Here are the references to Unicode in C# - DIS 23270

L2/02-123

Chapter 2 – Conformance: Page 19 – lines 28-30:

A conforming implementation of C# shall interpret characters in conformance with the Unicode Standard, Version 3.0 or later, and ISO/IEC 10646-1. Conforming implementations must accept Unicode source files encoded with the UTF-8 encoding form.

Chapter 3 – References: Page 21 – lines 20-22:

The Unicode Consortium. The Unicode Standard, Version 3.0, defined by: *The Unicode Standard, Version 3.0* (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), and Unicode Technical Report #15: *Unicode Normalization Form*s.

Chapter 4 – Definitions: Pages 24 – lines 14-15:

Source file — an ordered sequence of Unicode characters. Source files typically have a one-to-one correspondence with files in a file system, but this correspondence is not required.

Chapter 5 – Notational Conventions: Page 26 – lines 11-12:

for-statement:
for (; ;) *embedded-statement*
for (*for-initializer* ; ;) *embedded-statement*
for (; *for-condition* ;) *embedded-statement*
for(; ;*for-iterator*) *embedded-statement*
for (*for-initializer* ; *for-condition* ;) *embedded-statement*
for (; *for-condition* ; *for-iterator*) *embedded-statement*
for (*for-initializer* ; ;*for-iterator*) *embedded-statement*
for (*for-initializer* ; *for-condition* ; *for-iterator*) *embedded-statement*

All terminal characters are to be understood as the appropriate Unicode character from the ASCII range, as opposed to any similar-looking characters from other Unicode ranges.

Chapter 8 – Language Overview, Section 8.2.1: Page 33 -34:

8.2.1 Predefined types

C# provides a set of predefined types, most of which will be familiar to C and C++ developers.
The predefined reference types are object and string. The type object is the ultimate base type of all other types. The type string is used to represent Unicode string values. Values of type string are immutable. The predefined value types include signed and unsigned integral types, floating-point types, and the types bool, char,anddecimal. The signed integral types are sbyte, short, int,and long; the unsigned integral types are byte, ushort, uint,andulong; and the floating-point types are float and double.

The bool type is used to represent boolean values: values that are either true or false. The inclusion of bool makes it easier to write self-documenting code, and also helps eliminate the all-too-common C++ coding error in which a developer mistakenly uses "="when"==" should have been used. In C#, the example

```
int i = –;
F(i);
if (i = 0) // Bug: the test should be (i == 0)
G();
```

results in a compile-time error because the expression i = 0 is of type int,andif statements require an expression of type bool.

The char type is used to represent Unicode characters. A variable of type char represents a single 16-bit Unicode character.

The decimal type is appropriate for calculations in which rounding errors caused by floating point representations are unacceptable. Common examples include financial calculations such as tax computations and currency conversions. The decimal type provides 28 significant digits.

The table below lists the predefined types, and shows how to write literal values for each of them.

Type	**Description**	**Example**
object	The ultimate base type of all other types	object o = null;
string	String type; a string is a sequence of Unicodecharacters	string s = "hello";
sbyte	8-bit signed integral type	sbyte val = 12;
short	16-bit signed integral type	short val = 12;
int	32-bit signed integral type	int val = 12;
long	64-bit signed integral type	long val1 = 12;
		long val2 = 34L;
byte	8-bit unsigned integral type	byte val1 = 12;
ushort	16-bit unsigned integral type	ushort val1 = 12;
uint	32-bit unsigned integral type	uint val1 = 12;
		uint val2 = 34U;
ulong	64-bit unsigned integral type	ulong val1 = 12;
		ulong val2 = 34U;
		ulong val3 = 56L;
		ulong val4 = 78UL;
float	Single-precision floating point type	float val = 1.23F;
double	Double-precision floating point type	double val1 = 1.23;
		double val2 = 4.56D;
bool	Boolean type; a bool value is either true or false	bool val1 = true; bool val2 = false;
char	Character type; a char value is a Unicode character	char val = 'h';
decimal	Precise decimal type with 28 significant digits	decimal val = 1.23M;

Each of the predefined types is shorthand for a system-provided type. For example, the keyword int refers to the struct System.Int32. As a matter of style, use of the keyword is favored over use of the complete system type name.

Predefined value types such as int are treated specially in a few ways but are for the most part treated exactly like other structs. Operator overloading enables developers to define new struct types that behave

much like the predefined value types. For instance, a Digit struct can support the same mathematical operations as the predefined integral types, and can define conversions between Digit and predefined types. The predefined types employ operator overloading themselves. For example, the comparison operators == and != have different semantics for different predefined types:

- Two expressions of type int are considered equal if they represent the same integer value.
- Two expressions of type object are considered equal if both refer to the same object, or if both are null.
- Two expressions of type string are considered equal if the string instances have identical lengths and identical characters in each character position, or if both are null.

Chapter 9 – Lexical Structure – Page 65 – 82 – Many references to Unicode

Chapter 11 – Types – Page 106 – Sections 11.1.4 and 11.2.3

11.1.4 Integral types 28

C# supports nine integral types: sbyte, byte, short, ushort, int, uint, long, ulong,andchar. The integral types have the following sizes and ranges of values:

- The sbyte type represents signed 8-bit integers with values between –128 and 127.
- The byte type represents unsigned 8-bit integers with values between 0 and 255.
- The short type represents signed 16-bit integers with values between –32768 and 32767.
- The ushort type represents unsigned 16-bit integers with values between 0 and 65535.
- The int type represents signed 32-bit integers with values between –2147483648 and 2147483647.
- The uint type represents unsigned 32-bit integers with values between 0 and 4294967295.
- The long type represents signed 64-bit integers with values between –9223372036854775808 and 9223372036854775807.
- The ulong type represents unsigned 64-bit integers with values between 0 and 18446744073709551615.
- The char type represents unsigned 16-bit integers with values between 0 and 65535. The set of possible values for the char type corresponds to the Unicode character set. *[Note:* Although char has the same representation as ushort, not all operations permitted on one type are permitted on the other. *end not*e]

11.2.3 The string type

The string type is a sealed class type that inherits directly from object. Instances of the string class represent Unicode character strings.

Annex A – Page 349 – to the end – Several references