); // вывод результата
173
174     return 0;
175 }
```

Ответить



1. Руслан:

[1 июля 2020 в 16:35](#)

Используя компилятор G++, решил проверить код с дополнительными флагами (-std=c++17 -Wall -Wextra -Werror -Wpedantic -pedantic-errors) и параллельно сократить его. Исправил все появившиеся предупреждения, компиляция и линкинг прошли успешно, программа работает. Однако при пользовательском вводе числа больше 65535 (максимальное значение типа данных unsigned short) возникает ошибка, консоль выводит

сообщение "Ошибка сегментирования (стек памяти сброшен на диск)" (это в Linux, в других ОС аналогично).

Что я делаю не правильно, где может быть ошибка в коде?

```
1  #include <iostream>
2  typedef unsigned short int UNS;
3
4  bool range_check(UNS x) // функция проверки диапазона введенного числа
5  {
6      return (x <= 255);
7  }
8
9  UNS input(UNS x) // функция ввода числа
10 {
11     std::cout << "Enter a number from 0 to 255" << '\n';
12     std::cin >> x;
13     std::cout << '\a';
14     if (range_check(x))
15         return x;
16     else
17     {
18         std::cout << "Enter another number" << '\n';
19         return input(x);
20     }
21 }
22
23 void print(UNS x, UNS b128, UNS b64, UNS b32, UNS b16, UNS b8, UNS b4, UNS b2, UNS b1)
24 {
25     std::cout << "Your decimal number:" << '\t' << x << '\n';
26     std::cout << "is equal to a binary number:" << '\t' << b128 << b64 <<
27     std::cout << b8 << b4 << b2 << b1 << std::endl;
28 }
29
30 UNS check_bit(UNS x, UNS rank) // функция для проверки входных чисел: явл
31 {
32     return (x >= rank) ? 1 : 0;
33 }
34
35 UNS m_bit(UNS bit, UNS x, UNS rank) // функция проверки равен ли бит 1 и
36 {
37     return (bit == 1) ? (x -= rank) : x;
38 }
39
40 int main()
41 {
42     UNS x {}; // инициализация переменной для ввода
43     x = input(x); // пользовательский ввод значения переменной
44
45     UNS b128 = check_bit(x, 128); // инициализация и расчет бита
46     UNS x128 = m_bit(b128, x, 128); // инициализация и расчет временной пе
47
48 }
```

```

49     UNS b64 = check_bit(x128, 64);
50     UNS x64 = m_bit(b64, x128, 64);
51
52     UNS b32 = check_bit(x64, 32);
53     UNS x32 = m_bit(b32, x64, 32);
54
55     UNS b16 = check_bit(x32, 16);
56     UNS x16 = m_bit(b16, x32, 16);
57
58     UNS b8 = check_bit(x16, 8);
59     UNS x8 = m_bit(b8, x16, 8);
60
61     UNS b4 = check_bit(x8, 4);
62     UNS x4 = m_bit(b4, x8, 4);
63
64     UNS b2 = check_bit(x4, 2);
65     UNS x2 = m_bit(b2, x4, 2);
66
67     UNS b1 = check_bit(x2, 1);
68
69     print(x, b128, b64, b32, b16, b8, b4, b2, b1); // вывод результата
70
71     return 0;
}

```

Ответить



1. Руслан:

6 июля 2020 в 16:42

Решил переписав функцию ввода числа

```

1  UNS input(short x) // функция ввода числа
2  {
3      std::cout << "Enter a number from 0 to 255" << '\n';
4      std::cin >> x;
5      std::cout << '\a';
6
7      if (x < 0 || x == 32767) // если число больше диапазона типа дан
8      {
9          std::cout << "Wrong number! " << '\t' << "Your number will be
10         return 0;
11     }
12
13     if (range_check(x))
14         return x;
15     else
16     {
17         std::cout << "Enter another number" << '\n';
18         return input(x);
19     }
}

```

20 | }

[Ответить](#)

8. Виктор:

[5 июня 2020 в 20:20](#)

Интересно для общего ознакомления. На практике лучше использовать калькулятор программиста и не тратить время. Также сведя ошибки к минимуму.

[Ответить](#)

9. Nikita:

[28 апреля 2020 в 14:01](#)

Сделал так, чтобы пользователь сам выбирал длину числа =)

```
1  #include <iostream>
2  #include <cmath>
3
4  int getLength()
5  {
6      std::cout << "size (bit): " << std::endl;
7      int x;
8      std::cin >> x;
9
10     return x;
11 }
12
13 int getNumber(int length)
14 {
15     std::cout << "number [0; " << long long int (pow(2, length)) << "]" << std::endl;
16     int x;
17     std::cin >> x;
18
19     return x;
20 }
21
22 void convertNumber(int num, int len)
23 {
24     for (int i = len; i >= 0; --i)
25     {
26         if (num >= pow(2, i))
27         {
28             num -= pow(2, i);
29             std::cout << "1 ";
30         }
31         else
32         {
33             std::cout << "0 ";
34         }
35     }
36 }
```

```
35     }
36 }
37
38 int main()
39 {
40     int length = getLength();
41     int number = getNumber(length);
42     convertNumber(number, length);
43
44     return 0;
45 }
```

[Ответить](#)



10. Майя:

[12 марта 2020 в 20:43](#)

Я не понимаю, почему в предложенном ответе такой длинный вариант, потому что, мне кажется у меня проще

```
1  #include <iostream>
2
3  using namespace std;
4
5  void konvent(int f)
6  {
7      int k = 128; //максимальная возможная степень 2
8      while (k > 0) //будет выполняться пока не пройдёт единицу
9      {
10         if (f >= k)
11         {
12             cout << "1";
13             f -= k;
14         }
15         else
16             cout << "0";
17         k = k/2; // переходим к следующей степени 2
18     }
19 }
20
21 int main()
22 {
23     int n;
24     cout << "Please, enter a number of integer from 0 to 255: ";
25     cin >> n;
26     konvent(n);
27     return 0;
28 }
```

[Ответить](#)



11. Сергей:

[15 февраля 2020 в 23:27](#)

Спасибо за задание.

Циклы не использую. Старался поменьше кода

```
1  #include <iostream>
2  int fun1(int n, int s)
3  {
4      if (n>=s)
5      {
6          std::cout << 1;
7          n-=s;
8      }
9      else std::cout << 0;
10     return n;
11 }
12
13 int main()
14 {
15     std::cout << "intering number from 0 to 255" << std::endl;
16     int number, sum(128);
17     std::cin >> number;
18     std::cout << "Number " << number << " in binary: ";
19     number = fun1 (number, sum);
20     number = fun1 (number, sum/=2);
21     number = fun1 (number, sum/=2);
22     number = fun1 (number, sum/=2);
23     number = fun1 (number, sum/=2);
24     number = fun1 (number, sum/=2);
25     number = fun1 (number, sum/=2);
26     number = fun1 (number, sum/=2);
27     std::cout << std::endl;
28     return 0;
29 }
```

[Ответить](#)

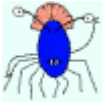


1. Dasha:

[14 апреля 2020 в 13:11](#)

Написать программу, которая читает массив из 8 элементов, заполненных 0 и 1 с клавиатуры. Полученный массив считается двоичным числом, которое в свою очередь переводится в десятичный код. Программа считает код слева направо, затем справа налево, получая 2 десятичных кода. При запуске программа подает 2 десятичных числа отсортированных по возрастанию на следующей строке.

[Ответить](#)



12. *Владимир:*

[17 января 2020 в 17:16](#)

Единственное, я не понял как мне это дело в обратном порядке показать.
Чтобы читалось слева направо. А не как у меня справа налево.

[Ответить](#)



13. *Константин:*

[10 декабря 2019 в 18:04](#)

У меня вот так получилось...

```
1  #include "pch.h"
2  #include <iostream>
3  #include <Windows.h>
4
5  void set_byte_num(int decimal)
6  {
7      int byte_num[8];
8      int num = 128;
9
10     for (int i = 0; i < 8; i++)
11     {
12         if (decimal >= num)
13         {
14             decimal -= num;
15             num /= 2;
16             byte_num[i] = 1;
17         }
18         else
19         {
20             num /= 2;
21             byte_num[i] = 0;
22         }
23         if (i == 4)
24         {
25             std::cout << " " << byte_num[i];
26         }
27         else
28         {
29             std::cout << byte_num[i];
30         }
31     }
32 }
33
34 int main()
35 {
36     setlocale(LC_ALL, "rus");
37     SetConsoleCP(1251);
```

```

38     SetConsoleOutputCP(1251);
39
40     std::cout << "Введите число от 0 до 255: ";
41     int num;
42     std::cin >> num;
43
44     if (!(num >= 0 && num <= 255))
45     {
46         std::cout << "Ошибка: Число не соответствует диапазону!" << std::endl;
47         return 0;
48     }
49     std::cout << "Ваше число в двоичной системе счисления: ";
50     set_byte_num(num);
51     std::cout << std::endl;
52 }

```

[Ответить](#)14. *Владимир:*[20 ноября 2019 в 13:11](#)

Подскажите, как выложить здесь код visual studio в оригинале (черное окно со всей разметкой и т.д.) как народ выкладывает в комментарях?

Есть важный вопрос по операторам, хотелось бы разобраться.

[Ответить](#)1. *Юрий:*[20 ноября 2019 в 15:39](#)

Просто вставляете код и всё)

[Ответить](#)15. *AleksTs:*[4 ноября 2019 в 06:25](#)

```

1  #include <iostream>
2
3  void printAndCovertNumbersIntoBit() {
4      // в консоле вводим число в диапазоне от 0 до 255
5      int number {0};
6      std::cin >> number;
7      // переводим number из десятичной системы исчисления в двоичную, где d = 2
8      int d {2}; // инициализируем переменную значением равным 2 в степени 7
9      while (d>=1) { // т.к. 2 в степени 0 = 1
10         if (number >= d) {
11             std::cout << 1;
12             number -= d; // уменьшаем number на значение d
13         } else {

```

```

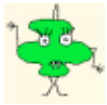
14         std::cout << 0;
15     }
16     // находим следующее число d в степени (n - 1) или кратное 2-м
17     d /= 2;
18     // по условию задачи разделяем пробелом следующую пару из 4-х битов
19     if (d == 8) std::cout << " ";
20 }
21 }
22
23 int main() {
24     std::cout << "Enter a number between 0 and 255:" << std::endl;
25
26     printAndCovertNumbersIntoBit();
27
28     return 0;
29 }

```

[Ответить](#)

1. [Алексей:](#)
[28 февраля 2020 в 19:31](#)

Бесподобно!!! Самый красивый вариант!

[Ответить](#)

16. [Alex:](#)
[27 октября 2019 в 19:45](#)

Немного упростил программу. Только тут ссылки, пока не изучали.

```

1  #include <iostream>
2  #include <cstdlib>
3
4  int input(int &x) {
5      std::cout << "Enter an integer between 0 and 255: ";
6      std::cin >> x;
7      return x;
8  }
9
10 int printandDecrementBit(const int &x, const int &pow)
11 {
12     if (x >= pow) { std::cout << "1"; return x - pow; }
13     else { std::cout << "0"; return x; }
14 }
15
16 int main()
17 {
18     int x = input(x);
19
20     x = printandDecrementBit(x, 128);

```

```

21     x = printandDecrementBit(x, 64);
22     x = printandDecrementBit(x, 32);
23     x = printandDecrementBit(x, 16);
24
25     std::cout << " ";
26
27     x = printandDecrementBit(x, 8);
28     x = printandDecrementBit(x, 4);
29     x = printandDecrementBit(x, 2);
30     x = printandDecrementBit(x, 1);
31
32     system("pause");
33     return 0;
34 }

```

[Ответить](#)1. *Sasha:*[10 января 2020 в 15:54](#)

Я бы не сказал, что стало проще. Ты просто ввод вынес в отдельную функцию и два if объединил. Читаемость ухудшилась

[Ответить](#)17. *armus1:*[25 сентября 2019 в 21:43](#)

Юрий, проверьте пожалуйста код, соответствует ли заданию:

```

1  #include <iostream>
2
3  using namespace std;
4
5  int beetNumber()
6  {
7      cout << "Enter an integer between 0 and 255:";
8      int x;
9      cin >> x;
10     return x;
11 }
12
13 int main(int x)
14 {
15     int a = beetNumber();
16
17     if (a >= 128) {
18         cout << "1";
19         a = (a - 128);
20     }
21     else {

```

```
22     cout << "0";
23 }
24 if (a >= 64) {
25     cout << "1";
26     a = (a - 64);
27 }
28 else {
29     cout << "0";
30 }
31 if (a >= 32) {
32     cout << "1";
33     a = (a - 32);
34 }
35 else {
36     cout << "0";
37 }
38 if (a >= 16) {
39     cout << "1";
40     a = (a - 16);
41 }
42 else {
43     cout << "0";
44 }
45
46 cout << " ";
47
48 if (a >= 8) {
49     cout << "1";
50     a = (a - 8);
51 }
52 else {
53     cout << "0";
54 }
55 if (a >= 4) {
56     cout << "1";
57     a = (a - 4);
58 }
59 else {
60     cout << "0";
61 }
62 if (a >= 2) {
63     cout << "1";
64     a = (a - 2);
65 }
66 else {
67     cout << "0";
68 }
69 if (a >= 1) {
70     cout << "1";
71     a = (a - 1);
72 }
73 else {
```

```

74         cout << "0";
75     }
76     return 0;
77 }

```

[Ответить](#)



18. *Saliwer:*

[30 августа 2019 в 18:06](#)

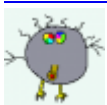
6 задание с использованием рекурсии:

```

1  #include <iostream>
2
3  int convertValue(int value, int exponent);
4
5  int main() {
6      std::cout << "Enter the value: ";
7      int x;
8      std::cin >> x;
9      // второй аргумент - 2^0 = 1
10     convertValue(x, 1);
11     std::cout << std::endl;
12     system("pause");
13     return 0;
14 }
15
16 int convertValue(int value, int exponent) {
17     //Условие выхода из рекурсии
18     if (value < exponent)
19         return value;
20     //Углубляемся до тех пор, пока exponent не станет больше value
21     int ret = convertValue(value, exponent * 2);
22     if ((ret - exponent) >= 0) {
23         std::cout << "1";
24         return ret - exponent;
25     }
26     else {
27         std::cout << "0";
28         return ret;
29     }
30 }

```

[Ответить](#)



19. *FolfieFluke:*

[15 августа 2019 в 11:49](#)

Сделал программку для получения двоичного числа с десятичных размером до 9 квинтиллионов (64 бита) с потенциалом до "бесконечного" расширения, только я не знаю как выделять больше памяти чем 64 бита для переменной, hex

Так как пока не знаю как делать правильные циклы, придумал "своеобразное" заикливание, не легко было сначала очень..)

Надеюсь будет интересно кому либо посмотреть :3

```
1  #include <iostream>
2  #include <Windows.h>
3  #include <cmath>
4  #include "ConstNamesNS.cpp"
5
6
7  // Для подсчёта количества бит используемых введённым значением
8  int bitCount{1};
9
10 // Объявление функций, решил не создавать отдельный файл в этот раз
11 long long mathMain();
12 long long userInput(int);
13 void text(int);
14 long long mathHBValue(unsigned long long, unsigned long long);
15 void numToBin(unsigned long long, unsigned long long, int, int);
16
17 // int main()
18 int main()
19 {
20     SetConsoleCP(1251);
21     SetConsoleOutputCP(1251);
22     // Приветственное сообщение с указанием действий, можно смело заменять на std
23     text(textnames::greetMSG);
24
25     // Вызов главной функции распределяющей задачи на несколько особых функций
26     mathMain();
27
28     return 0;
29 }
30 // Главная распределяющая функция НАЧАЛО
31 long long mathMain()
32 {
33     // Переменная для получения ближайшего двоичного номера(бита?) к введённому
34     // (определение положения по номеру бита введённого пользователем числа, и
35     long long highestBitValue{};
36
37     // Получение числа от пользователя с указанием значения выводимого текста
38     long long number = userInput(textnames::enterNum);
39
40     // 0 не полезен самочувствию программы
41     if (number == 0)
42         return 0;
43
44     // это использовалось для отладки программы, более не нужно, но пусть будет
45     // else std::cout << "\n*DebugInfo: in func \"mathMain\": var \"number\"
46
47     // Начало злой функции, которая заставила над собой хорошо подумать, съев
```

```

49 // Вычисления ближайшего к введённому пользователем числа относительно дво
50 // зной функцию (точнее её части, на самом деле их две, две следующих, про
51 // на 2, так как функции нравится делать ещё одно лишнее умножение, не буд
52 // программке
53 highestBitValue = mathHBValue(number, 1) / 2; // mathHBValue = math highest
54 // чтобы заиклить "правильн
55 // Ну очень хотелось сделать
56 // что то вот такое, так как
57 // В итоге вышло то что може
58 // квинтиллионов, так как ед
59
60 // Из за нрава функции mathHBValue делать лишнее умножение, получается и
61 bitCount -= 1;
62
63 // Тоже для отладки, нужно было понять что не так.. не нужно более
64 //std::cout << "\n*DIInf:func\"mathMain\":var\"highestBitValue\"'" << high
65
66 // Вызов второй части злой функции, тоже заикленная, может они и просты,
67 // реализовать. highestBitValue = результат прошлой функции. number = числ
68 // программа, это число используется для разделения пробелами групп по 4
69 // тоже что и выше, но используется для заполнения поля неиспользуе
70 // возможность выбирать её значение пользователю, но не сейчас.
71 numToBin(highestBitValue, number, 64, 64);
72 return 0;
73 }
74 // Главная распределяющая функция КОНЕЦ
75
76 // Начало повторяющихся функций
77 // Первая повторяющаяся функция начало
78 //repeating code
79 long long mathHBValue(unsigned long long number, unsigned long long tempValue
80 {
81     if (number >= tempValue) // Именно это делает лишнее умножение, (tempValue
82     {
83         // Тоже для отладки использовалось, потом там появился "bitCount++",
84         // пустой, и даже интересней. Но можно убрать, главное оставить "bitCo
85         std::cout << "\n*DIInf:f\"mathHBValue\":v\"number\"'" << number << "g
86         mathHBValue(number, tempValue * 2); // повторение функцией себя же по
87     }
88     else // конец повторения функции и возвращение значения
89     {
90         // Тоже для отладки, нужно было что бы понять куда деваются цифры и по
91         //std::cout << "\n*DIInf:in func\"mathHBValue\":var\"tempValue\"'" <<
92         return tempValue;
93     }
94 }
95 }
96 //repeating code end
97 // Первая повторяющаяся функция конец
98
99 // Вторая повторяющаяся функция начало
100

```



```

101 //repeating code
102 void numToBin(unsigned long long highestByte, unsigned long long number, int bitAndByteDivisor, int fillByZero)
103 {
104     // количество бит занимаемых числом, когда достигает 0 прекращает цикл
105     if (bitCount != 0)
106     {
107         // для пробелов между группами битов (по заданию)
108         if ((bitAndByteDivisor == 60) || (bitAndByteDivisor == 52) || (bitAndByteDivisor == 44) || (bitAndByteDivisor == 36) || (bitAndByteDivisor == 28) || (bitAndByteDivisor == 20) || (bitAndByteDivisor == 12) || (bitAndByteDivisor == 4))
109             std::cout << " ";
110         // для пробелов между байтами (для улучшения читабельности)
111         if ((bitAndByteDivisor == 56) || (bitAndByteDivisor == 48) || (bitAndByteDivisor == 40) || (bitAndByteDivisor == 32) || (bitAndByteDivisor == 24) || (bitAndByteDivisor == 16) || (bitAndByteDivisor == 8))
112             std::cout << " ";
113
114         // наполнение нолями неиспользуемых битов
115         if (bitCount < fillByZero--)
116         {
117             std::cout << "0";
118         }
119         else // после наполнения нолями начинается конвертация числа пользователя
120         {
121             fillByZero = -1000; // просто чтобы не случилось лишних нолей, хотя и не нужно
122
123             // А тут начинаются условия задания, где выполняется вычитания и деления
124             if (number >= highestByte)
125             {
126                 std::cout << "1";
127                 number -= highestByte;
128             }
129             else
130             {
131                 std::cout << "0";
132             }
133             highestByte /= 2; // деление ближайшего бита на два, чтобы получить следующий
134             bitCount--; // а тут воспользовался декрементом. Каждый цикл уменьшает количество битов на один
135         }
136         bitAndByteDivisor -= 1; // тоже можно было использовать декремент... но не нужно
137         numToBin(highestByte, number, bitAndByteDivisor, fillByZero); // Повторяется цикл
138     }
139     else
140     {
141         std::cout << "\n\n Всё\n"; // Конец, не нужная строка
142     }
143 }
144 //repeating code
145 // Вторая повторяющаяся функция конец
146 // Конец повторяющихся функций
147 // Номер от пользователя
148 long long userInput(int textVar) // long long, путём экспериментирования, выяснилось что long long лучше
149 {
150     text(textVar); // не важная строка
151     unsigned long long input{};

```

```

153     std::cin >> input;
154     std::cout << std::endl;
155     return input;
156 }
157 void text(int option) // Варианты текста, не важная часть
158 {
159     if (option == 1)
160         std::cout << "Я могу конвертировать твою циферку в единички и двоечки";
161     else
162         if (option == 2)
163             std::cout << "Пиши тут! - ";
164 }

```

Ответить

1. *Sasha:*
[10 января 2020 в 19:50](#)

Молодец, такую сложную программу написал, хотя еще циклы даже не прошел и о рекурсии не знаешь

Ответить

2. *Константин:*
[21 мая 2020 в 05:24](#)

Эй, добр молодец, а экран у тебя какой диагонали? Или здесь это не имеет значения? Запарился я комменты вычитывать!!!

Ответить

20. *Анастасия:*
[29 июля 2019 в 09:57](#)

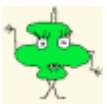
Задание чуть-чуть было усложнено -> этот код обрабатывает числа от 0 до 4 294 967 295. Без массива я думаю было бы громоздко, поэтому он здесь есть 😊

```

1  #include <iostream>
2
3
4  //Просим ввести целочисленное число
5  int inputInteger()
6  {
7      std::cout << "Please, enter any integer from 0 to 4 294 967 295 \n\t";
8      uint32_t x;
9      std::cin >> x;
10     return x;
11 }
12
13

```

```
14 //Основная функция, конвертирующая числа из десятичной системы счисления в двоичную
15 int conversion(uint32_t x)
16 {
17     //Проводим проверку над пользователем, чтобы не было переполнения
18     if (x < 0 || x > 4294967295)
19         std::cout << "Error...";
20     else {
21         std::cout << "Your number in binary representation is \n\t";
22
23         //Создаём массив данных о двоичном представлении числа
24         int d[32];
25
26         //Вычисляем и заносим данные
27         for (int i = 31; i >= 0; i--)
28         {
29             int r;
30             if (x != 0)
31             {
32                 r = x % 2;
33                 x = x / 2;
34                 d[i] = r;
35             }
36             else
37                 d[i] = 0;
38         }
39
40
41         //Выводим двоичное представление числа
42         for (int i = 0; i < 32; i++)
43         {
44             std::cout << d[i];
45             if (i == 3 || i == 7 || i == 11 || i == 15 || i == 19 || i == 23 || i == 27 || i == 31)
46                 std::cout << " ";
47         }
48
49
50     }
51     return 0;
52 }
53
54 int main()
55 {
56     int digit = inputInteger();
57     int inputDigit = conversion(digit);
58     std::cout << std::endl;
59     return 0;
60 }
```

[Ответить](#)

21. Алексей:

[9 июля 2019 в 12:32](#)

На самом деле это простейшая задача.

Суть деления уловить. И как-то упустил, что переменная себе выведет после получения значения.

[Ответить](#)



22. *Дмитрий:*

[12 июня 2019 в 21:30](#)

functions.h :

```
1 #ifndef FUNCTIONS_H
2 #define FUNCTIONS_H
3
4 int getNumber();
5 void checker(int num);
6
7 #endif
```

functions.cpp :

```
1 #include <iostream>
2
3 int getNumber()
4 {
5     std::cout << "Type number from 0 to 255: ";
6     int num;
7     std::cin >> num;
8     if (num > 255 || 0 > num)
9         std::cout << "Error: out of range number!" << std::endl;
10    else
11        return num;
12    return -1;
13 }
14
15 void conversion(int num, int exp)
16 {
17     int number = num;
18     int exponent = exp;
19     int position = pow(2, exponent--);
20     std::cout << bool(number != (number % position));
21     number %= position;
22     if (exponent >= 0)
23         conversion(number, exponent);
24     else
25         std::cout << "\nDone!" << std::endl;
26 }
27
28
```

```

29 void checker(int num)
30 {
31     if (num == -1)
32         std::cout << "Typed number should be no more than 255 and no less then
33     else
34         conversion(num, 7);
35 }

```

main.cpp :

```

1  #include <iostream>
2  #include "functions.h"
3
4  int main()
5  {
6      int num = getNumber();
7      checker(num);
8      return 0;
9  }

```

[Ответить](#)



1. *Дмитрий:*

[16 июня 2019 в 10:17](#)

XD чекер явно лишний, интересно почему раньше я этого не заметил? И комментарии все же были бы не лишними. В любом случае все что я хотел показать этим кодом, это то как использовать условный оператор if в качестве цикла путем заикливания. Идея не моя. Где-то когда-то читал и решил поделиться.

[Ответить](#)



23. *Анастасия:*

[28 мая 2019 в 21:06](#)

Вот и моя программа для бго упражнения:

```

1  // программа конвертирует введенное пользователем число от 0 до 255 в двоичную
2  #include <iostream>
3  #include <locale> // для корректного отображения кириллицы
4  #include <cmath> // для использования функции pow возведения в степень
5  using namespace std;
6
7  // результат работы функции - число от 0 до 255
8  unsigned short getNumber()
9  {
10     cout << "Введите число от 0 до 255: ";
11     int n{0};
12     cin >> n;
13
14     while ( n < 0 || n > 255)

```

```
15     {
16         cout << "Что-то пошло не так, попробуйте ещё раз. Введите число от
17         cin >> n;
18     }
19     return n;
20 }
21
22 // функция сравнивает уменьшаемое число с очередной степенью двойки, результат
23 void compare(unsigned short& number, const unsigned short m)
24 {
25     if (number >= pow(2,m) )
26     {
27         cout << "1";
28         number -= pow(2,m);
29     }
30     else cout << "0";
31 }
32 int main()
33 {
34     // для корректного отображения кириллицы
35     setlocale(LC_STYPE, "rus");
36
37     // получаем число от 0 до 255
38     unsigned short number{0};
39     number = getNumber();
40
41     // m - показатель степени двойки от 0 до 7 (2 в 7 степени = 128)
42     // запускаем цикл, который сравнивает искомое число со степенями двойки от
43     for ( short m = 7; m >= 0; --m)
44         compare (number, m);
45
46     return 0;
47 }
```

Ответить



1. Дмитрий:

[12 июня 2019 в 18:57](#)

Было бы куда интересней, и намного корректней, если бы человек выставяющий свой код только в целях повышения самооценки, хоть бы учитывал положения задачи в разделах. Ибо знакомство с оператором цикла "for" и "while" только впереди, а его уже в свой код суют. Конечно это всего лишь мое субъективное мнение, но я нахожу в этом лишь дурной тон. С другой стороны эта тенденция наблюдается у многих. Хотелось бы видеть не наличия операторов, которые в выполнения задачи вряд ли предусматривались, а наличия техники или навык импровизации!

Ответить



24. zvezdonom:

[5 апреля 2019 в 12:38](#)

Всем доброго времени суток. Добавлю и свою программку, которая переводит десятичное целое число, введенное пользователем, в самые распространенные позиционные системы счисления: 2 — двоичную, 3 — троичную, 8 — восьмеричную и 16 — шестнадцатеричную. Хотя может пересчитывать в любую позиционную. Добавил несколько проверок на вводимое число и на повторное вычисление. Компилятор — Qt Creator.

calculate.h

```
1 #ifndef CALCULATE_H
2 #define CALCULATE_H
3
4 // подключаем нужные библиотеки
5 #include <vector>
6 #include <string>
7
8 // объявляем прототипы наших функций
9 int readNumber();
10 std::vector<std::string> convertDecToNewSystem(int inputNumber, int num_system);
11 char enterFlag();
12
13 #endif // CALCULATE_H
```

calculate.cpp

```
1 #include <iostream>
2 #include <vector>
3 #include <string>
4
5 // получаем от пользователя целое десятичное число в диапазоне 0 - 2147483647
6 int readNumber()
7 {
8     long double inputNumber;
9     for(;;)
10     {
11         std::cout << "Please enter a number(integer) over the range 0 to 2 147
12         std::cin >> inputNumber;
13         if(inputNumber < 2147483648 && inputNumber > 0)
14             break;
15     }
16
17     return static_cast<int>(inputNumber);
18 }
19
20 // выполняем преобразование десятичного числа в другие системы счисления и воз
21 std::vector<std::string> convertDecToNewSystem(int inputNumber, int num_system)
22 {
23     // задание вектора, в котором будем хранить число в новой системе исчисле
24     std::vector<std::string> vector_newNumber;
25 }
```

```

26 // с помощью цикла определяем каждый символ нового числа и записываем в вектор
27 while(inputNumber != 0)
28 {
29     // находим i-ю цифру числа в новой системе счисления
30     int new_number_i = inputNumber%num_system;
31
32     // для 16-ной системы задаём соответствующие цифрам буквы
33     std::string new_number_string;
34     if(new_number_i == 10)
35         new_number_string = "A";
36     else if(new_number_i == 11)
37         new_number_string = "B";
38     else if(new_number_i == 12)
39         new_number_string = "C";
40     else if(new_number_i == 13)
41         new_number_string = "D";
42     else if(new_number_i == 14)
43         new_number_string = "E";
44     else if(new_number_i == 15)
45         new_number_string = "F";
46     else
47         new_number_string = std::to_string(new_number_i);
48     // добавляем в конец вектора i-ю цифру числа в новой системе счисления
49     vector_newNumber.push_back(new_number_string);
50     inputNumber /= num_system;
51 }
52
53 return vector_newNumber;
54 }
55
56 // эта функция возвращает символ, введённый пользователем, для проверки хочет ли повторить
57 char enterFlag()
58 {
59     std::cout << "\n\nPlease enter 'y' - if you want repeat conversion or 'n' - if not\n";
60     char flagRepeatOut;
61     std::cin >> flagRepeatOut;
62
63     return flagRepeatOut;
64 }

```

main.cpp

```

1  /* Программа, которая переводит десятичное целое число, введённое пользователем,
2   * в самые распространённые позиционные системы счисления:
3   * 2 - двоичную, 3 - троичную, 8 - восьмиричную и 16 - шестнадцатеричную.
4   */
5  #include <iostream>
6  #include "calculate.h"
7
8  int main()
9  {
10

```



```

11 // бесконечный цикл для того, чтобы пользователь мог выбрать продолжить ем
12 for(;;)
13 {
14     // очищаем экран
15     system("cls");
16
17     // получаем от пользователя число в десятичной системе счисления
18     int inputNumber = readNumber();
19
20     // перебираем в цикле все номера систем счисления, в которые хотим прео
21     for(int num_system = 2; num_system <= 16; num_system++)
22     {
23         // отбираем с помощью условия самые распространённые позиционные с
24         if(num_system < 4 || num_system == 8 || num_system == 16)
25         {
26             std::vector<std::string> vector_newNumber = convertDecToNewSys
27             std::cout << inputNumber << " convert to " << num_system << "
28             // перебираем вектор с полученным числом, но не сначала, а с ко
29             for(int i = vector_newNumber.size()-1; i >= 0; i--)
30                 std::cout << vector_newNumber[i];
31             std::cout << std::endl;
32         }
33     }
34
35     // собственно сама проверка введённых символов пользователя с клавиату
36     char flagRepeatOut = enterFlag();
37     if (flagRepeatOut == 'n')
38         break;
39     else if(flagRepeatOut != 'y')
40     {
41         for(;;)
42         {
43             // очищаем экран
44             system("cls");
45             flagRepeatOut = enterFlag();
46             if (flagRepeatOut == 'n' || flagRepeatOut == 'y')
47                 break;
48         }
49     }
50     if (flagRepeatOut == 'n')
51         break;
52 }
53
54 return 0;
}

```

Результат работы программы:

```

Please enter a number(integer) over the range 0 to 2 147 483 647: 1023
1023 convert to 2 numerical system: 111111111
1023 convert to 3 numerical system: 1101220

```

1023 convert to 8 numerical system: 1777

1023 convert to 16 numerical system: 3FF

Please enter

'y' - if you want repeat conversion or

'n' - if you want quit

[Ответить](#)



25. [Pere_Strelka:](#)

[17 февраля 2019 в 12:07](#)

Жесть, что только не выкладывают, хотя может и верно все это)

Сделал программку, которая переводит десятичное число в двоичное, причем использует всю мощь 4-х байтов 😊

Перевод осуществляется вторым способом, так как делал до этого урока.

Calc+.cpp:

```
1  #include "pch.h"
2  #include <iostream>
3  #include "functions.h"
4  using namespace std;
5
6  int main()
7  {
8      cout << "Chance a work (enter 1 to check for parity or enter 2 to convert to binary)" << endl;
9
10     Work:
11     int num;
12     cin >> num;
13
14     if (num == 0) {
15         return 0;
16     }
17     if (num == 1) {
18         cout << "Enter an integer: ";
19         cin >> num;
20         if (isEven(num))
21             cout << num << " is even.";
22         else
23             cout << num << " isn't even.";
24     }
25     if (num == 2) {
26         cout << "Enter an integer (please enter a number not exceeding 2147483647)";
27         cin >> num;
28         convertToBinary(num);
29     }
30
31     cout << "\n\nEnter 0 to exit or chance work again.\nEnter 1 to check for parity or enter 2 to convert to binary" << endl;
32     goto Work;
33 }
```

functions.h:

```
1 #pragma once
2 #ifndef FUNCTIONS_H
3 #define FUNCTIONS_H
4
5 bool isEven(int num);
6 void convertToBinary(int num);
7 bool approximatelyEqualAbsRel(double a, double b, double absEpsilon, double relEpsilon);
8
9 #endif FUNCTIONS_H
```

functions.cpp:

```
1 #include "pch.h"
2 #include <iostream>
3 using namespace std;
4
5 bool isEven(int num)
6 {
7     return num % 2 == 0;
8 }
9
10 void convertToBinary(int num)
11 {
12     // Сохранил изначальное число
13     int save = num;
14
15     int bin[32]; // Остаток для вывода
16     int y{ 1 }; // Частное
17     int x{ 0 }; // Считает кол-во символов в получившемся двоичном числе
18
19     // Цикл записывает значения типа 1-0 в массив путем деления с остатком на 2
20     while (y != 0) {
21         bin[x] = num % 2;
22         num = y = num / 2;
23         x++;
24     }
25
26     cout << "The number " << save << " in binary is ";
27
28     // Цикл последовательно выводит значения массива с конца
29     int i;
30     for (i = 1; i <= x; i++) {
31         cout << bin[x - i];
32     }
33
34     cout << ".\n";
35 }
```

[Ответить](#)



26. Денис:

[5 февраля 2019 в 17:07](#)

```
1 #include "pch.h"
2 #include <iostream>
3
4 int getNumber()
5 {
6     std::cout << "Enter a number from 0 to 255: \n";
7     int number;
8     std::cin >> number;
9     return number;
10 }
11
12 void printResult(int number)
13 {
14     if ((number / 128 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
15     else std::cout << "0 ";
16     if ((number / 64 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
17     else std::cout << "0 ";
18     if ((number / 32 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
19     else std::cout << "0 ";
20     if ((number / 16 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
21     else std::cout << "0 ";
22     std::cout << " ";
23     if ((number / 8 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
24     else std::cout << "0 ";
25     if ((number / 4 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
26     else std::cout << "0 ";
27     if ((number / 2 >= 1) && (number % 2 == 0)) { std::cout << "1 "; number =
28     else std::cout << "0 ";
29     if (number / 1 >= 1) std::cout << "1 ";
30     else std::cout << "0 ";
31 }
32
33 int main()
34 {
35     printResult(getNumber());
36     return 0;
37 }
```

[Ответить](#)

27. Владимир:

[28 ноября 2018 в 15:31](#)

То самое чувство, когда все просто выводили двоичное представления десятичного числа через cout, а ты запихивал его в int32_t...

[Ответить](#)



28. Евгений:

[22 ноября 2018 в 18:28](#)

Опять усложнил немного, считает любую цифру из десятичной в двоичную.

Основной файл:

```
1 #include "pch.h"
2 #include <iostream>
3 #include <cmath>
4 using namespace std;
5 int raschet(int x, int y)
6 {
7     while (x != 0)
8     {
9         if (x >= y)
10             cout << "1";
11         else
12             cout << "0";
13         if (x >= y)
14         {
15             x = x - y;
16             y = y / 2;
17         }
18         else
19         {
20             y = y / 2;
21             x = x;
22         }
23     }
24 }
25
26 if (y >= 1)
27 {
28     while (y > 1)
29     {
30         cout << "0";
31         y = y / 2;
32     }
33     cout << "0";
34 }
35
36 else
37
38 return 0;
39
40 }
```

Файл с функцией:

```
1 #include "pch.h"
```

```
2 #include <iostream>
3 #include <cmath>
4 using namespace std;
5
6 int raschet(int x, int y);
7 int main()
8 {
9     cout << "Vvedite 4islo: ";
10    int x;
11    cin >> x;
12
13    int ctepen = 2;
14    while (x>=ctepen)
15    {
16        ctepen = ctepen * 2;
17    }
18    int start = ctepen / 2;
19    raschet(x, start);
20
21
22
23    return 0;
24 }
```

Что думаете? Можно как то облегчить или замечания какие-нибудь?

[Ответить](#)



29. *Camio:*

[14 октября 2018 в 20:46](#)

С имеющимися знаниями вышло вот что:

```
1 #include "pch.h"
2 #include <iostream>
3 using namespace std;
4
5 int getNumber()
6 {
7     cout << "Enter integer number from 0 to 255: ";
8     int x;
9     cin >> x;
10    if (x > 255)
11    {
12        cout << "Wrong number\n";
13        getNumber();
14    }
15    else
16        return x;
17 }
18
19 void decToBin(int x)
```

```

20 {
21     int a = 128;
22     int c = 7;
23     int t = 0;
24     while (x > 0)
25     {
26         if (x >= a)
27         {
28             cout << "1";
29             if (x == a)
30             {
31                 while (c > 0)
32                 {
33                     cout << "0";
34                     c--;
35                 }
36             }
37             x -= a;
38         }
39         else
40             cout << "0";
41         t++;
42         if (t % 4 == 0)
43             cout << " ";
44
45         a /= 2;
46         c--;
47     }
48 }
49
50 int main()
51 {
52     int x = getNumber();
53     decToBin(x);
54     return 0;
55 }

```

[Ответить](#)



30. *Алексей:*

[26 сентября 2018 в 11:52](#)

```

1 #include <iostream>
2
3 using namespace std;
4
5 int getVal(); // Запрос числа
6 void checkPow(); // Сравнение числа со степенью двойки
7 int decrVal (int val, int pow); // Уменьшение числа
8 int checkVal (int val, int pow); // Сравнение + уменьшение
9

```

```
10
11 int main()
12 {
13     cout << "Hello!" << endl;
14     int val = getVal();
15     val = checkVal (val, 128);
16     val = checkVal (val, 64);
17     val = checkVal (val, 32);
18     val = checkVal (val, 16);
19     cout << " "; // Пробел (#### ####)
20     val = checkVal (val, 8);
21     val = checkVal (val, 4);
22     val = checkVal (val, 2);
23     val = checkVal (val, 1);
24
25     return 0;
26 }
27 int getVal ()
28 {
29     cout << "Enter an integer value between 0 and 255 please: ";
30     int val;
31     cin >> val;
32     return val;
33 }
34 void checkPow (int val, int pow)
35 {
36     (val >= pow)? (cout << "1"):(cout << "0");
37 }
38 int decrVal (int val, int pow)
39 {
40     if (val >= pow)
41         return val-pow;
42     else
43         return val;
44 }
45 int checkVal (int val, int pow)
46 {
47     checkPow (val, pow);
48     val = decrVal (val, pow);
49     return val;
50 }
```

[Ответить](#)

31. Константин:

[23 сентября 2018 в 20:40](#)

...рассмотрим 148 ещё раз. Какое наибольшее число, УМНОЖЕННОЕ НА 2(из ряда 1, 2, 4, 8, 16, 32, 64, 128, 256 и т.д.), меньше 148? Ответ: 128. Дык ить ежели 128 удвоить оно того 148мого на сто пудов большее будет!!! У меня тихо шифером шурша...

[Ответить](#)



1. *Константин:*

[21 мая 2020 в 22:59](#)

...стоп крыша!!! Это же элементарно, Ватсон! Например: 148 — это длина некоего отрезка в каких-то единицах длины. Для измерения последней в нашем распоряжении имеется набор линеек различной длины: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, ... и т. д. единиц этой самой длины. И требуется всего лишь подобрать такую комбинацию линеек, которая точно без избытка или недостатка будет покрывать интересующий отрезок. В данном случае 148 находится между 128 и 256, значит берём кладём линейку 128 на отрезок и пишем "1". От отрезка 148 остаётся $148 - 128 = 20$ единиц. Ага, $64 > 20$ — пишем "0"; $32 > 20$ — "0"; $16 < 20$ — кладём ещё одну линейку и пишем "1"; $20 - 16 = 4$, значит очередная линейка 8 пропускается и пишется "0", но за то следующая линейка полностью (без остатка) покрывает край отрезка: $4 - 4 = 0$ — измерение отрезка завершено! Отмечаем итоговую линейку "1" и дописываем не использованные линейки 2 — "0" и 1 — "0". Выписываем окончательную комбинацию: 1 0 0 1 0 1 0 0

128 64 32 16 8 4 2 1

[Ответить](#)



32. *Максим:*

[8 сентября 2018 в 01:30](#)

```

1  #include <iostream>
2
3  using namespace std;
4
5  int input()
6  {
7      int x;
8      cout<<"Enter a number from 0 to 255\n";
9      cin>>x;
10     return x;
11 }
12
13 void printOne()
14 {
15     cout<<"1";
16 }
17
18 void printNull()
19 {
20     cout<<"0";
21 }
22
23 int calc(int number,int catOfNum)
24 {
25     if (number>=catOfNum)
26     {
27         printOne();
28         number-=catOfNum;

```

```

29     }
30     else
31     printNull();
32     return number;
33 }
34
35 int main()
36 {
37     int Number(input());
38     Number=calc(Number,128);
39     Number=calc(Number,64);
40     Number=calc(Number,32);
41     Number=calc(Number,16);
42     Number=calc(Number,8);
43     Number=calc(Number,4);
44     Number=calc(Number,2);
45     Number=calc(Number,1);
46     return 0;
47 }

```

[Ответить](#)

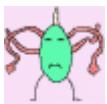


33. *Nikolai:*

[25 августа 2018 в 11:09](#)

Не надо писать фразы типа "является ли число x больше ОПРЕДЕЛЁННОГО ЧИСЛА, УМНОЖЕННОГО НА 2", очень размытый смысл у такой фразы. ОПРЕДЕЛЁННОЕ ЧИСЛО, умноженное на 2 может быть 50? $50 \cdot 2 = 100$. Или $40 \cdot 2 = 80$. Что за определённое число? Если имеется ввиду ряд чисел 2, 4, 8, 16, 32, 64, 128, 256, 512 и т.д., то эти числа лучше называть "степенью двойки".

[Ответить](#)



34. *Katerina:*

[29 июля 2018 в 06:07](#)

Огромное спасибо за ваши уроки! Учусь в университете в Германии, безумно сложно и этот сайт — просто спасение! Немного не поняла задание... вводит в заблуждение "число, умноженное на 2"... непонятно какое же число умножается на 2. Подсказку 2 поняла так, что надо отдельную функцию написать для первого числа. Похоже, неправильно)) Хотелось узнать имеет ли право на жизнь следующий код.

Файл Convert.h

```

1  #ifndef CONVERT_H
2  #define CONVERT_H
3
4  int getNumber();
5  int getFirstNumber(int x);
6  void compare(int x, int y);
7

```

```
8 | #endif // CONVERT_H
```

Convert.cpp

```
1 | #include <iostream>
2 |
3 | int getNumber()
4 | {
5 |     std::cout << "Enter your integer number from interval [0; 255]: ";
6 |     int x;
7 |     std::cin >> x;
8 |     return x;
9 | }
10 |
11 | int getFirstNumber(int x) {
12 |     int firstNumber = 0;
13 |     std::cout << x << " in binary system is ";
14 |     for (int y = 128; y >= 1; y /= 2) {
15 |         if (y == 8) {
16 |             std::cout << " ";
17 |         }
18 |         if (x >= y) {
19 |             std::cout << "1";
20 |             firstNumber = x - y;
21 |             break;
22 |         } else {
23 |             std::cout << "0";
24 |             continue;
25 |         }
26 |     }
27 |     return firstNumber;
28 | }
29 |
30 | void compare(int x, int y) {
31 |     for (int i = y; i >= 1; i /= 2) {
32 |         if (i == 8) {
33 |             std::cout << " ";
34 |         }
35 |         if (x >= i) {
36 |             std::cout << "1";
37 |             x -= i;
38 |             continue;
39 |         } else {
40 |             std::cout << "0";
41 |             continue;
42 |         }
43 |     }
44 |     std::cout << "\n";
45 | }
```

ConvertMain.cpp

```
1 #include "Convert.h"
2
3 int main()
4 {
5     int x = getNumber();
6     int firstNumber = getFirstNumber(x);
7     int y = (x - firstNumber)/2;
8     compare(firstNumber, y);
9     return 0;
10 }
```

И еще раз огромное спасибо за сайт!

[Ответить](#)



35. *YuriiJ:*

[29 июля 2018 в 01:29](#)

У меня получился такой код:

```
1 #include <iostream>
2 using namespace std;
3 int isNumberLessThanEven(int a)
4 {
5     if (a < 2)
6         return 2;
7     if (a < 4)
8         return 4;
9     if (a < 8)
10        return 8;
11    if (a < 16)
12        return 16;
13    if (a < 32)
14        return 32;
15    if (a < 64)
16        return 64;
17    if (a < 128)
18        return 128;
19    if (a < 256)
20        return 256;
21 }
22
23 void code(int a, int x)
24 {
25     cout << "Your code is " << endl;
26     do {
27         if (a >= x)
28             cout << "1";
29         a = a - x;
30
31         if (a < x)
32             cout << "0";
```

```

33     x = x / 2;
34     } while (x = 1);
35 }
36
37 int main()
38 {
39     cout << "Enter number from 0 to 255" << endl;
40     int a;
41     cin >> a;
42     int maxNum{ isNumberLessThanEven(a) };
43     code(a, maxNum);
44     system ("pause");
45     return 0;
46 }

```

Но прога зацикливается. Почему?

[Ответить](#)



1. *Saliwer:*

[30 августа 2019 в 20:30](#)

Наверное уже не очень актуально, но всё таки. У Вас в условии цикла while оператор присваивания =, а не проверка на равенство ==

[Ответить](#)



36. *master114:*

[4 мая 2018 в 13:00](#)

Не совсем понял вторую подсказку, поэтому фантазировал как мог.

Рад, что в итоге получилось, хоть и не так компактно как уже предложенные варианты, зато рабочий код -)))

Код — onlinegdb.com/Hkeby3KTG.

[Ответить](#)



1. *Юрий:*

[5 мая 2018 в 22:33](#)

Имелись в виду числа 1, 2, 4, 8, 16, 32, 64 и 128. Код рабочий, как вариант, может быть — у вас всё просто более развернуто получилось 😊

[Ответить](#)



37. *Anton:*

[20 февраля 2018 в 06:53](#)

```

1 #include "stdafx.h"
2 #include <iostream>

```

```

3 using namespace std;
4
5
6 int main()
7 {
8     cout << "Insert a number from 0 to 255: ";
9     int x = 0;
10    cin >> x;
11    while (x < 0 || x > 255) {
12        cout << "You took a wrong number, choose it again: ";
13        cin >> x;
14    }
15    cout << endl;
16    for (int i = 128; i > 0; i /= 2) {
17        if (x >= i) {
18            cout << "1";
19            x = x - i;
20        }
21        else cout << "0";
22    }
23    cout << endl;
24    return 0;
25 }

```

[Ответить](#)



1. **Юрий:**
[20 февраля 2018 в 14:00](#)

Да, можно и так.

[Ответить](#)



2. **Александр:**
[21 февраля 2018 в 19:05](#)

Я чуть дополнил , просто для интереса:

```

1 #include "stdafx.h"
2 #include <iostream>
3 #include <cmath> // для fabs()
4 #include <locale> // для корректного отображения кириллицы в консольном
5 using namespace std; // объявление пространства имён std
6 int main();
7
8 // x - число, которое будем тестировать
9 // pow - это множитель 2 (например, 128, 64, 32 и т.д.)
10 int printandDecrementBit(__int64 x, __int64 pow)
11 {
12     // Проверяем, является ли X больше определенного числа, умноженного на
13     if (x >= pow)

```

```
14         std::cout << "1";
15     else
16         std::cout << "0";
17
18     // Если x больше, чем число, умноженное на 2 - вычитаем его из значения
19     if (x >= pow)
20         return x - pow;
21     else
22         return x;
23 }
24
25 int decimalBinary()
26 {
27     setlocale(LC_ALL, "Russian");
28
29     std::cout << "\n";
30     std::cout << "\tПеревод десятичного числа в двоичное""\n";
31     std::cout << "Введите целое число от 0 до 1'099'511'627'775: ";
32     __int64 x;
33     std::cin >> x;
34     if (x > 1099511627775)
35     {
36         std::cout << "Введите допустимое значение""\n";
37         return decimalBinary();
38     }
39     if (x < 0)
40     {
41         std::cout << "Введите допустимое значение""\n";
42         return decimalBinary();
43     }
44     std::cout << "\n";
45     x = printandDecrementBit(x, 2147483648);
46     x = printandDecrementBit(x, 1073741824);
47     x = printandDecrementBit(x, 536870912);
48     x = printandDecrementBit(x, 268435456);
49     std::cout << " ";
50     x = printandDecrementBit(x, 134217728);
51     x = printandDecrementBit(x, 67108864);
52     x = printandDecrementBit(x, 33554432);
53     x = printandDecrementBit(x, 16777216);
54     std::cout << " ";
55     x = printandDecrementBit(x, 8388608);
56     x = printandDecrementBit(x, 4194304);
57     x = printandDecrementBit(x, 2097152);
58     x = printandDecrementBit(x, 1048576);
59     std::cout << " ";
60     x = printandDecrementBit(x, 524288);
61     x = printandDecrementBit(x, 262144);
62     x = printandDecrementBit(x, 131072);
63     x = printandDecrementBit(x, 65536);
64     std::cout << " ";
65     x = printandDecrementBit(x, 32768);
```

```

66 x = printandDecrementBit(x, 16384);
67 x = printandDecrementBit(x, 8192);
68 x = printandDecrementBit(x, 4096);
69 std::cout << " ";
70 x = printandDecrementBit(x, 2048);
71 x = printandDecrementBit(x, 1024);
72 x = printandDecrementBit(x, 512);
73 x = printandDecrementBit(x, 256);
74 std::cout << " ";
75 x = printandDecrementBit(x, 128);
76 x = printandDecrementBit(x, 64);
77 x = printandDecrementBit(x, 32);
78 x = printandDecrementBit(x, 16);
79 std::cout << " ";
80 x = printandDecrementBit(x, 8);
81 x = printandDecrementBit(x, 4);
82 x = printandDecrementBit(x, 2);
83 x = printandDecrementBit(x, 1);
84 std::cout << "\n";
85
86 return main();
87 }

```

[Ответить](#)

1. *Юрий:*
[21 февраля 2018 в 23:46](#)

Код не рабочий.

[Ответить](#)

2. *Роман:*
[21 июля 2018 в 20:42](#)

Боюсь дальше уже стоит использовать циклы, а также можно попробовать первый способ, на котором у меня уже есть такая прога, я в неё ещё впихну второй способ и скину сюда

[Ответить](#)

3. *master114:*
[4 мая 2018 в 12:56](#)

мне кажется на этапе цикла while может возникнуть ситуация бесконечного ввода
 Вдруг пользователь не понимает что от него хотят -))

Тогда нужно добавить какое-то количество попыток после чего программа автоматом должна прерываться

[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 «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

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