*lecture 9*

- *Header files*
- *Shared object libraries*
- *Matrix data structures in memory*
  - *column and row major orders*
  - *vector of vectors*
- *Hopefully continue with the Poor Man's matrix class*
    - *defining class operators for +,-,*,/*

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

## *header files (.hpp , .h)*

- *used to include declarations of classes, functions, and variables that are defined in other source files*
  - *the methods are implemented in separate source files*
- *C++ headers often contain templates, which can lead to naming conflicts with C headers*
  - *.hpp is more common for C++*
  - *.h is more common for C*
- *bad practice to include namespaces bc it forces the namespace on all files that include the header*

*header files (.hpp , .h)*

- ***include guards***
  - *shared code may only be defined one time*
    - *including the same header file more than once usually causes compilation error*
    - *this is prevented using the include guard*
- *the entire header file is enclosed in an* ***#ifndef*** *section*
  - *only when the macro checked is not defined is the header file included*

```
// Code.hpp
#ifndef CODE_HPP
#define CODE_HPP
// header file foo
#endif // CODE_HPP
```

*Kenneth J. Roche*
*University of Washington*

*header files (.hpp , .h)*

without actually allocating memory for the variable or generating code for the function

- *necessary to declare a **global variable** before it can be used in compilation unit outside the one defining it*
  - *place shared variable in header and mark it with keyword **extern***
  - *extern suggests the variable is initialized in another compilation unit*
- *functions are extern by default so don't need to be labeled such*
- *global variables and functions can be declared externally many times in a program, but defined only once*

If the same variable is defined multiple times in the program, the linker will report a *"**multiple definition**"* error. If a function is defined multiple times, the linker will report a *"**duplicate symbol**"* error.

```
// Code.hpp
extern int global_integer;

// Code.cpp
int global_integer = 0;
```

```cpp
// math_utils.cpp
#include <iostream>

int square(int x) {
    return x * x;
}
```

```cpp
// math_utils.hpp
#ifndef MATH_UTILS_HPP
#define MATH_UTILS_HPP

// extern is optional here, but shows that it's defined elsewhere
extern int square(int x);

#endif
```

```cpp
// main.cpp
#include <iostream>
#include "math_utils.hpp"

int main() {
    int value = 5;
    std::cout << "Square of " << value << " is " << square(value) << std::endl;
    return 0;
}
```

```
g++ main.cpp math_utils.cpp -o app
./app
```

*Kenneth J. Roche*
*University of Washington*

## *header files (.hpp , .h)*

- *header files shouldn't contain executable statements minus two exceptions*

```
inline int add(int a, int b)
{
    return a + b;
}
```

  - *1) **inline** shared class methods or functions*

    - *inline: compiler will (usually) replace inline function with actual code of the function at the point where it is called rather than function call and return instructions*

  - *2) shared templates*

    - *compiler needs access to template implementation to create type instances of it*
    - *generally put into the header file altogether*

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

```cpp
#include <iostream>
#include <vector>
#include <iomanip> // for std::setprecision

template <typename T>
std::vector<T> vector_addition(const std::vector<T> &vec1, const std::vector<T> &vec2)
{
    if (vec1.size() != vec2.size())
    {
        throw std::invalid_argument("Vector sizes must be equal for addition.");
    }

    std::vector<T> result;
    result.reserve(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
    {
        result.push_back(vec1[i] + vec2[i]);
    }

// KR: vector iterator - also used in other STLs such as list, deque, etc.
// for (auto it1 = vec1.begin(), it2 = vec2.begin(); it1 != vec1.end() && it2 != vec2.end(); ++it1, ++it2)
// {
//     result.push_back(*it1 + *it2);
// }

    return result;
}
```

```
bash-3.2$ g++ -std=c++17 -o xvadd1 vector-addition1.cpp
bash-3.2$ ./xvadd1
int vector addition:
5 7 9

float vector addition:
5.500000 7.700000 9.900000

double vector addition:
5.555000 7.777000 9.999000

bash-3.2$
```

```cpp
template <typename T>
void demo_vector_addition(const std::vector<T> &v1, const std::vector<T> &v2, const std::string &label)
{
    std::cout << label << " vector addition:\n";

    try
    {
        std::vector<T> result = vector_addition(v1, v2);
        for (const auto &val : result)
        {
            std::cout << std::fixed << std::setprecision(6) << val << " ";
        }
        std::cout << "\n\n";
    }
    catch (const std::invalid_argument &e)
    {
        std::cerr << "Error: " << e.what() << "\n\n";
    }
}

int main()
{
    std::vector<int> int_vec1{1, 2, 3};
    std::vector<int> int_vec2{4, 5, 6};

    std::vector<float> float_vec1{1.1f, 2.2f, 3.3f};
    std::vector<float> float_vec2{4.4f, 5.5f, 6.6f};

    std::vector<double> double_vec1{1.111, 2.222, 3.333};
    std::vector<double> double_vec2{4.444, 5.555, 6.666};

    demo_vector_addition(int_vec1, int_vec2, "int");
    demo_vector_addition(float_vec1, float_vec2, "float");
    demo_vector_addition(double_vec1, double_vec2, "double");

    return 0;
}
```

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

```cpp
#include <iostream>
#include <vector>
#include <iomanip> // for std::setprecision
//#include "vector_addition.hpp"

template <typename T>
void demo_vector_addition(const std::vector<T> &v1, const std::vector<T> &v2, const std::string &label)
{
    std::cout << label << " vector addition:\n";

    try
    {
        std::vector<T> result = vector_addition(v1, v2);
        for (const auto &val : result)
        {
            std::cout << std::fixed << std::setprecision(6) << val << " ";
        }
        std::cout << "\n\n";
    }
    catch (const std::invalid_argument &e)
    {
        std::cerr << "Error: " << e.what() << "\n\n";
    }
}

int main()
{
    std::vector<int> int_vec1{1, 2, 3};
    std::vector<int> int_vec2{4, 5, 6};

    std::vector<float> float_vec1{1.1f, 2.2f, 3.3f};
    std::vector<float> float_vec2{4.4f, 5.5f, 6.6f};

    std::vector<double> double_vec1{1.111, 2.222, 3.333};
    std::vector<double> double_vec2{4.444, 5.555, 6.666};

    demo_vector_addition(int_vec1, int_vec2, "int");
    demo_vector_addition(float_vec1, float_vec2, "float");
    demo_vector_addition(double_vec1, double_vec2, "double");

    return 0;
}
```

```
bash-3.2$ g++ -std=c++17 -o xvadd2 vector-addition2.cpp
vector-addition2.cpp:13:33: error: use of undeclared identifier 'vector_addition'
        std::vector<T> result = vector_addition(v1, v2);
                                ^
vector-addition2.cpp:37:5: note: in instantiation of function template specialization 'demo_vector_addition<int>' requested here
    demo_vector_addition(int_vec1, int_vec2, "int");
    ^
vector-addition2.cpp:13:33: error: use of undeclared identifier 'vector_addition'
        std::vector<T> result = vector_addition(v1, v2);
                                ^
vector-addition2.cpp:38:5: note: in instantiation of function template specialization 'demo_vector_addition<float>' requested here
    demo_vector_addition(float_vec1, float_vec2, "float");
    ^
vector-addition2.cpp:13:33: error: use of undeclared identifier 'vector_addition'
        std::vector<T> result = vector_addition(v1, v2);
                                ^
vector-addition2.cpp:39:5: note: in instantiation of function template specialization 'demo_vector_addition<double>' requested here
    demo_vector_addition(double_vec1, double_vec2, "double");
    ^
3 errors generated.
bash-3.2$
```

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

```cpp
#include <iostream>
#include <vector>
#include <iomanip> // for std::setprecision
#include "vector_addition.hpp"

template <typename T>
void demo_vector_addition(const std::vector<T> &v1, const std::vector<T> &v2, const std::string &label)
{
    std::cout << label << " vector addition:\n";

    try
    {
        std::vector<T> result = vector_addition(v1, v2);
        for (const auto &val : result)
        {
            std::cout << std::fixed << std::setprecision(6) << val << " ";
        }
        std::cout << "\n\n";
    }
    catch (const std::invalid_argument &e)
    {
        std::cerr << "Error: " << e.what() << "\n\n";
    }
}

int main()
{
    std::vector<int> int_vec1{1, 2, 3};
    std::vector<int> int_vec2{4, 5, 6};

    std::vector<float> float_vec1{1.1f, 2.2f, 3.3f};
    std::vector<float> float_vec2{4.4f, 5.5f, 6.6f};

    std::vector<double> double_vec1{1.111, 2.222, 3.333};
    std::vector<double> double_vec2{4.444, 5.555, 6.666};

    demo_vector_addition(int_vec1, int_vec2, "int");
    demo_vector_addition(float_vec1, float_vec2, "float");
    demo_vector_addition(double_vec1, double_vec2, "double");

    return 0;
}
```

```cpp
#ifndef VECTOR_ADDITION_HPP
#define VECTOR_ADDITION_HPP

#include <vector>

template <typename T>
std::vector<T> vector_addition(const std::vector<T> &vec1, const std::vector<T> &vec2)
{
    if (vec1.size() != vec2.size())
    {
        throw std::invalid_argument("Vector sizes must be equal for addition.");
    }

    std::vector<T> result;
    result.reserve(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
    {
        result.push_back(vec1[i] + vec2[i]);
    }

// for (auto it1 = vec1.begin(), it2 = vec2.begin(); it1 != vec1.end() && it2 != vec2.end(); ++it1, ++it2)
// {
//     result.push_back(*it1 + *it2);
// }

    return result;
}
#endif // VECTOR_ADDITION_HPP
```

```
bash-3.2$ g++ -std=c++17 -o xvadd2 -I. vector-addition2.cpp
bash-3.2$ ./xvadd2
int vector addition:
5 7 9

float vector addition:
5.500000 7.700000 9.900000

double vector addition:
5.555000 7.777000 9.999000

bash-3.2$
```

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

```
bash-3.2$ g++ -std=c++17 -o xvadd2 -I. vector-addition2.cpp
bash-3.2$ ./xvadd2
int vector addition:
5 7 9

float vector addition:
5.500000 7.700000 9.900000

double vector addition:
5.555000 7.777000 9.999000

bash-3.2$
```

```
bash-3.2$ mkdir include
bash-3.2$ mv *.hpp include/
bash-3.2$ g++ -std=c++17 -o xvadd2 -I. vector-addition2.cpp
vector-addition2.cpp:4:10: fatal error: 'vector_addition.hpp' file not found
#include "vector_addition.hpp"
         ^~~~~~~~~~~~~~~~~~~~~
1 error generated.
bash-3.2$ g++ -std=c++17 -o xvadd2 -I./include vector-addition2.cpp
bash-3.2$ ./xvadd2
int vector addition:
5 7 9

float vector addition:
5.500000 7.700000 9.900000

double vector addition:
5.555000 7.777000 9.999000

bash-3.2$ ls include/
vector_addition.hpp
bash-3.2$
```

```cpp
#include <iostream>
#include <vector>
// scale
template <typename T>
void scaleVector(std::vector<T> &vec, const T &scalar)
{
    for (auto &element : vec)
    {
        element *= scalar;
    }
}
// inner product
#include <numeric>
template <typename T>
T innerProduct(const std::vector<T> &vec1, const std::vector<T> &vec2)
{
    if (vec1.size() != vec2.size())
    {
        throw std::invalid_argument("Vectors must have the same size");
    }

    T result = 0;
    for (std::size_t i = 0; i < vec1.size(); ++i)
    {
        result += vec1[i] * vec2[i];
    }
    return result;
}
// normalize
#include <cmath>
template <typename T>
void normalizeVector(std::vector<T> &vec)
{
    T magnitude = 0;
    for (const auto &element : vec)
    {
        magnitude += element * element;
    }
    magnitude = std::sqrt(magnitude);

    for (auto &element : vec)
    {
        element /= magnitude;
    }
}
```

*single source file / compilation unit*

```cpp
int main(int argc, char *argv[])
{

    // test scale vector
    std::vector<int> v = {1, 2, 3, 4, 5};
    std::cout << "before scaling:" << std::endl;
    for (auto &element : v)
        std::cout << "v = " << element << std::endl;

    int scalar = 2;
    scaleVector(v, scalar);
    std::cout << "after scaling:" << std::endl;
    for (auto &element : v)
        std::cout << "v = " << element << std::endl;

    // test normalize vector
    std::vector<double> w = {3.0, 4.0};
    std::cout << "double vector before normalize:" << std::endl;
    for (auto &element : w)
        std::cout << "w = " << element << std::endl;
    normalizeVector(w);
    std::cout << "double vector after normalize:" << std::endl;
    for (auto &element : w)
        std::cout << "w = " << element << std::endl;

    // test inner product - double
    double dresult = innerProduct<double>(w, w);
    std::cout << "check length of normalized vector:" << std::sqrt(dresult) << std::endl;

    // test inner product - int
    std::vector<int> vec1 = {1, 2, 3};
    std::vector<int> vec2 = {4, 5, 6};
    int result = innerProduct(vec1, vec2);

    return 0;
}
```

*Compilation is a one-liner (but our nice functions are stuck in that source file)*

```
bash-3.2$ g++ -std=c++14 -o xlecture7 lecture7.cpp
bash-3.2$ ./xlecture7
before scaling:
v = 1
v = 2
v = 3
v = 4
v = 5
after scaling:
v = 2
v = 4
v = 6
v = 8
v = 10
double vector before normalize:
w = 3
w = 4
double vector after normalize:
w = 0.6
w = 0.8
check length of normalized vector:1
bash-3.2$
```

*mylibrary.hpp*

```
#ifndef MYLIBRARY_HPP
#define MYLIBRARY_HPP

#include "innerproduct.hpp"
#include "normalize.hpp"
#include "scale.hpp"

#endif // MYLIBRARY_HPP
```

```
#include <iostream>
#include <vector>
#include <cmath>
#include "mylibrary.hpp"
int main(int argc, char *argv[])
{

    // test scale vector
    std::vector<int> v = {1, 2, 3, 4, 5};
    std::cout << "before scaling:" << std::endl;
    for (auto &element : v)
        std::cout << "v = " << element << std::endl;

    int scalar = 2;
    scaleVector(v, scalar);
    std::cout << "after scaling:" << std::endl;
    for (auto &element : v)
        std::cout << "v = " << element << std::endl;

    // test normalize vector
    std::vector<double> w = {3.0, 4.0};
    std::cout << "double vector before normalize:" << std::endl;
    for (auto &element : w)
        std::cout << "w = " << element << std::endl;
    normalizeVector(w);
    std::cout << "double vector after normalize:" << std::endl;
    for (auto &element : w)
        std::cout << "w = " << element << std::endl;

    // test inner product - double
    double dresult = innerProduct<double>(w, w);
    std::cout << "check length of normalized vector:" << std::sqrt(dresult) << std::endl;

    // test inner product - int
    std::vector<int> vec1 = {1, 2, 3};
    std::vector<int> vec2 = {4, 5, 6};
    int result = innerProduct(vec1, vec2);

    return 0;
}
```

*let's work on better technique so that we can readily use our functions for various software efforts*

*Kenneth J. Roche*
*University of Washington*

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

*innerproduct.hpp*

```hpp
#ifndef INNERPRODUCT_HPP
#define INNERPRODUCT_HPP

#include <vector>

template<typename T>
T innerProduct(const std::vector<T>& vec1, const std::vector<T>& vec2);

#endif // INNERPRODUCT_HPP
```

*mylibrary.hpp*

```hpp
#ifndef MYLIBRARY_HPP
#define MYLIBRARY_HPP

#include "innerproduct.hpp"
#include "normalize.hpp"
#include "scale.hpp"

#endif // MYLIBRARY_HPP
```

*innerproduct.cpp*

```cpp
#include "innerproduct.hpp"

template<typename T>
T innerProduct(const std::vector<T> &vec1, const std::vector<T> &vec2)
{
    if (vec1.size() != vec2.size())
    {
        throw std::invalid_argument("Vectors must have the same size");
    }

    T result = 0;
    for (std::size_t i = 0; i < vec1.size(); ++i)
    {
        result += vec1[i] * vec2[i];
    }
    return result;
}

template int innerProduct(const std::vector<int>& vec1, const std::vector<int>& vec2);
template float innerProduct(const std::vector<float>& vec1, const std::vector<float>& vec2);
template double innerProduct(const std::vector<double>& vec1, const std::vector<double>& vec2);
```

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

*innerproduct.hpp*

```hpp
#ifndef INNERPRODUCT_HPP
#define INNERPRODUCT_HPP

#include <vector>

template<typename T>
T innerProduct(const std::vector<T>& vec1, const std::vector<T>& vec2);

#endif // INNERPRODUCT_HPP
```

*mylibrary.hpp*

```hpp
#ifndef MYLIBRARY_HPP
#define MYLIBRARY_HPP

#include "innerproduct.hpp"
#include "normalize.hpp"
#include "scale.hpp"

#endif // MYLIBRARY_HPP
```

*Note the explicit template instantiations at the end of the file*

*innerproduct.cpp*

```cpp
#include "innerproduct.hpp"

template<typename T>
T innerProduct(const std::vector<T> &vec1, const std::vector<T> &vec2)
{
    if (vec1.size() != vec2.size())
    {
        throw std::invalid_argument("Vectors must have the same size");
    }

    T result = 0;
    for (std::size_t i = 0; i < vec1.size(); ++i)
    {
        result += vec1[i] * vec2[i];
    }
    return result;
}

template int innerProduct(const std::vector<int>& vec1, const std::vector<int>& vec2);
template float innerProduct(const std::vector<float>& vec1, const std::vector<float>& vec2);
template double innerProduct(const std::vector<double>& vec1, const std::vector<double>& vec2);
```

*mylibrary.hpp*

```hpp
#ifndef MYLIBRARY_HPP
#define MYLIBRARY_HPP

#include "innerproduct.hpp"
#include "normalize.hpp"
#include "scale.hpp"

#endif // MYLIBRARY_HPP
```

*normalize.cpp*

```cpp
#include "normalize.hpp"

template<typename T>
void normalizeVector(std::vector<T> &vec)
{
    T magnitude = 0;
    for (const auto &element : vec)
    {
        magnitude += element * element;
    }
    magnitude = std::sqrt(magnitude);

    for (auto &element : vec)
    {
        element /= magnitude;
    }
}

template void normalizeVector(std::vector<float> &vec);
template void normalizeVector(std::vector<double> &vec);
```

*normalize.hpp*

```hpp
#ifndef NORMALIZE_HPP
#define NORMALIZE_HPP

#include <vector>
#include <cmath>

template <typename T>
void normalizeVector(std::vector<T> &vec);

#endif // NORMALIZE_HPP
```

*Kenneth J. Roche*
*University of Washington*

*mylibrary.hpp*

```cpp
#ifndef MYLIBRARY_HPP
#define MYLIBRARY_HPP

#include "innerproduct.hpp"
#include "normalize.hpp"
#include "scale.hpp"

#endif // MYLIBRARY_HPP
```

*scale.cpp*

```cpp
#include "scale.hpp"

template <typename T>
void scaleVector(std::vector<T> &vec, const T &scalar)
{
    for (auto &element : vec)
    {
        element *= scalar;
    }
}

template void scaleVector(std::vector<int> &vec, const int &scalar);
template void scaleVector(std::vector<float> &vec, const float &scalar);
template void scaleVector(std::vector<double> &vec, const double &scalar);
```

```cpp
#ifndef SCALE_HPP
#define SCALE_HPP

#include <vector>

template <typename T>
void scaleVector(std::vector<T> &vec, const T &scalar);

#endif // SCALE_HPP
```

*scale.hpp*

*Kenneth J. Roche*
*University of Washington*

*Compilation plan: combine functions into a relocatable shared object file or libary (PIC); link the library at compile time to resolve all symbol dependencies*

```
bash-3.2$ ls -lstr *.hpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  155 Apr 17 22:09 scale.hpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  144 Apr 17 22:20 mylibrary.hpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  207 Apr 18 00:18 innerproduct.hpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  187 Apr 18 00:18 normalize.hpp
bash-3.2$ ls -lstr *.cpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  2099 Apr 17 21:40 lecture7.cpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users  1272 Apr 17 22:00 lecture7a.cpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users   726 Apr 17 22:16 innerproduct.cpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users   490 Apr 17 22:17 normalize.cpp
8 -rw-r--r--@ 1 d3y402  PNL\Domain Users   449 Apr 17 22:18 scale.cpp
```

*Compilation plan: combine functions into a relocatable shared object file or libary (PIC); link the library at compile time to resolve all symbol dependencies*

```
bash-3.2$ cat l7.cmpl
rm *.o x* *.so;
date;
g++ -std=c++14 -c -fPIC normalize.cpp ;
g++ -std=c++14 -c -fPIC scale.cpp ;
g++ -std=c++14 -c -fPIC innerproduct.cpp
g++ -shared -o libmylibrary.so normalize.o scale.o innerproduct.o
echo "built shared object library";
ls -lstr libmylibrary.so;
g++ -std=c++14 -o xlecture7a -I. lecture7a.cpp -L. -lmylibrary
echo "built program binary";
ls -lstr xlecture7a;
```

```
bash-3.2$ source l7.cmpl
Tue Apr 18 00:59:20 PDT 2023
built shared object library
112 -rwxr-xr-x@ 1 d3y402  PNL\Domain Users  53832 Apr 18 00:59 libmylibrary.so
built program binary
144 -rwxr-xr-x@ 1 d3y402  PNL\Domain Users  70440 Apr 18 00:59 xlecture7a
```

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

*Compilation:*
*LD_LIBRARY_PATH* environment variable
(guidance for the runtime)

```
bash-3.2$ pwd
/Users/d3y402/Desktop/kr-code-training/cplusplus/lecture7
bash-3.2$ LD_LIBRARY_PATH=. ./xlecture7a
```

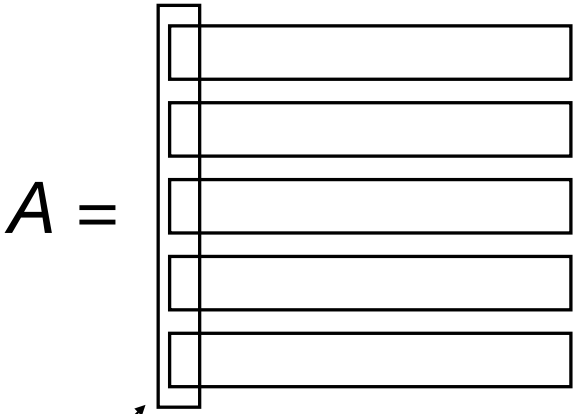
```
bash-3.2$ ./xlecture7a
before scaling:
v = 1
v = 2
v = 3
v = 4
v = 5
after scaling:
v = 2
v = 4
v = 6
v = 8
v = 10
double vector before normalize:
w = 3
w = 4
double vector after normalize:
w = 0.6
w = 0.8
check length of normalized vector:1
```

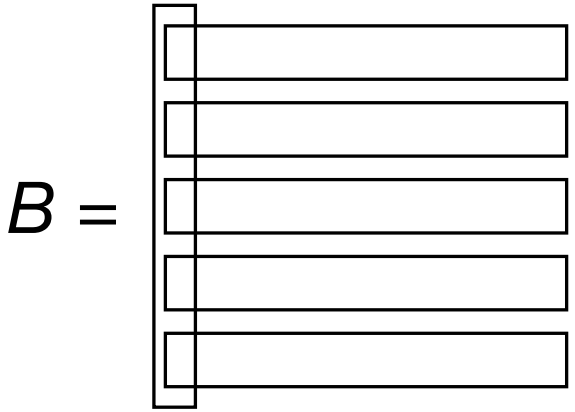
```
bash-3.2$ env | grep LD_LIBRARY_PATH
bash-3.2$ pwd
/Users/d3y402/Desktop/kr-code-training/cplusplus/lecture7
bash-3.2$ export LD_LIBRARY_PATH=/Users/d3y402/Desktop/kr-code-training/cplusplus/lecture7:${LD_LIBRARY_PATH}
bash-3.2$ env | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/Users/d3y402/Desktop/kr-code-training/cplusplus/lecture7:
```

# *matrix foo revisited ….*

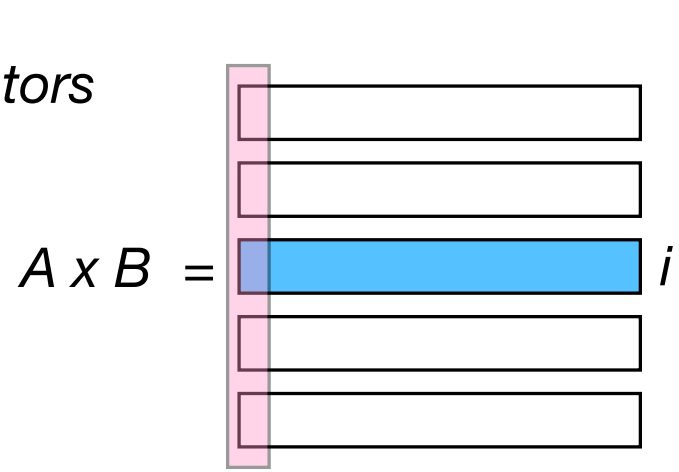
*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

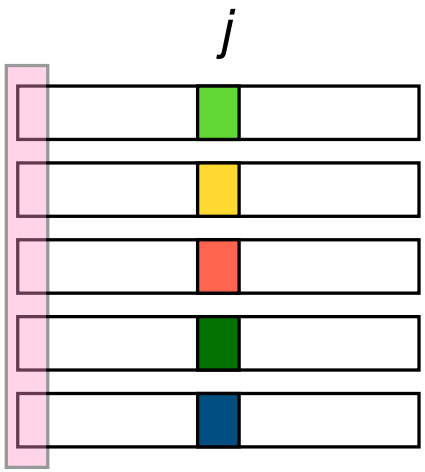
*matrix foo revisited ….*




*vector of vectors*




*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

# *mapping matrices column major format*

### 2.1.1 forward map

The column major index map $(i,j) \rightarrow k = i + j * m$ takes $\{a_{i,j}\}$ to $\{a_k\}$ for all combinations of $i \in [0, m-1], j \in [0, n-1]$, and assigns a specific order in memory based on the value of $k$, $k \in [0, mn-1]$. A simple example for the column major layout in memory for $m = 9, n = 11$ reads:

$$
\begin{aligned}
A &= \begin{pmatrix}
a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4} & a_{0,5} & a_{0,6} & a_{0,7} & a_{0,8} & a_{0,9} & a_{0,10} \\
a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} & a_{1,9} & a_{1,10} \\
a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} & a_{2,9} & a_{2,10} \\
a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} & a_{3,9} & a_{3,10} \\
a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & a_{4,9} & a_{4,10} \\
a_{5,0} & a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{5,10} \\
a_{6,0} & a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} & a_{6,9} & a_{6,10} \\
a_{7,0} & a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} & a_{7,9} & a_{7,10} \\
a_{8,0} & a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} & a_{8,9} & a_{8,10}
\end{pmatrix} \\
&= \begin{pmatrix}
a_0 & a_9 & a_{18} & a_{27} & a_{36} & a_{45} & a_{54} & a_{63} & a_{72} & a_{81} & a_{90} \\
a_1 & a_{10} & a_{19} & a_{28} & a_{37} & a_{46} & a_{55} & a_{64} & a_{73} & a_{82} & a_{91} \\
a_2 & a_{11} & a_{20} & a_{29} & a_{38} & a_{47} & a_{56} & a_{65} & a_{74} & a_{83} & a_{92} \\
a_3 & a_{12} & a_{21} & a_{30} & a_{39} & a_{48} & a_{57} & a_{66} & a_{75} & a_{84} & a_{93} \\
a_4 & a_{13} & a_{22} & a_{31} & a_{40} & a_{49} & a_{58} & a_{67} & a_{76} & a_{85} & a_{94} \\
a_5 & a_{14} & a_{23} & a_{32} & a_{41} & a_{50} & a_{59} & a_{68} & a_{77} & a_{86} & a_{95} \\
a_6 & a_{15} & a_{24} & a_{33} & a_{42} & a_{51} & a_{60} & a_{69} & a_{78} & a_{87} & a_{96} \\
a_7 & a_{16} & a_{25} & a_{34} & a_{43} & a_{52} & a_{61} & a_{70} & a_{79} & a_{88} & a_{97} \\
a_8 & a_{17} & a_{26} & a_{35} & a_{44} & a_{53} & a_{62} & a_{71} & a_{80} & a_{89} & a_{98}
\end{pmatrix} \\
&= \{a_0, a_1, a_2, \ldots, a_{97}, a_{98}\}.
\end{aligned}
$$

*Kenneth J. Roche*
*University of Washington*

*mapping matrices column major format*

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4} & a_{0,5} & a_{0,6} & a_{0,7} & a_{0,8} & a_{0,9} & a_{0,10} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} & a_{1,9} & a_{1,10} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} & a_{2,9} & a_{2,10} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} & a_{3,9} & a_{3,10} \\ a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & a_{4,9} & a_{4,10} \\ a_{5,0} & a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & a_{5,10} \\ a_{6,0} & a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} & a_{6,9} & a_{6,10} \\ a_{7,0} & a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} & a_{7,9} & a_{7,10} \\ a_{8,0} & a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} & a_{8,9} & a_{8,10} \end{pmatrix}$$

$$= \begin{pmatrix} a_0 & a_9 & a_{18} & a_{27} & a_{36} & a_{45} & a_{54} & a_{63} & a_{72} & a_{81} & a_{90} \\ a_1 & a_{10} & a_{19} & a_{28} & a_{37} & a_{46} & a_{55} & a_{64} & a_{73} & a_{82} & a_{91} \\ a_2 & a_{11} & a_{20} & a_{29} & a_{38} & a_{47} & a_{56} & a_{65} & a_{74} & a_{83} & a_{92} \\ a_3 & a_{12} & a_{21} & a_{30} & a_{39} & a_{48} & a_{57} & a_{66} & a_{75} & a_{84} & a_{93} \\ a_4 & a_{13} & a_{22} & a_{31} & a_{40} & a_{49} & a_{58} & a_{67} & a_{76} & a_{85} & a_{94} \\ a_5 & a_{14} & a_{23} & a_{32} & a_{41} & a_{50} & a_{59} & a_{68} & a_{77} & a_{86} & a_{95} \\ a_6 & a_{15} & a_{24} & a_{33} & a_{42} & a_{51} & a_{60} & a_{69} & a_{78} & a_{87} & a_{96} \\ a_7 & a_{16} & a_{25} & a_{34} & a_{43} & a_{52} & a_{61} & a_{70} & a_{79} & a_{88} & a_{97} \\ a_8 & a_{17} & a_{26} & a_{35} & a_{44} & a_{53} & a_{62} & a_{71} & a_{80} & a_{89} & a_{98} \end{pmatrix}$$

$$= \{a_0, a_1, a_2, \ldots, a_{97}, a_{98}\}.$$

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*

### 2.1.2 reverse map

To recover $i, j$ from $k$, let $(\forall k \in [0, mn-1])k \to (i = k \mod m, j = \frac{k-(k \mod m)}{m})$. **Ex.** Let $A \in \mathbb{C}^{3\times 4}$, then $k = 0 \to (i = 0 \mod 3 = 0, j = \frac{0-0}{3} = 0)$, $k = 7 \to (i = 7 \mod 3 = 1, j = \frac{7-1}{3} = 2)$, $k = 11 \to (i = 11 \mod 3 = 2, j = \frac{11-2}{3} = 3)$, etc.

*End Lecture 9*

*Kenneth J. Roche*
*University of Washington*
*AMATH 483 / 583 High Performance Scientific Computing*
*Spring Quarter 2025*