Joshua Llorin

Prof. Zia Ud Din

CP472 - Programming Languages

27 January 2024

Analysis on the Implementation of a Student Database System With and Without using Structs in C

This report covers an analysis on the 2 implementations of a student database system in the “C” language, one using structs and one not using structs. The analysis will be conducted on the basis of data organization, code maintainability and readability.

Data Organization can be interpreted both in terms of organizing the data within a container as well as how the data is organized in memory. When working with the struct data type the student data is grouped together within a single student data type with each attribute of the student data belonging to a specific field within the struct. In terms of memory the struct allocates more memory as more struct variables are declared. When not working with the struct data type, the implementation method that I used in this analysis was 3 individual lists that contained a single attribute for all the student data (student ID, name and age). The memory allocation for this however is significantly more complicated than the struct implementation. After launch the program first allocates 5 entries worth of data in memory using the *calloc* function (5 is a high number for safety) and fills the allocated memory with empty values to be filled. As the user fills in values for each of the attributes in the list the program then dynamically allocates enough

memory for 1 more entry each time a new entry is put in using the *realloc* function which extends the lists by 1 entry while preserving the position of each existing entry.

Code Maintainability for structs is much easier as to add more attributes to the student data type is as simple as adding in a new field within the struct itself. Alongside this the addStudent() function must also be updated to allow the user to input a value for the new attribute as well as use a pointer to store that new value into the student data type. For the implementation without the use of structures it is slightly more complicated as it involves initializing a new list at the beginning to act as the storage for all the data of that specific attribute as well as allocating more memory for the new list after the code launch. The addStudent() function must also be updated to allow for input and storage into the new list but there must also be another call to the *realloc* function to dynamically reallocate more memory for the list as new entries are added.

Readability for the implementation using structs is much simpler as all the attributes of the student data are contained within fields of a single struct data type as compared to the non-struct implementation which utilizes many lists to store the attributes of the data. The non-struct implementation also covers more data types as it utilizes individual lists of various types to store the various data types of each attribute.