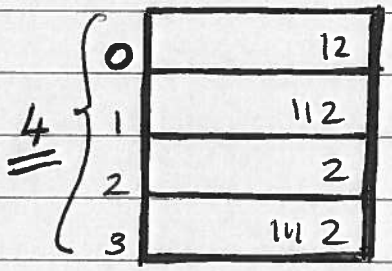
W6

CH.11

ARRAYS

■ EX: • Array of integers



• element type: int

• array size: 4

• Declaration: int intArray [4]

↑ Counting starts with 0

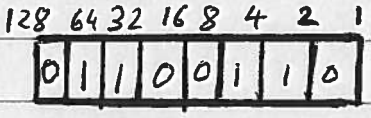
• Initialization:

```
/* Initialize array elements */
for (i=0; i<4; i++)
{ intArray[i] = 0;
}
```

↑ SELECTION of i^{th} element

■ Data Representation

- BIT: value is 0 or 1
- BYTE = 8 bits (can encode all characters)



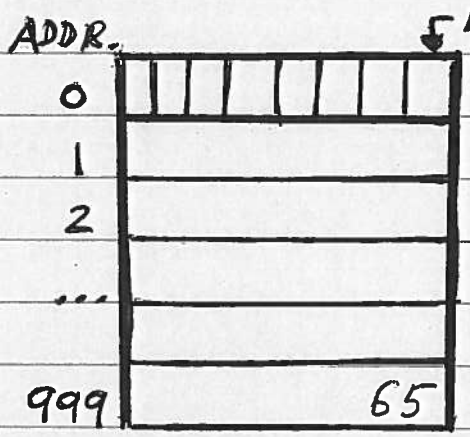
= 2+4+32+64 = 102

➡ $2^8 = 256$ bit combinations per byte

W6

- **WORD:** usually 2 or 4 bytes

- **MEMORY ADDR.:** refers to specific byte's address in computer memory

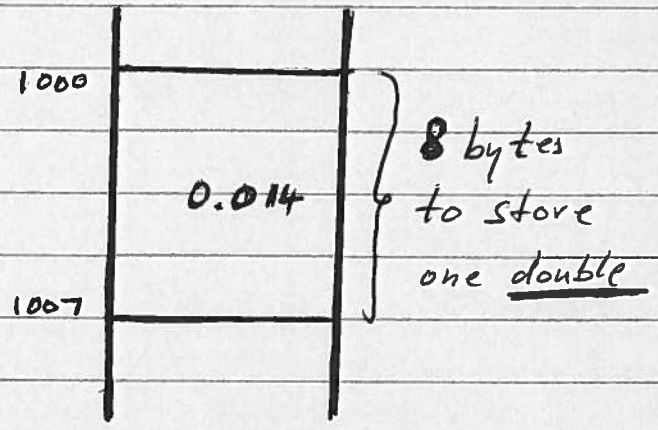


for example:

```
char ch;
ch = 'A'   (≙ 65)
```

- **DOUBLE:** requires more than 1 byte to store
  → stored in consecutive memory bytes

```
double d;
d = 0.014;
```

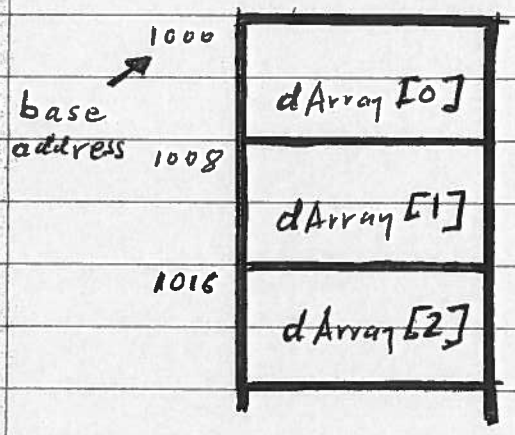


- `sizeof(int)`
  `sizeof(double)`
  `sizeof x`

  → returns no. of bytes used to store int, double, variable x

Examples

- char chArray[20];
  → 20 * sizeof(char) bytes provided
- double dArray[20];
  → 20 * sizeof(double) bytes provided



! DO NOT SELECT

dArray[-1]
dArray[21]

Arrays used as parameters

```
#define MaxNo 10

int intA[MaxNo];   /* allocated size is MaxNo */
                   /* effective size is size  */
                   /* actually being used     */
```

- EX: `void PrintIntArray(int array[MaxNo], int n);`

OR:

`void PrintIntArray(int array[], int n);`

BEHAVIOR LIKE GLOBAL VARIABLES:

CHANGING THE VALUE OF AN ARRAY ELEMENT IN A FUNCTION CHANGES THE VALUE OF THIS ELEMENT IN THE CALLING FUNCTION!

EX: "reverse.c" – Read integers, reverse order, print integers in reversed order

A[0]	2
A[1]	7
A[2]	8
A[3]	17

→ Print 17, 8, 7, 2

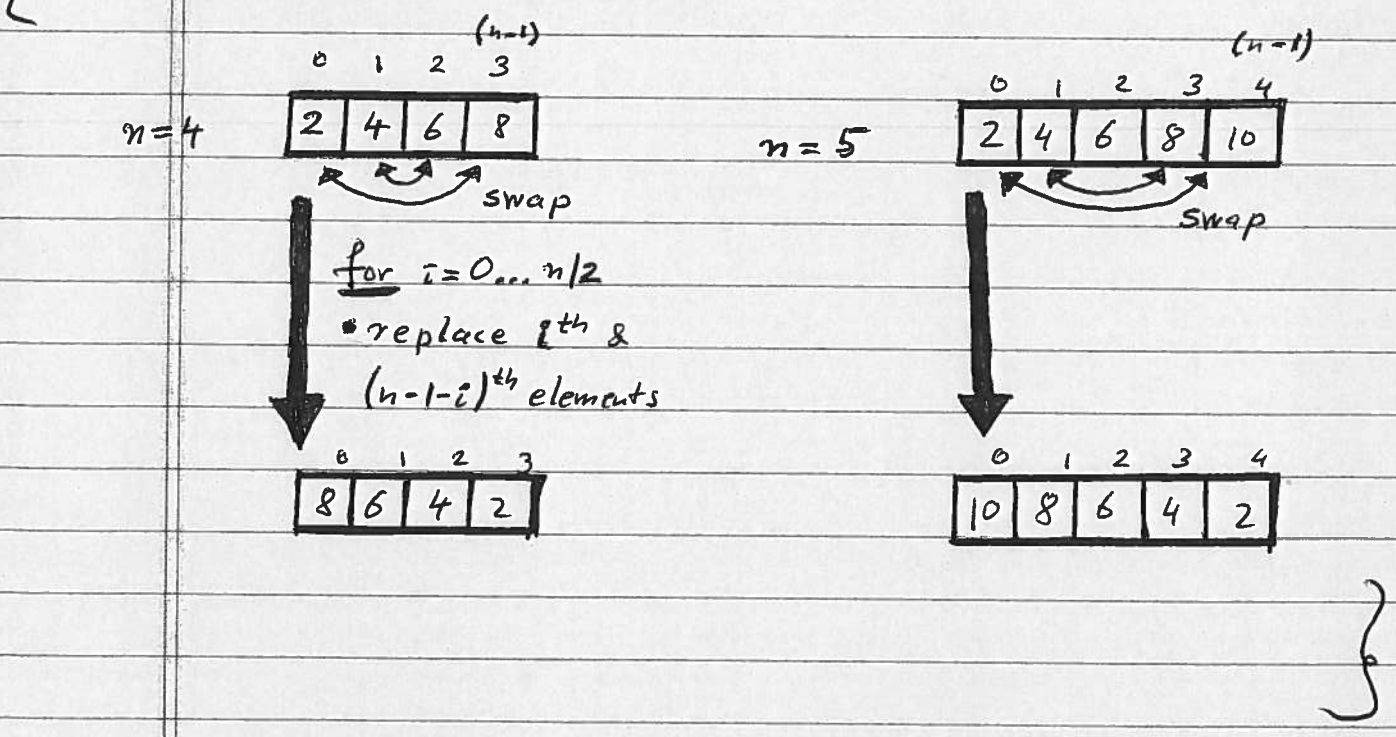
when printing from A[0] to A[3]

```
① main ()
{ int list[MaxNo], n;

  n = ② GetIntArray (list, MaxNo, end_sym);
  ③ ReverseIntArray (list, n);
  ④ PrintIntArray (list, n);
}
```

(2)
```
static int GetIntArray (int array[], int MaxNo, int end_sym)
{ int i, value;
  i = 0;
  while (TRUE)
  { printf ("Input no:");
    value = GetInteger ();
    if (value == end_sym) break;
    if (i == MaxNo) Error ("Out of bounds");
    array [i] = value;
    i++;
  }
  return (i);
}
```

→ REVERSING / SWAPPING STRATEGY:



③
```
static void ReverseIntArray (int array[], int n)
{ int i;
  for (i=0; i<n/2; i++)
  {
    /* SWAP ELEMENTS i AND (n-1-i) */
    ⑤ SwapIntElems (array, i, n-1-i);
  }
}
```

⑤
```
static void SwapIntElems (int array[],
                          int i1, int i2)
{ int tmp;

  tmp = array[i1];
  array[i1] = array[i2];
  array[i2] = tmp;
}
```

④
```
static void PrintIntArray (int array[], int n)
{int i;
 for (i=0; i<n; i++)
 {printf ("%d \n", array[i]);
 }
}
```

■ STATIC INITIALIZATION

• EX: `static int digit[10] = {0,1,2,3,4,5,6,7,8,9,};`

OR:

`static int digit[] = {0,1,2,3,4,5,6,7,8,9,};`

↑ size determined automatically

```
static string bigCity []
        = { "New York",
            "Los Angeles",
            "Chicago",
          };
```

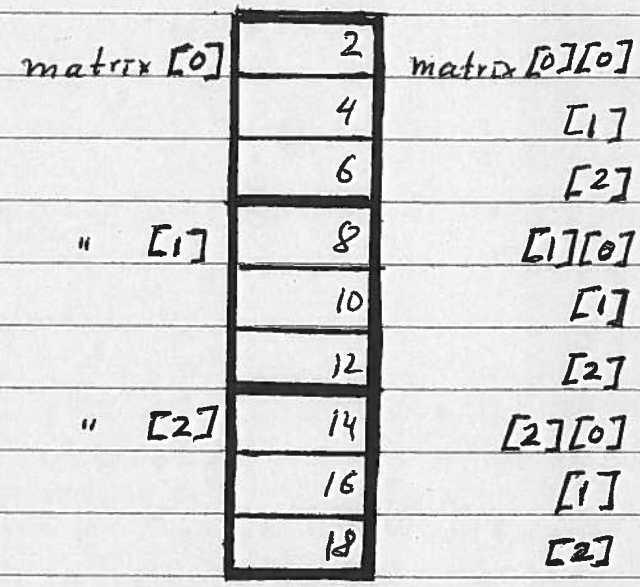
■ MULTIDIMENSIONAL ARRAYS

• EX: 2-dim. array — "matrix"

ROW \ COL	0	1	2
0	2	4	6
1	8	10	12
2	14	16	18

`int matrix[3][3];`

memory:



→ Passing this 2-dim. array as parameter:

```
static void PrintMatrix (int matrix[3][3])
                          /* OR: matrix [][3] */
                          /* only 1st dimension does */
                          /* not need to be specified. */

{ int row, col;
  for (row=0; row<3; row++)
  {
    for (col=0; col<3; col++)
      { printf ("  %d", matrix[row][col]);
      }
    printf("\n");
  }
}
```

→ Static Initialization:

```
static int mat[3][3]
        ={  {2,4,6},
            {8,10,12},
            {14,16,18}
        };
```