EÖTVÖS LORÁND
UNIVERSITY | BUDAPEST

Sequential Input Files

Object Oriented Programming | 2024 Spring
Practice 5

Presented by Tarlan Ahadli
Supervised by Prof. Teréz Anna Várkonyi

Sequential Input Files

- Denotion:
  - Infile(e) : e – data type
- Specification
  - Indexing
- Program : st,e,x : read
  - st – status of reading {norm/abnorm}
  - e – read element
  - x – infile
  - read
    - $x' = \{x_1, \ldots, x_n\}$ -> read(out $e = x_1$, out st, in x') -> $x = \{x_2, \ldots, x_n\}$

Concrete Problems

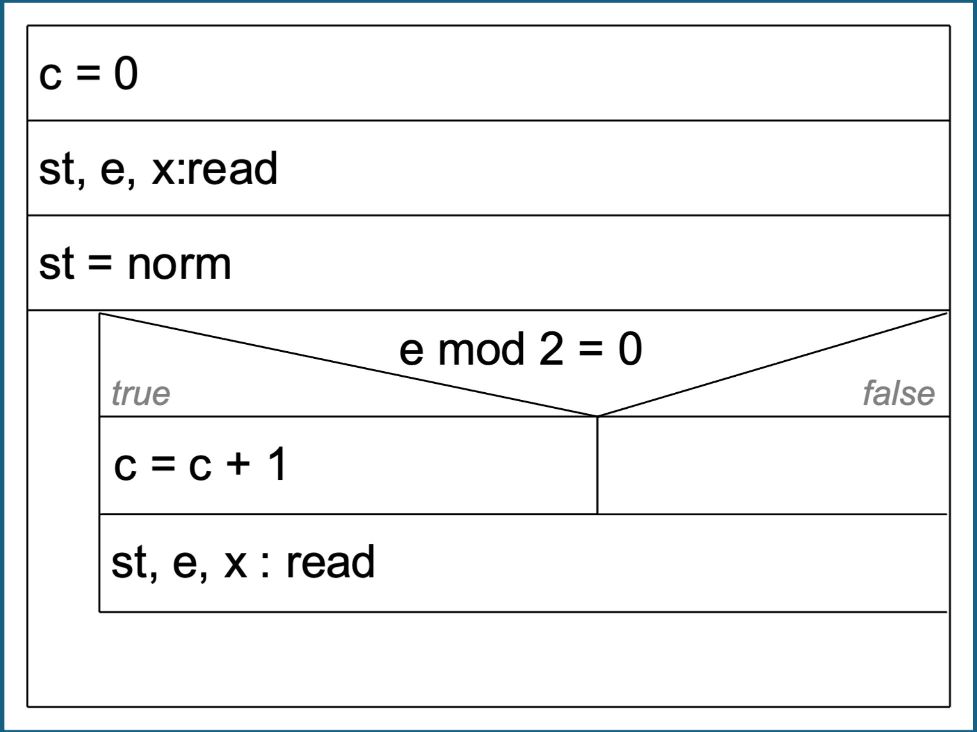
- $Count\ even\ numbers\ in\ the\ file\ given$
  - $A = (x{:}\, infile(Z), c{:}\, N)$
  - $Pre = (x = x') \mid x\ - out\ | x' - inp$
  - $Post = (c = \sum_{\substack{e\, \epsilon\, x' \\ 2\, |\, e}} 1)$

In the context of this programming task, reading a sequential input file and processing its elements changes the conceptual contents of the file from the program's perspective. While the physical file on disk remains unchanged, the elements that have been read are no longer part of the 'to be processed' set in the file as far as the program is concerned. This is why the initial state described by the precondition (an unchanged file) does not hold after processing, as described by the postcondition (a file with elements removed as they are read).

- **Postcondition (Post)**: After an element is read from the file, that element is 'cut' or removed from consideration in the file — conceptually, the file no longer contains that element, and subsequent reads will start from the next element. As the file is processed, it 'empties' until there are no elements left to read.
- **Precondition vs. Postcondition**: Since the file changes as it's read (elements are conceptually removed), the precondition (which assumes an unmodified file) is not true after the file has been processed. Therefore, the precondition doesn't hold in the postcondition.

Concrete Problems

- Analogy : Counting
  - $t : enor(E) \sim x : infile(Z)$
  - $cond(e) \sim 2|e$
- Operations for enums
  - First – st,e,x : read
  - Next - st,e,x : read
  - Current -> e
  - End -> st = abnormal



```
c = 0
st, e, x:read
st = norm
    e mod 2 = 0
    true:
        c = c + 1
    false:
    st, e, x : read
```

Concrete Problems

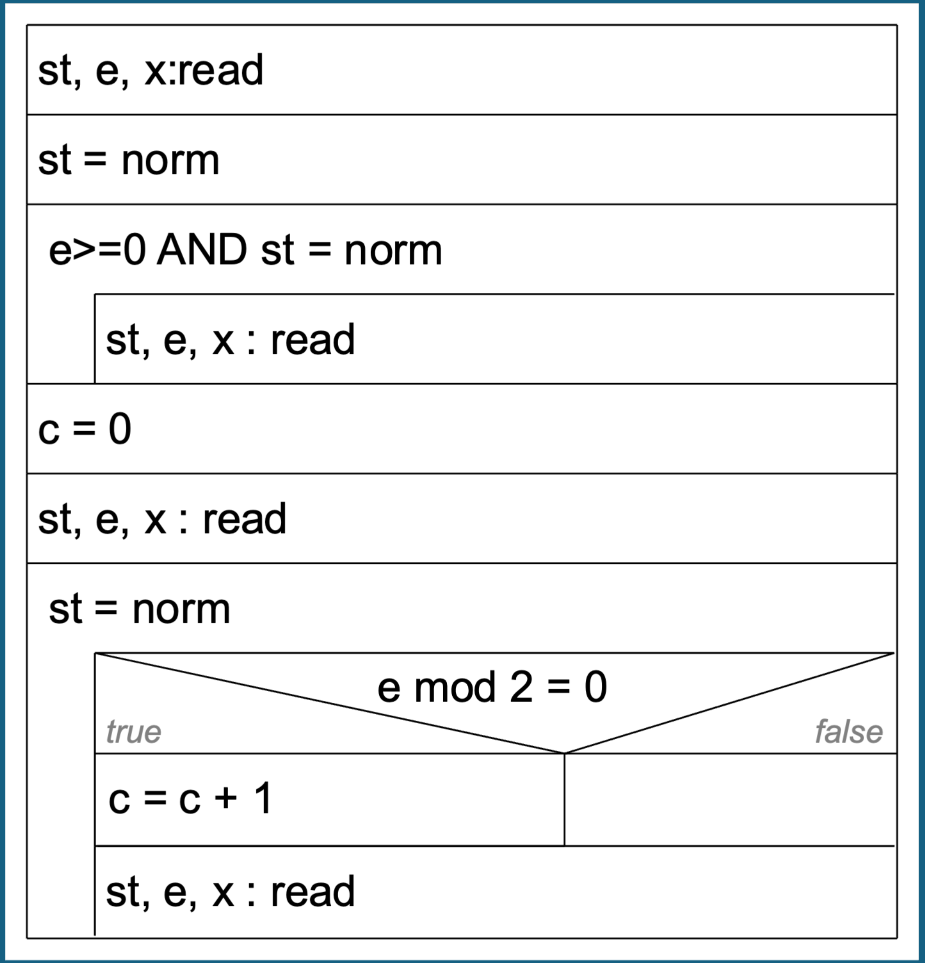
- Count even numbers after first negative number
  - $A = (x{:}\, infile(Z), c{:}\, N)$
  - $Pre = (x = x') \mid x\ - out\ |x' - inp$
  - $Post = \left(l, elem, (st'', e'', x'') = SEARCH_{e \in x'}(e <)\ \wedge c = \sum_{\substack{e \in x'' \\ 2 \mid e}} 1\right)$
  - $st''\ \ - read\ operation\ status\ when\ first\ negative\ element\ found$
  - $e'' - element\ to\ which\ current\ pointer\ points\ \ - elem$
  - $x'' - unread\ elements\ (remaining\ part\ of\ the\ x')$

Concrete Problems

- $Post = \left(l, elem, (st'', e'', x'') = SEARCH_{e \in x'}(e <)\ \wedge c = \sum_{\substack{e \in x'' \\ 2\,|\,e}} 1\right)$
- In the post condition given above, the variable l is redundant and has no use in our specific case. We can apply another methodology to address this.
- $Post = (e'', (st'', e'', x'') = SELECT_{(st,e) \in x'}(e < 0\ \vee st = abnormal\ (end\ of\ file) \wedge c = \sum_{\substack{e \in x'' \\ 2\,|\,e}} 1)$

Analogy | Selection and Counting

- $t : enum(E) \sim x : infile(Z)$
- $elem \sim e''$
- $cond(e) \sim (e < 0 \lor st = abnormal)$



```
st, e, x:read
st = norm
e>=0 AND st = norm
    st, e, x : read
c = 0
st, e, x : read
st = norm
    e mod 2 = 0
    true:  c = c + 1
    false:
    st, e, x : read
```

Practice:

- Count even numbers before negative number and after next negative number.
  Example: [1,2,4,-1,4,5,6,-2,3,6] - > SUM(2,4,6)