INDEX ORDER

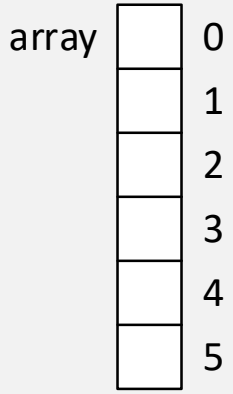
Does the order matter?

Delroy A. Brinkerhoff

ARRAY DEFINITION AND MEMORY ALLOCATION

```
int array[2][3];
int array[3][2];
```

rows × cols = cols × rows



TRADITION

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{bmatrix}$$

- Mathematics
  - The first index increases down columns
  - The second index increases along rows
- Other programming languages
  - FORTRAN: `real A(3,2)`
  - ALGOL: `REAL A[0:2,0:1]`
  - C++ continues the practice

INITIALIZER LIST ORDER

```
char array[3][2] = { 'A', 'B', 'C', 'D', 'E', 'F' };

for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 2; j++)
        cout << setw(2) << array[i][j];
    cout << endl;
}
```

```
A B
C D
E F
```

INITIALIZER LIST ORDER

```
char array[2][3] = { 'A', 'B', 'C', 'D', 'E', 'F' };

for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 2; j++)
        cout << setw(2) << array[i][j];
    cout << endl;
}
```

```
A B
D E
ä
```

INITIALIZER LIST ORDER

```
char array[3][2] = { 'A', 'B', 'C', 'D', 'E', 'F' };

for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 3; j++)
        cout << setw(2) << array[i][j];
    cout << endl;
}
```

```
A B C
C D E
```

INITIALIZER LIST ORDER

```
char array[2][3] = { 'A', 'B', 'C', 'D', 'E', 'F' };

for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 3; j++)
        cout << setw(2) << array[i][j];
    cout << endl;
}
```

```
A B C
D E F
```

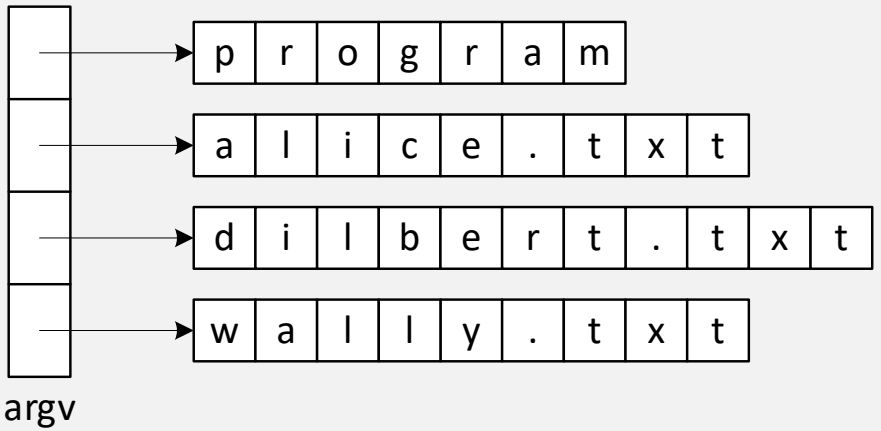
EXTRACTING ROWS

```
void print_row(char* row, int size)
{
    for (int i = 0; i < size; i++)
        cout << setw(2) << row[i];
}

char array[][3] = { 'A', 'B', ‘C’,
                    'D', 'E', 'F’,
                    'G', 'H', ‘I’,
                    'J', 'K', 'L’ };

print_row(array[2], sizeof(array[2]) / sizeof(char));
```

EXTRACTING ROWS

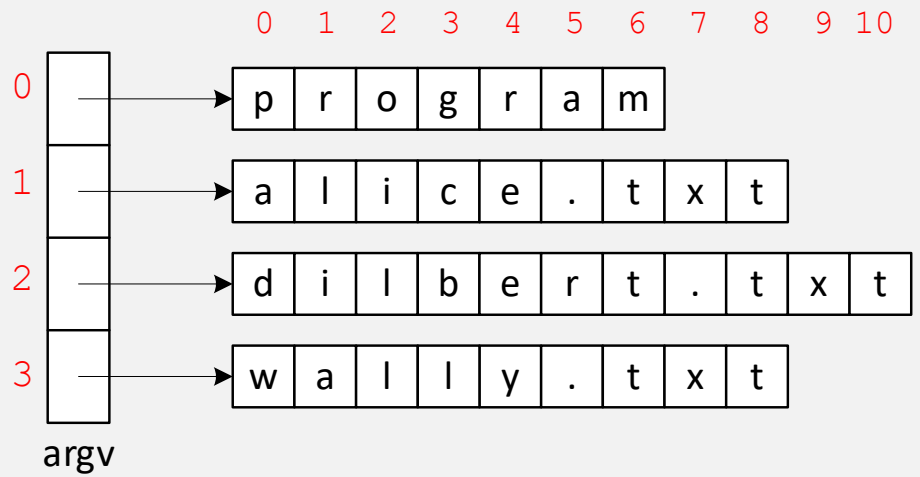
```
char array[][4] = {
    'A', 'B', 'C', '\0',
    'D', 'E', 'F', '\0',
    'G', 'H', 'I', '\0',
    'J', 'K', 'L', '\0'
};

cout << array[2] << endl;
```

COMMAND-LINE ARGUMENTS



- Command line arguments
  - `char* argv[]`
  - `char** argv`
  - Come from the operating system
  - Are an array of strings
  - Program access arguments with one index and characters with two: `[row][col]`

COMMAND-LINE ARGUMENTS

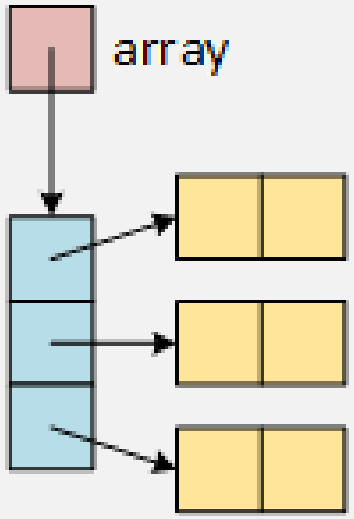


- Command line arguments
  - `char* argv[]`
  - `char** argv`
  - Come from the operating system
  - Are an array of strings
  - Program access arguments with one index and characters with two: `[row][col]`

- `argv[2][5] is 'r'`

CONSISTENCY WITH JAVA

PUBLIC INT[][] ARRAY = NEW INT[3][2];



PUBLIC INT[][] ARRAY = NEW INT[2][3];