Topic 1: Strongly- vs weakly-typed programming languages

1. URI 1: https://www.geeksforgeeks.org/strongly-typed-vs-loosely-typed-languages/

URI 2: www.tutorialspoint.com/cprogramming/c\_type\_casting.htm

URI 3: https://www.programiz.com/c-programming/c-operators#type-conversion

1. Précis:

The distinction between strongly- and weakly-typed languages is centered on how strictly that language enforces its typing rules, in particular the implicit conversions between mismatched types. A strongly-typed language will resist or forbid operations between incompatible types, which usually results in an error. On the other hand, a weakly-typed language is more permissive and might perform implicit type conversions (coercion) in those cases. Based on the present sources, C appears to be weakly-typed. This is because of the extensive use of implicit conversions and permission of manipulation. The language trusts the programmer to manage correctly rather than preventing them at the compiler level.

1. Summary:

The main difference between the two types is how restrictive the compiler enforces variable types and their conversions. A strongly-typed language requires adherence to the type rules, often preventing operations between incompatible types and requiring explicit conversion. In contrast, a weakly-typed language allows implicit/automatic conversions.

Based on this, C is classified as weakly-typed. It includes implicit arithmetic conversions between char, int, float, in expressions without requiring casts from the programmer. The *void* pointer type can be assigned without type checks. It also has explicit casting power to force a change to another type. This weak-typing grants flexibility and low level control over resources.

Topic 2: Arrays

1. Hoover

URI 2: https://www.w3schools.com/c/c\_arrays.php

URI 3: https://www.fresh2refresh.com/c-programming/c-array/

1. Précis:

An array is a fundamental data structure in C that allows the storage of multiple values of the same type in a single, contiguous block of memory under one variable name. It’s purpose is to efficiently manage and organize related data, enabling access to individual elements via numerical index. This structure facilitates processing data collections through loops, replacing the need for numerous individual variables. This requires management, as C does not prevent access beyond the array’s allocated bounds, which can lead to memory corruption and unstable behavior.

1. Summary:

An array is a collection of elements of the same type that are stored sequentially in memory and accessed through a variable name. They group related data together, allowing the use of an index to directly read/modify any element in the collection. This enables the efficiency of a loop across large sets of values. In this, they prevent the need to create eparate variables and are the foundation for implementing processes like algorithms. An array should be used when there is a fixed number of items of the same type. For example, the grades of my 45 students, an inventory, or data points for a set of research subjects.

Topic 3: Strings in C

1. Hoover

URI 2: https://www.guru99.com/c-strings.html

URI 3: https://codeforwin.org/c-programming/string-programming-exercises-and-solutions-in-c

1. Précis:

A string is a not a primitive data type, but rather a convention implemented as an array of char elements, terminated by a null character (‘\0’). Their function is to store and manipulate text data. Strings are created either by declaring a char array and initializing it with a string literal (which automatically adds the null terminator) or by manually assigning individual characters and appending the ‘\0’.

The special commands for strings are the functions provided by the C Standard Library; they all rely on the null terminator to identify the end of the string and perform their operations safely. A string should be used whenever a program needs to process text, such as accepting user input. It must always be large enough to hold the text and the null terminator to prevent buffer overflows.

1. Summary:

A string is an array of characters (char) that ends with a special nill terminator character: ‘\0’; this signals the end of the string. They are used to represent and manipulate textual data, such as words, sentences, or any sequence of characters. They allow programs to work woth text input from users, output text to a screen, among other uses.

They can be created by declaring with an initialization: char string[] = “Text”; with the complier automatically adding the ‘\0’ and sizing the array. They can also be made manually by declaring an array and assigning each character, but the terminator must be added manually here. They do have special commands such as strlen(), strcpy(), and srtcat(). These all automatically look for the null terminator to know when to stop. A string should be used when a program needs to handle text as in the situations mentioned above.