

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.

This document was produced by using OpenOffice.

Variable

```
int a;
```

a can hold an integer

address

data

&a

a

```
a = 100;
```

a holds an integer

address

data

&a

a = 100

* and & Operator

* address

means the **value**
that is stored at the address

& variable

returns the **address** of a
location where the
variable's **value** is stored

address

data

&p

p = &a

&a

a = 100

*p = 100

address

data

&a

a = 100

Pointer (1)

```
int * p;
```

p can hold an address

```
p = &a;
```

p holds the address of a

address

data

&p

p

address

data

&p

p = &a

&a

a = 100

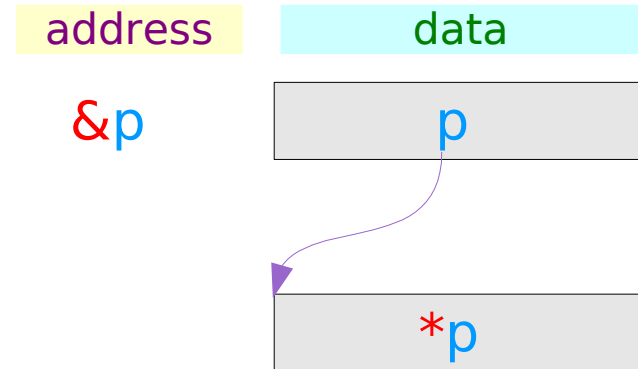
Pointer (2)

```
int * p;
```

`p` can hold an address

```
int *p;
```

`*p` can hold an integer



Variable Assignment

```
int a = 100 ;
```

```
int b = a ;
```

`p` can hold an address

`p` is initialized with the address of integer variable `a`

address

data

`&a`

`a = 100`

`&b`

`b = 100`

`a` and `b` have the same integer value

Address Assignment - Pointer

```
int    a = 100 ;
```

```
int *  p = &a ;
```

p can hold an address

p is initialized with the address of integer variable a

address

data

&p

p = &a

&a

a = 100

a and *p have the same integer value, since &a and p have the same address

Address Assignment - Reference in C++

int

a = 100 ;

int &

b = a ;

b's address is initialized
with a's address

b acts like an integer
variable

b holds an integer

address

data

&a =
&b

a = b

variable b is
an alias of a

a and b have the same
integer value, since
&a and &b have the same
address

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