

# Pointer (1A)

---

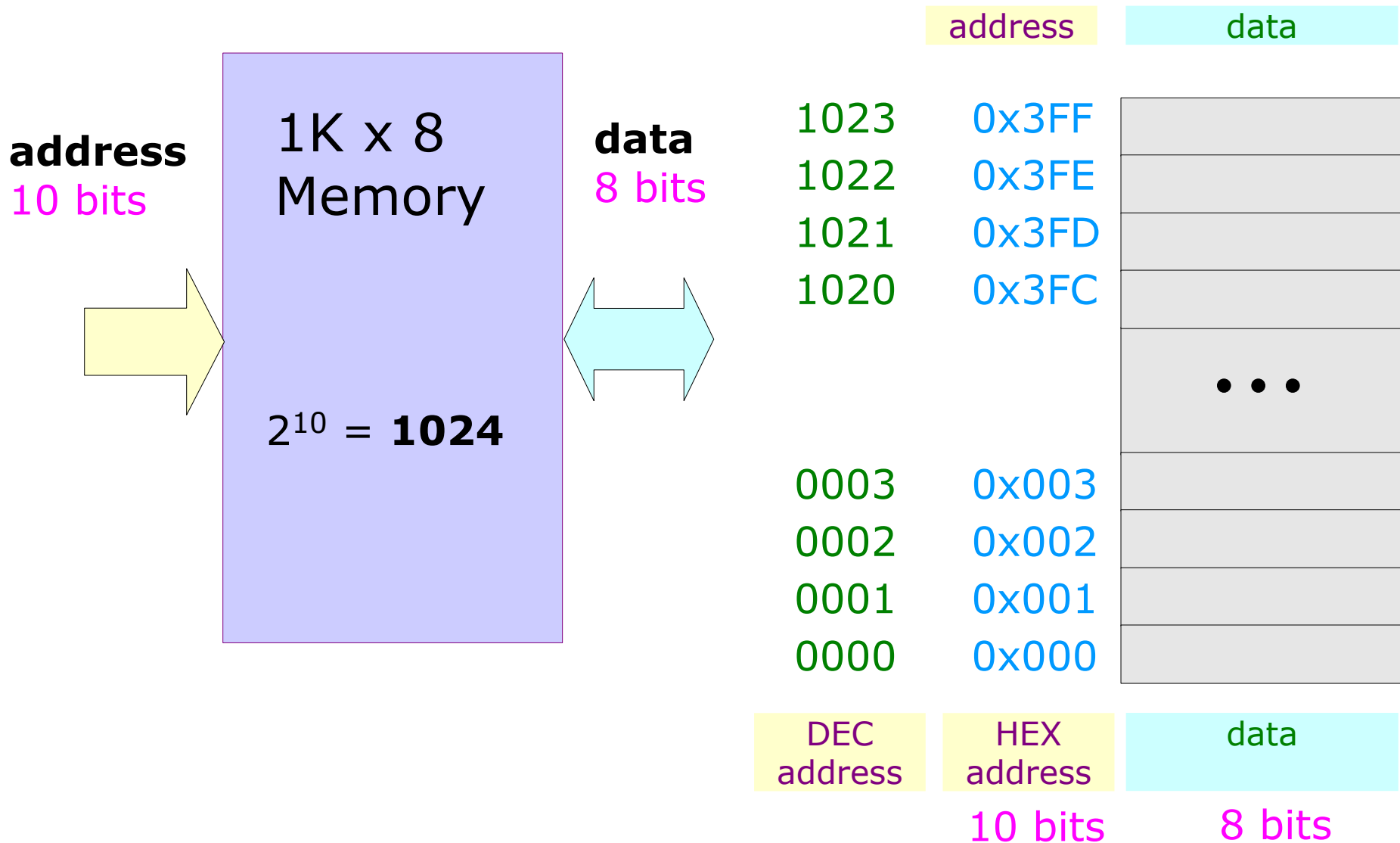
Copyright (c) 2010 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice.

# Address and Data in a Memory



# Variable

```
int a;
```

a can hold an *integer*

address

data

&a

a

```
a = 100;
```

a holds an *integer* 100

address

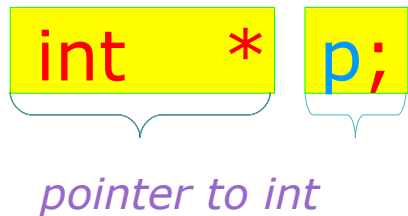
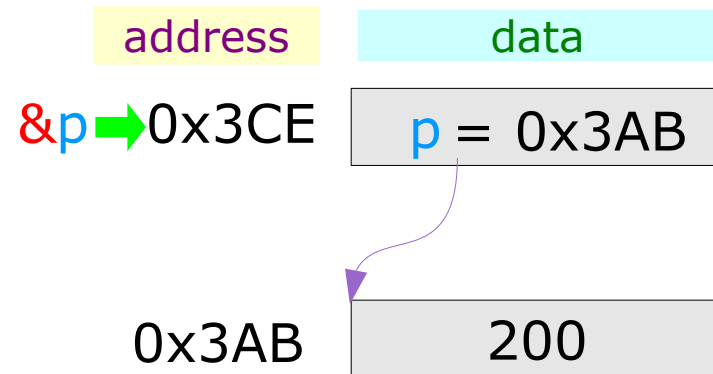
data

&a

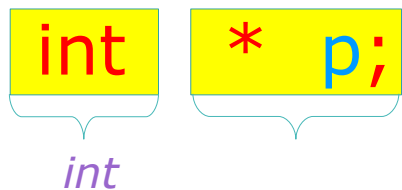
a ← 100

# Pointer Variable

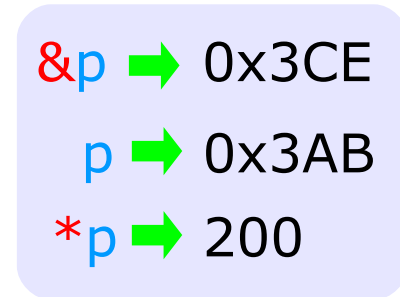
```
int * p;  
  
p can hold an address
```



p holds an address of Int type.



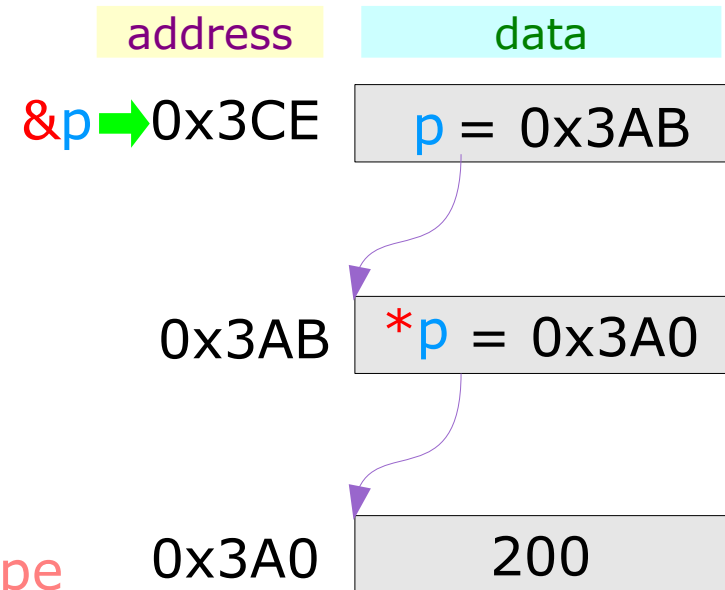
\*p holds an integer



# Pointer to Pointer Variable

```
int ** p;
```

p can hold an address



```
int ** p;
```

pointer to  
pointer to int

p holds an address  
of **Pointer to Int** type

```
int * *p;
```

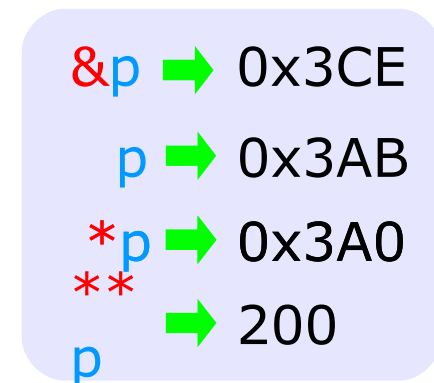
pointer to int

\*p holds an address  
of **Int** type

```
int **p;
```

int

\*\*p holds an integer



# Integer Pointer Examples (1)

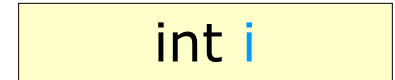
```
int    i;  
int *  pi;  
int ** qi;
```

**i** holds an integers

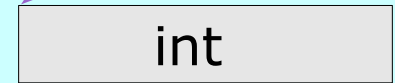
**pi** holds an address  
of **int** type

**qi** holds an address  
of **Pointer to int** type

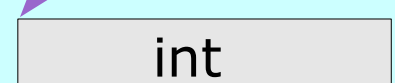
int type



int \* type



int \*\* type



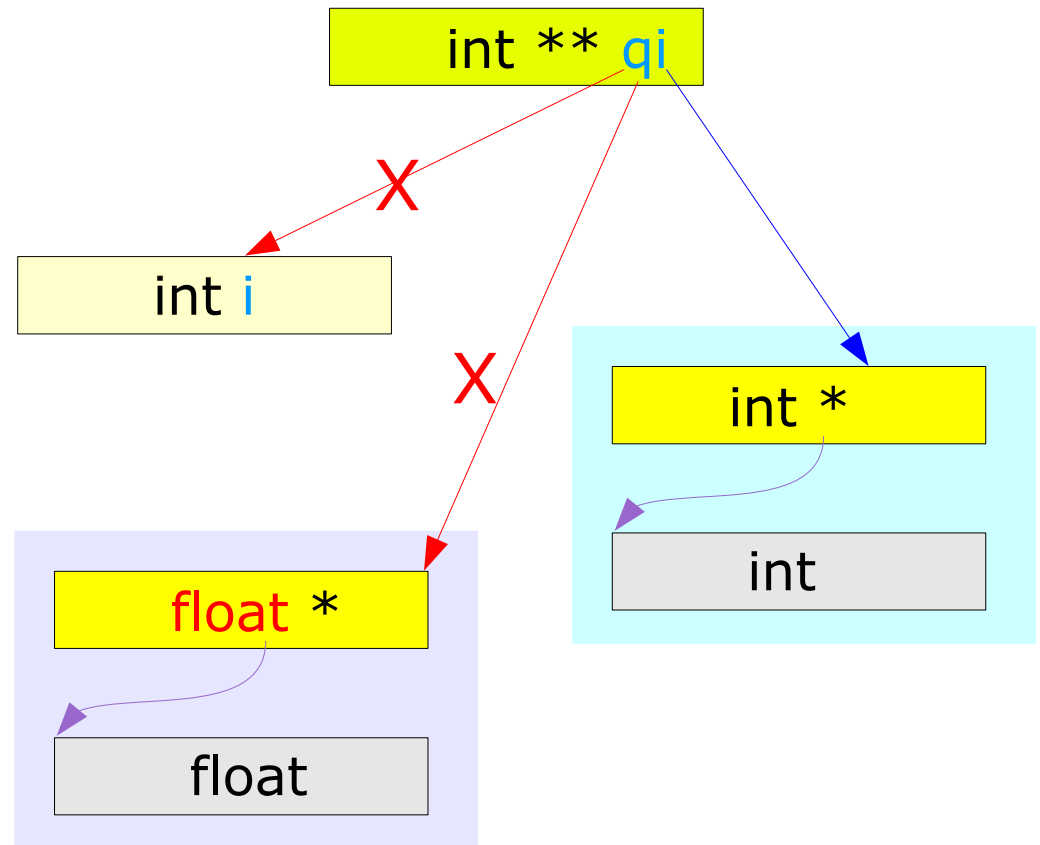
# Integer Pointer Examples (2)

```
int    i;  
int *  pi;  
int ** qi;
```

**i** holds an integers

**pi** holds an address  
of **int** type

**qi** holds an address  
of **Pointer to int** type





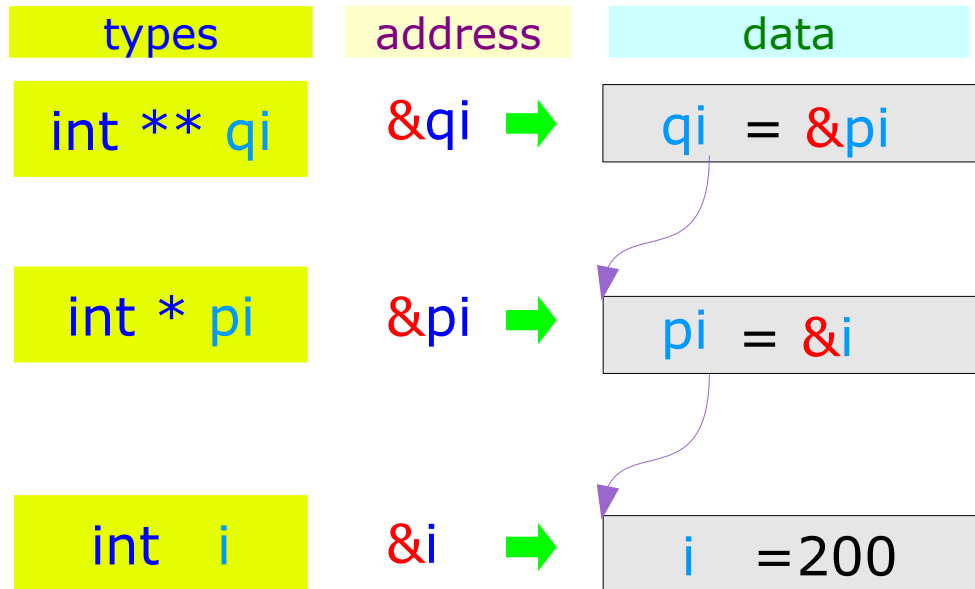
# Integer Pointer Examples (3)

```
int    i = 200;
int *  pi = &i;
int ** qi = &pi;
```

**i** holds an integers

**pi** holds an address  
of **int** type

**qi** holds an address  
of **Pointer to int** type



`*qi = pi`

`*pi = i`

`**qi = *pi = i`

# Array of Pointers (1)

```
int    a [4];
```

```
int *  b [4];
```

Array name **a** holds the starting address

**int** **a** **[4]**

*No. of elements = 4*

*Type of each element*

Array name **b** holds the starting address

**int \*** **a** **[4]**

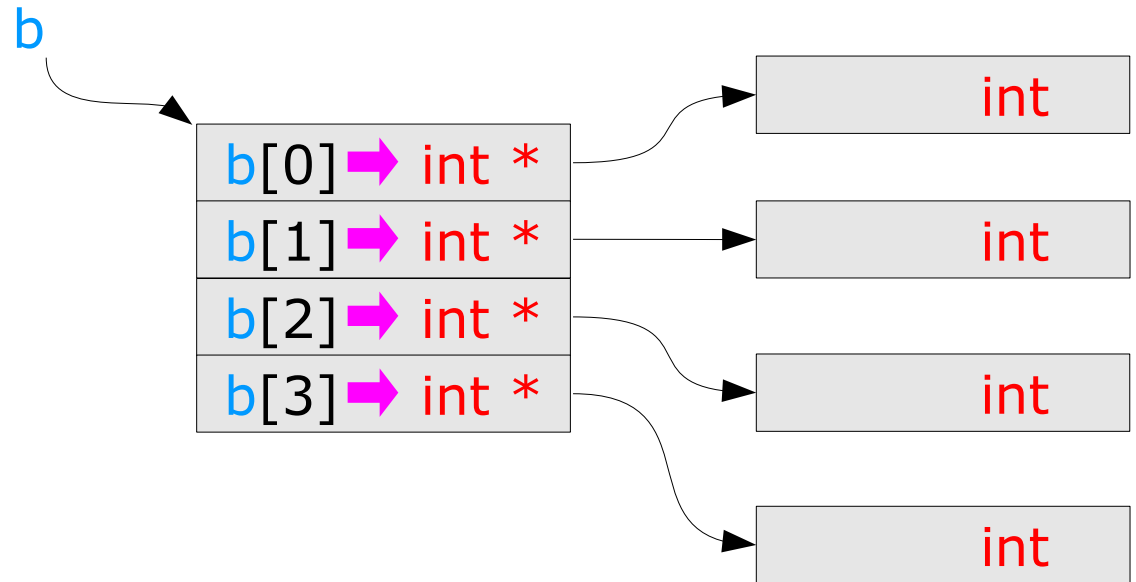
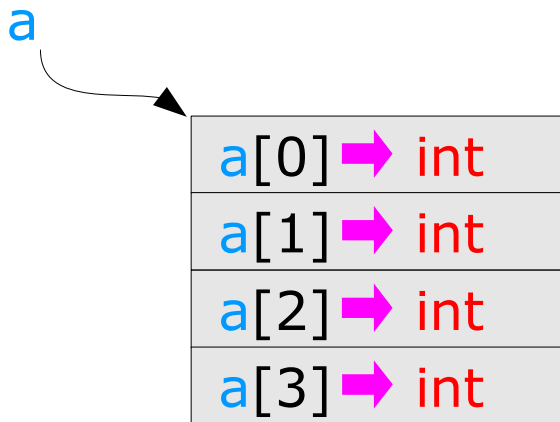
*No. of elements = 4*

*Type of each element*

# Array of Pointers (2)

```
int    a [4];
```

```
int *  b [4];
```



# 2-D Array (1)

```
int    a [4];  
int    c [4] [4];
```

Array name **a** holds the starting address

**int** **a** **[4]**

*No. of elements = 4*

*Type of each element*

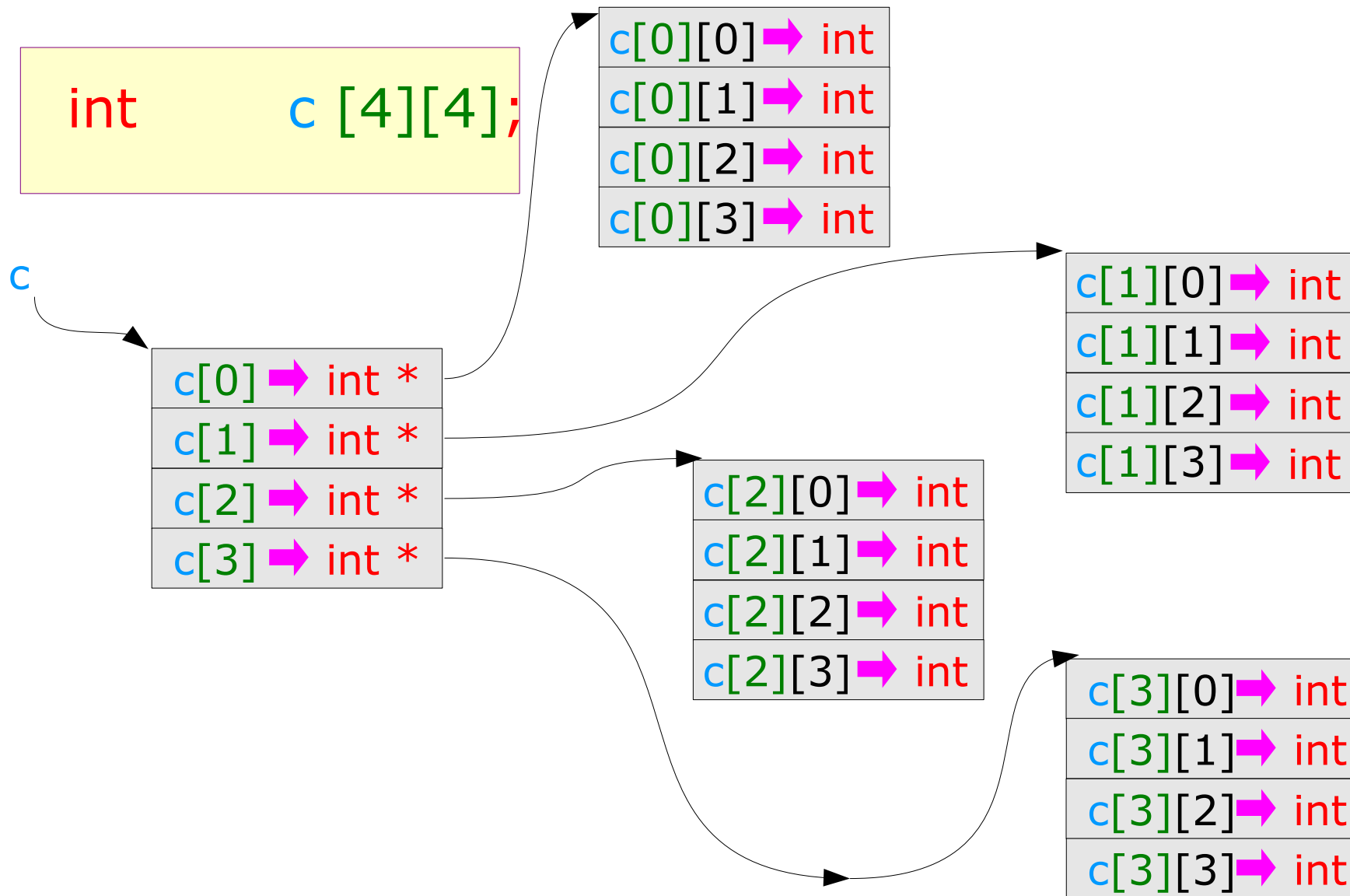
**c[0], c[1], c[2], c[3]** hold the starting address

**int** **c[4]** **[4]**

*No. of elements = 4*

*Type of each element*

# 2-D Array (2)



# 2-d Array

---

## References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun