Fig. 5.4.9.1. Sample program CIF.IN. See text for explanation.

'#$ not in dictionary'. This applies to both looped and single
data items.

A more complex example of writing a CIF is given in the pro-
gram cif2cif available with the CIFtbx release. A similar program
that reads a CIF and writes an XML file is cif2xml, also available
with the CIFtbx release.

5.4.11. Error-message glossary

The CIFtbx routines will generate explicit error messages in the
printout or in the created CIF if requested to do so (e.g. during
dictionary checks). If data processing cannot continue (i.e. fatal
errors), an appropriate error message is placed in the printout and
execution terminates. However, the default approach is to remain
mute and for error detection to be monitored by the application
program via the CIFtbx functions returning .true. or .false.
values that tell the application program whether the command
was performed correctly. This places the primary responsibility for
error checking on the application software. The importance of this
approach is that it enables the local application to respond to run-
time problems in a controlled way and to take corrective action
if it is possible. However, some types of processing errors, such
as exceeding the dimensions of critical CIFtbx arrays, do require
appropriate messages to be issued and for execution to cease.
5. APPLICATIONS

```plaintext
data_mumbo_jumbo

_audit_creation_date 91-03-20
_audit_creation_method from_xtal_archive_file_using_CIFIO
_audit_update_record
91-04-09 text and data added by Tony Willis.
91-04-15 rec’d by co-editor with diagram as manuscript HL7
_dummy_test "rubbish to see what dict_ says"

_chemical_name_systematic trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-1,3-oxazolidin-5-one
_chemical_formula_moiety 'C18 H25 N O3'
_chemical_formula_weight 303.40
_chemical_melting_point ?

### cell_length_a 5.959(1)
cell_length_b 14.956(1)
cell_length_c 19.737(3)
cell_measurement_theta_min 25
cell_measurement_theta_max 31
_symmetry_cell_setting orthorhombic

loop
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
s .20200 .79800 .91667 .030(3) Uij ?
o .49800 .49800 .66667 .02520 Uiso ?
c1 .48800 .09600 .03800 .03170 Uiso ?

loop _blat1 _blat2 1 2 3 4 5 6 a b c d 7 8 9 0

Fig. 5.4.9.2. Example CIF read by the sample program CIF\_IN shown in Fig. 5.4.9.1.

```
5.4. CIFTBX: FORTRAN TOOLS FOR MANIPULATING CIFS

C....... Open a new CIF
400 if(pfile_('test.new')) go to 450
write(6,'(/a/)') ' Output CIF by this name exists already!'
go to 500

C....... Request dictionary validation check
450 if(dict_('cif.core.dic','valid')) go to 460
write(6,'(/a/)') ' Requested Core dictionary not present'

C....... Insert a data block code
460 f1 = pdta_({'whoops_a_daisy'})
C....... Enter various single data items to show how
f1 = pchar_({'audit_creation_method','using CIFTbx'})
f1 = pchar_({'audit_creation_extra2','Terry O’Connell''})
f1 = pchar_({'audit_creation_extra3','Terry O’Convertl''})
f1 = ptext_({'audit_creation_record',' Text data may be '})
f1 = ptext_({'audit_creation_record',' or in a loop.'})
f1 = pnumb_({'cell_measurement_temperature', 293., 0.})
f1 = pnumb_({'cell_volume', 1759.0, 13.})
f1 = pnumb_({'cell_length_b', 8.75354, 0.003})
f1 = pnumb_({'cell_length_c', 19.737, 0.003})

C....... Enter some looped data
f1 = ploop_({'atom_type_symbol'})
f1 = ploop_({'atom_type_oxidation_number'})
f1 = ploop_({'atom_type_number_in_cell'})
do 470 i=1,3
f1 = pchar_({'alpha(1:i)'})
f1 = pnumb_({'float(i),float(i)*0.1'})
470 f1 = pnumb_({'float(i)*8.64523,0.0'})

C....... Do it again but as contiguous data with text data
f1 = ploop_({'atom_site_label'})
f1 = ploop_({'atom_site_occupancy'})
f1 = ploop_({'some_silly_text'})
do 480 i=1,2
f1 = pchar_({'alpha(1:i)'})
f1 = pnumb_({'float(i),float(i)*0.1'})
480 f1 = ptext_({'', 'Hi Ho the diddly oh!''})
call close_

Fig. 5.4.10.1. Sample program to create a CIF.

533
However, the message

More than MAXBOOK bookmarks requested

is not ‘fatal’, in the sense that the function bkmrk_returns .false. to permit appropriate action before termination. This is effectively a fatal error for which recompilation with a larger value of MAXBOOK is necessary. However, this is usually the result of a logic error in the application, and the error has been made non-fatal to allow the programmer to insert debugging code, if desired. The application should clean up and exit promptly.

5.4.11.2. Fatal errors: data sequence, syntax and file construction

Dict_ must precede ocif_

Dictionary files must be loaded before an input CIF is opened because some checking occurs during the CIF loading process.

Illegal tag/value construction

Data name (i.e. a ‘tag’) and data values are not matched (outside a looped list). This usually means that a data name immediately follows another data name, or a data value was found without a preceding data name. The most likely cause of this error is the failure to provide ‘;’ or ‘|’ for missing or unknown data values or a failure to declare a 1oop_ when one was intended.

Item miscount in loop

Within a looped list the total number of data values must be an exact multiple of the number of data names in the 1oop_ header.

Prior save-frame not terminated

Save-frame terminator found out of context. Save frames must start with save_framecode and end with save_. These messages will be issued if this does not occur.

Syntax construction error

Within a data block or save frame the number of data values does not match the number of data names (ignoring loop structures). This message should occur only if there is an internal logic error in the library. Normally the program will terminate on Item miscount in loop first.

Unexpected end of data

When processing multi-line text the end of the CIF is encountered before the terminal semicolon.

5.4.11.3. Fatal errors: invalid arguments

The following messages are generated by calls with invalid arguments.

Call to find_ with invalid arguments
Internal error in putnum

5.4.11.4. Warnings: input errors

Category <cat-code> first implicitly defined in cif

The category code in the DDL2 data name is not matched by an explicit definition in the dictionary. This may be intentional but usually indicates a typographical error in the CIF or the dictionary.

Data name <name> not in dictionary!
The data item name <name> was used in the CIF but could not be found in the dictionary.

Data block header missing
No data_ or global_ was found when expected.

Duplicate data item <name>
Two or more identical data names <name> have been detected in a data block or save frame.

Exponent overflow in numeric input
Exponent underflow in numeric input

The numeric value being processed has an exponent that cannot be processed on this machine. If the string involved is not intended as a number, then surrounding it with quotes may resolve the problem.

Heterogeneous categories in loop <new cat-code> vs <old cat-code>

Looped lists should not contain data items belonging to different categories. This error occurs if the category of a new data item fails to match the category of a prior data item. A special category (none) is used to denote item names for which no category has been declared. Warnings are not issued on this level for a loop for which all data items have no declared category.

Input line length exceeds line_

Non-blank characters were found beyond the value given by the variable line_. The default value for line_ is 80 (optionally increased to 2048 in CIFtbx 2.7 and later for CIF 1.1 compliance). The extra characters in column positions line_+1 through MAXBUF will be processed but the input file may need to be reformatted for use with other CIF-handling programs.

Missing loop_ items set as DUMMY

A looped list of output items was truncated with an incomplete loop packet (i.e. the number of items did not match the number of 1oop_ data names). The missing values were set to the character string ‘DUMMY’.

Numb type violated <name>
The data item <name> has been processed with an explicit dictionary type numb, but with a non-numeric value. Note that the values ‘?’ or ‘.’ will not generate this message.

Quoted string not closed
Character values may be enclosed by bounding quotes. The strict definition of a ‘quoted-string’ value is that it must start with a <wq> digraph and end with a <qw> digraph, where w is a white-space character blank or tab and q is a single or double quote, and the same type of quote mark is used in the terminal digraph as was used in the initial digraph. This message is issued if these conditions are not met.

5.4.11.5. Warnings: output errors

Converted pchar_ output to text for <string>

An attempt was made to write a string with pchar_ instead of ptext_, but the string contains a combination of characters for which ptext_ must be used.

ESD less than precision of machine
Overflow of esd
Underflow of esd

A call to pnumb_ or numb_ was made with values of the number and standard uncertainty (e.s.d) which cannot be presented properly on this machine. A bounding value of 0 or 99999 is used for the e.s.d.

Invalid value of esdlim_ reset to 19
In processing numeric output, a value of esdlim_ less than 9 or greater than 99999 was found. esdlim_ is then set to 19.
5.4. CIFTBX: FORTRAN TOOLS FOR MANIPULATING CIFS

Missing loop_name set as _DUMMY
Missing loop_items set as _DUMMY
In processing a loop, a dummy string has been inserted for a missing header or value.
Output CIF line longer than line
In outputting a line, the data exceed the limit specified in line. This occurs only if a single data name or a value exceeds this limit.
Out-of-sequence call to end text block
The termination of a text block has been invoked before a text block has been started. This can only occur with irregular use of the CIFTbx routines rather than the standard interface routines.
Output prefix may force line overflow
A prefix string placed in prefix exceeds line less the allowed length of tags.
Prefix string truncated
A prefix string specified to prefix is longer than the maximum line length allowed. The prefix string is truncated and processing continues.

5.4.11.6. Warnings: dictionary checks
Aliases and names in different loops; only using first alias
If a DDL2 dictionary contains a loop of alias declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first alias name is used.
Attempt to redefine category for item
Attempt to redefine type for item
If a DDL2 dictionary contains a category or type for a data item that conflicts with an earlier declaration, the first is used.
Categories and names in different loops
Types and names in different loops
If a DDL2 dictionary contains a loop of category or type declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first category name or type is used.
Category id does not match block name
In a DDL2 dictionary, the save-frame code is expected to start with the category name. If a category name within the frame is not within a loop, it is checked against that in the frame code and a warning is issued if these do not match.
Conflicting definition of alias
A DDL2 dictionary contains a new declaration of a data-name alias which is in conflict with a previous alias definition. The first alias declaration is used.
Duplicate definition of same alias
A DDL2 dictionary contains a new declaration of an alias for a data name which duplicates a previously defined alias pair.
Item name <name> does not match category name
If category checking is enabled and the category assigned to an item name does not match the initial characters of the item name, this message is issued. This may indicate a typographical error or a deprecated item in the dictionary.
Item type <type-code> not recognised
The DDL2 dictionary type codes are translated to the DDL type codes ‘numb’, ‘char’ and ‘text’. If an unrecognized type code is found no translation occurs.

Multiple DDL category definitions
Multiple DDL name definitions
Multiple DDL type definitions
Multiple DDL related item definitions
Multiple DDL related item functions
DDL1 and DDL2 declarations for categories, data names, data types and related items are used in the same data block or save frame.
Multiple categories for one name
Multiple types for one name
A dictionary contains a loop of category or type definitions and an unlooped declaration of a single data name. The first category or type definition is used.
No category defined in block <name> and name <name> does not match
A DDL2 dictionary contains no category for the defined data item and it was not possible to derive an implicit category from the block name. This usually indicates a typographical error in the dictionary.
No category specified for name <name>
A dictionary contains categories and category checking is enabled but no category is defined for the named data item.
No name defined in block
No name in the block matches the block name
These messages are issued if a dictionary save frame or data block contains no name definition or if all the names defined fail to match the block name.
No type specified for name <name>
A type code is missing from a dictionary and type checking was requested in the dict_invocation.
One alias, looped names, linking to first
A DDL2 dictionary may contain a list of data names and a single alias outside this loop. In this case, the correct name to which to link the alias must be derived implicitly. If the save-frame code matches the first name in the loop no warning is issued, because the use of the block name was probably the intended result, but if no such match is found this warning is issued.

5.4.12. Internals and programming style
CIFTbx is programmed in a highly portable Fortran programming style. However, on some older systems, some adaptation may be necessary to allow compilation. Implementors should be aware of the extensive use of variables in common blocks to transmit information and control execution (programming by side-effects), the use of the INCLUDE statement, the use of the ENDDO statement, the names of routines used internally by the package, the use of names longer than six characters and the use of names including the underscore character.

Some aspects of the internal organization of the library to deal with characteristics of CIFS are worth noting. CIFTbx copies an input CIF to a direct-access (i.e., random-access) file, but writes an output CIF directly. All data names are converted to lower case to deal with the case-insensitive nature of CIF. A hierarchy of parsing routines is used to deal with processing white space.

The major issues of programming style and internals are summarized here. See the Primer on the CD-ROM for more information.