Lab9: Pointers and Arrays

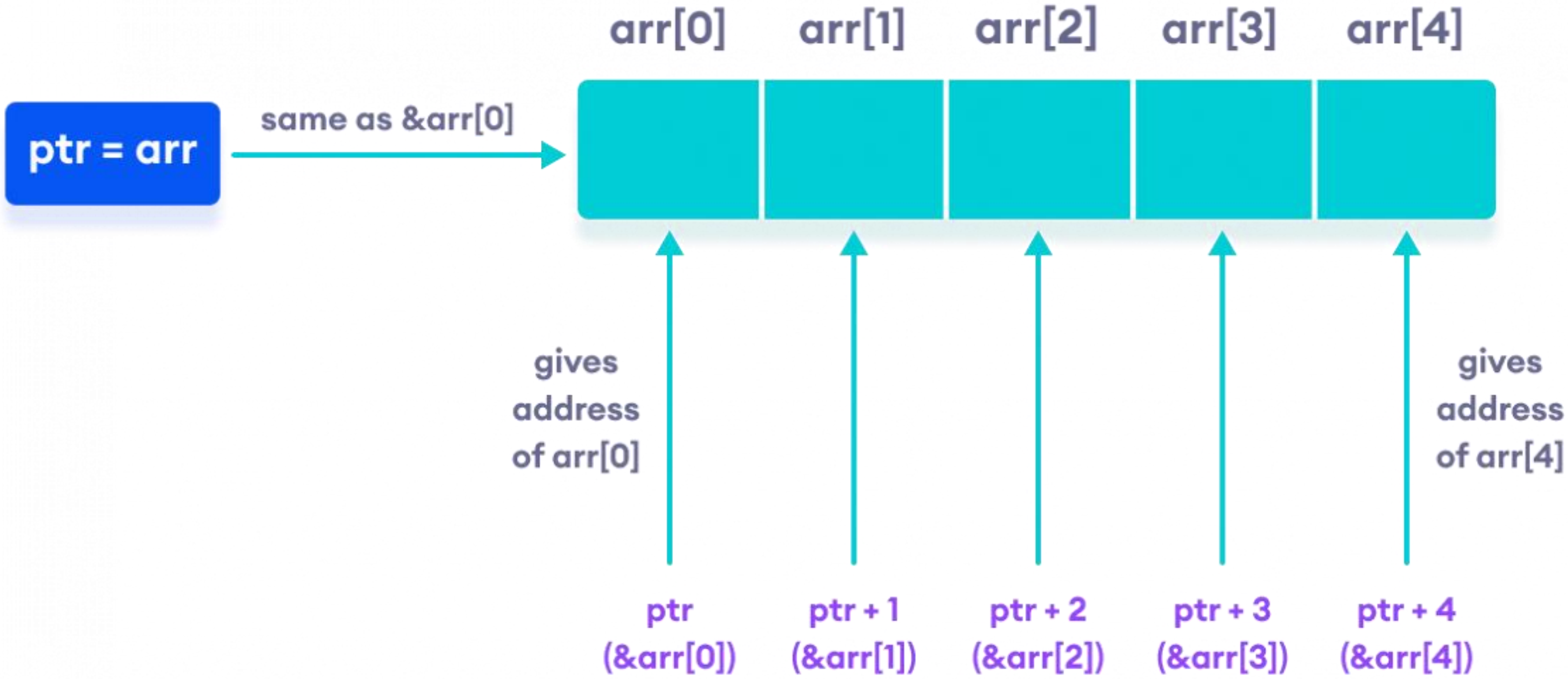
Yasin Zamani Dian-Lun Lin (Luan)

Department of Electrical and Computer Engineering

University of Utah, Salt Lake City, UT

Pointer

Array as Pointer!

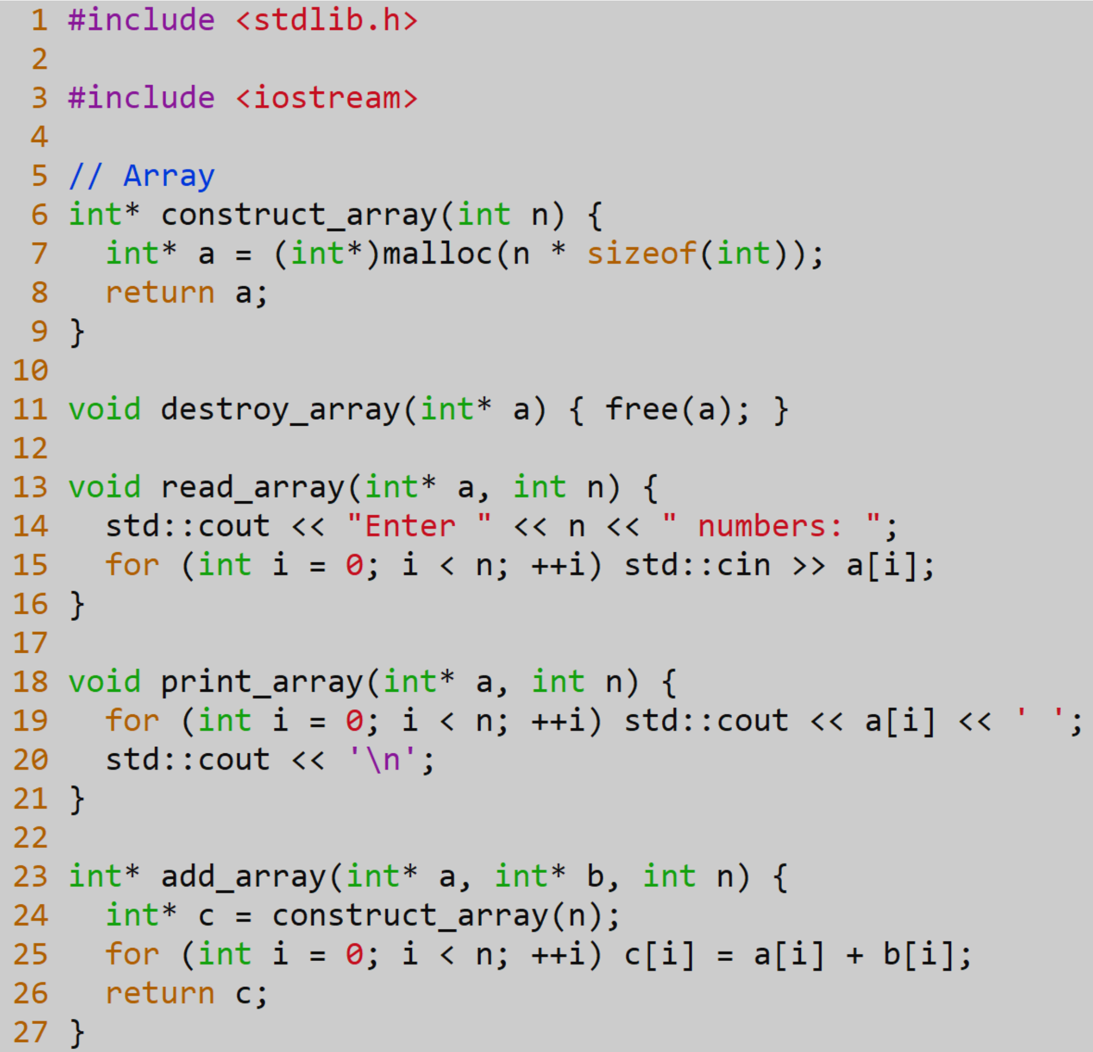


Problem

Array Library

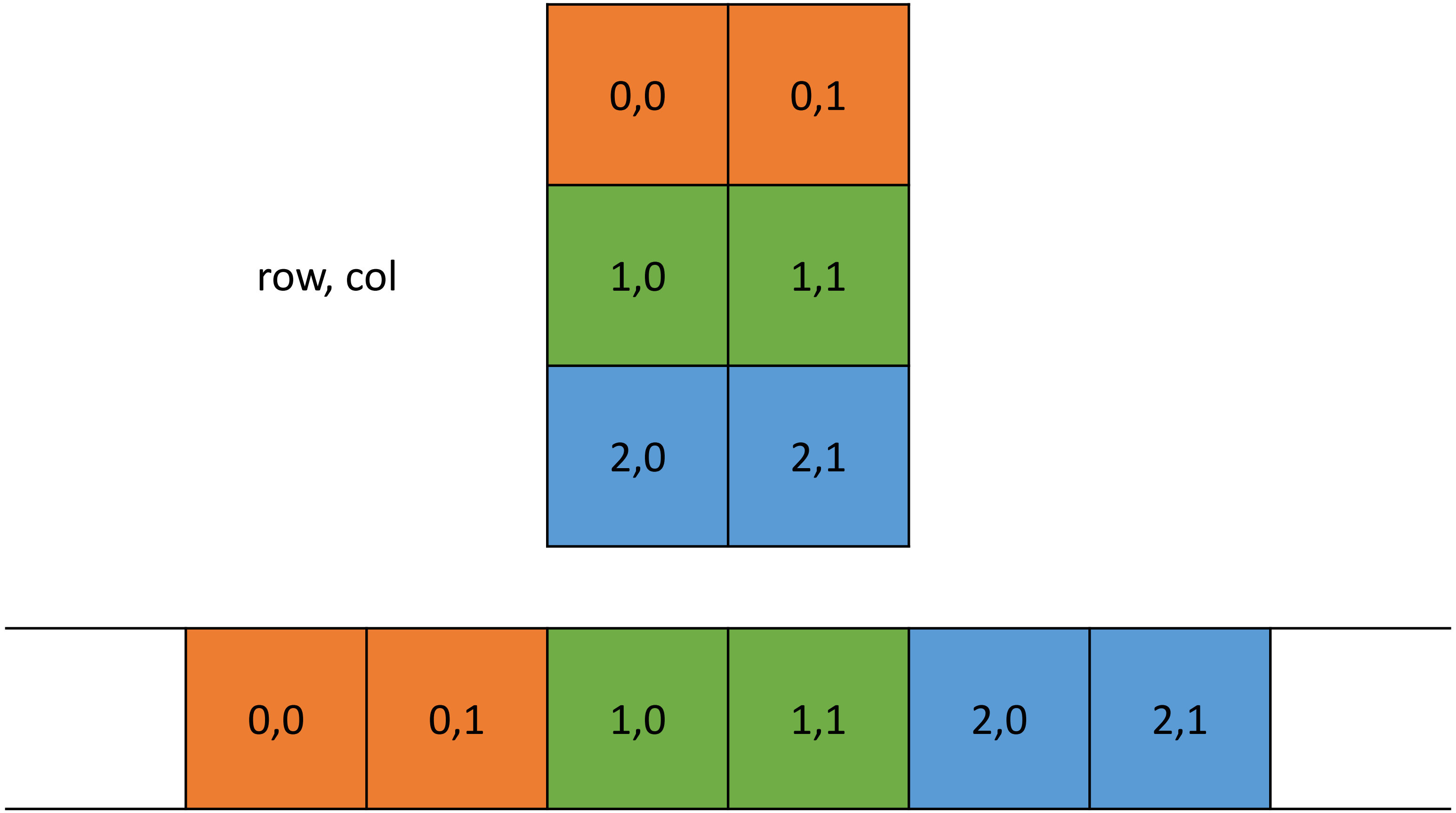
- Construct
- Destroy
- Read
- Print
- Add

Solution



```
#include <stdlib.h>

#include <iostream>

// Array
int* construct_array(int n) {
  int* a = (int*)malloc(n * sizeof(int));
  return a;
}

void destroy_array(int* a) { free(a); }

void read_array(int* a, int n) {
  std::cout << "Enter " << n << " numbers: ";
  for (int i = 0; i < n; ++i) std::cin >> a[i];
}

void print_array(int* a, int n) {
  for (int i = 0; i < n; ++i) std::cout << a[i] << ' ';
  std::cout << '\n';
}

int* add_array(int* a, int* b, int n) {
  int* c = construct_array(n);
  for (int i = 0; i < n; ++i) c[i] = a[i] + b[i];
  return c;
}
```

Matrix as Array!



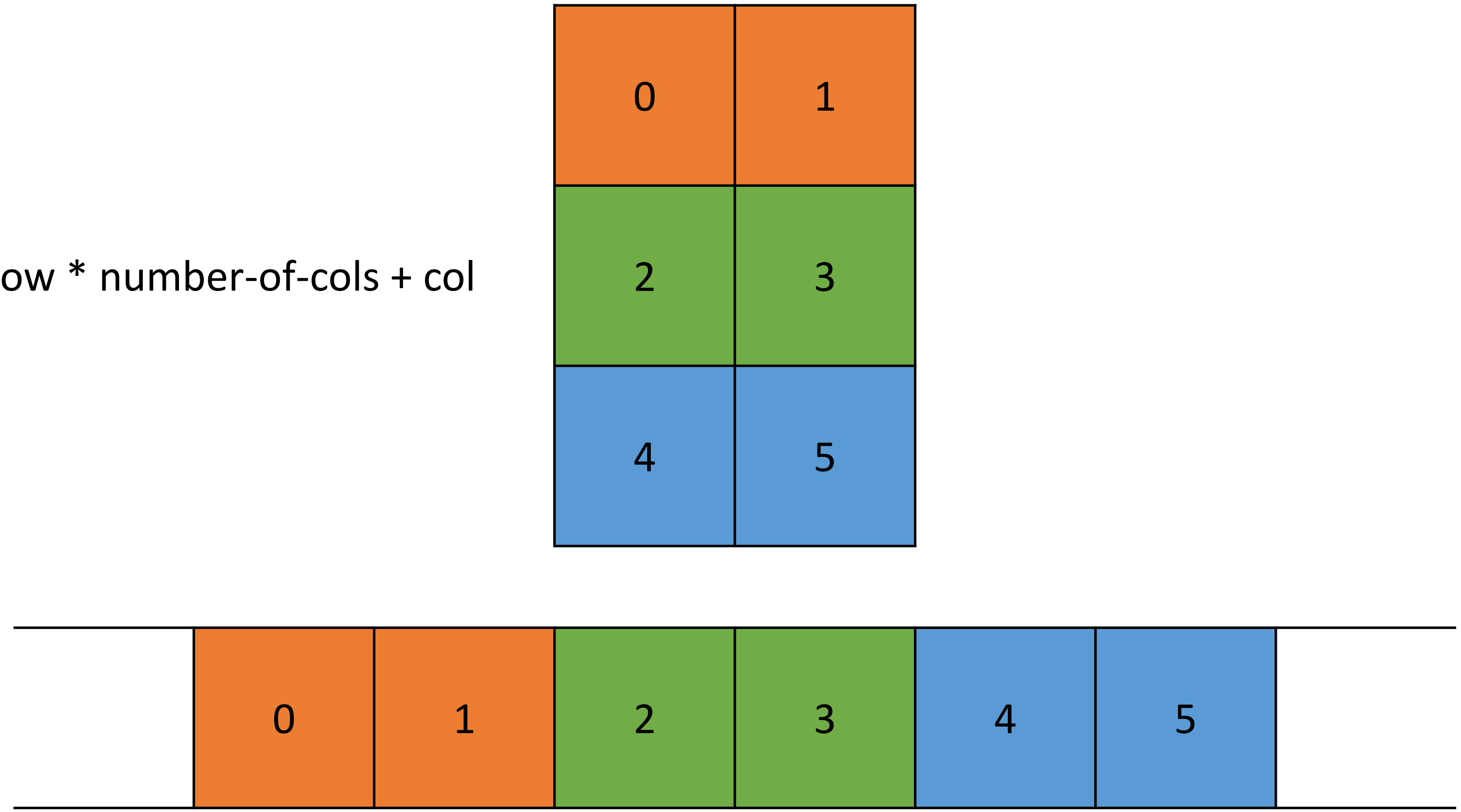
row, col

0,0	0,1
1,0	1,1
2,0	2,1

0,0	0,1	1,0	1,1	2,0	2,1

Matrix as Array!

ind = row * number-of-cols + col



0	1
2	3
4	5

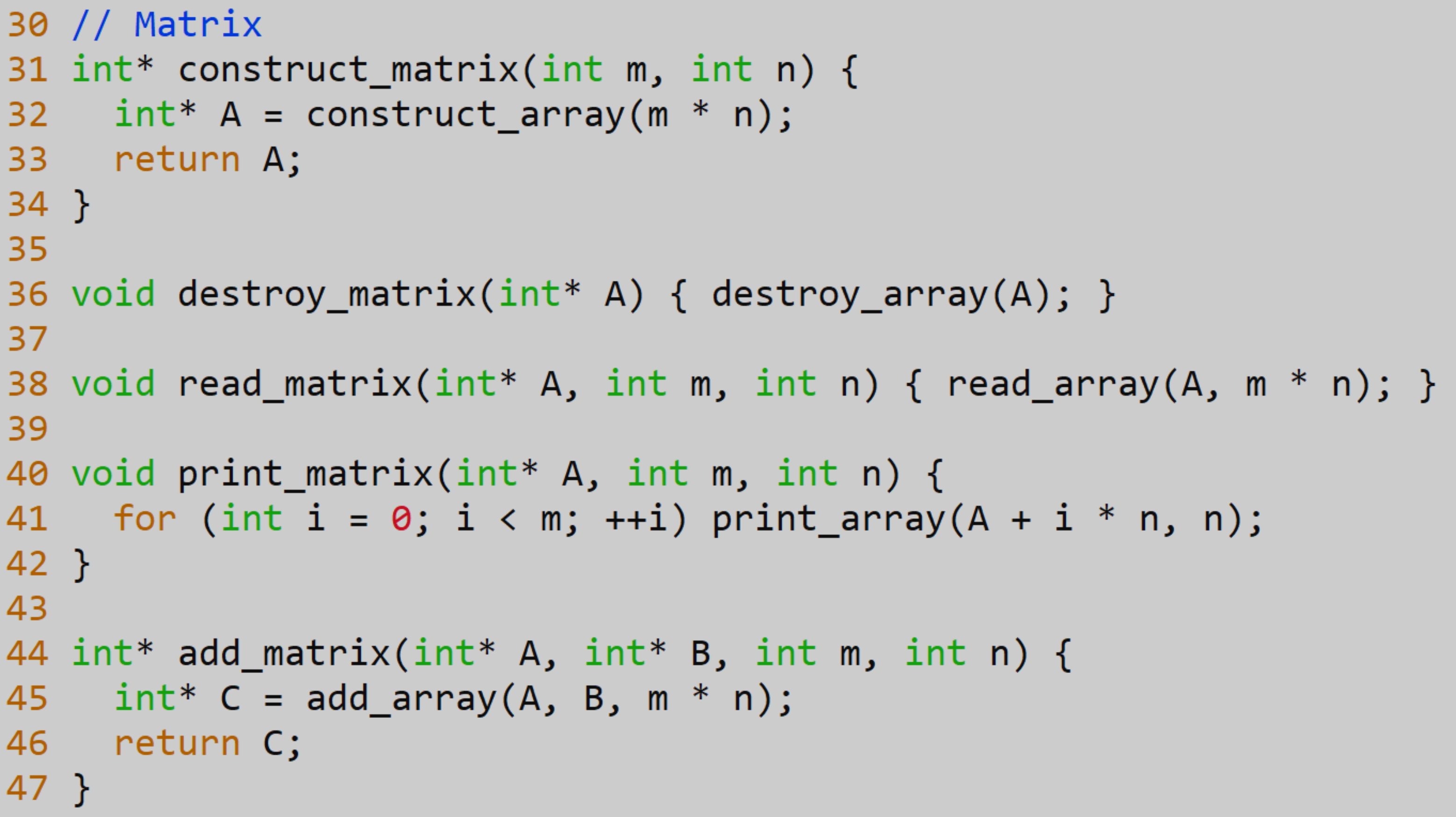
0	1	2	3	4	5

Problem

Matrix Library

- Construct
- Destroy
- Read
- Print
- Add

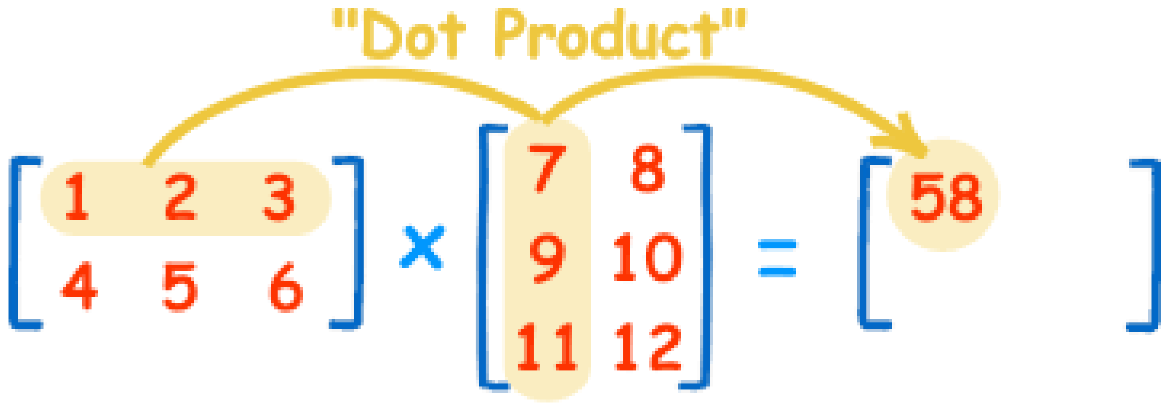
Solution



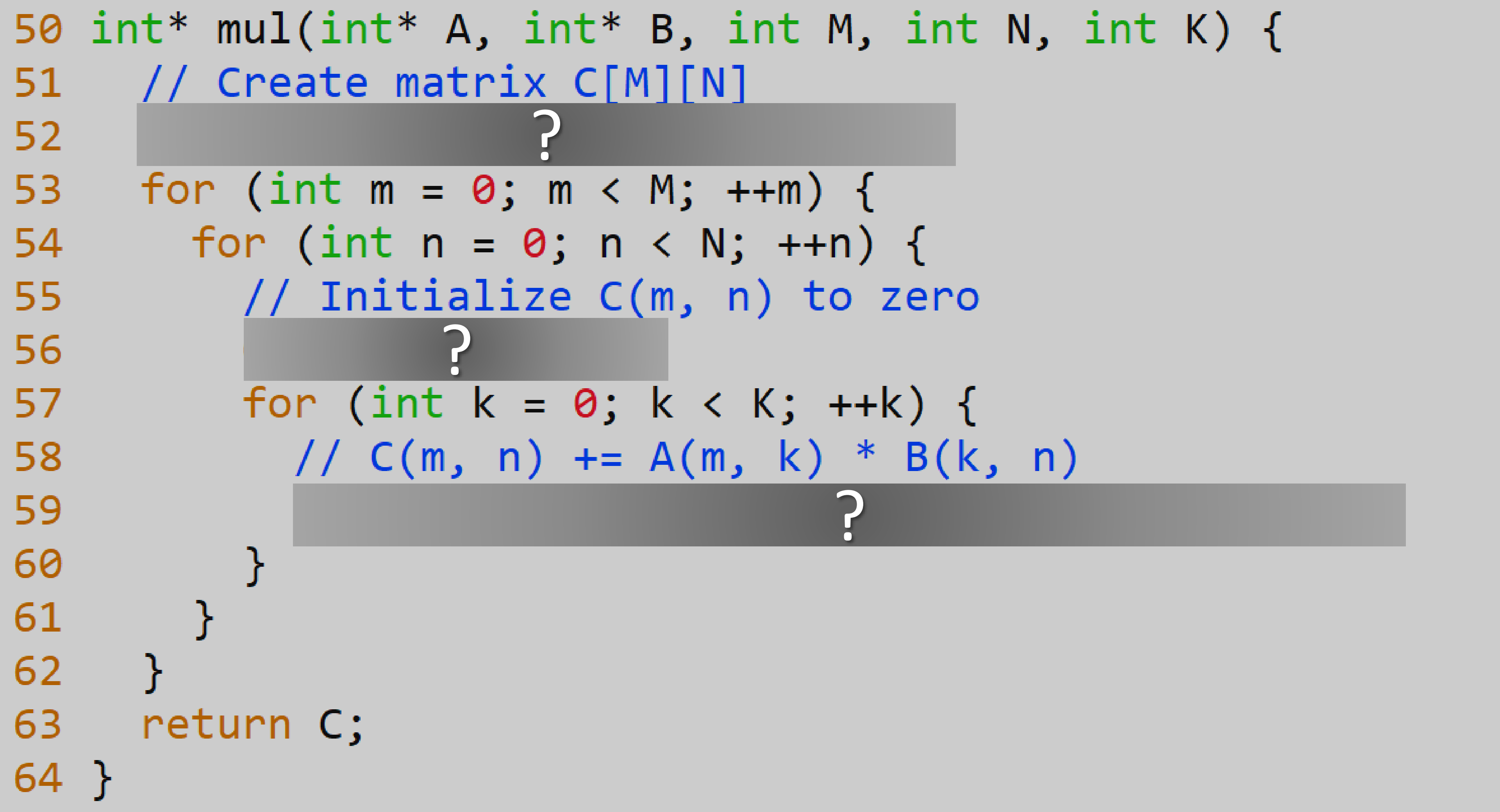
```
// Matrix
int* construct_matrix(int m, int n) {
  int* A = construct_array(m * n);
  return A;
}

void destroy_matrix(int* A) { destroy_array(A); }

void read_matrix(int* A, int m, int n) { read_array(A, m * n); }

void print_matrix(int* A, int m, int n) {
  for (int i = 0; i < m; ++i) print_array(A + i * n, n);
}

int* add_matrix(int* A, int* B, int m, int n) {
  int* C = add_array(A, B, m * n);
  return C;
}
```

Assignment

Matrix Multiplication



Solution



```
int* mul(int* A, int* B, int M, int N, int K) {
  // Create matrix C[M][N]
                ?
  for (int m = 0; m < M; ++m) {
    for (int n = 0; n < N; ++n) {
      // Initialize C(m, n) to zero
              ?
      for (int k = 0; k < K; ++k) {
        // C(m, n) += A(m, k) * B(k, n)
                            ?
      }
    }
  }
  return C;
}
```