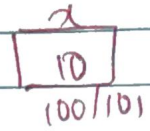
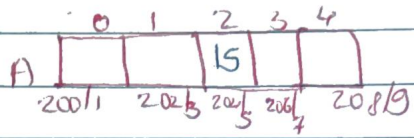
* Section 6: Arrays Representation

79. Introduction to Array

Scalar → int x = 10;



Vector → int A[5];
A[2] = 15;

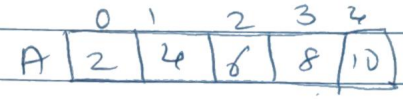


80. Declarations of Array

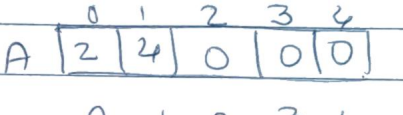
i) int A[5];

garbage method

ii) int A[5] = {2,4,6,8,10};

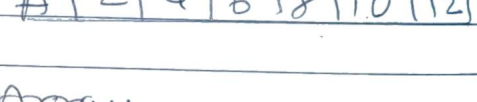


iii) int A[5] = {2,4};



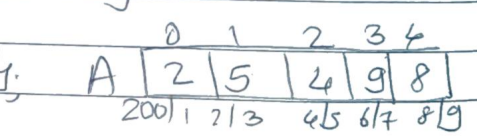
iv) int A[5] = {0};

v) int A[] = {2,4,6,8,10,12};



* Accessing an elements in Array

int A[5] = {2,5,4,9,8};



traversing in array

```
for (i=0; i<5; i++) {
        printf("%d", A[i]);
}
```

for traversing

```
printf("%d", A[2]);
printf("%d", 2[A]);
printf("%d", *(A+2));
```

for printing

82. Static vs Dynamic Array

```
void main(){
    int A[5];
    int n;
    cin>>n;
    int B[n];

    int *p;
    p=new int [5];
    p=(int*)malloc(5*
                sizeof(int));
    A[0]=5;
    delete [] p;
    free(p);
```

~~only~~ for C

for C++

for C++

for C

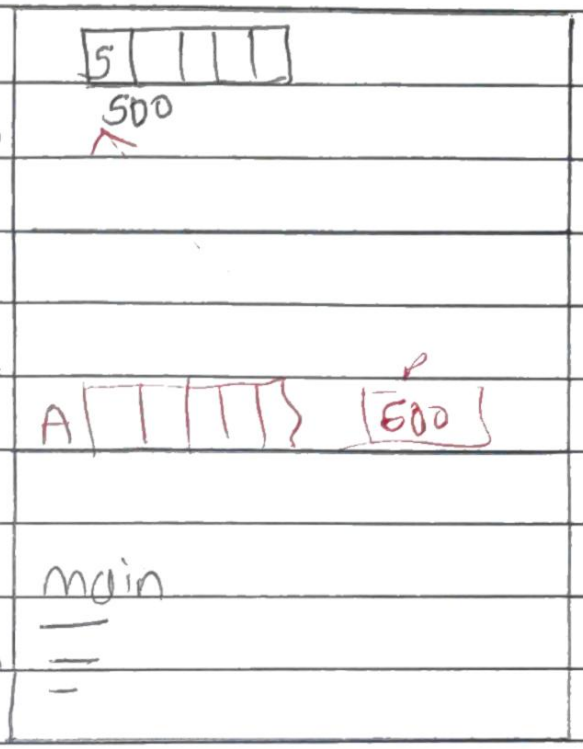
for C++

for C

Heap

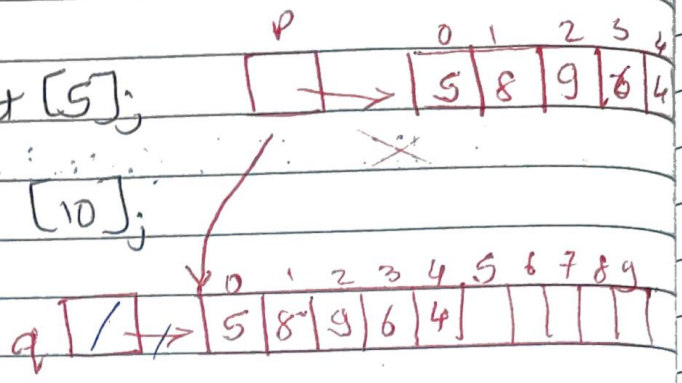
Stack

code Section



84. How to increase Array size

```
int *p = new int [5];
int *q = new int [10];
```



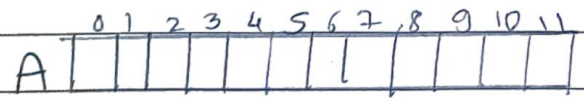

```
for ( i=0; i<5; i++)
    q[i] = p[i];
delete [] p;
p = q;
q = NULL
```

86. 2D Arrays

1) int A[3][4] = {{1,2,3,4}, {2,4,6,8}, {3,5,7,9}};
3 x 4 ||| Everything in stack

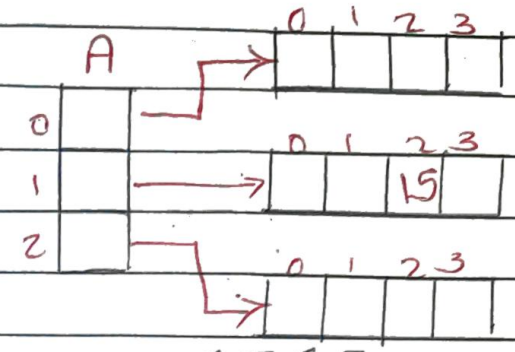
	0	1	2	3
A0	200 1	2 3	4 5	6 7
1	8 9		15	
3				

A[1][2] = 15;



2) int *A[3]; // partially in stack and heap

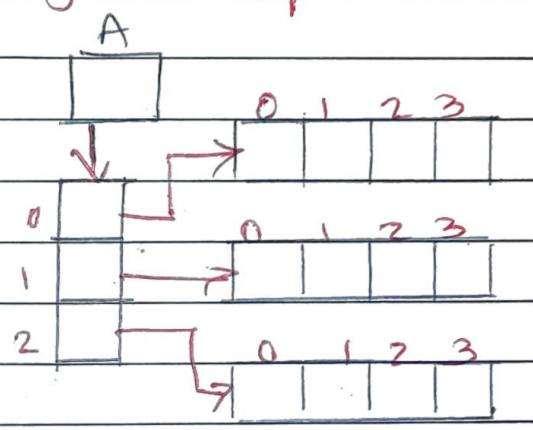
A[0] = new int [4];
A[1] = new int [4];
A[2] = new int [4];



A[1][2]=15;

3) int ** A; // Everything in Heap

A = new int *[3];
A[0] = new int [4];
A[1] = new int [4];
A[2] = new int [4];



* Accessing 2D Array

For changing row → for (i=0; i<3; i++) {
    for (j=0; j<4; j++) { } for accessing elements of a row
        A[i][j] = ...
    }
}

88. Array representation by compiler

1) int A[5] = {3, 5, 8, 4, 2};

	0	1	2	3	4
A	3	5	8	4	2
	200/1	2/3	4/5	6/7	8/9

L_0

base address

index

A[3] = 10;

sizeof

Add(A[3]) = 200 + 3 * 2 = 206

Add (A[3]) = L_0 + 3 * 2

* Relative Formula

$\boxed{\text{Add}(A[i]) = L_0 + i * w}$ → size of data type

base address index Faster

2) int A[1...5]

	1	2	3	4	5
A	3	5	8	4	2
	200/1	2/3	4/5	6/7	8/9

L_0

A[3] = 10;

Addr (A[3]) = 200 + (3-1) * 2 = 204

$\boxed{\text{Addr}(A[i]) = L_0 + (i-1) * w}$ → slower

89. Row Major Formula for 2D Array

int A[3][4];

m n

	0	1	2	3
A 0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

	0	1	2	3	4	5	6	7	8	9	10	11
A	a_{00}	a_{01}	a_{02}	a_{03}	a_{10}	a_{11}	a_{12}	a_{13}	a_{20}	a_{21}	a_{22}	a_{23}
	200/1	2/3	4/5	6/7	8	10	212	14	16	18	20	222/23

L_0

row 0 | row 1 | row 2

A[i][2] = 10;

Add(A[i][2]) = 200 + [1×4+2] * 2

= 200 + 6 * 2 = 212

Add (A[2][3] = 200 + [2×4+3] * 2

= 222

$$\boxed{Add(A[i][j]) = L_0 + [i * n + j] * w}$$

Faster

int A[1...3][1...4]

$$\boxed{Addr(A[i][j]) = L_0 + [(i-1) * n + (j-1)] * w}$$

Slower

90 Column Major Formula for 2D Arrays

int A[3][4];
m × n

	0	1	2	3	4	5	6	7	8	9	10	11
A	a_{00}	a_{10}	a_{20}	a_{01}	a_{11}	a_{21}	a_{02}	a_{12}	a_{22}	a_{03}	a_{13}	a_{23}
	200/1	2	4	6	20 8/9	10	12	14	216	18	20	222/23
Lo	Col 0			Col 1			Col 2			Col 3		

Addr(A[1][2]) = 200 + [2*3 + 1] * 2 = 214

Addr(A[1][3]) = 200 + [3*3 + 1] * 2 = 220

$$\boxed{Addr(A[i][j]) = Lo + [j * m + i] * w}$$

91. Formulas for nD Arrays

Type A[d1][d2][d3][d4];
For 4d Array

Row-major

$$\boxed{Addr(A[i_1][i_2][i_3][i_4]) = Lo + [i_1 * d_2 * d_3 * d_4 + i_2 * d_3 * d_4 + i_3 * d_4 + i_4] * w}$$

For n dimension array

$$\boxed{= Lo + \left[\sum_{p=1}^{n} i_p * \prod_{q=p+1}^{n} d_q\right] * w}$$

nD = n-1 + ... + 3 + 2 + 1

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

4D = 3+2+1

5D = 4+3+2+1

$= O(n^2)$

Column-major;

$$Addr(A[i_1][i_2][i_3][i_4]) = L_0 + [i_4 * d_1 * d_2 * d_3 + i_3 * d_1 * d_2 + i_2 * d_1 + i_1] * w$$

~~For 4 dimension array~~

~~$= d_0 +$~~

Row major Optimized
Horner's Rule

$$= L_0 + [i_4 + i_3 * d_4 + i_2 * d_3 * d_4 + i_1 * d_2 * d_3 * d_4] * w$$

$$= L_0 + [i_4 + d_4 * [i_3 + i_2 * d_3 + i_1 * d_2 * d_3]] * w$$

$$= L_0 + [i_4 + d_4 * [i_3 + d_3 * [i_2 + i_1 * d_2]] * w$$

4D → 3
5D → 4
nD → n-1 → O(n)

92. Formulas for 3D Arrays

int A[l][m][n];

Row major

Addr(A[i][j][k]) =
Lo + [i * m * n + j * n + k] * w

Column major

Addr(A[i][j][k]) =
Lo + [k * l * m + j * l + i] * w

93. Solutions for Quiz 2

1) Question 1

A[1...10][1...15] as Integer
m × n

Lo = 100
w = 1
Add(A[i][j]) = Lo + (i-1) * n + (j-1)
= 100 + (i-1) * 15 + j-1
= 100 + 15i - 15 + j-1
= 15i + j + 84

2) Question 2

int x[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}}
size of (int) = 4

x+3 → 2036
*(x+3) → 2036
*(x+2)+3 → 2036

X	0	1	2
2000			
0	1 (0)	2 (4)	3 (8)
1	4 (12)	5 (16)	6 (20)
2	7 (24)	8 (28)	9 (32)
3	10 (36)	11 (40)	12 (44)

1 12

3) Question 3
A and B are matrices
for A×B which mapping effect is efficient

Row-major → Lo + i*n + j

Col-major → Lo + j*m + i

Independent Representation

4) Question 4
3D Array ? x[?][?][?]
int 2 bytes
float 4 bytes

$t_0 = i * 1024$
$t_1 = j * 32$
$t_2 = k * 4$
$t_3 = t_0 + t_1$
$t_4 = t_2 + t_3$
$t_5 = x[t_4]$

type x[l][m][n]

Add (x[i][j][k])

$= L_0 + [i*m*n + j*n + k]*w$

$= L_0 + i*m*n*w + j*n*w + k*w$

$w = 4$ $\quad n = 8$ $\quad m = 32$

$n*w = 32$ $\quad m*n*w = 1024$

$n = \frac{32}{4} = 8$ $\quad m = \frac{1024}{32} = 32$

float x[?][32][8]