CHAPTER 2

ARRAYS

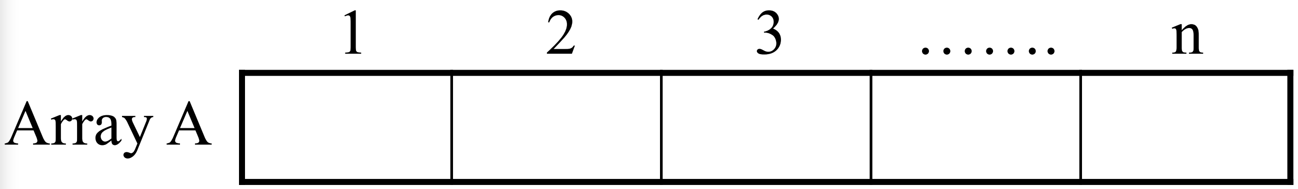
All the programs in this file are selected from

Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed
“Fundamentals of Data Structures in C”,
Computer Science Press, 1992.

Arrays

Definition:

(1) A set of index and value
(2) Using consecutive memory
(3) Support "Random Access " and "Sequential Access"
(4) Insert/Delete an element: O(n)



Arrays in C

int list[5], *plist[5];

list[5]: five integers
list[0], list[1], list[2], list[3], list[4]
*plist[5]: five pointers to integers
plist[0], plist[1], plist[2], plist[3], plist[4]

implementation of 1-D array

list[0]	base address = l_0
list[1]	l_0+ sizeof(int)
list[2]	l_0+ 2*sizeof(int)
list[3]	l_0+ 3*sizeof(int)
list[4]	l_0+ 4*size(int)

Arrays in C *(Continued)*

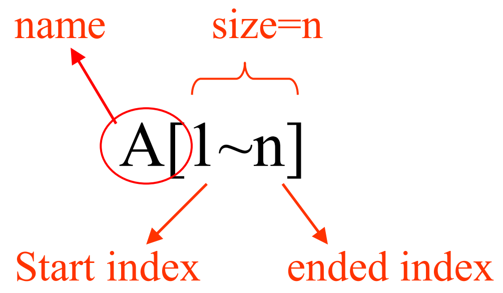
Compare int *list1 and int list2[5] in C.

Same: list1 and list2 are pointers.
Difference: list2 reserves five locations.

Notations:
list2 - a pointer to list2[0]
(list2 + i) - a pointer to list2[i]=(&list2[i])

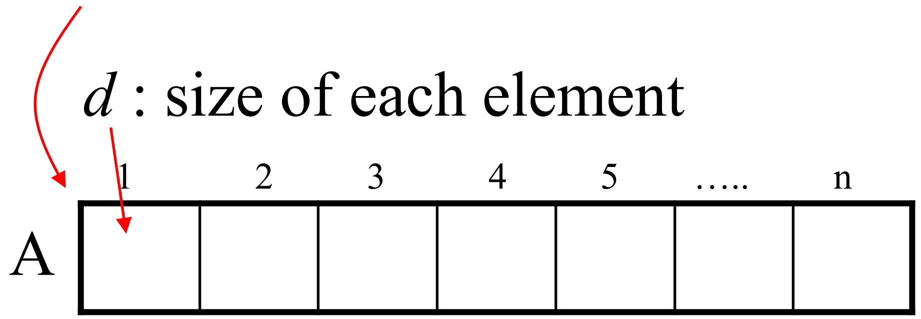
Addressing(1-Dimension Arrays)

Declare:



suppose l_0 : start address

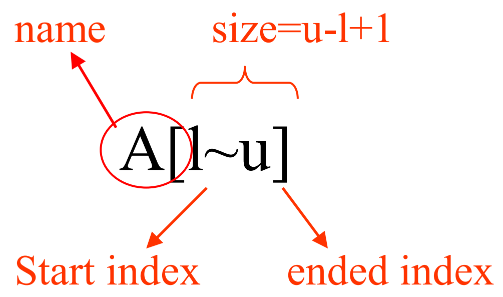
d : size of each element



Then $\boxed{A[i] = l_0 + (i-1) \times d}$

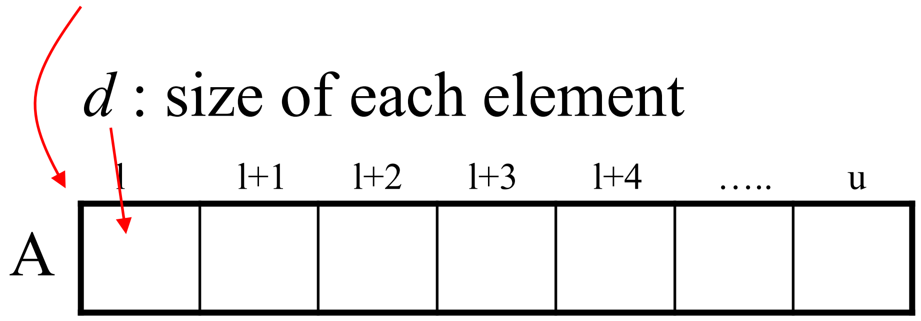
Addressing(1-Dimension Arrays) (General case)

Declare :



suppose l_0 : start address

d : size of each element



Then $\boxed{A[i] = l_0 + (i - l) \times d}$

Addressing(2-Dimension Arrays)

Declare:

$$\underset{\text{name}}{A}\,[\overbrace{1\sim m,\ 1\sim n}^{\text{m row, n column, size=m*n}}]$$

suppose l_0 : start address

d : size of each element

Row-Major: $\boxed{A[i,j] = l_0 + ((i-1)\times n + (j-1))\times d}$

Column-Major: $\boxed{A[i,j] = l_0 + ((j-1)\times m + (i-1))\times d}$

Addressing(2-Dimension Arrays) (General case)

Declare:

name $\quad$ (u_1-l_1+1) row, (u_2-l_2+1) column

$$\bigcirc\!\!\!\!\!A\,[\underbrace{l_1\sim u_1,\ l_2\sim u_2}]$$

suppose l_0 : start address

d : size of each element

Row-Major: $\boxed{A[i,j]=l_0+((i-l_1)\times(u_2-l_2+1)+(j-l_2))\times d}$

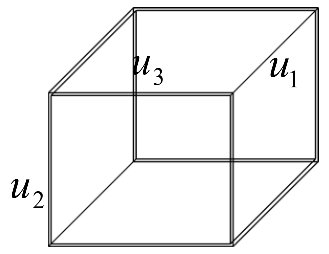
Column-Major:

$$\boxed{A[i,j]=l_0+((j-l_2)\times(u_1-l_1+1)+(i-l_1))\times d}$$

Addressing(3-Dimension Arrays)

Declare:

A[1~u1, 1~u2, 1-u3]



suppose l_0 : start address

d : size of each element

Row-Major:

$$\boxed{A[i,j,k] = l_0 + (((i-1)\times u_2 \times u_3) + ((j-1)\times u_3) + (k-1))\times d}$$

Column-Major:

$$\boxed{A[i,j,k] = l_0 + (((k-1)\times u_2 \times u_1) + ((j-1)\times u_1) + (i-1))\times d}$$

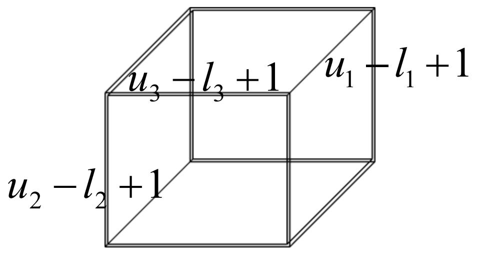
Addressing(3-Dimension Arrays) (General case)

Declare:

$$A[l_1 \sim u_1,\ l_2 \sim u_2,\ l_3 - u_3]$$

Row-major →

← column-major



suppose l_0 : start address
d : size of each element

Row-Major:

$$\boxed{\begin{aligned} A[i,j,k] = l_0 + ((i-l_1)\times(u_2-l_2+1)\times(u_3-l_3+1) \\ +(j-l_2)\times(u_3-l_3+1)+(k-l_3))\times d \end{aligned}}$$

Column-Major:

$$\boxed{\begin{aligned} A[i,j,k] = l_0 + ((k-l_3)\times(u_2-l_2+1)\times(u_1-l_1+1) \\ +(j-l_2)\times(u_1-l_1+1)+(i-l_1))\times d \end{aligned}}$$

Addressing(n-Dimension Arrays)

Declare:

$$\xrightarrow{\text{Row-major}}$$

A[1~u₁, 1~u₂,… 1-uₙ]

suppose l_0 : start address

d : size of each element

Row-Major: $A[a_1, a_2 \ldots a_n] = l_0 + (((a_1 - 1) \times u_2 \times u_3 \times \ldots \times u_n) +$

$$((a_2 - 1) \times u_3 \times u_4 \times \ldots \times u_n) +$$

$$((a_3 - 1) \times u_4 \times u_5 \times \ldots \times u_n) +$$

$$\vdots$$

$$(a_n - 1)) \times d$$

Addressing(n-Dimension Arrays)

Declare:

A[1~u1, 1~u2,… 1-un]

$\xleftarrow{\hspace{12em}}$ column-major

suppose l_0 : start address

d : size of each element

Column-Major: $$A[a_1, a_2 \ldots a_n] = l_0 + (((a_n - 1) \times u_{n-1} \times u_{n-2} \times \ldots \times u_1) + ((a_{n-1} - 1) \times u_{n-2} \times u_{n-3} \times \ldots \times u_1) + ((a_{n-2} - 1) \times u_{n-3} \times u_{n-4} \times \ldots \times u_1) + \vdots (a_1 - 1)) \times d$$

Ex)

A[-4~3, -3~2], l_0=100, d=1

(a) 若為 Row-major，則 A[1, 1]=?

(b) 若為 Column-major，則 A[1, 1]=?

中山資工87)

A[-2~7, -4~10, -2~1, -3~2, 1~10], l_0=38, d=8

(a) 若為 Row-major，則 A[0, 8, 0, 1, 8]=?

輔大資管90)

Suppose there is an integer array M[5, 8], the address of M[0, 0], M[0, 3], and M[1, 2] are 100, 106, and 120 respectively. What is the address of M[4, 5]

交大資科90)

Assume in a byte machine. A is an array declared as A[-1~m, 2~n], and each element occupied 3 bytes. The address of A[3, 5] is at 180, and A[5, 3] is at 138

(1) Find the address of the element A[-1, 2]

(2) Find the value of m or n

雲科大資工90)

有一個二維陣列A，假設A[1, 1]的位址是644，而A[3, 3]的位址是676，請問A(14, 14)的位址為何？

Summary:

二維陣列中，若給兩個已知的位址，則若是row-major 則代表要求l_0與***n***，若是column-major則代表要求l_0與***m***
若給三個已知位址則代表可能需求***d***

Ex)

A[3, 3]=121, A[6, 4]=159, d=1, 求A[4, 5]=?

Ex)

A[1, 1]=2, A[2, 3]=18, A[3, 2]=28, 求A[4, 5]=?

Sparse Martix

$$\begin{array}{c|ccc} & \text{col 1} & \text{col 2} & \text{col 3} \\ \hline \text{row 1} & -27 & 3 & 4 \\ \text{row 2} & 6 & 82 & -2 \\ \text{row 3} & 109 & -64 & 11 \\ \text{row 4} & 12 & 8 & 9 \\ \text{row 5} & 48 & 27 & 47 \end{array}$$

5*3

$$\begin{array}{c|cccccc} & \text{col1} & \text{col2} & \text{col3} & \text{col4} & \text{col5} & \text{col6} \\ \hline \text{row1} & 15 & 0 & 0 & 22 & 0 & -15 \\ \text{row2} & 0 & 11 & 3 & 0 & 0 & 0 \\ \text{row3} & 0 & 0 & 0 & -6 & 0 & 0 \\ \text{row4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{row5} & 91 & 0 & 0 & 0 & 0 & 0 \\ \text{row6} & 0 & 0 & 28 & 0 & 0 & 0 \end{array}$$

6*6

(a) 15/15

(b) 8/36

↑

sparse matrix
data structure?

Sparse Martix

Represent:

(1) Represented by a two-dimensional array
  → Sparse matrix wastes space
(2) 3-Tuple (Triples)
  → Each element is characterized by <row, col, value>
(3) Double Link-list
  →We will discuss it in chapter 4

Triples

# of rows = m

# of columns = n

# of nonzero terms = k

	row	col	value
[0]	6	6	8
[1]	1	1	15
[2]	1	4	22
[3]	1	6	-15
[4]	2	2	11
[5]	2	3	3
[6]	3	4	-6
[7]	5	1	91
[8]	6	3	28

Row-major

$$\begin{array}{c|cccccc|} & \text{col1} & \text{col2} & \text{col3} & \text{col4} & \text{col5} & \text{col6} \\ \text{row1} & 15 & 0 & 0 & 22 & 0 & -15 \\ \text{row2} & 0 & 11 & 3 & 0 & 0 & 0 \\ \text{row3} & 0 & 0 & 0 & -6 & 0 & 0 \\ \text{row4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{row5} & 91 & 0 & 0 & 0 & 0 & 0 \\ \text{row6} & 0 & 0 & 28 & 0 & 0 & 0 \end{array}$$

Transposing a Matrix

$$A_{m\times n} \rightarrow A^t_{n\times m} \qquad A_{ij} \rightarrow A^t_{ji}$$

Ex)

$$\begin{bmatrix} 0 & 7 \\ 1 & 8 \\ 3 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 3 \\ 7 & 8 & 2 \end{bmatrix}$$

Transposing a Matrix (Algorithm 1)

It's not row-major

```
For i=0 to k
  put A[i, j, value] into At[j, i, value]
End
```

	row	col	value
[0]	6	6	8
[1]	1	1	15
[2]	4	1	22
[3]	6	1	-15
[4]	2	2	11
[5]	3	2	3
[6]	4	3	-6
[7]	1	5	91
[8]	3	6	28

Transposing a Matrix (Algorithm 2)

```
For j=1 to n do
  For i=1 to k do
    if (A[i].col=j) then
    put A[i, j, value] into A^t[j, i, value]
  End
End
```

Time Complexity: $O(n \times k)$
where $k \approx m \times n$

→ when it's not "sparse" anymore….

	row	col	value
[0]	6	6	8
[1]	1	1	15
[2]	1	5	91
[3]	2	2	11
[4]	3	2	3
[5]	3	6	28
[6]	4	1	22
[7]	4	3	-6
[8]	6	1	-15

Compared with matrix transport using 2-D array

```
For i=1 to m do
  For j=1 to n do
    put A[i, j] into $A^t$[j, i]
  End
End
```

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

Time Complexity: $O(m \times n)$

Transposing a Matrix (Algorithm 3)

Step1:

```
For i=1 to k do
  column_element[A[i].col]++
End
```

Step2:

```
For i=2 to n do
  row_start[i]=row_start[i-1]+column_element[i-1]
End
```

Step3:

```
For i=1 to k do
  A^t[row_start[A[i].col]]=A[i]
  row_start[A[i].col]++
End
```

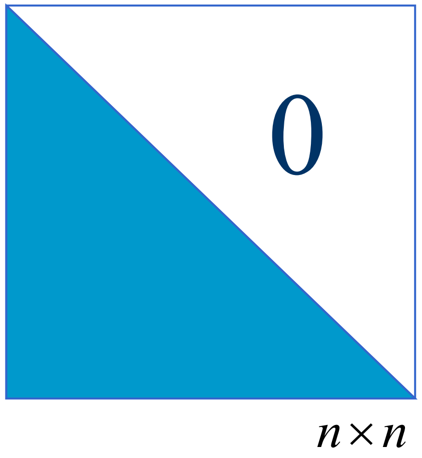
Time Complexity: O(k+n) or O(Max(k, n))

Lower Triangular Matrix (row-major)

Store:

```
count=1
For i=1 to n do
  For j=1 to i do
    B[count]=A[i, j]
    count++
  End
End
```

$A_{n \times n}$, 其中 $A_{ij}=0$ if $i<j$



Access:

```
if (i<j) then return 0
else
    $k = \frac{i(i-1)}{2} + j$
return B[k]
```

	1	2	3	4	5	…..	n(n+1)/2
B	(1,1)	(2,1)	(2,2)	(3,1)	(3,2)		(n,n)

Lower Triangular Matrix (column-major)

$A_{n \times n}$, 其中 $A_{ij}=0$ if $i<j$

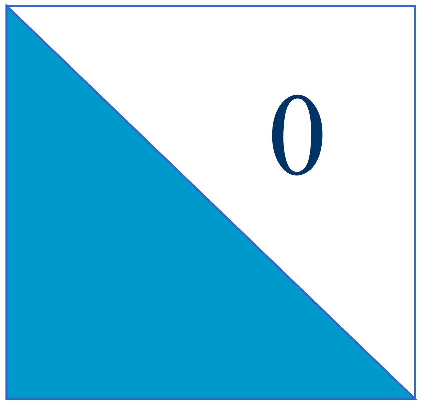
Access:

$$k=(n+(n-1)+\ldots(n-(j-1)+1))+(i-j+1)$$

$$=\frac{(2n-j+2)(j-1)}{2}+(i-j+1)$$

$$=\frac{(2n-j)(j-1)}{2}+i$$

$$=n(j-1)-\frac{j(j-1)}{2}+i$$



	1	2	3	4	5	…..	n(n+1)/2
B	(1,1)	(2,1)	(3,1)	(4,1)			(n,n)

Upper Triangular Matrix

Row-major:

$$k = n(i-1) - \frac{i(i-1)}{2} + j$$

Column-major:

$$k = \frac{(j-1)\times j}{2} + i$$

Symmetric Matrix

Definition:

$A_{n \times n}$, 其中 $A_{ij} = A_{ji}$

(1) Square Matrix

(2) Band Matrix $A_{n,a,b}$