because `_A1` occurs in the first data block encountered; the output from *cif2cif* in this example will be

```
data_A
    loop_     _A1
               a1   aa1
#             ---end-of-data-block---
```

The behaviour of the program differs from *QUASAR* in two other small ways. When the request list forces the output data stream to contain the same data-block header more than once, an error message is posted to the standard error channel and the data-block headers in the output stream are annotated with a comment of the form '`#<---- duplicate data block`'. In this case the output file does *not* conform to the CIF syntax rules.

When a data name is requested but no matching data item appears in the output file, *cif2cif* writes an error message to the standard error channel. However, unlike *QUASAR*, which inserts the requested data name in the output stream with an associated value of '`?`' (for *unknown*), *cif2cif* produces *no* output for the requested data item.

5.3.5.2.1.5. *Other features*

Some additional features are of use in special circumstances.

The user may preserve the layout of the contents of looped lists exactly as in the input file, or may ask the program to adjust the layout to a more visually pleasing tabular form.

The user may enable recognition of data-name aliases in the dictionaries used for validation. When the relevant command-line argument is set to *true*, user-supplied data names will be transformed to the canonical forms in the validating dictionary. This would permit, for example, a small-molecule CIF using the core dictionary definitions to be converted to mmCIF format.

The user may prefix each line of output with an identical character string. A typical reason for so doing would be to include a fragment of CIF listing within the body of an email message or some other document. Such an output would not conform to the syntax rules for CIF.

5.3.5.2.2. *Invocation of the program*

cif2cif is another application of the *CIFtbx* library by the same author, and so has a similar user interface to that of *CYCLOPS* (Section 5.3.4.1.2). Under a Unix-like operating system, the program is typically called with a command such as

`cif2cif -i` *infile* `-o` *outfile* [`-q` *reqfile*]

where *infile* is the name of the input file, *outfile* is the output file and *reqfile* is an optional file containing a request list for a subset of the original contents.

A more complete set of options available in a Unix-like operating environment is

`cif2cif` [`-i` *infile*] [`-o` *outfile*] [`-d` *dictfile*] [`-q` *reqfile*] [`-f` *cmndfile*] [`-c` *catck*] [`-a` *alias*] [`-t` *tab*] [`-e` *sulim*] [`-p` *prefix*]

where the options are as follows:

-i specifies the name of the input file, *infile*.

-o specifies the name of the output file, *outfile*.

-d specifies the name of a dictionary file, *dictfile*, against which the existence, type and category of data names are checked. The dictionary file may be *either* a CIF dictionary or a list of file names. That is, it may contain dictionary definitions in DDL format or (if the file begins with the characters `#DICT`) it may contain a list of dictionary file names to be entered. Thus, multiple dictionaries may be specified to the program.

-q specifies the name of the request file, *reqfile*, containing a list of data names (with associated data-block directives) that should be extracted as a subset of the contents of the original file.

-f specifies the name of a command file *cmndfile* that contains additional directives to the program.

-c is a flag indicating whether an error message should be raised if a data name has been assigned a category different from the leading portion of the data name itself. The Boolean variable *catck* may take the values 't', '1' or 'y' for *true*, 'f', '0' or 'n' for *false*.

-a is a flag indicating whether data-name aliases in the validating dictionary should be used to replace user-supplied names by their canonical forms. The Boolean variable *alias* may take the same values for *true* or *false* as above.

-t is a flag indicating whether the output should be reformatted with tabs to produce a regular table layout within looped lists. The Boolean variable *tab* takes the same values as above. If *true*, text is reformatted; if *false*, the original formatting is retained.

For the flags expecting Boolean values, the default is 'f' (*false*).

-e specifies the precision to retain in rounding standard uncertainty values. The permitted integer values are 9, 19 (the default) and 29.

-p takes a string value which is prefixed to every line of output. Every occurrence of the underscore character '_' in the prefix is changed to a space on output.

If no input or output file names are specified, the program will read from the standard input channel or write to standard output, respectively. The special character hyphen ('-') may also be supplied as an argument in place of a file name to indicate standard input or standard output as appropriate.

Finally, if the operating system supports the passing of environment variables to a program, the name of the input file may be passed as the value of `$cif2cif_INPUT_CIF`, and likewise the output file, `$cif2cif_OUTPUT_CIF`, dictionary file, `$cif2cif_CHECK_DICTIONARY`, and request file, `$cif2cif_REQUEST_LIST`, may be specified.

5.3.5.3. *ciftex*: translating to a typesetting language

The program *ciftex* (McMahon, 1993) was developed to create files for typesetting the journal *Acta Crystallographica* using the text-formatting language TEX (Knuth, 1986). Details of its use in the journal production process are given in Chapter 5.7. It is discussed here as an example of translating a CIF to some output format where data values are annotated with different text depending on their accompanying data names.

5.3.5.3.1. *Basic operation of ciftex*

The program is designed to act as a filter, typically in a Unix-style environment, reading a CIF on the standard input channel and outputting a modified data stream to standard output. The output is a file of TEX code that is processed by the TEX program to produce a device-independent file describing the content of a formatted typeset document. Further post-processing allows the formatted document to be viewed on the screen or printed.

Each input token (number, character or text string; data name; `loop_` or `data_` keywords) is transformed as it is identified; there is no lookahead and minimal retention of context. The data stream is treated purely syntactically; no transformations are applied on the basis of the supposed meaning of any of the file contents.