Project 3 specifications

September 2024

1 Overview and purpose

This project has you implement a number of array operations.

Populating an array with geometrically increasing or decreasing values is necessary in several applications. In financial modeling, it helps simulate compound interest and growth scenarios over time. In computer graphics, it is used to create perspective effects and gradient shading by adjusting color or depth values geometrically. In audio processing, it generates logarithmic scales for frequency representation, aligning with human perception. In machine learning, it assists in hyper-parameter tuning and learning rate schedules, improving model training efficiency. In network protocols, it supports exponential backoff algorithms, optimizing re-transmission strategies. In digital signal processing, it facilitates the design of filters with geometrically spaced frequency bands. In physics simulations, it models exponential decay or growth processes, such as radioactive decay or population dynamics. Overall, geometric progression in arrays is essential for accurately representing exponential relationships, optimizing algorithms, and enhancing realism in simulations.

Cross-correlation is widely used in various real-world applications to measure the similarity between signals or datasets. In signal processing, it identifies the time delay between signals for applications such as echo location and radar detection. In image processing, it helps in template matching, where a small image or pattern is located within a larger image. In telecommunications, it aids in synchronizing the receiver with the transmitted signal, improving data integrity. In finance, cross-correlation analyzes the relationship between different financial instruments, helping in portfolio management and risk assessment. In neuroscience, it studies the relationship between neuronal spike trains to understand brain connectivity. In machine learning, cross-correlation is used in convolutional neural networks (CNNs) for feature extraction and pattern recognition. Overall, cross-correlation is essential for comparing and aligning signals, improving detection accuracy, and extracting meaningful information across diverse fields.

An algorithm that moves all duplicate entries to the end of an array while ensuring the first n entries are unique and in the same order as they appear in the original list can be highly useful in situations where data integrity and order are crucial. For instance, in a customer management system, this algorithm could be used to ensure that each customer's unique information is processed first, preventing duplication in crucial initial operations such as verification or reporting. By maintaining the original order of unique entries, we preserve the sequence in which data was entered or received, which is often important for time-sensitive processes. This approach can also be beneficial in optimizing search operations within the unique subset, reducing the complexity of handling duplicates and enhancing overall system efficiency.

Having a single function that sets all entries of a dynamically allocated primitive array to zero, deletes the array, and assigns the local variable storing the array address to `nullptr` ensures safer and more efficient memory management in C++. This approach encapsulates the memory cleanup process, reducing the risk of human error and code duplication. It prevents potential issues like memory leaks and dangling pointers by ensuring that these operations are always performed correctly and in the right order. By consolidating these tasks into a single function, code readability and maintainability are improved, making it easier to manage resources, debug issues, and enforce consistent memory management practices across the codebase. This function serves as a reliable and reusable utility, enhancing the overall robustness and stability of the software.

2 Project description

The first function `double *geometric( double a, double ratio, std::size_t cap )` returns the address of a dynamically allocated array of `double` that stores the values a, ar, ar^2, ar^3, up to ar^{n-1} where r is the ratio and n is the capacity of the array.

The second function

```
double *cross_correlation(
  double array0[], std::size_t cap0, double array1[], std::size_t cap1
);
```

will return a dynamically allocated array of capacity `cap0 + cap1 - 1`. With each entry of this new array initialized to zero, you will then calculate each pair of products `array0[i]*array1[j]` and add this to the result at index `i + j`. You may assert that `cap0 + cap1 >= 1`.

The third function `std::size_t shift_duplicates( int array[], std::size_t capacity )` will move duplicate entries to the end of the array. The unique entries will be moved to the front of the array in the same order that they first appear, while the duplicate entries will be moved to the end of the array, but the order does not matter. The value returned will be the number of unique entries.

The fourth function `void deallocate( double *&ptr, bool is_array, std::size_t capacity = 0 )` proceed as follows:

1. If `is_array` is true, then you will loop through the array setting each entry to zero. The third argument will contain the capacity of the array. You will then deallocate that memory.
2. Otherwise, it is not an array, so it is just a pointer to one instance of type `double`. You will set that entry to zero and the third argument will be ignored.

In both cases, the last action you will take is to set `ptr` to `nullptr`.

2.1 Testing

In Project 1, we described many of the tests that could be used to verify that your project is working correctly. In this project, like for Project 2, you will be required to author your own tests. We will, however, give you some hints.

1. You could write a function
   `bool test_geometric( double array[], std::size_t capacity, double a, double ratio )` which checks that the first entry of the array is `a` and that each subsequent entry is `ratio` times the previous entry.
2. Given two arrays, you can calculate the cross-correlation on paper, and then check the answer yourself. You should, however, realize that if one array has a capacity equal to one, then the cross correlation is simply each entry in the other array multiplied by that value. This should be your first test. Also, if one array has all zeros, then the cross-correlation is also an array of zeros.

3. Cross-correlation has a special cases when one array capacity zero. It should still generate an array, with capacity as described, consisting of only zeros.

4. For shifting duplicates, you can create an array `int data[10]{ 5, 4, 2, 2, 2, 4, 5, 1, 4, 3 };` and then print the resulting array:

   ```
   for ( std::size_t k{ 0 }; k < 10; ++k ) {
     std::cout << data[k] << " ";
   }
   ```

   and make sure the output starts with `5 4 2 1 3` and that the remaining five entries contain `2 2 4 5 4` in some order.

5. The last function is actually easier to write, but much more difficult to test. Perhaps here it is easier, once you are finished your solution, to convince a colleague that your implementation is correct.