```
1   MODULE KINDERREIMMOD
2       IMPLICIT NONE
3       PRIVATE
4       PUBLIC :: CHILD_DATA, CHILD, START, BUILD_CYCLE, LAST_ONE,
        PUT_CYCLE, DEL_NEXT, LENGTH, PRINT_CHILD
5
6       TYPE CHILD_DATA
7           CHARACTER(LEN=10)     :: NAME
8           INTEGER               :: AGE
9       END TYPE
10
11      TYPE CHILD
12          TYPE(CHILD_DATA)      :: DATA
13          TYPE(CHILD), POINTER :: NEXT
14      END TYPE
15
16      TYPE START
17          TYPE(CHILD), POINTER :: TOP
18      END TYPE
19
20      CONTAINS
21
22      SUBROUTINE INIT_CYCLE (S)                ! sets S into NULL
        state
23          TYPE(START), INTENT(OUT) :: S
24          NULLIFY(S%TOP)
25      END SUBROUTINE
26
27      SUBROUTINE BUILD_CYCLE(FILENAME, LIST)   ! creates an cyclic
        list of children (incl. reading from file)
28          CHARACTER(LEN=*), INTENT(IN) :: FILENAME
29          TYPE(START), INTENT(INOUT)   :: LIST
30          TYPE(CHILD_DATA)             :: READCHILD
31          INTEGER                      :: ios, err
32
33          ! put list in empty state
34          CALL INIT_CYCLE(LIST)
35
36          OPEN(UNIT=33, FILE=TRIM(FILENAME), IOSTAT=ios,
            ACTION="READ")
37
38          IF (ios == 0) THEN
39
40              ! reading children
41              DO
42                  READ(33, *, IOSTAT=err) READCHILD
43
44                  IF (err == 0) THEN
45                      CALL INS_TAIL(LIST, READCHILD)
46                  ELSE
47                      EXIT
48                  END IF
49
50              END DO
51
52          ELSE
53              WRITE(*,*) "ERROR: Cannot open the input file."
54          END IF
55
56          CLOSE(UNIT=33)
57
```

```
58          ! connect tail with head of list
59          CALL CONNECT(LIST)
60
61      END SUBROUTINE BUILD_CYCLE
62
63      SUBROUTINE INS_TAIL(LIST, TO_INS)          ! insert element at
        the end (tail) of a list
64          TYPE(START), INTENT(INOUT) :: LIST
65          TYPE(CHILD_DATA)           :: TO_INS
66          TYPE(CHILD), POINTER       :: LIST_ELEM, TAIL
67
68          ALLOCATE(LIST_ELEM, TAIL)
69
70          IF (EMPTY(LIST)) THEN
71
72              LIST_ELEM%DATA = TO_INS          ! put child data into
                an element of list
73              LIST%TOP       => LIST_ELEM
74              NULLIFY(LIST_ELEM%NEXT)
75
76          ELSE
77              TAIL => LIST%TOP
78
79              DO WHILE(ASSOCIATED(TAIL%NEXT))
80                  TAIL => TAIL%NEXT
81              END DO
82
83              LIST_ELEM%DATA = TO_INS
84              TAIL%NEXT      => LIST_ELEM
85              NULLIFY(LIST_ELEM%NEXT)
86
87          END IF
88
89      END SUBROUTINE INS_TAIL
90
91      SUBROUTINE CONNECT(LIST)                   ! connect tail with
        head of list
92          TYPE(START), INTENT(INOUT) :: LIST
93          TYPE(CHILD), POINTER       :: TAIL
94
95          ALLOCATE(TAIL)
96          TAIL => LIST%TOP
97
98          DO WHILE (ASSOCIATED(TAIL%NEXT))
99              TAIL => TAIL%NEXT
100         END DO
101
102         TAIL%NEXT => LIST%TOP
103
104         DEALLOCATE(TAIL)
105     END SUBROUTINE CONNECT
106
107     FUNCTION LAST_ONE(LIST)                    ! checks if one child
        is remaining
108         TYPE(START), INTENT(IN)  :: LIST
109         LOGICAL                  :: LAST_ONE
110
111         LAST_ONE = ASSOCIATED(LIST%TOP%NEXT, LIST%TOP)
112     END FUNCTION LAST_ONE
113
114
```

```
115     FUNCTION EMPTY(LIST)                    ! checks if the list
        is empty (no children)
116         TYPE(START), INTENT(IN)  :: LIST
117         LOGICAL                  :: EMPTY
118
119         EMPTY = .NOT. ASSOCIATED(LIST%TOP)
120     END FUNCTION EMPTY
121
122     SUBROUTINE PUT_CYCLE(LIST,CURR_CHILD)   ! prints the list,
        starting with CURR_CHILD
123         TYPE(START), INTENT(IN)  :: LIST
124         TYPE(CHILD), POINTER     :: CURR_CHILD
125         TYPE(CHILD), POINTER     :: HELP
126
127         WRITE(*,*) "NAME: ", CURR_CHILD%DATA%NAME, " ALTER: ",
            CURR_CHILD%DATA%AGE
128         HELP => CURR_CHILD%NEXT
129
130         DO WHILE(.NOT. ASSOCIATED(CURR_CHILD, HELP))
131             WRITE(*,*) "NAME: ", HELP%DATA%NAME, " ALTER: ",
                HELP%DATA%AGE
132             HELP => HELP%NEXT
133         END DO
134
135     END SUBROUTINE PUT_CYCLE
136
137     SUBROUTINE DEL_NEXT(LIST, CURR_CHILD)   ! deletes element
        following the CURR_CHILD
138         TYPE(START), INTENT(INOUT) :: LIST
139         TYPE(CHILD), POINTER       :: CURR_CHILD, HELP
140
141         ALLOCATE(HELP)
142
143         HELP%NEXT => CURR_CHILD%NEXT%NEXT
144         IF (ASSOCIATED(CURR_CHILD%NEXT, LIST%TOP)) THEN
145             LIST%TOP => HELP%NEXT
146         END IF
147
148         CURR_CHILD%NEXT => HELP%NEXT
149
150         DEALLOCATE(HELP)
151
152     END SUBROUTINE DEL_NEXT
153
154     FUNCTION LENGTH(LIST)                   ! returns the length
        of the list
155         TYPE(START), INTENT(IN) :: LIST
156         TYPE(CHILD), POINTER    :: HELP
157         INTEGER                 :: LENGTH
158
159         LENGTH = 1
160         HELP => LIST%TOP
161
162         DO WHILE(.NOT. ASSOCIATED(HELP%NEXT, LIST%TOP))
163             LENGTH = LENGTH + 1
164             HELP => HELP%NEXT
165         END DO
166
167     END FUNCTION LENGTH
168
169
```

```
170     SUBROUTINE PRINT_CHILD(CURR_CHILD)      ! prints data of
        current child
171         TYPE(CHILD), POINTER :: CURR_CHILD
172
173         WRITE(*,*) "NAME: ", CURR_CHILD%DATA%NAME, ", ALTER: ",
            CURR_CHILD%DATA%AGE
174     END SUBROUTINE PRINT_CHILD
175
176 END MODULE KINDERREIMMOD
```