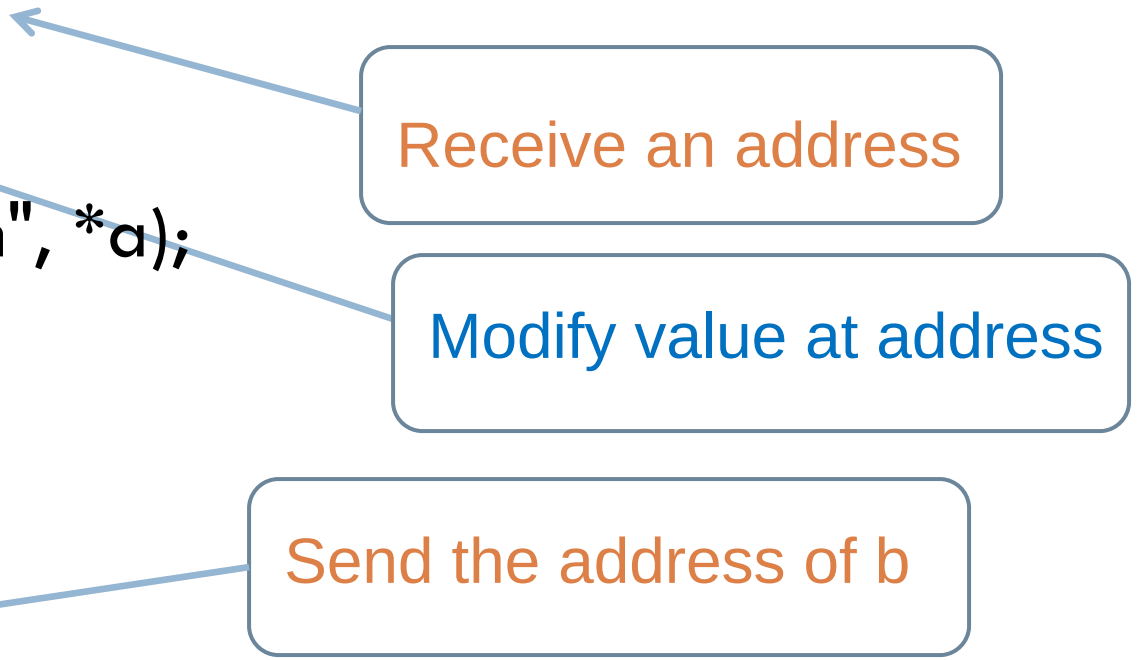
ARRAYS AND POINTERS IN C

CSSE 120 — Rose-Hulman Institute of Technology

Reminder: Using a pointer to change the value of an actual parameter

```
void foo(int *a){
        *a = 7;
        printf("%d\n", *a);
}

int b = 3;
foo(&b);
printf("%d\n", b);
```



From the last homework:

- **swap**: a function to exchange the values of two variables

- Let's look at some possible wrong approaches and why they would not work

```
void swap1(int x, int y) {
   x = y;
   y = x;
}

void swap2(int x, int y) {
   int temp;
   temp = y;
   y = x;
   x = temp;
}


void swap3(int *x, int *y) {
   int *temp;
   temp = y;
   y = x;
   x = temp;
}
```

Q1

C Arrays

- C Arrays are like Python lists
- But there are limitations on how they can be mutated

An example using lists in Python

- Consider the following Python Code:
  - list = [1, “spam”, 4, “U”]
  - list.append(2)
  - list.remove(“U”)
  - length = len(list)
- What do these statements tell us about Python lists?
  - Type does not matter
  - They are dynamically allocated
  - Can be expanded or shrunk
  - Size not specified

List in Python vs Array in C

- No built-in list type in C
- Array is closest data structure to Python's list
- Consider this C code

```
int SIZE = 4;
int nums[SIZE];
int i;
for(i = 0; i < SIZE; i++)
    nums[i] = i * i;
```

- How is this similar to lists in Python?
- Different?

Q2, 3

Initialization and access

- How do we initialize a list or array?
  - Python list: **a = [1, 3, 5]**
  - C array: **int a[ ] = {1, 3, 5};**
- How do we access an element?
  - Python list: **x = a[i]**
  - C array: **x = a[i];**
- How do we access the last element?
  - Python list: **x = a[-1]**
  - C array: **x = a[SIZE - 1];** // the array doesn't know its size.

Quiz: Write countEvens

```
int countEvens(int nums[], int size) {
   // Returns the count of even numbers in the nums array.
   // TODO: complete this function…
   return count;
}
int main() {
   int SIZE = 7;
   int a[] = {16, 5, 23, 19, 42, 17, 12};
   int evens = countEvens(a, SIZE);
   printf("The number of even numbers is %d.\n", evens);
   return 0;
}
```

Working with arrays

1. Checkout the ***ArraysAndRefs*** project from SVN
2. In function **main()** declare a variable, **scores**, to store an array of integers.
3. Implement the function **readScores()** that initializes an array of integers
4. Test the function by invoking it in **main()** and using function **printArray()** to print the values stored in the array
5. If time permits, also enter your **countEvens()** function from the quiz and test it

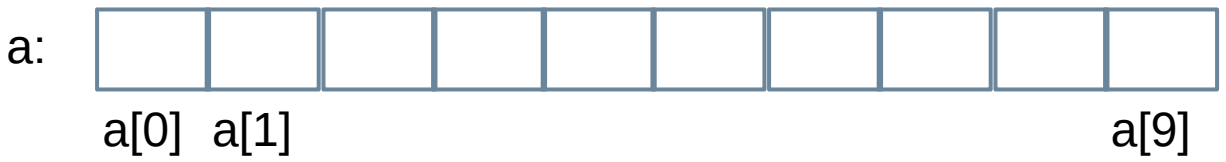
Arrays and Pointers

- In C there is a strong relationship between arrays and pointers
  - An array occupies a fixed location in memory
  - Its address cannot be changed
- Any operation that can be achieved by indexing (e.g., a[i]) can be done with pointers
- The pointer version will be
  - a bit more challenging to implement at first
  - but faster in some cases

How arrays and pointers relate

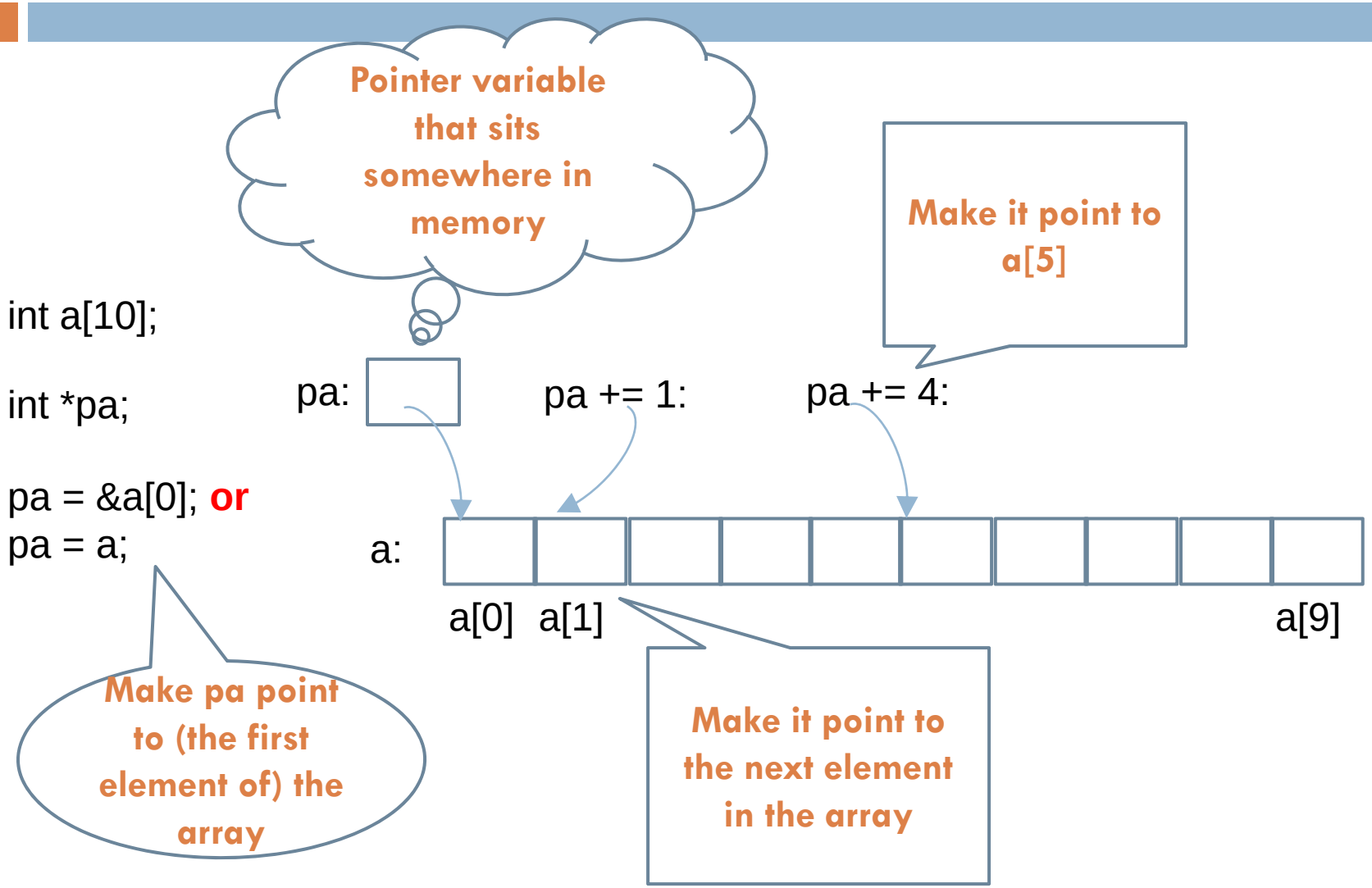
Each element in the array is accessed using the notation a[i] where i is the index of the ith element.

int a[10];



int a[10]; defines an array of size 10, i.e., a block of 10 consecutive integers named a[0], a[1], …, a[9]. a is really the starting address of the array.

How arrays and pointers relate



Summary of arrays and pointers

- int *pa; declares a pointer to an integer
- Set pa to point to array a
  - pa = a;
  - pa = &a[0];
- Copy the content of a[0] into x
  - int x = a[0];
  - int x = *pa;

Summary of arrays and pointers (2)

- Point to the second element in the array
  - pa + 1
  - &a[1]
- Copy the content of a[1] into y
  - int y = a[1];
  - int y = *(pa + 1);

Arrays as function parameters

- **int** [] and **int** * are equivalent, when used as formal parameters in a function definition, e.g., …
  - **void f (int a[], int count) { …**
  - **void f (int *a, int count) { …**
- Note that in neither case can we know the size of the array, unless it is passed in as a separate parameter.
- In either case, the 6th element of **a** can be equivalently accessed as
  - a[5]
  - *(a+5) // treating array a as a pointer

Using pointers with arrays

- How do we modify printArray() so that it uses pointers instead of array indexing?
- Implement:
  - `void printArrayThePointerWay(int* a, int m) {…}`
- Test the function by invoking it in **main()**, like so:
  - `printArrayThePointerWay(scores, size)`

HW Warm-up: Thinking of a Sort

- Homework asks you to imagine you are a real estate agent who is helping potential home buyers to analyze the prices of homes in Vigo county.
- In order to analyze those prices you may need to sort the prices.
- Given:

```
double ratings[] = {2.4, 5.0, 4.4, 3.2, 0.1};
```

- What would we do to sort **`ratings`** in ascending order?