Topic: 2.2.1 Data types

A data type is a method of interpreting a pattern of bits.

Intrinsic data types

Intrinsic data types are the data types that are defined within a particular programming language.

There are numerous different data types. They are used to make the storage and processing of data easier and more efficient. Different databases and programming systems have their own set of intrinsic data types, but the main ones are:

- Integer
- Real
- Boolean
- String
- Character
- Date
- Container

Integer

An integer is a positive or negative number that does not contain a fractional part. Integers are held in pure binary for processing and storage. Note that some programming languages differentiate between short and long integers (more bytes are used to store long integers).

Real

A real is a number that contains a decimal point. In many systems, real numbers are referred to as singles and doubles, depending upon the number of bytes in which they are stored.

Boolean

A Boolean is a data-type that can store one of only two values – usually these values are "True" or "False". Booleans are stored in one byte – True being stored as 11111111 and False as 00000000.

String

A string is a series of alphanumeric characters enclosed in quotation marks. A string is sometimes just referred to as 'text'. Any type of alphabetic or numeric data can be stored as a string: "Birmingham City", "3/10/03" and "36.85" are all examples of strings. Each character within a string will be stored in one byte using its ASCII code; modern systems might store each character in two bytes using its Unicode. The maximum length of a string is limited only by the available memory.

Topic: 2.2.1 Data types

Notes:

- if dates or numbers are stored as strings then they will not be sorted correctly; they will be sorted according to the ASCII codes of the characters – "23" will be placed before "9";
- Telephone numbers must be stored as strings or the initial zero will be lost.

Character

A character is any letter, number, punctuation mark or space, which takes up a single unit of storage (usually a byte).

Dates

In most computer systems dates are stored as a 'serial' number that equates to the number of seconds since January 1st, 1904 (thus they also contain the time). Although the serial numbers are used for processing purposes, the results are usually presented in one of several 'standard' date formats – for example, dd/mm/yyyy, or ddMonthName, yyyy. Dates usually take 8 bytes of storage.

Comparison of the common data types:

Data Type	Example value	Storage Required
Integer	42	4 bytes
Real	23.1	4 or 8 bytes
Boolean	True	1 byte
String	"Hello World"	1 byte for each character (if using Unicode, then 2 bytes are required for each character)
Character	"X"	1 byte
Date	12/11/2009	8 bytes

Topic: 2.2.1 Data types

Estimating the size of a file:

The basic formula for estimating the size of a file is:

> Size of file = [size of each record] × [number of records] + [a little bit more!]

If we consider a file with 200 records, which stores the details of an organization's customers:

CUSTOMER(RefCode, Name, PostCode, Telephone, DoB, Age)

We can estimate the size of the record as follows:

Attribute	Data type	Extreme Example	Size of field (bytes)
RefCode	Integer	99 999	4
Name	String	Margaret Edwards	20
PostCode	String	WC12 16AA	9
Telephone	String	(0203) 9898 1234	16
DoB	Date	31-12-76	8
Age	Real	104	4
		Total	**62**

Thus 200 records would require:

$$62 \times 200 = 12400 \text{ bytes}$$
$$= \frac{12400}{1024} \text{Kbytes}$$
$$= 12.1 + 1.21\,(10\%)$$
$$= 13.3 \text{ Kbytes}$$

Note that to determine the maximum field length, an extreme case was considered and several bytes added to play safe.

Topic: 2.2.1 Data types

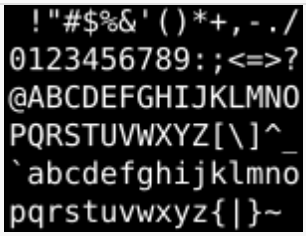
Character sets

ASCII Character sets:

The 104-key PC US English QWERTY keyboard layout evolved from the standard typewriter keyboard, with extra keys for computing.

ASCII normally uses 8 bits (1 byte) to store each character. However, the 8th bit is used as a check digit, meaning that only 7 bits are available to store each character. This gives ASCII the ability to store a total of

2^7 = 128 different values.



The 95 printable ASCII characters, numbered from 32 to 126 (decimal)

ASCII values can take many forms:

- Numbers
- Letters (capitals and lower case are separate)
- Punctuation (?/|\£$ etc.)
- non-printing commands (enter, escape, F1)

Topic: 2.2.1 Data types

Take a look at your keyboard and see how many different keys you have. The number should be 104 for a windows keyboard, or 101 for traditional keyboard. With the shift function values (a, A; b, B etc.) and recognising that some keys have repeated functionality (two shift keys, the num pad). We roughly have 128 functions that a keyboard can perform.

Binary	Dec	Hex	Abbr	Binary	Dec	Hex	Glyph	Binary	Dec	Hex	Glyph	Binary	Dec	Hex	Glyph
000 0000	0	00	NUL	010 0000	32	20	?	100 0000	64	40	@	110 0000	96	60	`
000 0001	1	01	SOH	010 0001	33	21	!	100 0001	65	41	A	110 0001	97	61	a
000 0010	2	02	STX	010 0010	34	22	"	100 0010	66	42	B	110 0010	98	62	b
000 0011	3	03	ETX	010 0011	35	23	#	100 0011	67	43	C	110 0011	99	63	c
000 0100	4	04	EOT	010 0100	36	24	$	100 0100	68	44	D	110 0100	100	64	d
000 0101	5	05	ENQ	010 0101	37	25	%	100 0101	69	45	E	110 0101	101	65	e
000 0110	6	06	ACK	010 0110	38	26	&	100 0110	70	46	F	110 0110	102	66	f
000 0111	7	07	BEL	010 0111	39	27	'	100 0111	71	47	G	110 0111	103	67	g
000 1000	8	08	BS	010 1000	40	28	(	100 1000	72	48	H	110 1000	104	68	h
000 1001	9	09	HT	010 1001	41	29	)	100 1001	73	49	I	110 1001	105	69	i
000 1010	10	0A	LF	010 1010	42	2A	*	100 1010	74	4A	J	110 1010	106	6A	j
000 1011	11	0B	VT	010 1011	43	2B	+	100 1011	75	4B	K	110 1011	107	6B	k

Topic: 2.2.1 Data types

000 1100	12	0C	FF	010 1100	44	2C	,	100 1100	76	4C	L	110 1100	108	6C	l
000 1101	13	0D	CR	010 1101	45	2D	-	100 1101	77	4D	M	110 1101	109	6D	m
000 1110	14	0E	SO	010 1110	46	2E	.	100 1110	78	4E	N	110 1110	110	6E	n
000 1111	15	0F	SI	010 1111	47	2F	/	100 1111	79	4F	O	110 1111	111	6F	o
001 0000	16	10	DLE	011 0000	48	30	0	101 0000	80	50	P	111 0000	112	70	p
001 0001	17	11	DC1	011 0001	49	31	1	101 0001	81	51	Q	111 0001	113	71	q
001 0010	18	12	DC2	011 0010	50	32	2	101 0010	82	52	R	111 0010	114	72	r
001 0011	19	13	DC3	011 0011	51	33	3	101 0011	83	53	S	111 0011	115	73	s
001 0100	20	14	DC4	011 0100	52	34	4	101 0100	84	54	T	111 0100	116	74	t
001 0101	21	15	NAK	011 0101	53	35	5	101 0101	85	55	U	111 0101	117	75	u
001 0110	22	16	SYN	011 0110	54	36	6	101 0110	86	56	V	111 0110	118	76	v
001 0111	23	17	ETB	011 0111	55	37	7	101 0111	87	57	W	111 0111	119	77	w
001 1000	24	18	CAN	011 1000	56	38	8	101 1000	88	58	X	111 1000	120	78	x
001 1001	25	19	EM	011 1001	57	39	9	101 1001	89	59	Y	111 1001	121	79	y
001 1010	26	1A	SUB	011 1010	58	3A	:	101 1010	90	5A	Z	111 1010	122	7A	z

Topic: 2.2.1 Data types

001 1011	27	1B	ESC	011 1011	59	3B	;	101 1011	91	5B	[	111 1011	123	7B	{
001 1100	28	1C	FS	011 1100	60	3C	<	101 1100	92	5C	\	111 1100	124	7C	|
001 1101	29	1D	GS	011 1101	61	3D	=	101 1101	93	5D	]	111 1101	125	7D	}
001 1110	30	1E	RS	011 1110	62	3E	>	101 1110	94	5E	^	111 1110	126	7E	~
001 1111	31	1F	US	011 1111	63	3F	?	101 1111	95	5F	_	111 1111	127	7F	DEL

Topic: 2.2.1 Data types

If you look carefully at the ASCII representation of each character you might notice some patterns. For example:

Binary	Dec	Hex	Glyph
110 0001	97	61	a
110 0010	98	62	b
110 0011	99	63	c

As you can see, a = 97, b = 98, c = 99. This means that if we are told what value a character is, we can easily work out the value of subsequent or prior characters.

Topic: 2.2.1 Data types

Unicode:

The problem with ASCII is that it only allows you to represent a small number of characters (~128 or 256 for Extended ASCII). This might be OK if you are living in an English speaking country, but what happens if you live in a country that uses a different character set? For example:

- Chinese characters 汉字
- Japanese characters 漢字
- Cyrillic Кири́ллица
- Gujarati ગુજરાતી
- Urdu اردو

You can see that we quickly run into trouble as ASCII can't possibly store these hundreds of thousands of extra characters in just 7 bits. What we do instead is use **unicode**. There are several versions of unicode, each with using a different number of bits to store data:

Name	Descriptions
UTF-8	8-bit is the most common unicode format. Characters can take as little as 8-bits, maximizing compatibility with ASCII. But it also allows for variable-width encoding expanding to 16, 24, 32, 40 or 48 bits when dealing with larger sets of characters
UTF-16	16-bit, variable-width encoding, can expand to 32 bits.
UTF-32	32-bit, fixed-width encoding. Each character takes exactly 32-bits

Topic: 2.2.1 Data types

With over a million possible characters we should be able to store every character from every language on the planet, take a look at these examples:

code point	glyph*	character	UTF-16 code units (hex)
U+007A	z	LATIN SMALL LETTER Z	007A
U+6C34	水	CJK UNIFIED IDEOGRAPH-6C34 (water)	6C34
U+10000	𐀀	LINEAR B SYLLABLE B008 A	D800, DC00
U+1D11E	𝄞	MUSICAL SYMBOL G CLEF	D834, DD1E

A **data structure** is a collection of different data items that are stored together in a clearly defined way. Two common data structures are arrays and records.