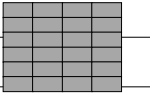


### Multi-dimensional Arrays



- **Have seen 1-dimensional arrays**
  - E.g., `double ar[10];` -- one subscript
- **What about higher-order arrays?**
  - Two-dimensional -- two subscripts
    - Often known as a matrix
  - n-dimensional -- n subscripts
- **Does C++ provide them?**
  - Yes and no

---

---

---

---

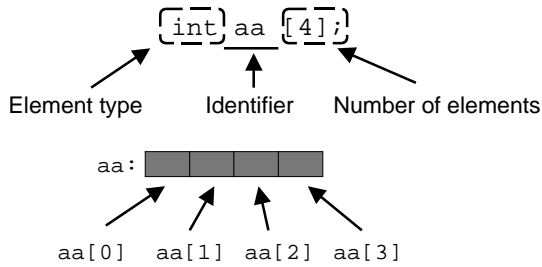
---

---

---

---

### Array Declaration (1-D)




---

---

---

---

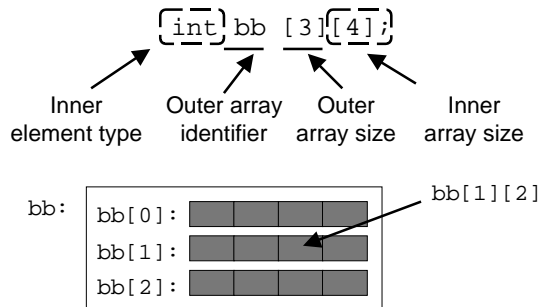
---

---

---

---

### Array Declaration (2-D)




---

---

---

---

---

---

---

---

What About This?

```
int cc [3,4];
```

Number of elements in outer array, inner array?

Unfortunately, this is not permitted in C/C++, primarily because the comma operator (',') means "sequential evaluation". Even for class types, we can't overload operator[] with more than one argument.

The Bottom Line

- **No C++ multi-dimensional arrays!**
  - Only arrays of arrays
- **Arrays of arrays not very flexible**
  - Must declare sizes at compile time
  - Same problems as simple arrays
    - Can't assign, return, etc.
- **An alternative would be nice**

Matrix Notation

Matrix	Matrix element
$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$	$a_{i,j}$ <p>i = row subscript j = column subscript</p>

m = number of rows  
n = number of columns

Note: assumes 1-origin matrix subscripts

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---



Element Access Function

```
double& Matrix::Elem (unsigned int r,
                     unsigned int c)
{
  assert ((r > 0) && (r <= nrows));
  assert ((c > 0) && (c <= ncols));

  return array[(r-1)*ncols+(c-1)];
}
```

Returns a reference to a matrix element that can be used to inspect or modify its value.

---

---

---

---

---

---

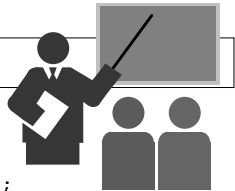
---

---

---

---

Matrix Class Usage



```
Matrix bb(3,4);
Matrix dd(5,5);

bb.Elem(1,2) = 1.5;
dd.Elem(2,2) = bb.Elem(2,4);
```

“Elem” member function permits element access. But this is pretty clumsy! Will the following alternative work?

```
bb[1,2] = 1.5;
dd[2,2] = bb[2,4];
```

---

---

---

---

---

---

---

---

---

---

Overloading the () Operator

```
class Matrix
{
public:
  ...
  double& operator()(unsigned int r,
                    unsigned int c);
  ...
};
```

Operator() is called the “function call” operator. Can be overloaded with any number of arguments. Example: “b(1,2) = 1.5;”

---

---

---

---

---

---

---

---

---

---

More Matrix Operations

- **Change size**
  - Subscript origins?
- **Math operations**
  - Addition
    - +, +=
    - Matrix/scalar?
  - Subtraction
  - Multiplication
  - Inversion
- **Set elements**
  - Single value
  - Identity matrix
  - Transpose
- **Class operations**
  - Copy constructor
  - Destructor
  - Assignment
- **Anything else?**

---

---

---

---

---

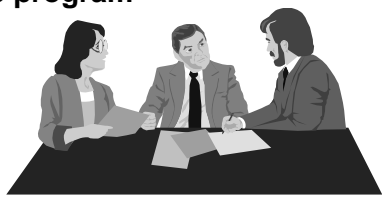
---

---

---

Class Design Exercise

- **Complete Matrix definition**
- **Member function implementation**
- **Example program**




---

---

---

---

---

---

---

---