

Section 7.1 pointer

# What is pointer

- Each variable has
  - type (what kind of data to hold), for example, int, double
  - Name (identify different variables), for example, i, sum
  - Value (current value of a variable), and
  - Address (memory location).
- A pointer denotes the memory location of a variable. That is, a pointer is a variable to hold the address of another variable.

# Why Pointer

In C++, pointers are important for several reasons.

- Pointers allow sharing of values stored in variables in a uniform way.
- Pointers can refer to values that are allocated on demand (dynamic memory allocation).
- Pointers are necessary for implementing polymorphism, an important concept in object-oriented programming (later)

# Pointer syntax

```
char *p; //declares a char pointer
```

```
int *q; //declares an int pointer
```

```
float *r; //declares a float ptr
```

```
string *s; //declares a string ptr
```

```
int *p, q; //only p is a pointer variable;  
           //q is an int variable
```

```
int *p, *q; //to declare two pointers,  
           //attach the * to each variable's name
```

# Usage of pointer

```
int *p; //p points to an int, code link
```

```
int a = 2;
```

```
p = &a; //&a is the address of a,
```

//p saves the address of a, or p points to a

```
cout << *p << endl; // *p is like to say "the guy whose address is p".
```

//after p = &a, \*p is an alias of a

//ie, \*p and a are the same

```
*p = 5;
```

```
cout << a << endl;
```

Memory State after `int *p = &x;` and `int *q = &y;`

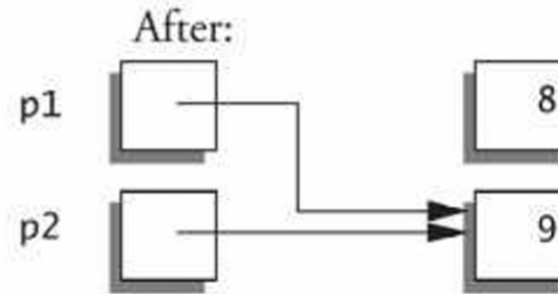
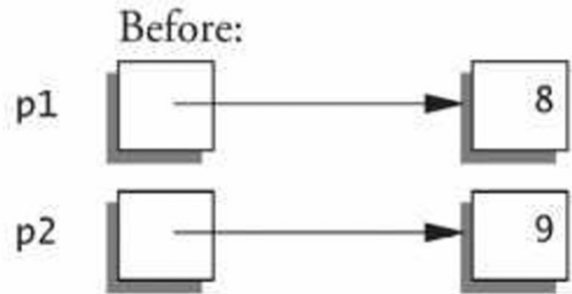
Type	Name	Address	Data
...	...	...	...
int	x	0x12345670	5
int	y	0x12345674	8
int pointer	p	0x12345678	0x12345670
int pointer	q	0x1234567C	0x12345674
...	...	...	...

The diagram illustrates the memory state after the execution of the code. It features a table with four columns: Type, Name, Address, and Data. The table contains five rows of data, with the first and last rows representing memory locations before and after the variables. The second row shows variable 'x' of type 'int' at address '0x12345670' with data '5'. The third row shows variable 'y' of type 'int' at address '0x12345674' with data '8'. The fourth row shows variable 'p' of type 'int pointer' at address '0x12345678' with data '0x12345670'. The fifth row shows variable 'q' of type 'int pointer' at address '0x1234567C' with data '0x12345674'. Two orange arrows originate from the 'Data' column: one from '0x12345670' pointing to the 'Name' column 'x', and another from '0x12345674' pointing to the 'Name' column 'y'. Two pink arrows originate from the 'Name' column: one from 'p' pointing to the 'Address' column '0x12345678', and another from 'q' pointing to the 'Address' column '0x1234567C'.

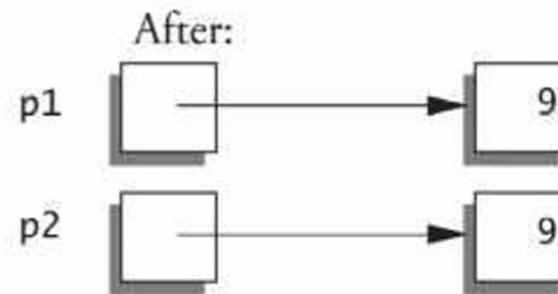
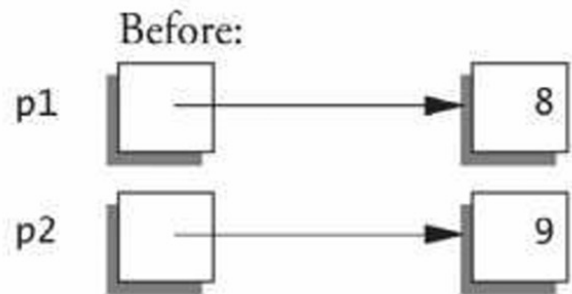
```
int a1 = 8, a2 = 9;  
int *p1 = &a1, *p2 = &a2;
```

#### Display 10.1 Uses of the Assignment Operator with Pointer Variables

```
p1 = p2;
```



```
*p1 = *p2;
```



# Summary

- Each variable has a name, address, and value.
- Pointer variable is declared by  
`type* pointer_variable;`
- A pointer variable saves the address of a variable of its type. Or we say the pointer points to the variable.
  - For example, int pointer saves the address of an int, double pointer saves the address of a double variable.
- To access the variable pointed by a pointer, use \* operator before a pointer variable.