

Matrices and 2-D Arrays

Example

```
#include <iostream>
#include <iomanip>
using namespace std;

int main(){
    const int M = 5;
    const int N = 7;
    int A[M][N];
    int B[M][N] = { { 11, 12, 13, 14 },
                   { 21, 22, 23, 24 },
                   { 31, 32, 33, 34 } };

    int m = 3, n = 4;
    for (int i = 0; i < m; ++i){
        for (int j = 0; j < n; ++j)
            cout << setw(4) << B[i][j] << ' '; //See ALT.
        cout << endl;
    }
    return 0;
}

/* Output:
  11   12   13   14
  21   22   23   24
  31   32   33   34
*/
// ALT: cout << setw(4) << *((B+i)+j) << ' ';
```

Matrices and 2-D Arrays

- Just like a 1-D array, except that you need two indices to specify an element: $A[i][j]$ is the variable in “row i ” and “column j .”
- Filled from the “upper left” corner toward the “lower right” corner. E.g., $B[][]$ above is visualized as follows.

```
11  12  13  14  ??  ??  ??
21  22  23  24  ??  ??  ??
31  32  33  34  ??  ??  ??
??  ??  ??  ??  ??  ??  ??
??  ??  ??  ??  ??  ??  ??
```

- An “array of arrays,” stored **contiguously** by row in C++. E.g., $B[][7]$ above is stored as follows.

```
11 12 13 14 ?? ?? ?? 21 22 23 24 ?? ?? ?? 31 32 ...
```

- **Pointers to pointers.** Type $\text{int } [] [N]$ is the same as $\text{int } (*) [N]$ and is more specific than int^{**} . Given an array A defined as $\text{int } A[M] [N] \dots;$,

$A[i][j]$ means $*(A[i] + j)$ or $*(*(A+i)+j)$
or $A[0][i*N + j]$,

where N is the number of columns of storage in A .

$A[i][j]$ is the ij th element.

$A[i]$ is the address of $A[i][0]$.

A is the address of $A[0]$.

- When used as a function parameter, a 2-D array’s column dimension must be specified explicitly.
 $\text{int } [] [5]$ is not the same type as $\text{int } [] [7]$.

Example: Generate an $N \times N$ multiplication table and store it in a 2-D array.

```
#include <iostream>
#include <iomanip>
#include <cassert>
using namespace std;

const int N_ROWS_A = 20; // max number of rows of data
const int N_COLS_A = 20; // max number of columns of data

void makeMultTable( int A[][ N_COLS_A ], int N ){

    if ( N <= N_COLS_A && N <= N_ROWS_A)
        for (int i=0; i<N; ++i)
            for (int j=0; j<N; ++j)
                A[i][j] = (i+1)*(j+1);
    else
        cerr << "Error in makeMultTable(): table size "
              << "exceeds array dimensions.\n\n";
}
```

```

void printTable( int A[][ N_COLS_A ], int N,
                ostream& os=cout, int fieldWidth=5 ){

    if ( N <= N_COLS_A && N <= N_ROWS_A)
        for (int i=0; i<N; ++i){
            for (int j=0; j<N; ++j)
                os << setw(fieldWidth) << A[i][j] << ' ';
            os << endl;
        }
    else
        cerr << "Error in printTable(): table size "
              << "exceeds array dimensions.\n\n";
}

int main(){
    static int A[ N_ROWS_A ][ N_COLS_A ];
    cout << "Enter table size N (N <= "
          << N_COLS_A << "): ";
    int N;  cin >> N;
    assert (N > 0 && N <= 20);
    makeMultTable( A, N );
    printTable( A, N );
    return 0;
}

/* Sample I/O:
Enter table size N (N <= 20): 4
    1     2     3     4
    2     4     6     8
    3     6     9    12
    4     8    12    16
*/

```