CSE 220 – C Programming

Array and Pointers

Arrays and Pointers

- Consider the following declarations

```
int a[10], *p;
```


```
p = &a[0]     //Makes p point to the 1st element of a
*p = 10;      //Sets the value that p points to to 5
p = &a[1];    //Makes p point to the 2nd element of a
*p = 21;      //Sets a[1] to 21
```

What is the value of x?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
int x = *ptr;
```

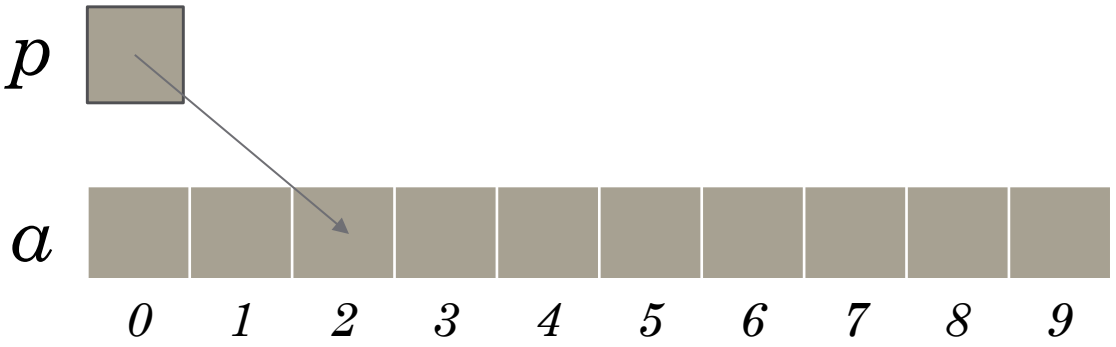
1. Error
2. 1
3. 3
4. I don't know

Pointer Arithmetic

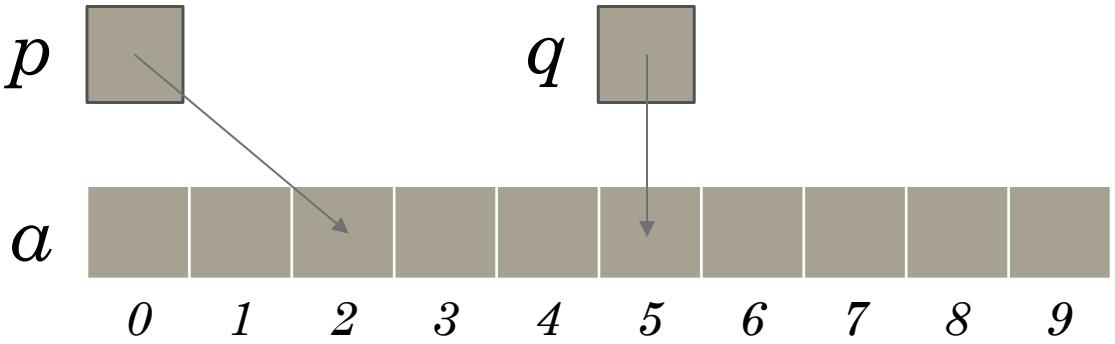
- Access the array by performing pointer arithmetic
- C allows:
  - Adding an integer to a pointer
  - Subtracting an integer from a pointer
  - Subtracting one pointer from another
- Note: such operations only have meaning for pointers with addresses in an array.

Adding an Integer to a Pointer

p = &a[2]

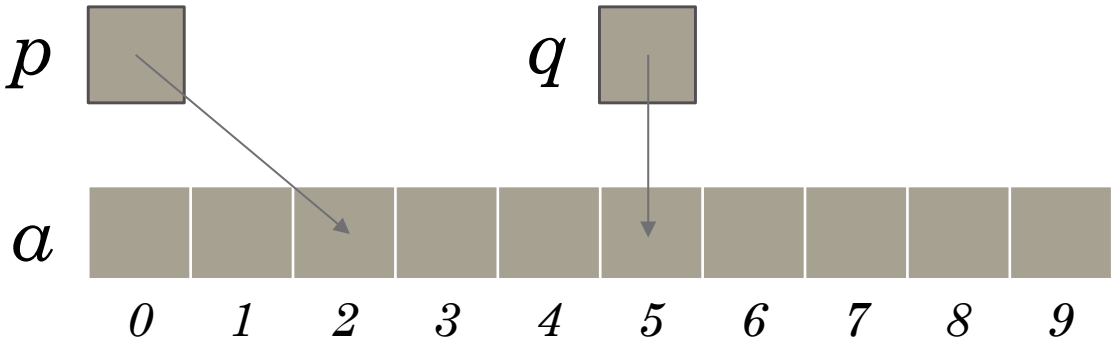


q = p + 3

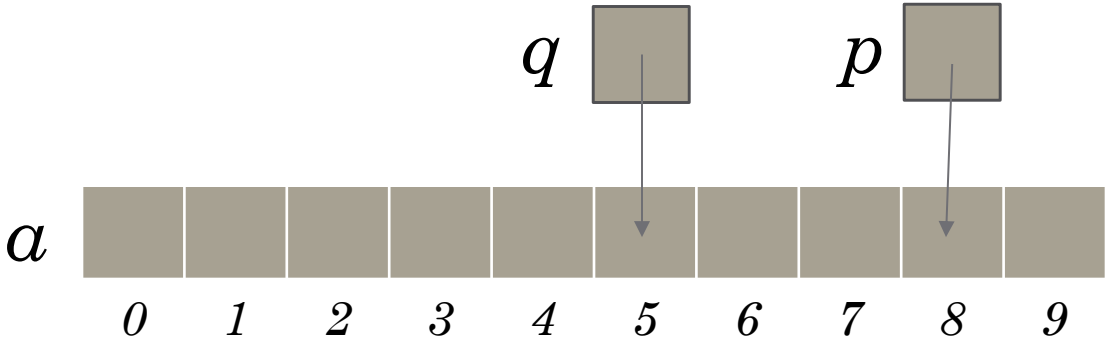


Adding an Integer to a Pointer

q = p + 3

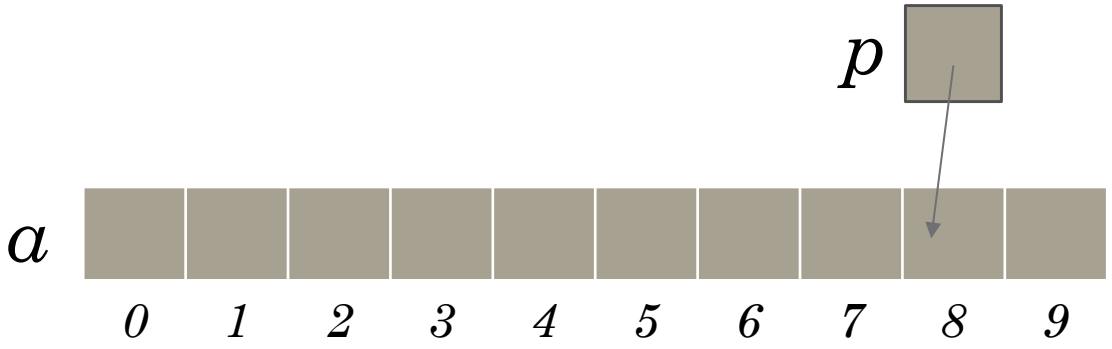


p += 6

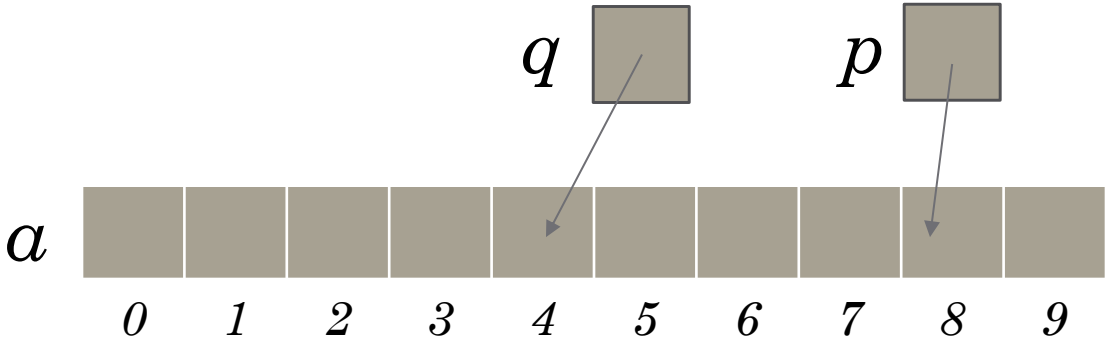


Subtracting an Integer from a Pointer

p = &a[8]

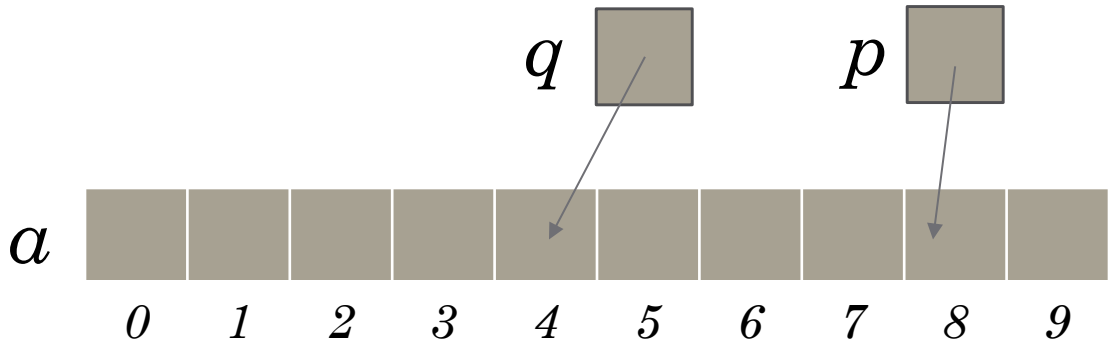


q = p - 4

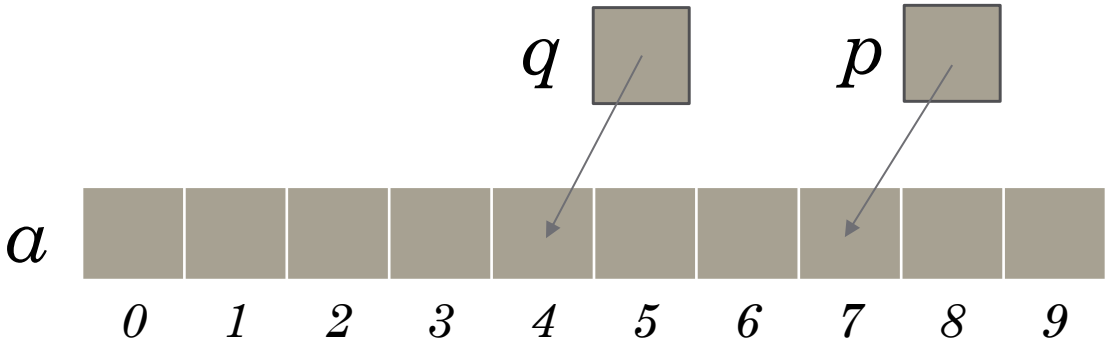


Subtracting an Integer from a Pointer

q = p - 4



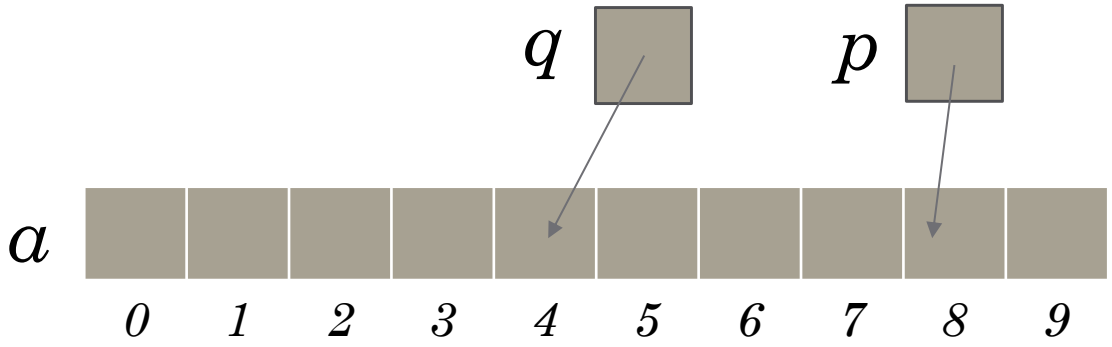
p--



Subtracting a Pointer from a Pointer

Result = The distance (measured in array elements) between the pointers

```
p = &a[8]
q = &a[4]
```




```
int i = p – q;          //i is 4
int j = q – p;          //j is -4
```

What is the value of x?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
ptr++;
ptr -= 2;
int x = ptr - &array[0];
```

1. Error
2. 0
3. 1
4. I don't know

Comparing Pointers

- Can compare pointers using:
  - Relational operators: <, <=, >, >=
  - Equality operators: !=, ==
- With relational operators:
  - the result is meaningful if both pointer point to elements of the same array
  - p < q: the element that p points to comes before the element that q points to in the array
- Equality Operators:
  - p == q: p and q point to the same variable
  - p != q : p and q point to different variables.

Pointers and Arrays

- Use a pointer to visit elements of an array

```
int sum = 0, a[10];
…       //initialize array content
for (int *p=&a[0]; p<&a[10]; p++){
    sum += *p;
}
```

A[10] does not exist.
The last element is a[9].
But this is safe, since the loop is not trying to read the content of a[10]

What is the value of x?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
int x = ptr > &array[0];
```

1. Error
2. 0
3. 1
4. I don't know

Combining * and ++

Consider the statement:

```
a[i++] = j;

//Assign j to a[i], increments the index i
```

The equivalent statement using a pointer:

```
int *p = &a[i];

*(p++) = j
```

Different from:

```
(*p)++  //Increments the value that p points
        //to p remains unchanged
```

What is the state of the array?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
*(ptr++) = 99;
```

1. Error
2. 1, 3, 99, 7
3. 1, 99, 5, 7
4. I don't know

What is the state of the array?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
(*ptr)++;
```

1. Error
2. 1, 3, 5, 7
3. 1, 4, 5, 7
4. I don't know

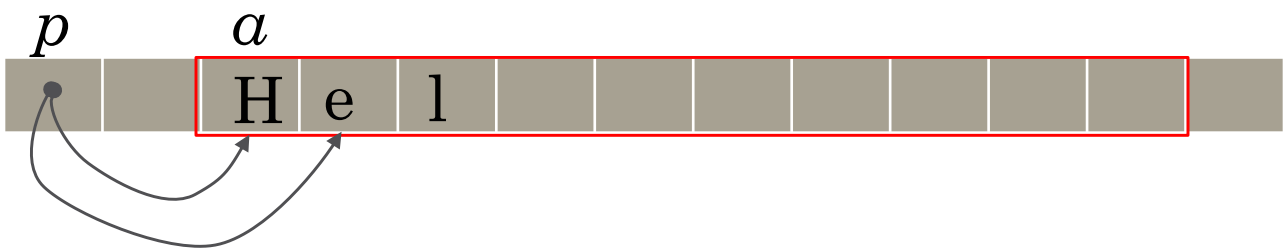
Arrays as pointers

- The name of the array is a pointer to the first element:

```
char a[10]; int i = 2;
```


```
*a = 'H';          //puts H in a[0]
*(a + 1) = 'e'; //puts e in a[1]
*(a + i) = 'l'; //puts 'l' in a[i]
```

Arrays as pointers




```
char *p = a; //declare a pointer to a[0]
p++;  //p points to the next element (a[1]);
```

Cannot change the location that *a* points to

```
a = p + 5;          WRONG
```

What is the state of the array?

```
int array[] = {1, 3, 5, 7};
int * ptr = &array[1];
array = ptr;
```

1. Error
2. 3, 5, 7
3. 1, 3, 5
4. I don't know

Pointers and Arrays

- Use a pointer to visit elements of an array

```
int sum = 0, a[10];
…        //initialize array content
for (int *p=&a[0]; p<&a[10]; p++){
    sum += *p;
}
```

A[10] does not exist.
The last element is a[9].
But this is safe, since the loop is not trying to read the content of a[10]

Pointers and Arrays

- Use a pointer to visit elements of an array

```
int sum = 0, a[10];
…        //initialize array content
for (int *p=a; p<(a+10); p++){
    sum += *p;
}
```

Pointers and Arrays

- Repeat until element with value 0:

```
int a[10];
…
while (*a != 0) {
   a++;
}
```

Wrong

```
int a[10];
…
int *p = a;
while (*p != 0) {
   p++;
}
```

Correct

Exercise

- Consider the following declarations:

```
int a[ ] = {2, 3, 5, 7, 0, 1,
-4, -17, 76, 8, 41};
int *p = &a[5], *q;
```

Set q to point to the second element of a: `q = a+1`

What is the value of *(p+1)? `-4`

Advance p by two positions: `p += 2`

Is p < q? `false`

Is *p < *q? `true`

Arrays as Arguments

- When passed to a function, array name is treated as a pointer

```
void resetValues(int array[], int n) {
   for (int i = 0; i<n; i++)
      array[i] = -1;
}
…
int totals[] = {100, 52, 71, 98};
resetValues(totals, 4);
```

Arrays as Arguments

- Can pass the array as a pointer

```
void resetValues(int *array, int n) {
   for (int i = 0; i<n; i++)
      array[i] = -1; //*(array + i) = -1
}
…
int totals[ ] = {100, 52, 71, 98};
resetValues(totals, 4);
```

Arrays as Arguments

- Changes to the array made inside the function persist outside the function
- Passing a large array does not take more time than passing a small array
- Can pass an array starting at any index, not necessarily from the first element

```
int doSomething(int *array, int n);
      …
}

int val[20] = {1, 2, 3, …};
doSomething(val+5, 7);
```

What does this function do?

```
void abc(int array[], int size) {
  for (int * ptr = array;
       ptr < (array + size);
       ++ptr) {
    *ptr = 0;
  }
}
```

1. It makes all of the elements of the array become 0.
2. It changes my career choices.
3. It causes an error.
4. It confuses me.

Practice Problems

What are the elements of array?

```
int array[3];
for (int *ptr = array; ptr < (array + size); ++ptr) {
    *ptr = ptr - array;
}
```

1. Error
2. 0, 1, 2
3. 0, 1, 2, 3
4. 0, 0, 0

What are the elements of array?

```
int array[] = {1, 2, 3, 5, 7, 11, 13};
int *ptr = array + 5;
++(*ptr);
ptr -= 2;
*(++ptr) = 99;
```

1. Error
2. 1, 2, 3, 4, 99, 12, 13
3. 1, 2, 3, 99, 7, 12, 13
4. 1, 2, 3, 5, 7, 11, 13

What are the elements of array?

```
int array[] = {1, 2, 3, 5, 7, 11, 13};
array += 5;
++(*array);
array -= 2;
*(++array) = 99;
```

1. Error
2. 1, 2, 3, 4, 99, 12, 13
3. 1, 2, 3, 99, 7, 12, 13
4. 1, 2, 3, 5, 7, 11, 13