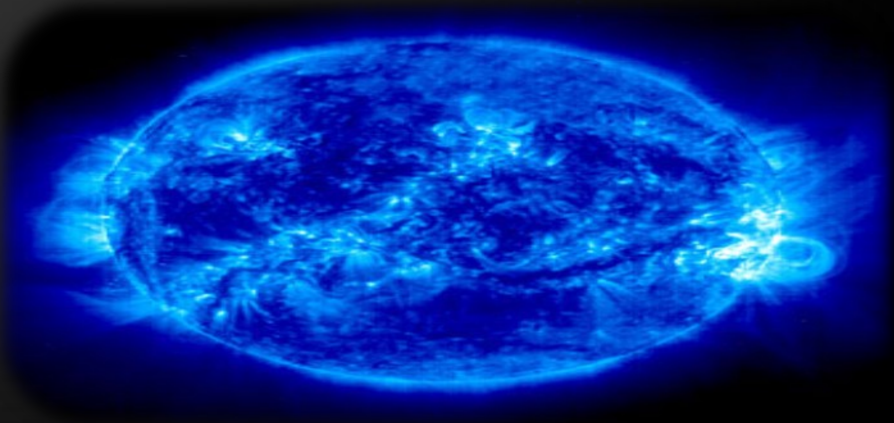
Operators and Expressions

Performing Simple Calculations with C#

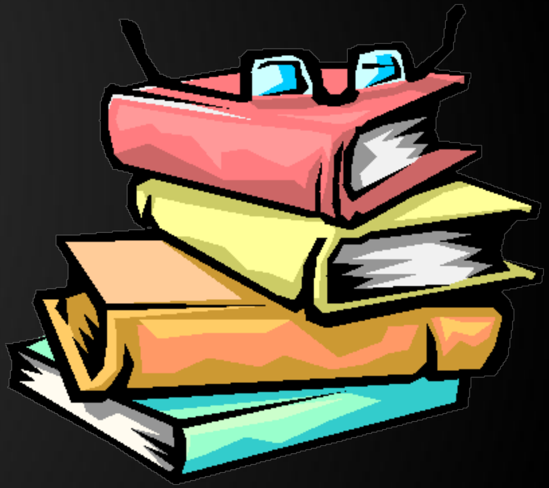


Svetlin Nakov

Telerik Corporation

www.telerik.com

Table of Contents



Operators in C#

Arithmetic, Logical, Comparison, Assignment, Etc.

What is an Operator?

- **Operator** is an operation performed over data at runtime
  - Takes one or more arguments (operands)
  - Produces a new value
- Operators have precedence
  - Precedence defines which will be evaluated first
- **Expressions** are sequences of operators and operands that are evaluated to a single value

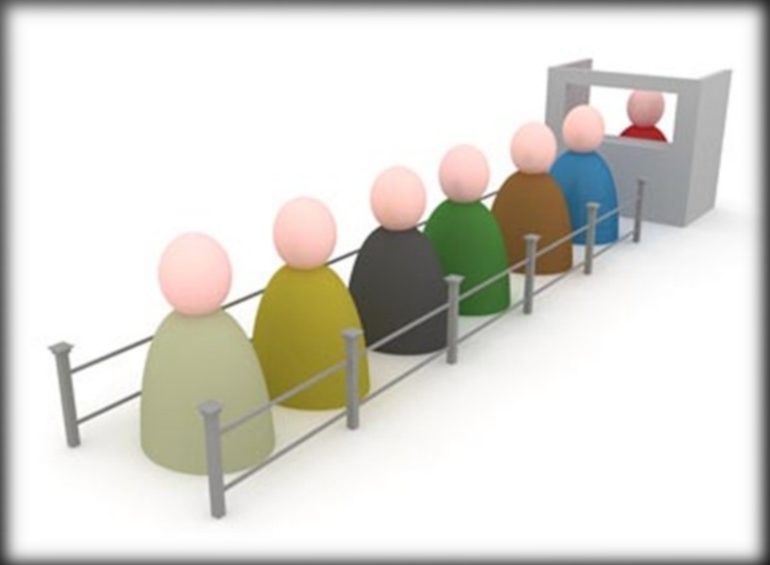
Operators in C#

- Operators in C# :
  - Unary – take one operand
  - Binary – take two operands
  - Ternary (**?:**) – takes three operands
- Except for the assignment operators, all binary operators are left-associative
- The assignment operators and the conditional operator (**?:**) are right-associative

Categories of Operators in C#

Category	Operators
Arithmetic	`+ - * / % ++ --`
Logical	`&& || ^ !`
Binary	`& | ^ ~ << >>`
Comparison	`== != < > <= >=`
Assignment	`= += -= *= /= %= &= |= ^= <<= >>=`
String concatenation	`+`
Type conversion	`is as typeof`
Other	`. [] () ?: new`

Operators Precedence



Operators Precedence

Precedence	Operators
Highest	`++ --` (postfix) `new typeof`
	`++ --` (prefix) `+ -` (unary) `! ~`
	`* / %`
	`+ -`
	`<< >>`
	`< > <= >= is as`
	`== !=`
	`&`
Lower	`^`

Operators Precedence (2)

Precedence	Operators
Higher	|
	&&
	||
	?:
Lowest	= *= /= %= += -= <<= >>= &= ^= |=

- Parenthesis operator always has highest precedence
- Note: prefer using parentheses, even when it seems stupid to do so

Arithmetic Operators

Arithmetic Operators

- Arithmetic operators **+**, **-**, ***** are the same as in math
- Division operator **/** if used on integers returns integer (without rounding) or exception
- Division operator **/** if used on real numbers returns real number or `Infinity` or `NaN`
- Remainder operator **%** returns the remainder from division of integers
- The special addition operator **++** increments a variable

telerik

Arithmetic Operators – Example

```
int squarePerimeter = 17;
double squareSide = squarePerimeter/4.0;
double squareArea = squareSide*squareSide;
Console.WriteLine(squareSide); // 4.25
Console.WriteLine(squareArea); // 18.0625

int a = 5;
int b = 4;
Console.WriteLine( a + b ); // 9
Console.WriteLine( a + b++ ); // 9
Console.WriteLine( a + b ); // 10
Console.WriteLine( a + (++b) ); // 11
Console.WriteLine( a + b ); // 11

Console.WriteLine(11 / 3); // 3
Console.WriteLine(11 % 3); // 2
Console.WriteLine(12 / 3); // 4
```

Arithmetic Operators

Live Demo

Logical Operators

Logical Operators

- Logical operators take boolean operands and return boolean result
- Operator `!` turns `true` to `false` and `false` to `true`
- Behavior of the operators `&&`, `||` and `^` (`1` == `true`, `0` == `false`) :

Operation	||	||	||	||	&&	&&	&&	&&	^	^	^	^
Operand1	0	0	1	1	0	0	1	1	0	0	1	1
Operand2	0	1	0	1	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	0	0	1	0	1	1	0

Logical Operators – Example

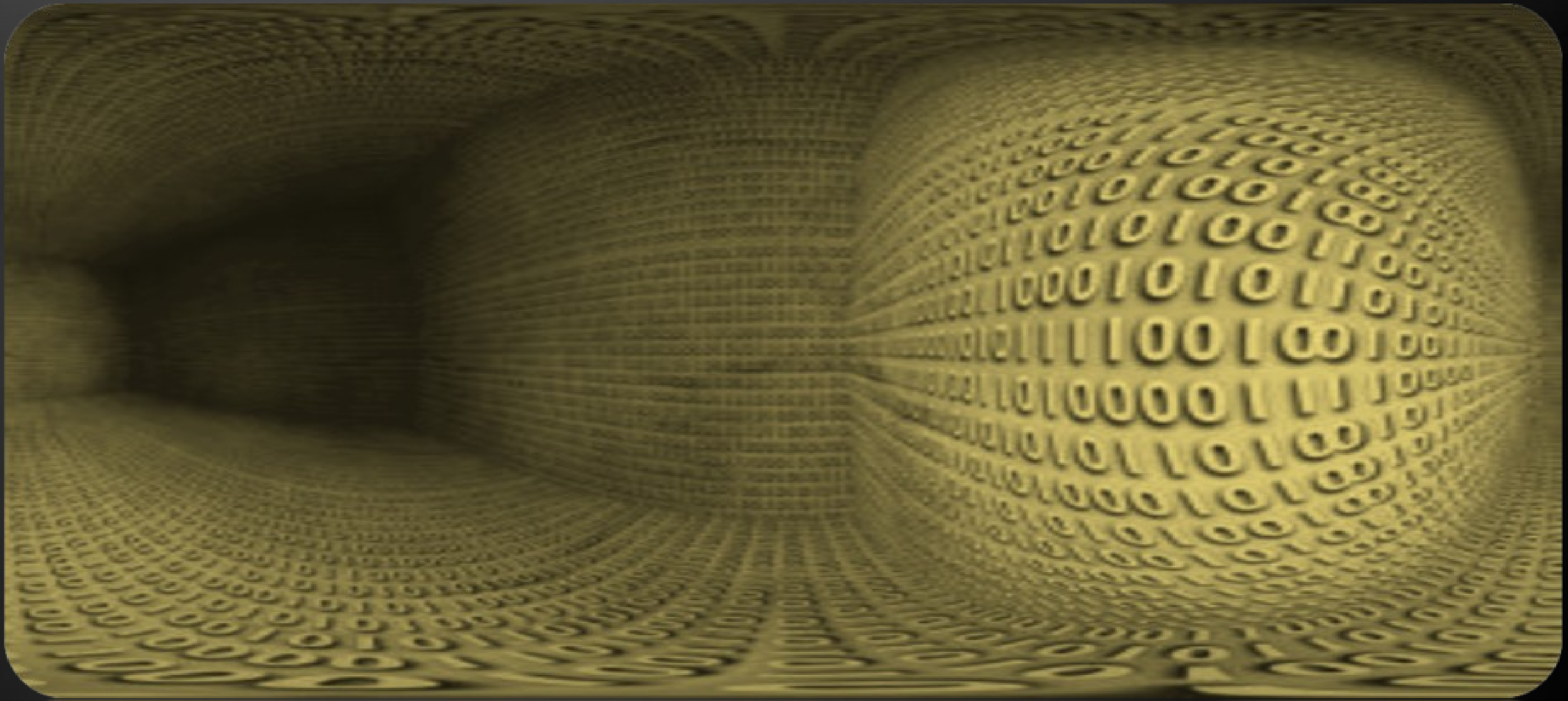
- Using the logical operators:

```
bool a = true;
bool b = false;
Console.WriteLine(a && b); // False
Console.WriteLine(a || b); // True
Console.WriteLine(a ^ b); // True
Console.WriteLine(!b); // True
Console.WriteLine(b || true); // True
Console.WriteLine(b && true); // False
Console.WriteLine(a || true); // True
Console.WriteLine(a && true); // True
Console.WriteLine(!a); // False
Console.WriteLine((5>7) ^ (a==b)); // False
```

Logical Operators

Live Demo

Bitwise Operators



Bitwise Operators

- Bitwise operator `~` turns all `0` to `1` and all `1` to `0`
  - Like `!` for boolean expressions but bit by bit
- The operators `|`, `&` and `^` behave like `||`, `&&` and `^` for boolean expressions but bit by bit
- The `<<` and `>>` move the bits (left or right)
- Behavior of the operators `|`, `&` and `^`:

Operation	|	|	|	|	&	&	&	&	^	^	^	^
Operand1	0	0	1	1	0	0	1	1	0	0	1	1
Operand2	0	1	0	1	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	0	0	1	0	1	1	0

Bitwise Operators (2)

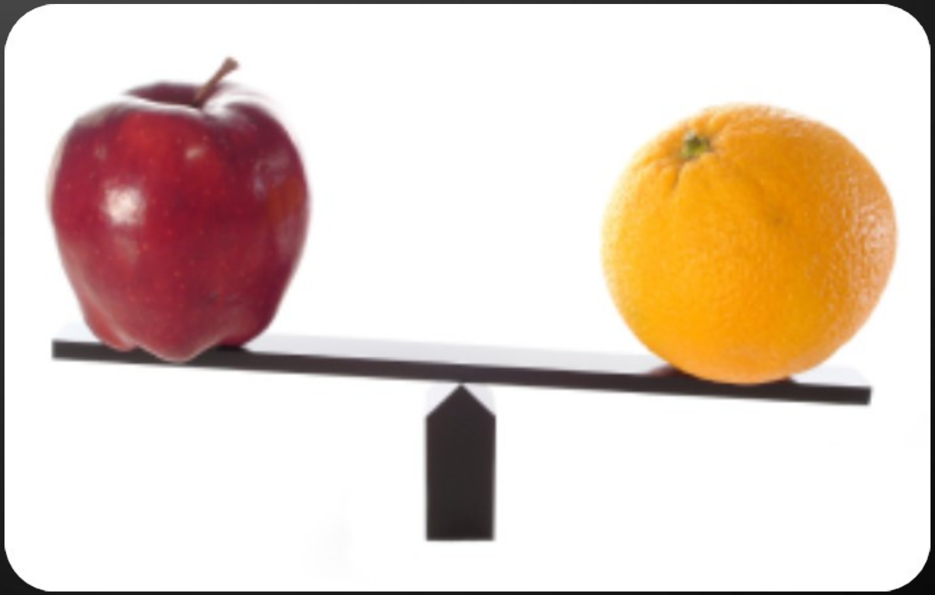
- Bitwise operators are used on integer numbers (byte, sbyte, int, uint, long, ulong)
- Bitwise operators are applied bit by bit
- Examples:

```
ushort a = 3;                  // 00000000 00000011
ushort b = 5;                  // 00000000 00000101
Console.WriteLine( a | b); // 00000000 00000111
Console.WriteLine( a & b); // 00000000 00000001
Console.WriteLine( a ^ b); // 00000000 00000110
Console.WriteLine(~a & b); // 00000000 00000100
Console.WriteLine( a<<1 ); // 00000000 00000110
Console.WriteLine( a>>1 ); // 00000000 00000001
```

Bitwise Operators

Live Demo

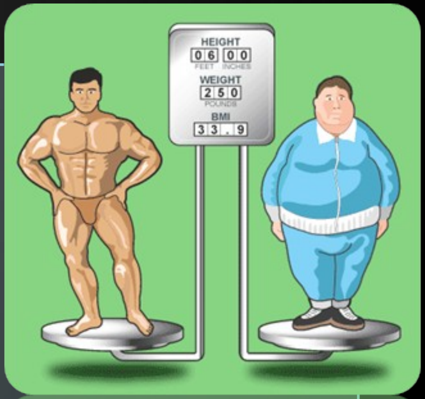
Comparison and Assignment Operators



Comparison Operators

- Comparison operators are used to compare variables
  - ==, <, >, >=, <=, !=
- Comparison operators example:

```
int a = 5;
int b = 4;
Console.WriteLine(a >= b); // True
Console.WriteLine(a != b); // True
Console.WriteLine(a == b); // False
Console.WriteLine(a == a); // True
Console.WriteLine(a != ++b); // False
Console.WriteLine(a > b); // False
```



telerik

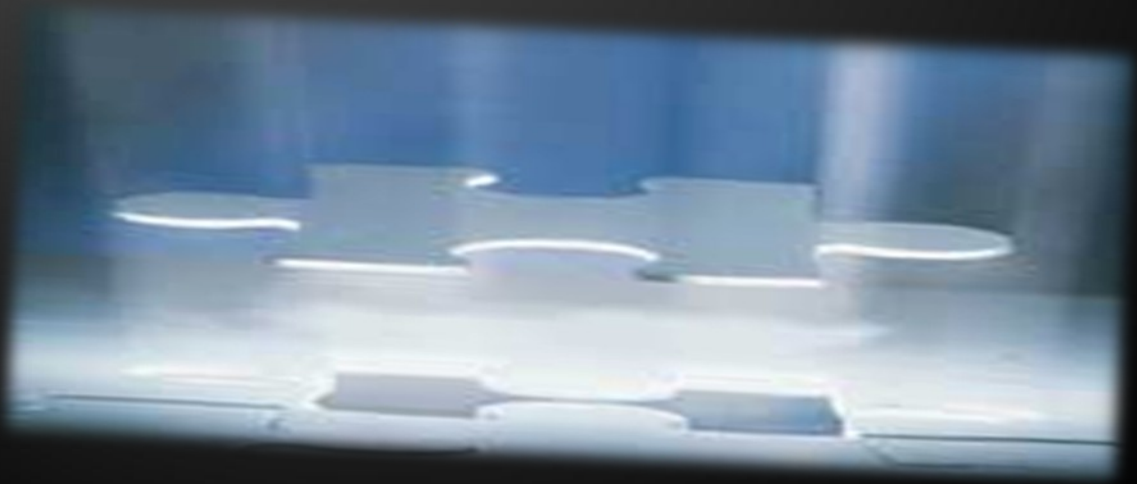
Assignment Operators

- Assignment operators are used to assign a value to a variable ,
  - =, +=, -=, |=, ...
- Assignment operators example:

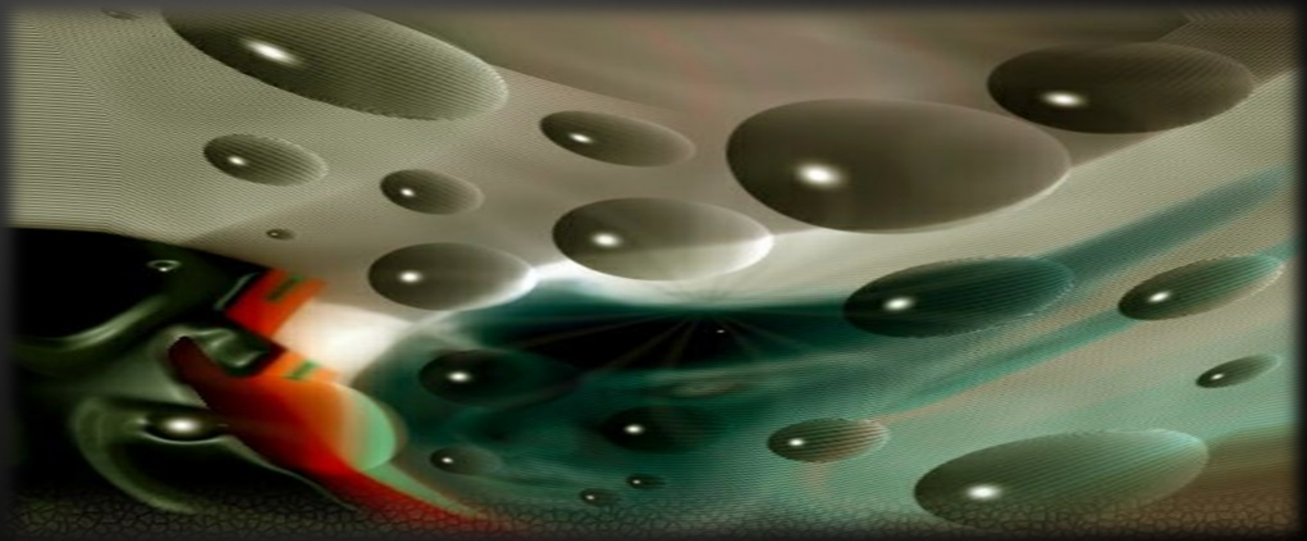
```
int x = 6;
int y = 4;
Console.WriteLine(y *= 2); // 8
int z = y = 3; // y=3 and z=3
Console.WriteLine(z); // 3
Console.WriteLine(x |= 1); // 7
Console.WriteLine(x += 3); // 10
Console.WriteLine(x /= 2); // 5
```

Comparison and Assignment Operators

Live Demo



Other Operators



Other Operators

- String concatenation operator **+** is used to concatenate strings
- If the second operand is not a string, it is converted to string automatically

```
string first = "First";
string second = "Second";
Console.WriteLine(first + second);
// FirstSecond
string output = "The number is : ";
int number = 5;
Console.WriteLine(output + number);
// The number is : 5
```

Other Operators (2)

- Member access operator `.` is used to access object members
- Square brackets `[]` are used with arrays indexers and attributes
- Parentheses `( )` are used to override the default operator precedence
- Class cast operator `(type)` is used to cast one compatible type to another

Other Operators (3)

- Conditional operator `?:` has the form

```
b ? x : y
```

  (if `b` is true then the result is `x` else the result is `y`)

- The `new` operator is used to create new objects
- The `typeof` operator returns `System.Type` object (the reflection of a type)
- The `is` operator checks if an object is compatible with given type

Other Operators – Example

- Using some other operators:

```
int a = 6;
int b = 4;
Console.WriteLine(a > b ? "a>b" : "b>=a"); // a>b
Console.WriteLine((long) a); // 6

int c = b = 3; // b=3; followed by c=3;
Console.WriteLine(c); // 3
Console.WriteLine(a is int); // True
Console.WriteLine((a+b)/2); // 4
Console.WriteLine(typeof(int)); // System.Int32

int d = new int();
Console.WriteLine(d); // 0
```

Other Operators

Live Demo

Implicit and Explicit Type Conversions

Implicit Type Conversion

- Implicit Type Conversion
  - Automatic conversion of value of one data type to value of another data type
  - Allowed when no loss of data is possible
    - "Larger" types can implicitly take values of smaller "types"
  - Example:

```
int i = 5;
long l = i;
```

Explicit Type Conversion

- Explicit type conversion
  - Manual conversion of a value of one data type to a value of another data type
  - Allowed only explicitly by `(type)` operator
  - Required when there is a possibility of loss of data or precision
  - Example:

```
long l = 5;
int i = (int) l;
```

Type Conversions – Example

- Example of implicit and explicit conversions:

```
float heightInMeters = 1.74f; // Explicit conversion
double maxHeight = heightInMeters; // Implicit

double minHeight = (double) heightInMeters; // Explicit

float actualHeight = (float) maxHeight; // Explicit

float maxHeightFloat = maxHeight; // Compilation error!
```

- Note: Explicit conversion may be used even if not required by the compiler

Type Conversions

Live Demo

Expressions

Expressions

- Expressions are sequences of operators, literals and variables that are evaluated to some value
- Examples:

```
int r = (150-20) / 2 + 5; // r=70

// Expression for calculation of circle area
double surface = Math.PI * r * r;

// Expression for calculation of circle perimeter
double perimeter = 2 * Math.PI * r;
```

telerik

Expressions (2)

- Expressions has:
  - Type (integer, real, boolean, ...)
  - Value
- Examples:

Expression of type int. Calculated at compile time.

Expression of type int. Calculated at runtime.

```
int a = 2 + 3; // a = 5
int b = (a+3) * (a-4) + (2*a + 7) / 4;  // b = 12
bool greater = (a > b) || ((a == 0) && (b == 0));
```

Expression of type bool. Calculated at runtime.

Expressions

Live Demo

Summary

- We discussed the operators in C#:
  - Arithmetic, logical, bitwise, comparison, assignment and others
  - Operator precedence
- We learned when to use implicit and explicit type conversions
- We learned how to use expressions

telerik

Operators and Expressions

Questions?

Exercises

1. Write an expression that checks if given integer is odd or even.
2. Write a boolean expression that checks for given integer if it can be divided (without remainder) by 7 and 5 in the same time.
3. Write an expression that calculates rectangle's area by given `width` and `height`.
4. Write an expression that checks for given integer if its third digit (right-to-left) is `7`. E. g. `1732` → `true`.
5. Write a boolean expression for finding if the bit `3` (counting from `0`) of a given integer is `1` or `0`.
6. Write an expression that checks if given point (`x`, `y`) is within a circle K(0, 5).

Exercises (2)

1. Write an expression that checks if given positive integer number n (n ≤ 100) is prime. E.g. 37 is prime.

E Write an expression that calculates trapezoid's area by given sides a and b and height h.

i Write an expression that checks for given point (x, y) if it is within the circle K((1,1), 3) and out of the rectangle R(top=1, left=-1, width=6, height=2).

Write a boolean expression that returns if the bit at position p (counting from 0) in a given integer number v has value of 1. Example: v=5; p=1 → false.

Exercises (3)

s Write an expression that extracts from a given integer `i` the value of a given bit number `b`. Example: i=5; b=2 → value=1.

2. We are given integer number `n`, value `v` (`v`=0 or 1) and a position `p`. Write a sequence of operators that modifies `n` to hold the value `v` at the position `p` from the binary representation of `n`.

   Example: n = 5 (00000101), p=3, v=1 → 13 (00001101)

   n = 5 (00000101), p=2, v=0 → 1 (00000001)

Exercises (4)

1. Write a program that exchanges bits 3, 4 and 5 with bits 24,25 and 26 of given 32-bit unsigned integer.
2. * Write a program that exchanges bits {p, p+1, …, p+k-1) with bits {q, q+1, q+k-1} of given 32-bit unsigned integer.