Should you start from 0 or 1?

Kaminski / cs3310

This concept is important to computer people since we sometimes start with 0, sometimes 1, depending on. . . physical reality at times (arrays start at [0] in C#) and "logical" convention (assign InvoiceNumber starting at 1). The reason this is an issue in this course is because it's an easy place to make a mistake in programming – i.e., an edge case or off-by-one case or inconsistency-between-modules case. Most of the following are things we all know, but a few issues and cautions are noted, particularly when using a new programming language, when working with users/clients/programmers from other countries, when you're working on a team project where consistency is needed.

- Counters are initialized to 0, then incremented for every item -- obviously! There are 3 A's in KALAMAZOO. There are 0 A's in WESTERN. But always check your output for the correct result since it's not uncommon for a programmer to put the `count++` in the wrong spot.

- Position (ordinal) numbering generally starts with 1, not 0 – i.e., 1st, 2nd, 3rd, . . (Zeroth is in the dictionary, but 0th isn't commonly used by humans). For example, "Stadium Dr. is the 2nd right turn after leaving Parkview BTR". Parkview Ave. is the 1st street, Stadium Dr. is the 2nd street, there is no 0th street.

- Numbering schemes start with 1, not 0, in the real world, generally. Does anyone have a room# of 0, or a WIN or SSN of 000-00-000, license plate number of WMU 0000, a phone number of 000-0000, an area code of 000, a house number of 0, . . . Imagine the "suppress-leading-zeroes" problem in programs if these were stored as integers. [Programmers should always consider the "suppress-leading-zeroes" problem and avoid it with proper formatting codes – e.g., when WIN's were first used at WMU, some students appeared to have 8-digits WIN's].

- Numbering schemes sometimes don't even start at 1 – e.g., SSN of 000-00-0001 (even before computer processing in 1936). Numbers sometimes have internal meanings for sub-groups of digits – e.g., SSNs are of the form AAA-GG-SSSS (Area issued, Group, Serial#) and none of these 3 sub-groups could be all 0's.

- Not all position numbering starts at 1 – as for example with floor numbering in a building. Typically in the USA (and some other countries), the floor you walk in on from the outside front entrance is the 1st floor – e.g., CEAS Building at Parkview has room C-123 on the 1st floor, and D-201 on the 2nd floor. However, much of Europe starts numbering at 0, in effect, calling the main entrance floor the ground floor, and the next level up the 1st floor – so they'd probably label things as: room C-023 on the ground floor, and D-101 on the 1st floor. [See "Storey" in Wikipedia]. (So, if you're booked into a room on the 3rd floor of a hotel in London, how many flights of stairs do you climb? 3, not 2 like you would in Chicago).

- So, is numbering scheme a somewhat arbitrary concept? In the normal everyday world, no, it's context-specific and we use the "everybody knows" rule. But in programming, the numbering scheme starting point IS often arbitrary – it's a "logical" concept (a human-imposed conceptual idea, a pretense). But of course, everyone on the project must agree to use the same convention. So, it's the system designer who gets to decide (based on either client/user needs/wishes or on the designer's judgment), and of course, it's clearly spell it out in the requirements specifications (which the programmers all follow).

- Array subscripts (indexes) in C# start at 0. So . . .
    - An array of size N has N elements which are referred to as element[0] to element[N-1].
    - `for` loops which handle arrays go from `i = 0` to `i < N` (not to `i = N` or to `i <= N`) -- this is already a habit for most programmers, so it's probably not a common mistake to make.
    - The 3rd element in an array is element[2] -- which takes an "extra cpu cycle of human thought" (like Ada writing the first computer program in the mid- 19th century, which was in 1843). So be careful when referring to a particular element in an array (including the last one). . .
    - This is a PHYSICAL ("really real") concept, so you can't just arbitrarily change it – but if necessary, in order to please the user, for example, who may want a list of songs to be printed out as 1,2,3,…N, you could easily print `i+1` for the songNumber to correspond to `song[i]` in a loop.
    - In order to better map the data storage to the real-world problem (for clarity, to reduce the likelihood of programmer error) one could set up an array of size N+1, and then just not use location 0 – as for example, in storing the average monthly sales totals, where "everybody knows" that 1 is January. . . 12 is December.

.

- Array subscripts don't start at 0 in all programming languages. COBOL (and other older languages), for example, starts at 1, and so elements go from [1] to [N]. This is a caution when using a unfamiliar language or when converting a program from one language to another. This is particularly important if the "bounds checking" switch is not enabled (which it SHOULD be) – and hence locations outside 1 to N in COBOL or outside 0 to N-1 in 0-starting languages contain data. It's just that the data is whatever happens to be stored in memory immediately before or after (respectively) the array – it's garbage, perhaps "good looking garbage".

- The auto-increment feature in database management systems generally starts with 1, by default – for example, in generating unique primary keys for a table's rows, e.g., for InvoiceNumber in the Invoice table. This is an arbitrary starting point, conceptually, but it's part of the "everybody knows" rule

- What about random access files (a focus in this course) – do the relative record numbers (RRN's) start with 0 or 1? Such files are implemented using the concept of a "relative file" – i.e., relative to the start of the file, which record is being referred to.
    - In C/C++/C#/…, a relative file is just a logical concept – a human (programmer) conceptualization of the file which is really (physically) just a stream of bytes. (Even the concept of a file being a collection of records is just a logical view imposed on the file by the programmer, in C/C++/C#/…). Since it's just a logical concept, the designer could decide to start with 0 or 1. The argument for staring at 1 is that RRN is a positional concept, so one is likely to refer to "the 1st record is in the 1st location, which is RRN 1" rather than "the 1st record is in the 1st location which is RRN 0". (The argument for starting at 0 is that programmers are used to array subscripts staring at 0, and so will be less error-prone).
    - In COBOL (and other languages), a relative file is a physical concept (like arrays), and so the 1st location in the file is specified as Relative Key (i.e., RRN) 1. There IS no Relative Key 0.
    - In C/C++/C#/… RRN is not specifically used by the language's built-in I/O methods for reading and writing. Nor is RRN directly used for positioning the file-position-pointer before a subsequent read or write – that's done by the `seek` in C# FileStream (or `seekp` and `seekg` in C++, or `seek` in C). A seek uses the byte-offset relative to the start of the file, where the 1st byte of the file is byte 0. So if the designer specifies that:
        - RRNs start at 1, then `offset = (RRN – 1) * sizeOfRecord;`
        - RRNs start at 0, then `offset = RRN * sizeOfRecord;`