John Rollinson

W7252829

2/19/2020

Chapter 10 Review Questions:

25. How do smart pointers differ from regular pointers?

A pointer is known as a smart pointer when it has the ability to delete the dynamically allocated memory automatically which is no longer in use.

In other words, smart pointers keep track of the owners of resources and automatically deallocate the resources when the last owner goes out of scope.

Smart pointers remove the problem of dangling pointers and memory leaks in a program.

Smart pointers also remove the problem of double deletion which occurs when one part of the program deletes a pointer that has already been deleted.

26. Name the header file that needs to be included in a program that uses smart pointers.

Smart pointers keep track of the owners of resources and automatically de-allocate the resources when the last owner goes out of scope.

Three types of smart pointers are defined here:

- unique\_ptr

- shared\_ptr

- weak\_ptr

The classes of smart pointers are defined in the memory header file so the user has to include the following statement to use the smart pointers. #include

27. What happens when a unique\_ptr that is managing an object is assigned the nullptr value?

Smart pointers keep track of the owners of resources and automatically de-allocate the resources when the last owner goes out of scope.

Three types of smart pointers are defined as: unique\_ptr, shared\_ptr, weak\_ptr. Here, when a unique\_ptr that is managing an object is assigned the nullptr value then the managed object is de-allocated.

28. What does the get() method of the unique\_ptr class do?

Smart pointers keep track of the owners of resources and automatically de-allocate the resources when the last owner goes out of scope.

The unique\_ptr class has a number of instance member functions. Get() method is also a member function of unique\_ptr class.

The get() member function returns the raw pointer to the object managed by this pointer. It is useful if such a pointer needs to be passed to a function that does not know how to handle smart pointers.

30. List three different operations that are permitted on raw pointers but not on unique\_ptr objects.

The unique pointer is a kind of smart pointer. Smart pointers in c++ is an object that wraps an ordinary pointer to an owned object. Smart pointer classes are parameterized by the type of object pointed to. There are some operations which will be supported by raw pointer but not supported by unique pointers.

**Operation 1**: Pointer arithmetic operation can be supported by a raw pointer, but not supported by a smart pointer.

Consider the following example:

Int \*p = new int;

Unique\_ptr<int> uptr(p);

Uptr++ // it will generate an error

In the above code segment, uptr is a smart pointer and generate an error when an arithmetic operation increment is performed on that pointer.

**Operation 2**: User can initialize a raw pointer with the value of another raw pointer but in case of unique pointer, user can not initialize it with the value of another unique pointer.

Consider the following example:

Unique\_ptr<int> uptr1(new int);

Unique\_ptr<int> uptr2 = uptr; // it will generate an error

In the above code segment, uptr1 is a unique pointer and gives an error when it is initialized with the value of another unique pointer.

**Operation 3**: User can pass a raw pointer to function by value but in case of unique pointer, user can not directly pass a unique pointer to a function by value because pass by values involves copying the actual parameter.

Consider the following example:

Void fun(unique\_ptr<int> uptrParam)

{

Cout << “hello” << end1;

}

Int main()

{

Unique\_ptr<int> uptr(new int);

\*uptr = 10;

Fun(uptr); // this will generate an error

}