

# Unique Pointer Exercises

- Briefly describe how the `unique_ptr` type is implemented
- In terms of memory usage and efficiency, how does a `unique_ptr` instance compare to a traditional pointer?

- Explain how `unique_ptr` follows the principles of RAII

- Give some examples of traditional pointer operations that are supported by `unique_ptr`
- Give an example of a traditional pointer operation that is not supported by `unique_ptr`

- Write a simple program that creates and initializes an instance of `unique_ptr` and performs some operations on it
- What changes would you need to make your program compile under C++11?
- (Optional) Put your compiler into C++11 mode and check your answer to the previous question

- Explain what is meant by transfer of ownership in the context of `unique_ptr`
- Why is transfer of ownership useful when returning a `unique_ptr` from a function?

- Write a function which creates a `unique_ptr` local variable which is returned by the function
- Write a program that calls the function and prints out the data in the returned `unique_ptr`