Lab 1.5- Abdirahim Abdullahi

1-userstaticarrary.c

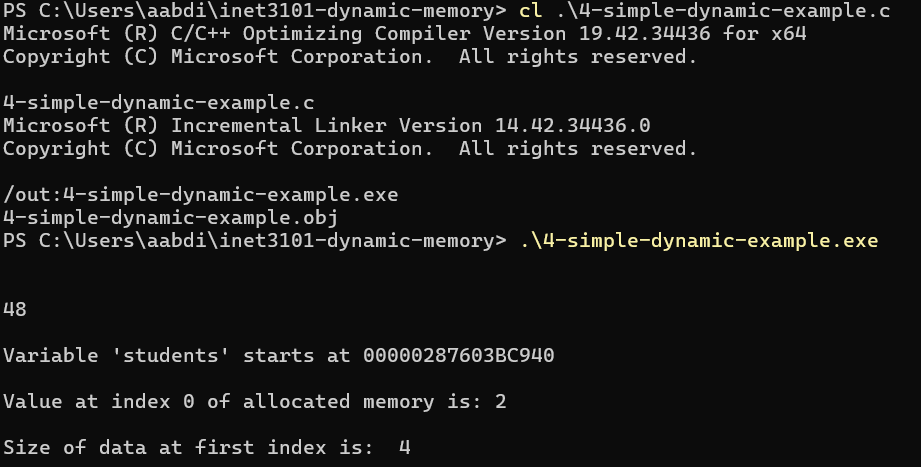
A screenshot of a computer screen

AI-generated content may be incorrect.

3-dynamically-allocating

Originally, the code used malloc() to allocate a set amount of memory, so it couldn't expand dynamically when the user wished to add more integers than the initial allocation. I replaced the for-loop with a while-loop for continuous user input and applied dynamic memory reallocation with realloc() to address this. Reallocation doubles the memory size once the array fills its capacity, therefore guaranteeing effective expansion and low memory operations. As W3Schools' realloc() lesson explains—which emphasizes realloc() as a fundamental ability for enlarging memory blocks without losing previously stored values—this method follows best practices in dynamic memory management. Using this approach lets the software dynamically grow depending on user input while preserving memory economy. After every malloc() and realloc() call, error handling is also used to prevent crashes resulting from memory allocation problems; memory is then correctly released at the end to stop leaks.

4-simple-dynamic



5-python list

An object in object-oriented programming (OOP) is a class instance that captures both data and behavior, that is, functions or methods working on the data. Objects help to enable abstraction, encapsulation, and reuse, so simplifying and controlling code. Unlike a basic C array, a Python list is an object with built-in capabilities allowing it to be dynamically scaled, sorted, altered, and expanded rather than being a collection of data. Calling list.append(10) for instance automatically controls memory allocation and resizing; list.sort() sorts the elements without having the programmer to personally apply sorting logic. These actions are copies from the list class that show how Python simplifies data management by applying OOP ideas.

6-linked list

Using a linked list offers flexible memory management without requiring a fixed-size array, therefore helping to tackle some of the same issues we want to overcome with dynamic memory allocation. Resizing in conventional arrays calls for wasteful copying data and building a fresh array. Linked lists, on the other hand, allow elements to be added or deleted without moving other elements in memory using dynamically allocated nodes made with malloc(). For regular insertions and deletions, linked lists so are more efficient than dynamic arrays. But since items aren't kept contiguously in memory like arrays, they also bring extra memory cost from storing pointers and can have longer access times.