ARRAYS AND POINTERS IN C

CSSE 120 — Rose-Hulman Institute of Technology

Recall: The three notations for pointers in C

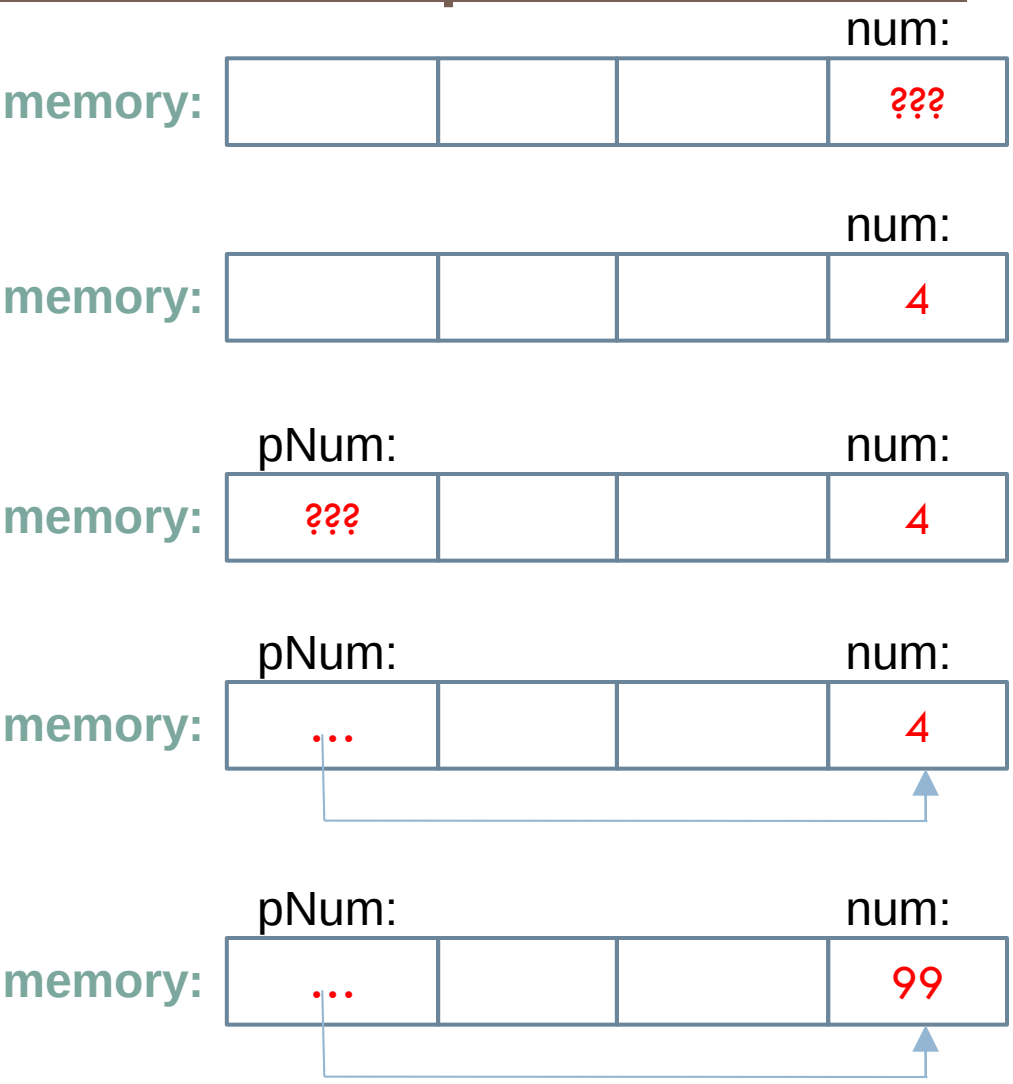
```
int num;

num = 4;

int* pNum;

pNum = &num;

*pNum = 99;
```



pNum is a ***pointer*** to an int

pNum is set to the ***address*** of num

The ***thing at pNum*** is set to 99

`pNum` is the *pointer* and `num` is the *pointee*.
`*pNum` *deferences* the pointer, which means that it obtains the pointee.

Q1-8

Using pointers as parameters
Box and Pointer Diagrams

```
int b;
foo(&b);

void foo(int* a) {
    ...
    *a = 7;
}
```

Send the address of b

Receive an address

Modify value at address

Now `b` has the value 7 that was established in `foo`!

This is useful for:

- sending data back from a function via the parameters, and for
- passing large amounts of data to a function.

Thus pointers in C give us the same advantages as references-to-objects in Python.

Pointer Pitfalls

- Don't try to dereference an unassigned pointer:

```
int* p;
*p = 5;          /* (usually) immediate crash! */
```

- Pointer variables must be assigned *address* values.

```
int x = 3;
int* p;
p = x;           /* this statement is the error */
...
... *p ...   /* but the crash won't occur until now */
```

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
  - Size not specified
  - Can be expanded or shrunk

```
int main() {
    int size = 7;
    int a[size];

    initializeArray(a, size);

    return EXIT_SUCCESS;
}

void initializeArray(int a[], int size) {
    int k;

    for (k = 0; k < size; ++k) {
        a[k] = 100;
    }
}
```

Declare the array : type and size. Allocate space, uninitialized. Size cannot change. Can initialize elements with: `int a[] = {…};`

Pass the array to a function – just the array name. Must also send size; no *len* function.

Get an array as a parameter – array name plus empty brackets. Must also send size; no *len* function.

Loop through array. Reference array elements like in Python – square brackets with index, indices start at 0. NO CHECK that references stay within the array!

Checkout 26-Arrays1. Do the TODO's.

C Arrays

- C Arrays are like Python lists
  - Reference elements using **foo[index]** as in Python
- But for arrays:
  - All elements of the array must be of the same type (e.g. all int's or all double's).
  - You must specify the size of the array when it is constructed. Allocates space, with garbage values.
  - You cannot change the size of the array later.
    - But we will see how to accomplish this using dynamic arrays, later.
  - Arrays don't know their size. You must carry along a separate variable that keeps track of their size.
  - Arrays don't check that their references are within the bounds of the array. Referencing out of bounds can cause a crash or even change the values of other variables in the program!
  - Arrays are closer to the hardware than lists; hence they are somewhat faster than lists.

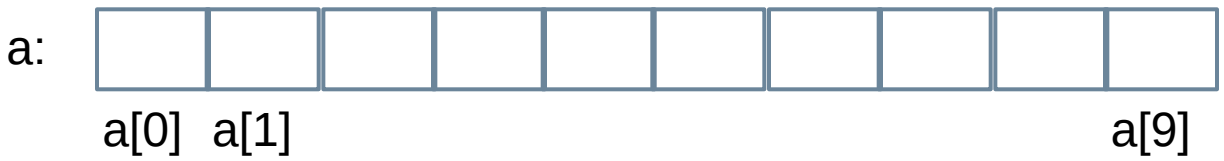
Arrays and Pointers

- 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

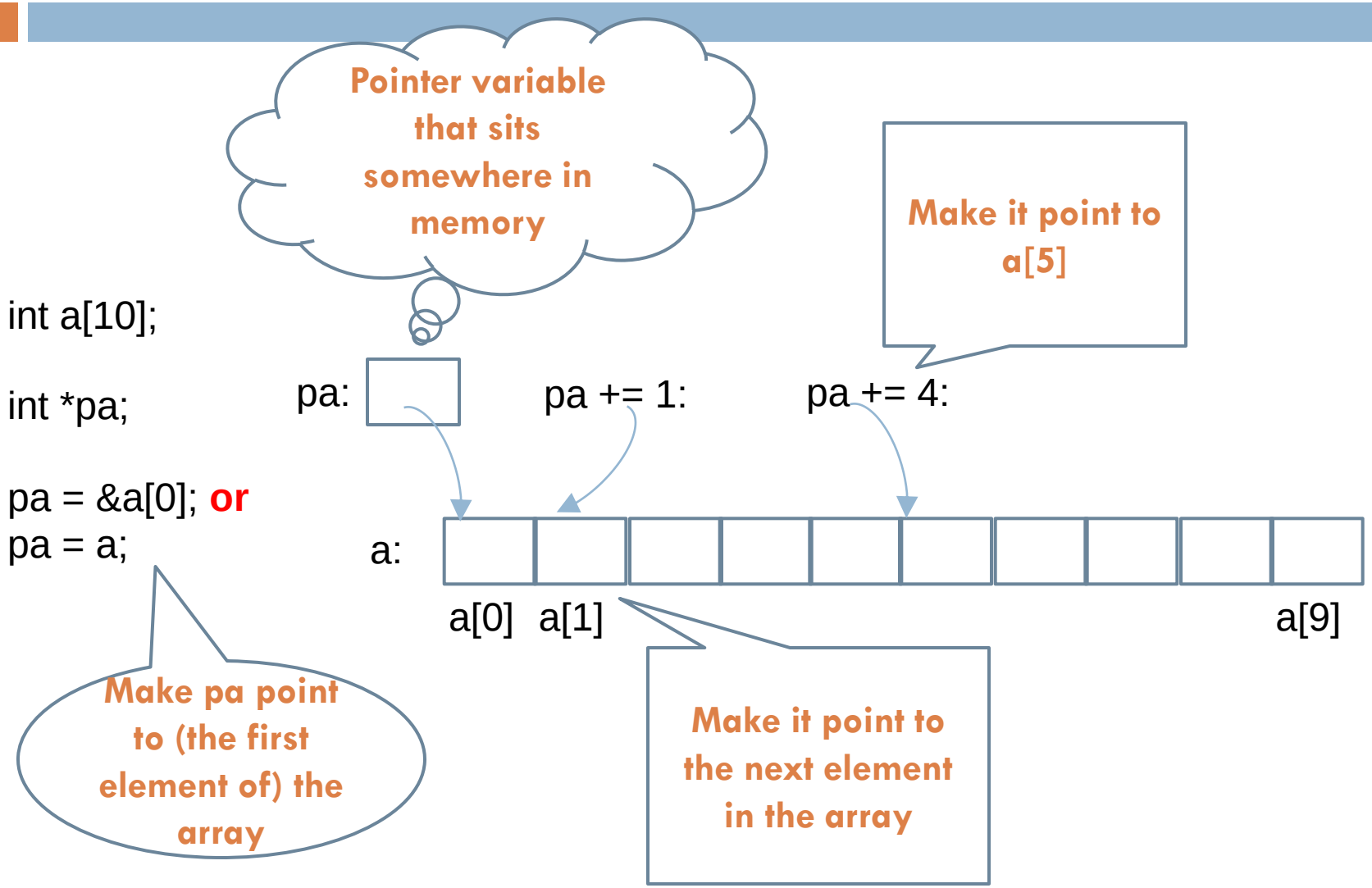
Each element in the array is accessed using the notation a[i] where i is the index of the ith element.

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[0];` or `pa = a;` (your choice)
- Refer to array elements (given above assignment)
  - `a[0]` or `*pa` (your choice)
- Pointer arithmetic
  - Can increment pointers, so the following are equivalent:

```
                pa = &a[0];          pa = &a[0];
a[k]            *(pa + k)            pa = pa + k;
                                     *pa
```

Array notation vs. Pointer notation

```
void initializeArray(int a[], int size) {
    int k;

    for (k = 0; k < size; ++k) {
        a[k] = 100;
    }
}
```

```
void initializeArray(int* a, int size) {
    int* p;

    for (p = a; p < a + size; ++p) {
        *p = 100;
    }
}
```

Do TODO

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?