(v) A **data block** is the highest-level component of a CIF, containing data items or (in the case of dictionary files only) save frames. A data block is identified by a **data-block header**, which is an isolated character string (that is, bounded by white space and not forming part of a data value) beginning with the case-insensitive reserved characters `data_`. A **block code** is the variable part of a data-block header, *e.g.* the string *foo* in the header `data_foo`.

(vi) A **looped list** of data is a set of data items represented as a table or matrix of values. The data names are assembled immediately following the word `loop_`, each separated by white space, and the associated data values are then listed in strict rotation. The table of values is assembled in row-major order; that is, the first occurrence of each of the data items is assembled in sequence, then the second occurrence of each item, and so forth. In a CIF, looped lists may not be nested.

2.2.3. The syntax of a CIF

The essential syntax rules for a CIF data file are discussed alongside an example (Fig. 2.2.3.1), which is an extract from a file used to exemplify the reporting of a small-molecule crystal structure to *Acta Crystallographica Section C*. The following discussion is tutorial in nature and is intended to give an overview of the syntactic features of CIF to the general reader. The special use of save frames in dictionary files is not discussed in this summary. Software developers will find the full specification at the end of the chapter. If there are any real or apparent discrepancies between the two treatments, the full specification is to be taken as definitive.

A CIF contains only ASCII characters, organized as lines of text.

Tokens (the discrete components of the file) are separated by white space; layout is not significant. Thus, in the list of atom-site coordinates in Fig. 2.2.3.1, the hydrogen-atom entries are cosmetically aligned in columns, but the non-aligned entries for the other atoms are equally valid. Indeed, there is no requirement that each cluster of looped data values be confined to a separate row; contrast the cosmetic ordering of the atom-sites loop with the loop of symmetry-equivalent positions, where entries run on the same or following lines indiscriminately.

A comment is a token introduced by a hash character `#` and extending to the end of the line. Comments are considered to have no portable information content and may freely be discarded by a parser. However, revision 1.1 of the CIF specification introduces a *recommendation* that a CIF begin with a comment taking the form

```
#\#CIF_1.1
```

where the 1.1 is a version identifier of the reference CIF specification. This is primarily for the benefit of general file-handling software on current operating systems (*e.g.* graphical file managers that associate software applications with files of specific type), and its presence or absence does not guarantee the integrity of the file with respect to any particular revision of the CIF specification.

The first non-comment token of a CIF must be a data-block header, which is a character string that does not include white space and begins with the case-insensitive characters `data_`.

The file may be partitioned into multiple data blocks by the insertion of further data-block headers. Data-block headers are case-insensitive (that is, two headers differing only in whether corresponding letter characters are upper or lower case are considered identical). Within a single data file identical data-block headers are not permitted.

```
data_99107abs

# Chemical data
_chemical_name_systematic
; 3-Benzo[b]thien-2-yl-5,6-dihydro-1,4,2-oxathiazine
  4-oxide
;
_chemical_formula_moiety         "C11 H9 N O2 S2"
_chemical_formula_weight          251.31

# Crystal data
_symmetry_cell_setting            orthorhombic
_symmetry_space_group_name_H-M    'P 21 21 21'

loop_
   _symmetry_equiv_pos_as_xyz
 'x, y, z' 'x+1/2, -y+1/2, -z' '-x, y+1/2, -z+1/2'
 '-x+1/2, -y, z+1/2'

_cell_length_a                    7.4730(11)
_cell_length_b                    8.2860(11)
_cell_length_c                    17.527(2)
_cell_angle_alpha                 90.00
_cell_angle_beta                  90.00
_cell_angle_gamma                 90.00

# Atomic coordinates and displacement parameters
loop_
  _atom_site_label
  _atom_site_type_symbol
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
S4 S 0.32163(7) 0.45232(6) 0.52011(3) 0.04532(13)
S11 S 0.39642(7) 0.67998(6) 0.29598(2) 0.04215(12)
O1 O -0.00302(17) 0.67538(16) 0.47124(8) 0.0470(3)
O4 O 0.2601(2) 0.28588(16) 0.50279(10) 0.0700(5)
N2 N 0.14371(19) 0.66863(19) 0.42309(9) 0.0402(3)
C3 C 0.2776(2) 0.57587(19) 0.43683(9) 0.0332(3)
C5 C 0.1497(3) 0.5457(3) 0.57608(11) 0.0498(5)
C6 C -0.0171(3) 0.5529(2) 0.52899(12) 0.0460(4)
C12 C 0.4215(2) 0.57488(19) 0.38139(9) 0.0344(3)
C13 C 0.5830(2) 0.4995(2) 0.38737(10) 0.0386(4)
C13A C 0.6925(2) 0.5229(2) 0.32123(10) 0.0399(4)
C14 C 0.8631(3) 0.4608(3) 0.30561(13) 0.0532(5)
C15 C 0.9423(3) 0.4948(3) 0.23709(15) 0.0644(7)
C16 C 0.8563(3) 0.5917(3) 0.18349(14) 0.0667(7)
C17 C 0.6901(3) 0.6568(3) 0.19729(12) 0.0546(5)
C17A C 0.6090(3) 0.6204(2) 0.26670(10) 0.0396(4)
H5A   H    0.1284    0.4834    0.6221    0.060
H5B   H    0.1861    0.6537    0.5908    0.060
H6A   H   -0.0374    0.4490    0.5050    0.055
H6B   H   -0.1186    0.5762    0.5617    0.055
H13   H    0.6182    0.4397    0.4297    0.046
H14   H    0.9218    0.3972    0.3414    0.064
H15   H    1.0548    0.4527    0.2262    0.077
H16   H    0.9127    0.6130    0.1373    0.080
H17   H    0.6340    0.7227    0.1616    0.066
```

Fig. 2.2.3.1. Typical small-molecule CIF.

Data names are character strings that begin with an underscore character _ and do not contain white-space characters. Data names serve to index data values and are case-insensitive.

Where a data name indexes a single data value, that value follows the data name separated by white space.

Where a data name indexes a set of data values (conceptually a vector or table column), the relevant data items are preceded by the case-insensitive string `loop_` separated by white space.

The examples of Fig. 2.2.3.1 show the use of `loop_` to specify a vector or one-dimensional list of values (the symmetry-equivalent positions) and a tabular or matrix list (the atom-site positions).

Again it should be emphasized that the rows and columns of the table are identified by parsing each value and referring back to the sequence in the header of its identifying tag, and not by a conventional layout of items. It is usual for CIF writers to lay out two-dimensional data arrays as well formatted tables for ease of inspection in text editors or other viewers, but CIF readers must never rely on layout alone to identify a tabulated value.

A corollary of this is that the number of values in a looped list must be an exact integer multiple of the number of data names declared in the loop header.

Within a single data block the same data name may not be repeated. Thus if a data item may have multiple values, these items must be collected together within a looped list, the data name itself being given once only in the loop header.

A data value is a string of characters. CIF distinguishes between numerical and character data in a broad sense (see Section 2.2.5.2 below). Numerical values may not contain white space (and indeed are constrained to a limited character set and ordering, essentially encompassing a small range of ways in which numbers are characteristically represented in printed form). Because tokens are separated by white space, character data that include white-space characters must be quoted. If the data value does not extend beyond the end of a line of text, it may be quoted by matching single-quote (apostrophe, `'`) or double-quote (quotation mark, `"`) characters. If the data value does extend beyond the end of a line of text, then paired semicolon characters `;` *as the first character of a line* may be used as delimiters. The closing semicolon is the first character of a line immediately following the close of the data-value string. Fig. 2.2.3.1 shows examples of all three types of delimited character values that include white space.

Character strings that begin with certain other characters must also be quoted. These leading characters are those which introduce tokens with special roles in a STAR File (such as underscore _ at the start of a data name, hash `#` at the start of a comment and dollar `$` identifying a save-frame reference pointer). Likewise, the STAR File reserved words **`loop_`**, **`stop_`** and **`global_`** must be quoted if they represent data values, as must any character string beginning with **`data_`** or **`save_`**.

Lines of text are restricted to 2048 characters in length and data names are restricted to 75 characters in length. These are increases over the original values of 80 and 32 characters, respectively.

2.2.4. Portability and archival issues

The CIF format is designed to be independent of operating system (OS) and programming language. Nevertheless, variations in the way that each OS specifies and handles character sets mean that care must be taken to ensure that CIF software is portable across different computer platforms. There are also constraints on the application of these specifications in order to maintain compatibility between archival systems. These issues are discussed briefly here. More details are given in the formal CIF specification (see Section 2.2.7). In general, compatibility and portability considerations for different OSs are of little importance to users of CIFs, but they need to be well understood by software developers.

2.2.4.1. Character set

The characters permitted in a CIF are in effect the printable characters in the ASCII character set. However, a CIF may also be constructed and manipulated using alternative single-character byte mappings such as EBCDIC, and multi-byte or wide character encodings such as Unicode, provided there is a direct mapping to the permitted ASCII characters. Accented characters, characters in non-Latin alphabets and mathematical or special typographic symbols may not appear as single characters in a CIF, even if a host OS permits such representations.

White space (used to separate CIF tokens and within comments or quoted character-string values) is most portably represented by the printable space character (decimal value 32 in the ASCII character set). In an ASCII environment, white space may also be indicated by the control characters denoted HT (horizontal tab, ASCII decimal 9), LF (line feed, ASCII decimal 10) and CR (carriage return, ASCII decimal 13). To ease problems of translation between character encodings, the characters VT (vertical tab, ASCII decimal 11) and FF (form feed, ASCII decimal 12) are explicitly excluded from the CIF character set; this is a restriction that is not in the general STAR File specification (Chapter 2.1).

2.2.4.2. Line terminators

Given that the STAR File is built on the premise of a line-oriented text file, it is difficult in practice to provide a complete and portable description of how to identify the start or end of a line of text. The difficulty arises for two reasons.

First, some OSs or programming languages are record-oriented; that is, the OS is able to keep track of a region of memory associated with a specific record. It is usually appropriate to associate each such record with a line of text, but the ‘boundaries’ between records are managed by low-level OS utilities and are not amenable to a character-oriented discussion. Such systems, where records of fixed length are maintained, may also give rise to ambiguities in the interpretation of padding to the record boundary following a last printable character – is such padding to be discarded or treated as white space? It is for this reason that the elision of trailing white space in a line is permitted (but not encouraged) in the full CIF syntax specification [Section 2.2.7.1.4(17)].

The second complication arises because current popular OSs support several different character-based line terminators. Historically, applications developed under a specific OS have made general use of system libraries to handle text files, so that the conventions built into the system libraries have in effect become standard representations of line terminators for all applications built on that OS. For a long time this created no great problem, since files were transferred between OSs through applications software that could be tuned to perform the necessary line-terminator translations in transit. The best-known such application is undoubtedly the ‘text’ or ‘ascii’ transmission mode of the typical ftp (file transfer protocol) client.

Increasingly, however, a common network-mounted file system may be shared between applications running under different OSs and the same file may present itself as ‘valid’ to one user but not to another because of differences in what the applications consider a line terminator.

The problem of handling OS-dependent line termination is by no means unique to STAR File or CIF applications; any application that manipulates line-oriented text files must accommodate this difficulty. The specification notes the practice of designing applications that treat equivalently as a line terminator the characters LF (line feed or newline), CR (carriage return) or the combination CR followed by LF, since these are the dominant conventions under the prevailing Unix, MacOS and DOS/Windows OSs of the present day. While this will be a sensible design decision for many CIF-reading applications, software authors must be aware that the CIF specification aims for portability and archivability through a more general understanding of what constitutes a line of text.