ECE 220 Computer Systems & Programming

Arrays

ECE ILLINOIS

ILLINOIS

Arrays

Array

- A list of values of **uniform type** arranged sequentially in memory
- Example: a list of telephone numbers
- Expression **`a[4]`** refers to the 5th element of the array **`a`**

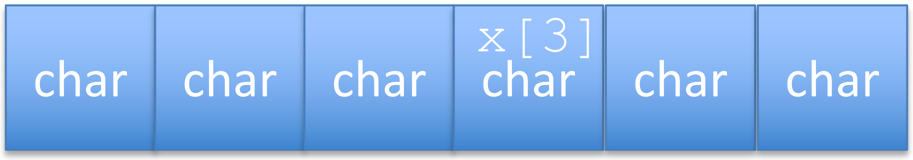
Arrays

- **Allocate a group of memory locations: character string, table of numbers**
- **Declaring and using Arrays**

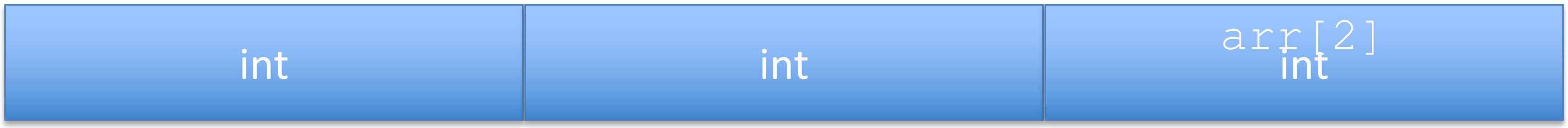
```
int grid[10] = {0,1,2,3,4,5,6,7,8,9};
grid[6] = grid[3] + 1;
int i;
for(i=0;i<2;i++)
{
   grid[i+1] = grid[i] + 2;
}
```

Array Layout

`char x[6]`



`int arr[3]`



Pointer Review

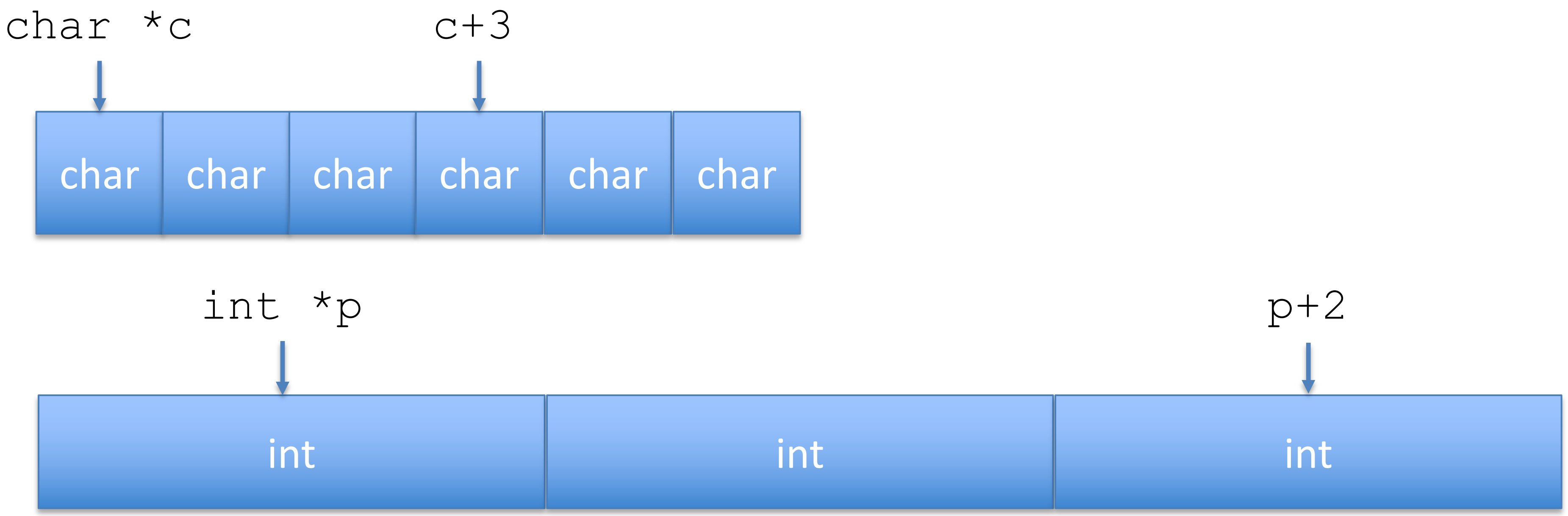
Pointer

- Address of a variable in memory
- Allows us to indirectly access variables (in other words, we can talk about its **address** rather than its **value**)
- Pointers carry type information

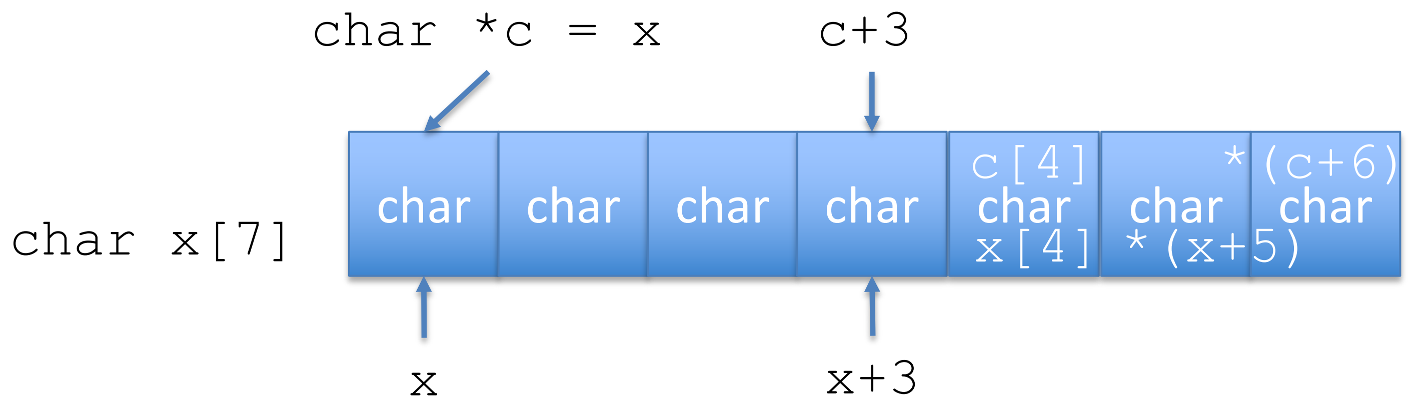
& - address operator: ‘&x’ returns the address of variable x

*** - indirection (dereference) operator: ‘*ptr’ returns the value pointed to by ptr**

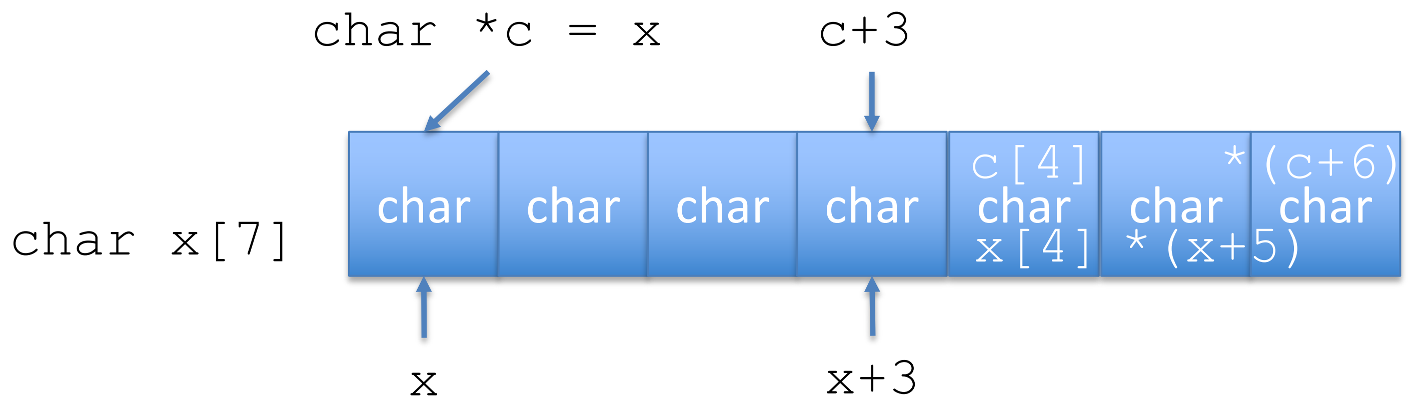
Pointer Arithmetic

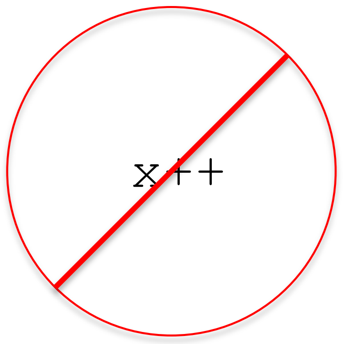


Pointer/Array Duality



Duality Limits





Array identifiers are not l-values

Passing Array as Arguments

C passes arrays by reference

- the address of the array (i.e., address of the first element) is written to the function's activation record
- otherwise, would have to copy each element

```
int main(){
  int array[10];
  int result;
  result = average(array);
  return 0;
}

int average(int array[10]);
/* int average(int array[]); */
/* int average(int *array); */
```