Module 3 Critical Thinking

Dennis Weddig

Colorado State University Global

CSC450: Programming III

Dr. Jack Li

8/31/2025 11:59pm

Module 3 Critical Thinking

Repository location for Module 3 Critical Thinking assignment:

https://github.com/denniswed/csc450/tree/main/Module3/critthink

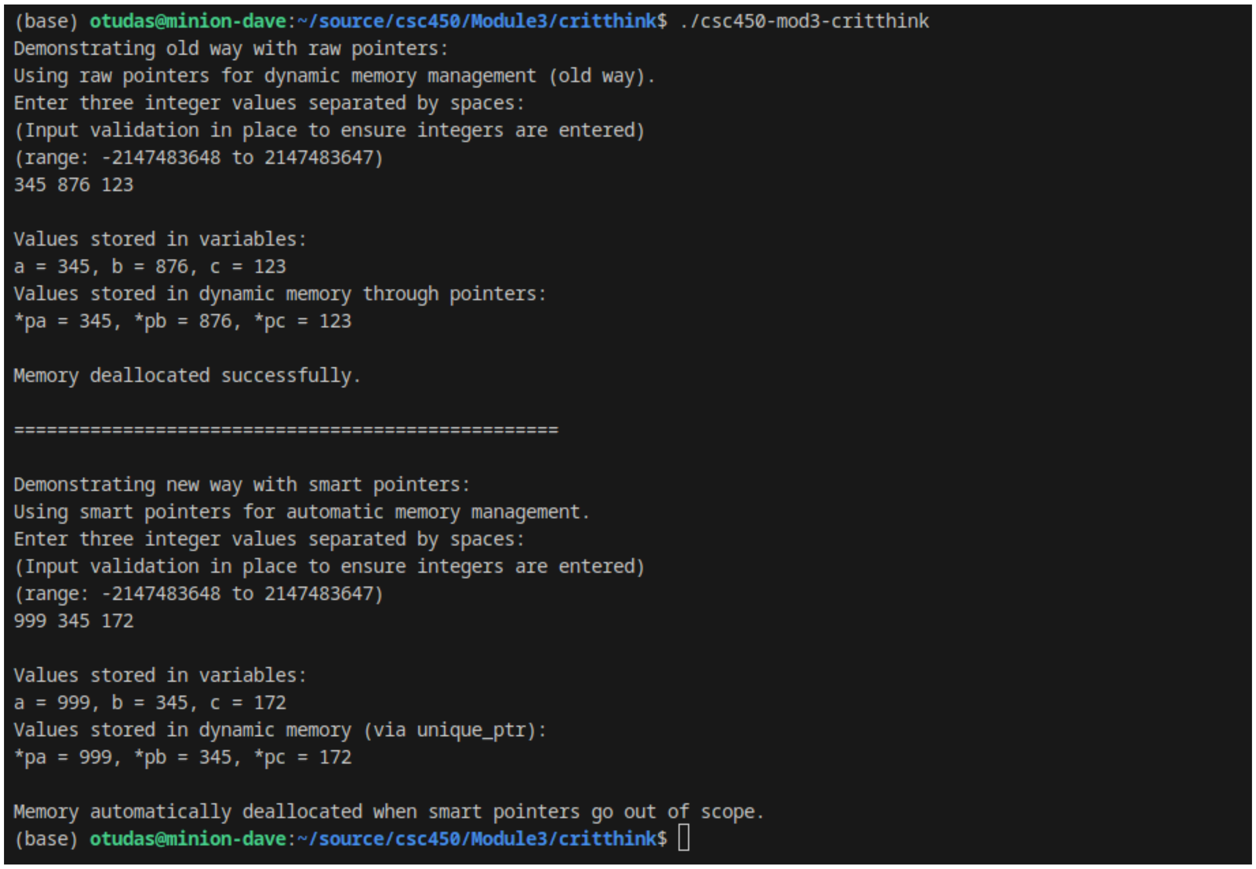
Code:

```cpp
#include <iostream>

#include <limits> // for std::numeric_limits
#include <memory> // for std::unique_ptr, std::make_unique

int oldway() {
std::cout << "Using raw pointers for dynamic memory management (old way).\n";
int a{}, b{}, c{}; // Initialize to 0 to prevent undefined behavior

// Input validation loop to ensure safe integer input
std::cout << "Enter three integer values separated by spaces: \n";
std::cout << "(Input validation in place to ensure integers are entered) \n";
std::cout << "(range: -2147483648 to 2147483647)\n";
while (!(std::cin >> a >> b >> c)) {
std::cin.clear(); // clear error state
std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n'); // discard invalid input
std::cout << "Invalid input. Please enter three integers: ";
}

// Dynamically allocate memory for three integers
int *pa = nullptr;
int *pb = nullptr;
int *pc = nullptr;

try {
pa = new int(a); // allocate and initialize with value
pb = new int(b);
pc = new int(c);
} catch (const std::bad_alloc &e) {
std::cerr << "Memory allocation failed: " << e.what() << '\n';
return 1; // terminate program safely
}

// Display values stored in variables
std::cout << "\nValues stored in variables:" << std::endl;
std::cout << "a = " << a << ", b = " << b << ", c = " << c << '\n';
```

```
// Display values stored in dynamically allocated memory
std::cout << "Values stored in dynamic memory through pointers:" << std::endl;
std::cout << "*pa = " << *pa << ", *pb = " << *pb << ", *pc = " << *pc
<< '\n';

// Clean up allocated memory
delete pa;
delete pb;
delete pc;

// Avoid dangling pointers
pa = pb = pc = nullptr;

std::cout << "\nMemory deallocated successfully.\n";
return 0;
}

int newway() {
std::cout << "Using smart pointers for automatic memory management.\n";
int a{}, b{}, c{}; // safely initialized

// Input validation
std::cout << "Enter three integer values separated by spaces: \n";
std::cout << "(Input validation in place to ensure integers are entered) \n";
std::cout << "(range: -2147483648 to 2147483647)\n";
while (!(std::cin >> a >> b >> c)) {
std::cin.clear();
std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
std::cout << "Invalid input. Please enter three integers: ";
}

// Use smart pointers (automatic memory management, no delete needed)
auto pa = std::make_unique<int>(a);
auto pb = std::make_unique<int>(b);
auto pc = std::make_unique<int>(c);

// Display values stored in variables
std::cout << "\nValues stored in variables:\n";
std::cout << "a = " << a << ", b = " << b << ", c = " << c << '\n';

// Display values stored via smart pointers
std::cout << "Values stored in dynamic memory (via unique_ptr):\n";
std::cout << "*pa = " << *pa << ", *pb = " << *pb << ", *pc = " << *pc
<< '\n';
```

```
// No explicit delete — memory is released automatically when pa, pb, pc go
// out of scope
std::cout << "\nMemory automatically deallocated when smart pointers go out "
"of scope.\n";
return 0;
}

int main() {
std::cout << "Demonstrating old way with raw pointers:\n";
if (oldway() != 0) {
return 1; // exit if oldway failed
}

std::cout << "\n" << std::string(50, '=') << "\n\n";

std::cout << "Demonstrating new way with smart pointers:\n";
if (newway() != 0) {
return 1; // exit if newway failed
}

return 0;
}
```

Screenshot of above compile and execution:



```
(base) otudas@minion-dave:~/source/csc450/Module3/critthink$ ./csc450-mod3-critthink
Demonstrating old way with raw pointers:
Using raw pointers for dynamic memory management (old way).
Enter three integer values separated by spaces:
(Input validation in place to ensure integers are entered)
(range: -2147483648 to 2147483647)
345 876 123

Values stored in variables:
a = 345, b = 876, c = 123
Values stored in dynamic memory through pointers:
*pa = 345, *pb = 876, *pc = 123

Memory deallocated successfully.

==================================================

Demonstrating new way with smart pointers:
Using smart pointers for automatic memory management.
Enter three integer values separated by spaces:
(Input validation in place to ensure integers are entered)
(range: -2147483648 to 2147483647)
999 345 172

Values stored in variables:
a = 999, b = 345, c = 172
Values stored in dynamic memory (via unique_ptr):
*pa = 999, *pb = 345, *pc = 172

Memory automatically deallocated when smart pointers go out of scope.
(base) otudas@minion-dave:~/source/csc450/Module3/critthink$
```

References

cppreference.com. (n.d.). Smart pointers. In Cppreference.com. Retrieved August 26, 2025, from https://en.cppreference.com/w/cpp/memory

Fertig, A. (2024, September 3). Understanding the inner workings of C++ smart pointers – The shared_ptr. Andreas Fertig blog. Retrieved August 26, 2025, from https://andreasfertig.com/blog/2024/09/understanding-the-inner-workings-of-cpp-smart-pointers-the-shared_ptr/

Microsoft. (2021, August 3). Smart pointers (Modern C++). In Microsoft Learn. Retrieved August 26, 2025, from https://learn.microsoft.com/en-us/cpp/cpp/smart-pointers-modern-cpp?view=msvc-170