C# Basics

Variables

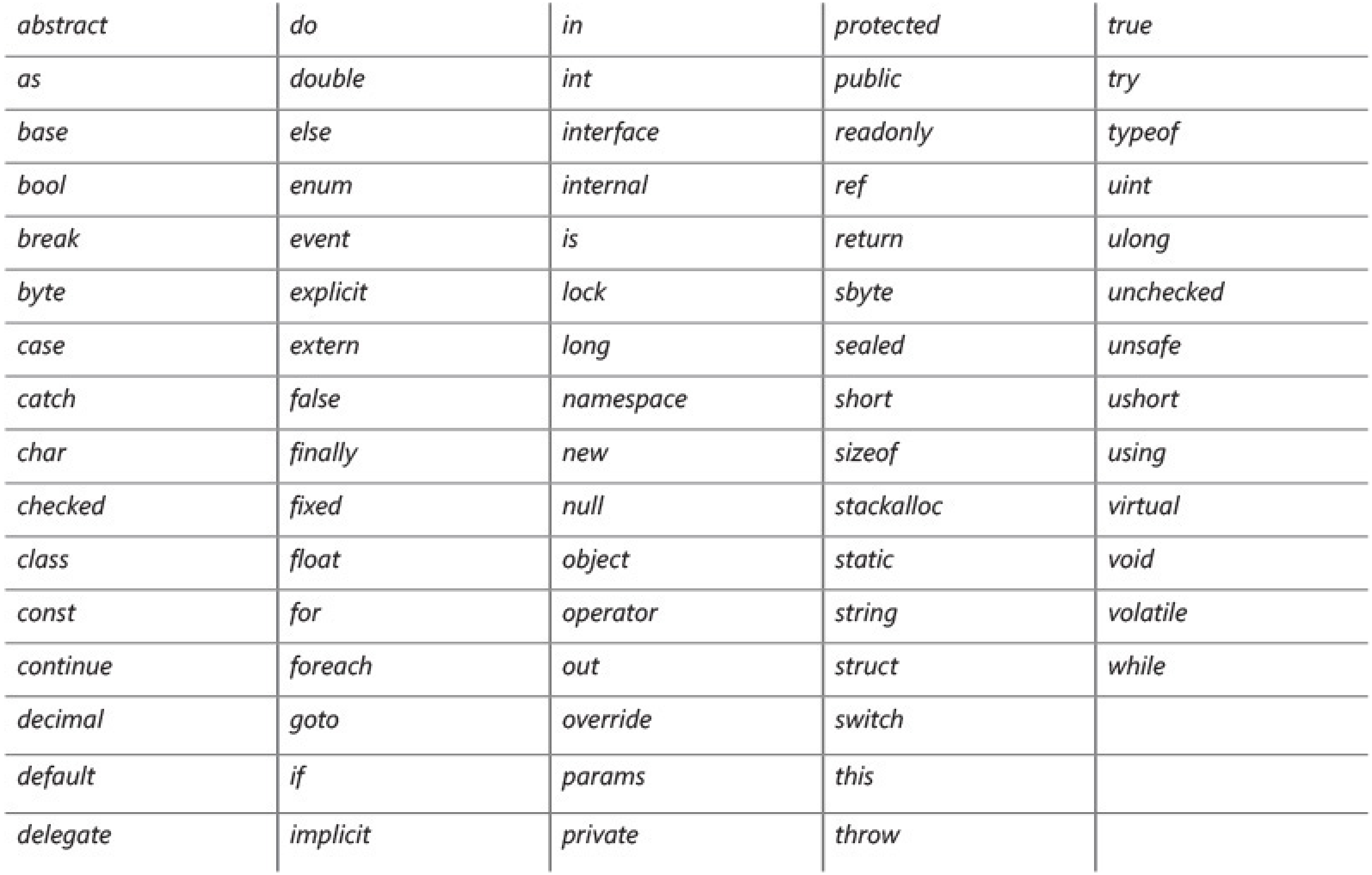
Using identifiers

- Identifiers are the names that you use to identify the elements in your programs, such as namespaces, classes, methods, and variables. In C#, you must adhere to the following syntax rules when choosing identifiers:
    - You can use only letters (uppercase and lowercase), digits, and underscore characters.
    - An identifier must start with a letter or an underscore.

      For example, result, _score, footballTeam, and plan9 are all valid identifiers, whereas result%, footballTeam$, and 9plan are not.
- C# is a case-sensitive language: *footballTeam* and *FootballTeam* are two different identifiers.

Reserved Keyword

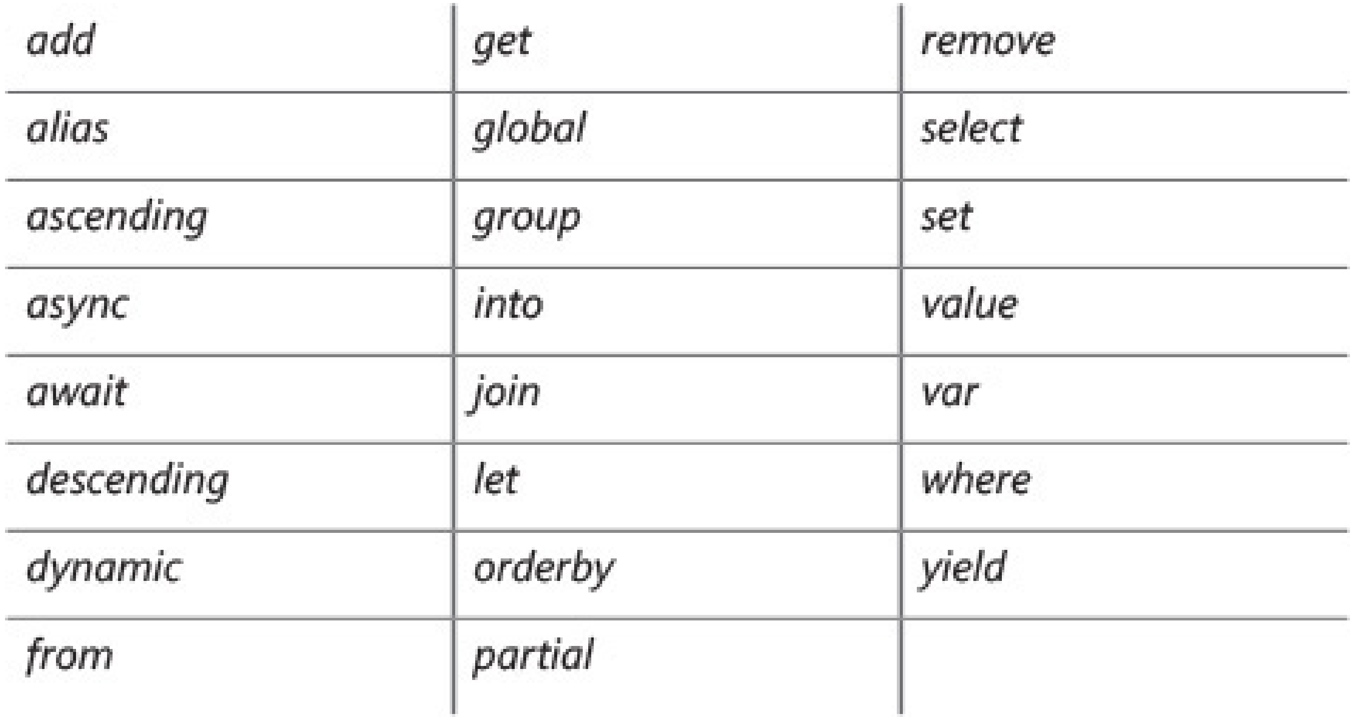
- The C# language reserves certain identifiers for its own use, and you cannot reuse these identifiers for your own purposes.



abstract	do	in	protected	true
as	double	int	public	try
base	else	interface	readonly	typeof
bool	enum	internal	ref	uint
break	event	is	return	ulong
byte	explicit	lock	sbyte	unchecked
case	extern	long	sealed	unsafe
catch	false	namespace	short	ushort
char	finally	new	sizeof	using
checked	fixed	null	stackalloc	virtual
class	float	object	static	void
const	for	operator	string	volatile
continue	foreach	out	struct	while
decimal	goto	override	switch	
default	if	params	this	
delegate	implicit	private	throw	

Keyword

- C# also uses the following identifiers. These identifiers are **not reserved** by C#, which means that you can use these names as identifiers for your own methods, variables, and classes, but you should **avoid doing so if at all possible**.

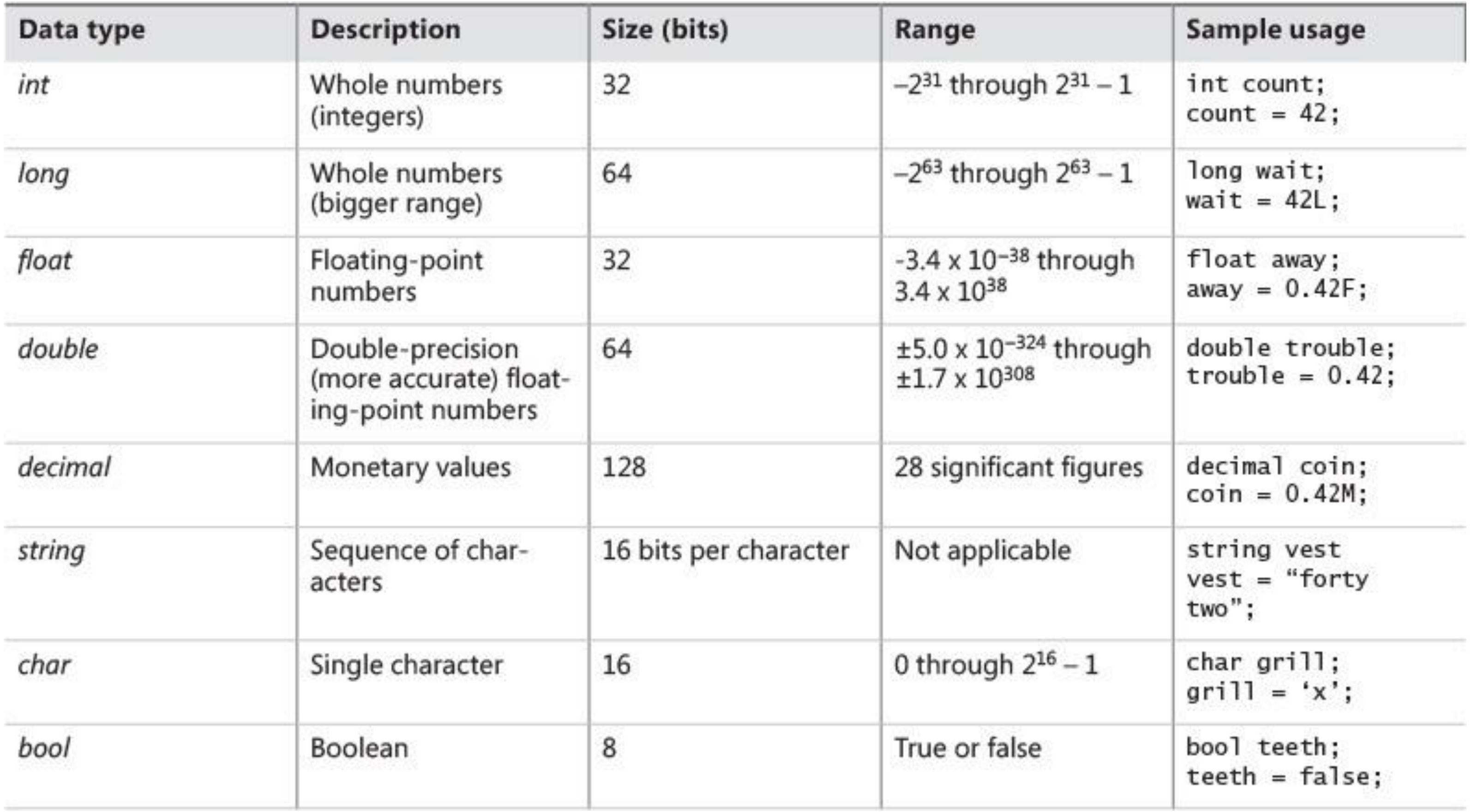


add	*get*	*remove*
alias	*global*	*select*
ascending	*group*	*set*
async	*into*	*value*
await	*join*	*var*
descending	*let*	*where*
dynamic	*orderby*	*yield*
from	*partial*	

Important C# Variable Types

- Core C# variable types start with a lowercase character
  - bool
  - int
  - float
  - char
  - string
  - class

C# Built-In Primitive Data Types



Data type	Description	Size (bits)	Range	Sample usage

int	Whole numbers (integers)	32	-2^{31} through $2^{31} - 1$	`int count;`
				`count = 42;`

long	Whole numbers (bigger range)	64	-2^{63} through $2^{63} - 1$	`long wait;`
				`wait = 42L;`

float	Floating-point numbers	32	-3.4×10^{-38} through 3.4×10^{38}	`float away;`
				`away = 0.42F;`

double	Double-precision (more accurate) floating-point numbers	64	$\pm 5.0 \times 10^{-324}$ through $\pm 1.7 \times 10^{308}$	`double trouble;`
				`trouble = 0.42;`

decimal	Monetary values	128	28 significant figures	`decimal coin;`
				`coin = 0.42M;`

string	Sequence of characters	16 bits per character	Not applicable	`string vest`
				`vest = “forty two”;`

char	Single character	16	0 through $2^{16} - 1$	`char grill;`
				`grill = ‘x’;`

bool	Boolean	8	True or false	`bool teeth;`
				`teeth = false;`

Important C# Variable Types

- **bool** – A 1-bit True or False Value
  - Short for Boolean
  - Named after George Boole (an English mathematician)
  - bools in C# actually use more than 1-bit of space
    - The smallest addressable memory chunk on a 32-bit system is 32 bits.
    - The smallest on a 64-bit system is 64 bits.
  - Literal examples: true   false
  - `bool verified = true;`

Important C# Variable Types

- **int** – A 32-bit Integer
  - Stores a single integer number
    - Integers are numbers with no fractional or decimal element
  - int math is very fast and accurate
  - Can store numbers between –2,147,483,648 and 2,147,483,647
  - 31 bits used for number and 1 bit used for sign
  - Literal examples: 1 34567 -48198
  - `int nonFractionalNumber = 12345;`

Important C# Variable Types

- **float** – A 32-bit Decimal Number
  - Stores a floating-point number with a decimal element
    - A floating-point number is stored in something like *scientific notation*
    - Scientific notation is numbers in the format $a*10^b$: 300 is $3*10^2$
  - Floating-point numbers are stored in the format $a*2^b$
    - 23 bits are used for the significand (the a part)
    - 8 bits are used for the exponent (the ^{b} part)
    - 1 bit determines whether the number is positive or negative
  - Floats are *inaccurate* for large numbers and for numbers between -1 and 1
    - There is no accurate float representation for 1 / 3
  - Literal examples: 3.14f   123f   123.456f
  - `float notPreciselyOneThird = 1.0f / 3.0f;`

Important C# Variable Types

- **char** – A 16-bit Character
  - Single character represented by 16 bits of information
  - Uses Unicode values for the characters
    - Unicode represents 110,000 different characters from over 100 different character sets and languages
  - Floats are *inaccurate* for large numbers and for numbers between -1 and 1
    - There is no accurate float representation for 1 / 3
  - Uppercase and lowercase letters are different values!
  - char literals are surrounded by single quotes
  - Literal examples: 'A' 'a' '\t'
  - `char theLetterA = 'A';`

Important C# Variable Types

- **string** – A Series of 16-bit Characters
  - Stores from no characters ("") to an entire novel
    - Max length is 2 billion chars; 12,000 times the length of Hamlet
  - string literals are surrounded by double quotes
  - Literal examples: `"Hello"` `""` `"\tTab"`
  - `string theFirstLineOfHamlet = "Who's there?";`
  - You can access individual characters via *bracket access*
    - `char theCharW = theFirstLineOfHamlet[0];`
    - `char questionMark = theFirstLineOfHamlet[11];`
  - The length of a string is accessed via .Length
    - `int len = theFirstLineOfHamlet.Length;`
      - Sets len to 12

Important C# Variable Types

- **class** – A Collection of Functions and Data
  - A class creates a new variable type
  - Covered extensively in later Modules
  - Already used in the HelloWorld project

    Everything between the braces { } is part of the class