Tassadaq Hussain

Microsoft Barcelona Supercomputing Center
Universitat Politécnica de Catalunya
Barcelona, Spain

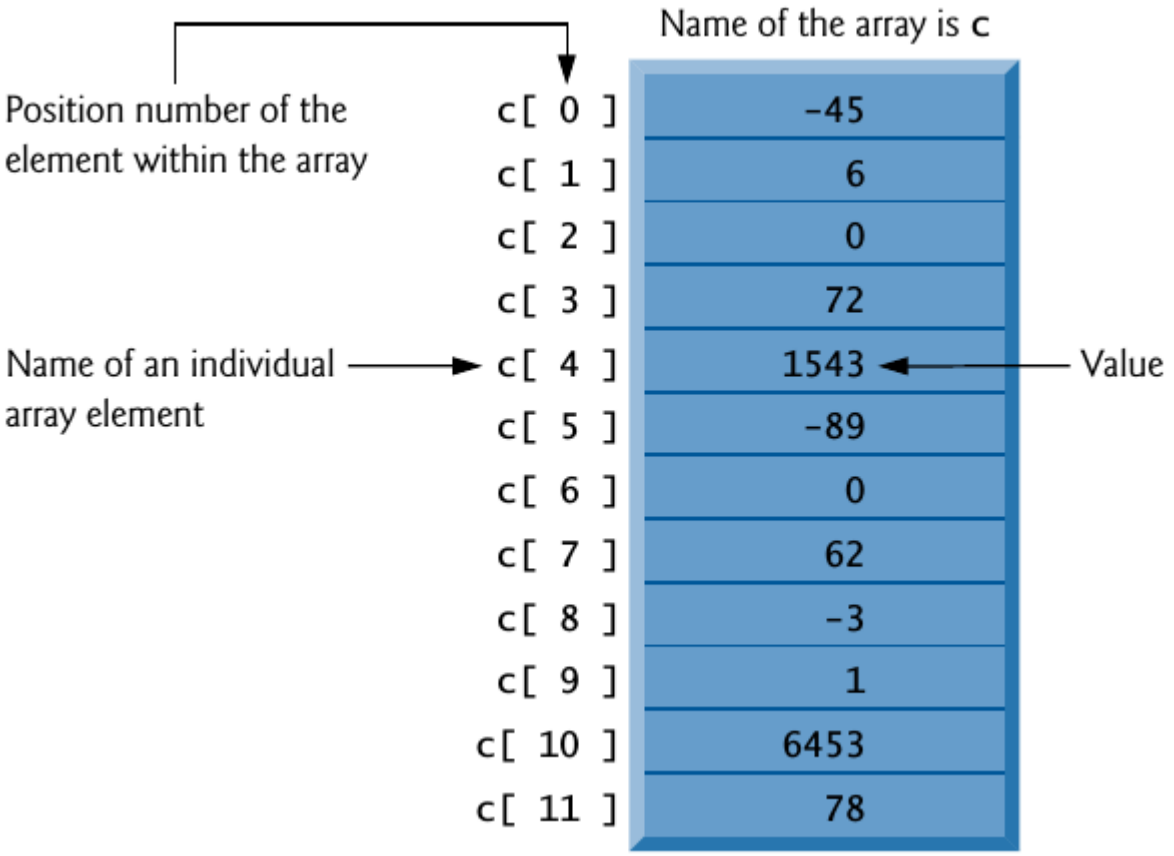
Arrays and Vectors

Arrays

An array is a consecutive group of memory locations that all have the same type.

type arrayName[arraySize];

int c[11];



Operators	Associativity	Type
`::` `()`	*[See parentheses caution in Fig. 2.10]*	scope resolution
`()` `[]`	left to right	function call/array access
`++` `--` `static_cast<type>(operand)`	left to right	unary (postfix)
`++` `--` `+` `-` `!`	right to left	unary (prefix)
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<<` `>>`	left to right	insertion/extraction
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`&&`	left to right	logical AND
`||`	left to right	logical OR
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment
`,`	left to right	comma

Two dimensional array

Int a[2][2];

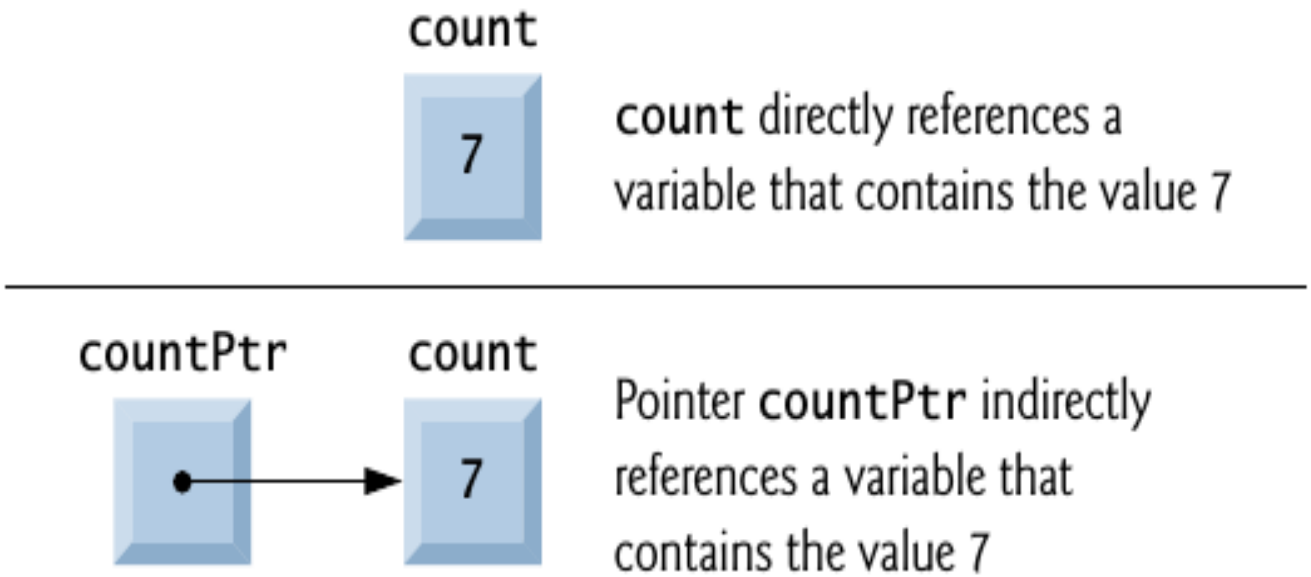
Pointers

Pointers

Pointer variables contain memory addresses as their values.

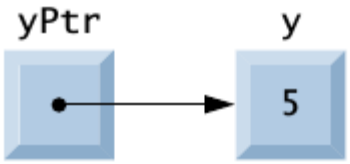
Normally, a variable directly contains a specific value.

A pointer contains the memory address of a variable that, in turn, contains a specific value.


count
7
count directly references a variable that contains the value 7
countPtr
count
7
Pointer countPtr indirectly references a variable that contains the value 7

```c
int y = 5; // declare variable y
int *yPtr; // declare pointer variable yPtr
```

```
yPtr = &y; // assign address of y to yPtr
```


yPtr
y
5