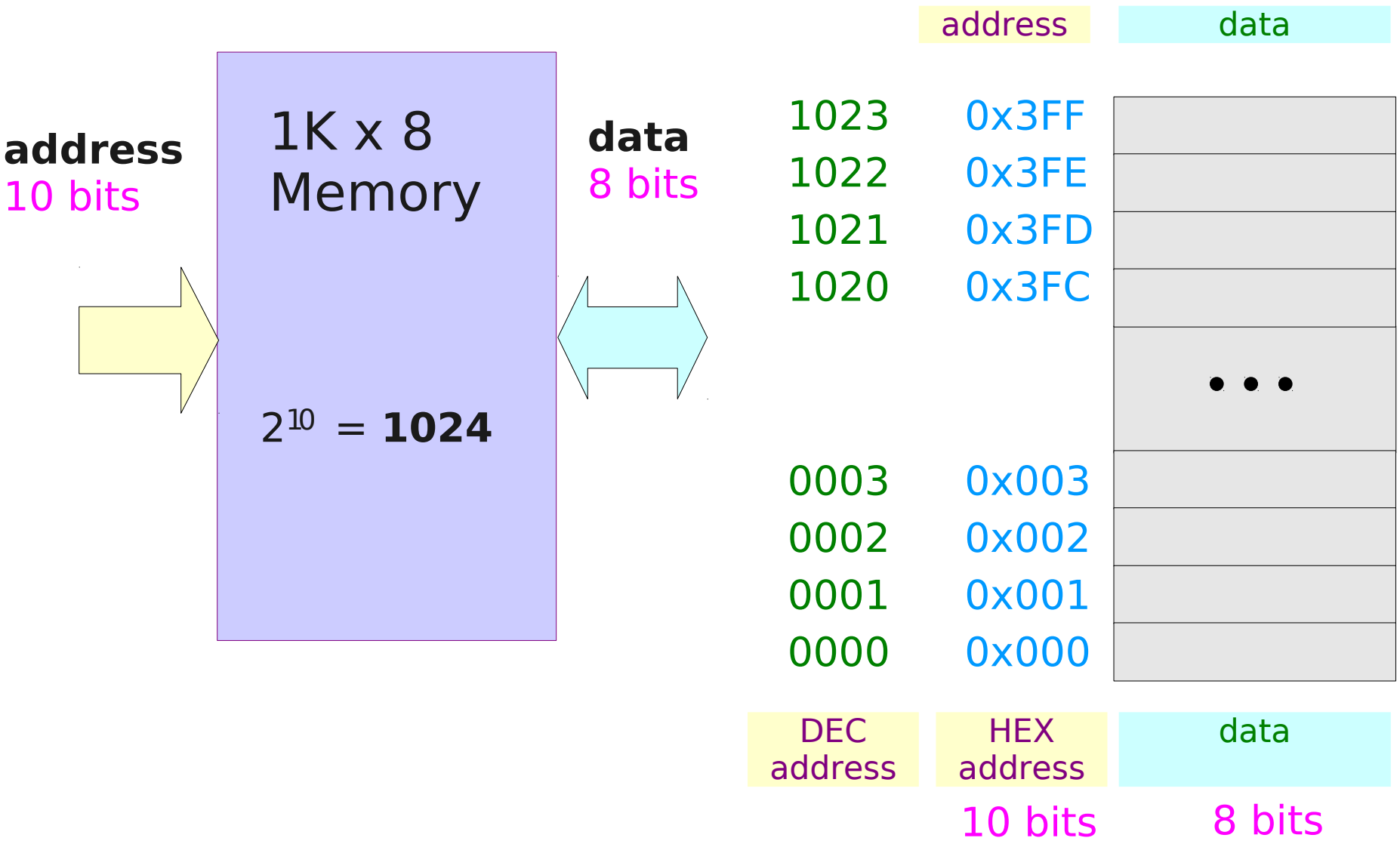
Pointer (2A)

Please send corrections (or suggestions) to 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;
```

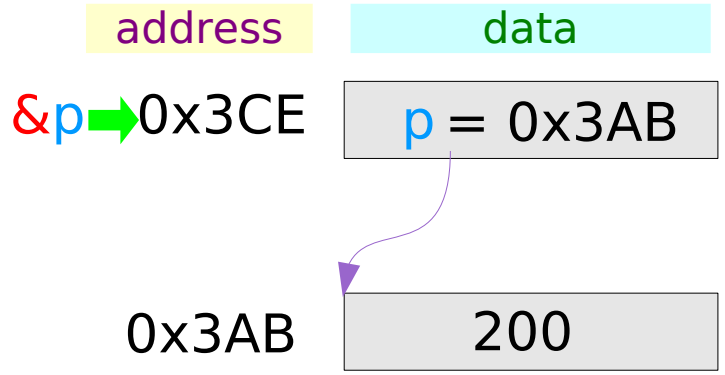
a holds an *integer* 100

address	data
&a	a ← 100

Pointer Variable

```
int *    p;
```

p can hold an *address*




```
int   *   p;
```

p can hold an *address*

pointer to int

```
int   *   p;
```

*p can hold an *integer*

int

&p ➡ 0x3CE

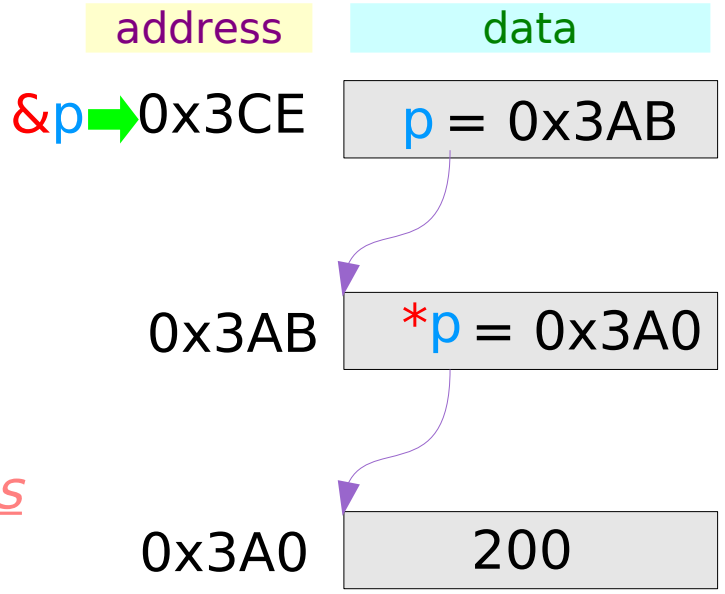
p ➡ 0x3AB

*p ➡ 200

Pointer to Pointer Variable

```
int **    p;
```

p can hold an *address*




```
int  **   p;
```

p can hold an *address*

pointer to pointer to int

```
int  *    *p;
```

*p can hold an *address*

pointer to int

```
int       **p;
```

**p can hold an *integer*

int

&p ➡ 0x3CE

p ➡ 0x3AB

*p ➡ 0x3A0

**p ➡ 200

Integer Pointer Examples (1)

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

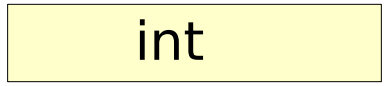
i can hold an *integers*

pi can hold an *address*

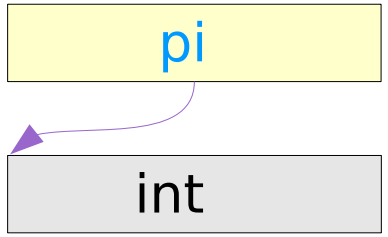
qi can hold an *address*

int

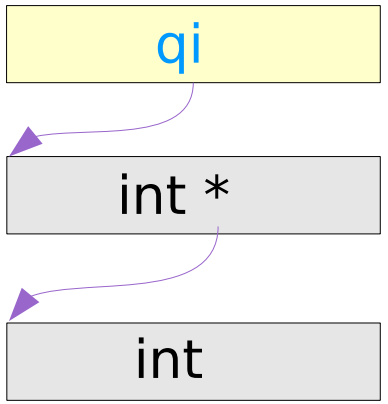
int type



Int * type

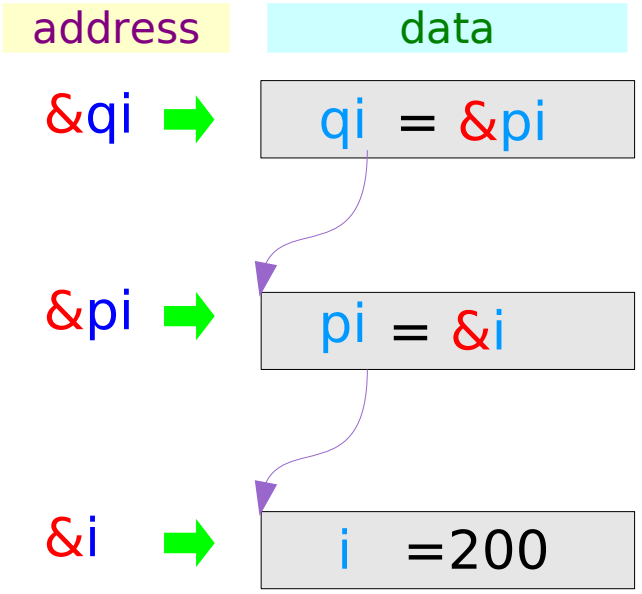


Int ** type



Integer Pointer Examples (2)

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



i can hold an *integers*

pi can hold an *address*

qi can hold an *address*

```
*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]

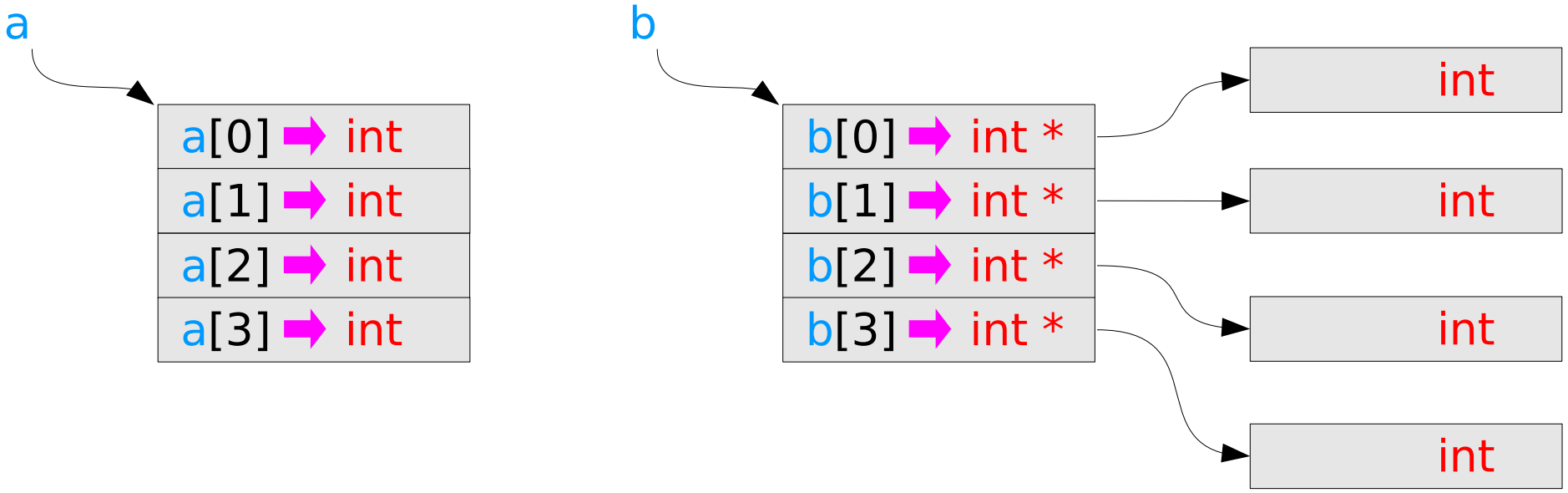
No. of elements = 4

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

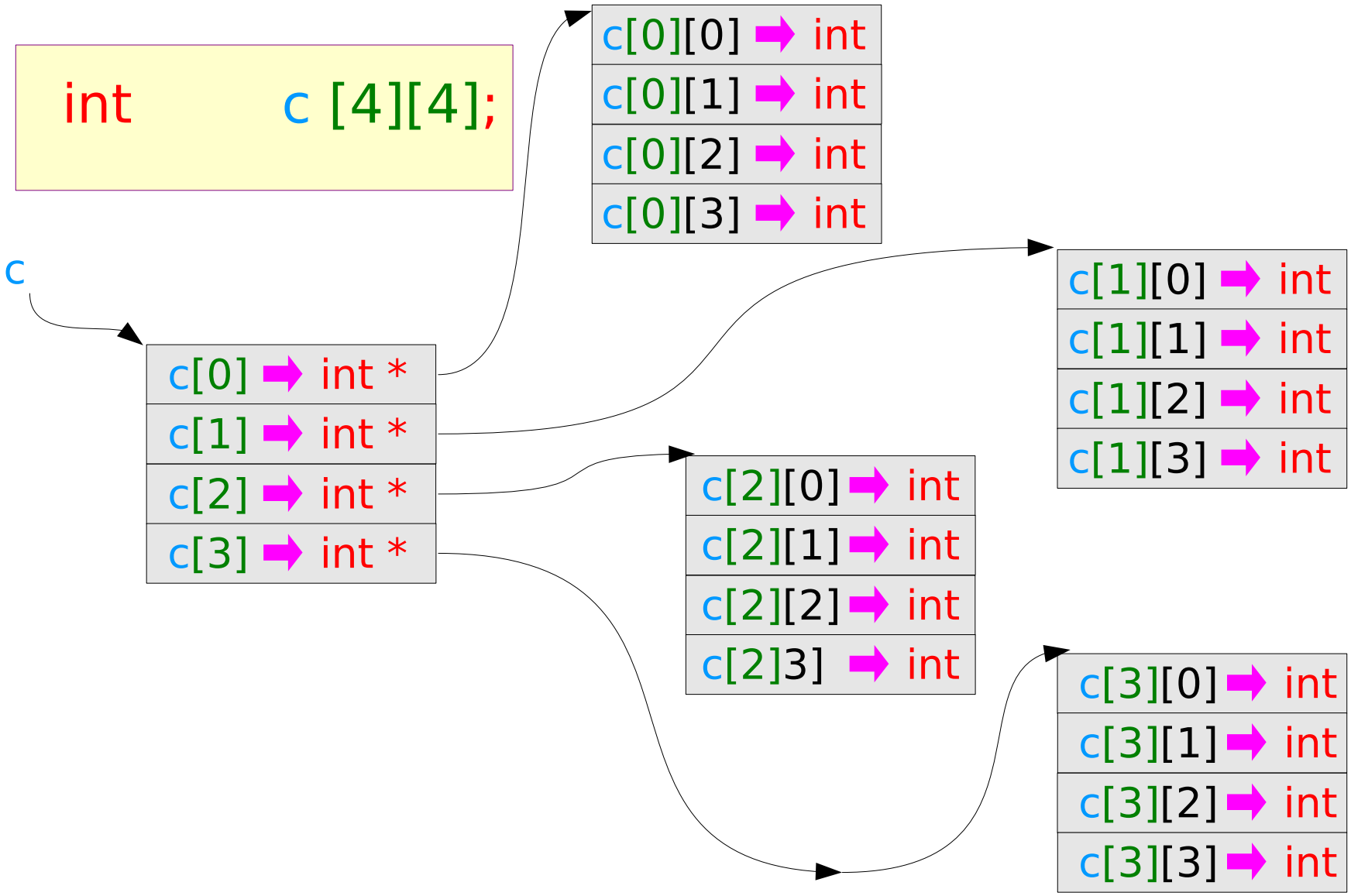
c[0], c[1], c[2], c[3] hold the starting *address*

int c[4] [4]

No. of elements = 4

Type of each element

2-D Array (2)


int c [4][4];
c
c[0] ➡ int *
c[1] ➡ int *
c[2] ➡ int *
c[3] ➡ int *
c[0][0] ➡ int
c[0][1] ➡ int
c[0][2] ➡ int
c[0][3] ➡ int
c[1][0] ➡ int
c[1][1] ➡ int
c[1][2] ➡ int
c[1][3] ➡ int
c[2][0] ➡ int
c[2][1] ➡ int
c[2][2] ➡ int
c[2]3] ➡ int
c[3][0] ➡ int
c[3][1] ➡ int
c[3][2] ➡ int
c[3][3] ➡ int

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