APS105 Lecture 27
10.1

Dan Zingaro

March 29, 2010

Feedback from Reading Quiz

- "How many comparisons will selection sort make for an array with 10 elements?"
  - I gave you the formula, but you can figure it out even without that
- "I don't understand the difference between comparison and iteration."
  - Comparison: check if one value is less, equal to, or greater than another
- "How do you choose the best possible pivot value when you have no idea what the values of the array are? Can you do a sum of all the array elements and then divide by the total number of elements to find the average ..."
  - That's a good way to find a reasonable pivot, but we're just going to take the rightmost value and hope for the best

Structures vs. Arrays

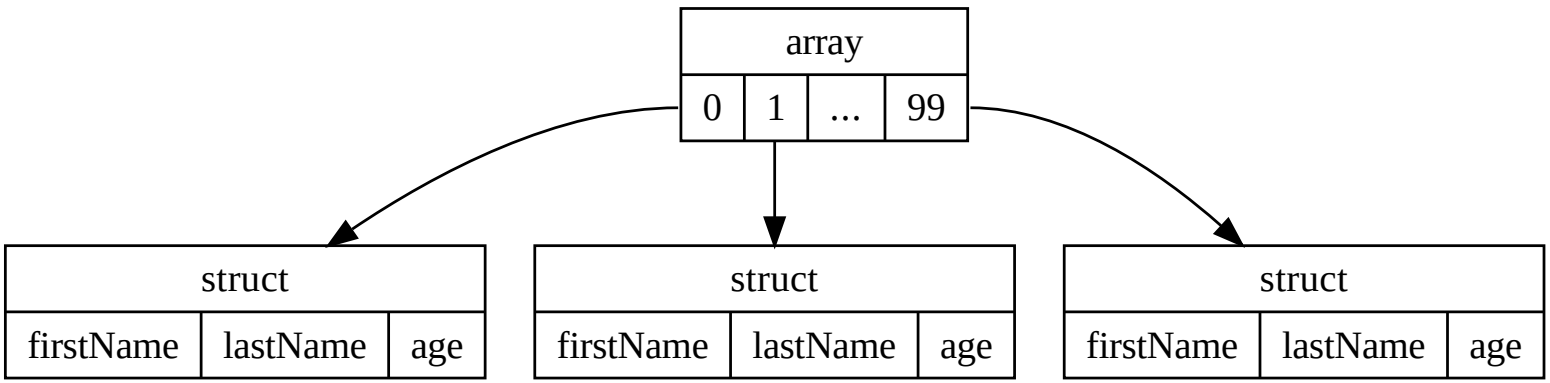
- To collect data of the same type, use an array
  - e.g. to store 1000 marks between 0 and 100
- To collect data of different types, use a structure
  - e.g. to store info about a person: first name, last name, age, male/female
- To collect small amounts of data of the same type, where each piece of data has a unique name, use a structure
  - e.g. to store a date: year, month, day

ConcepTest

Assume we want to store information about 100 people. For each person, we will store their first name, last name, and age. How should we store this data?

- A. As an array, each of whose elements is a structure
- B. As a structure, each of whose components is an array
- C. As an array, each of whose components is a string
- D. As a structure, each of whose components is a string

ConcepTest: Diagram



ConcepTest

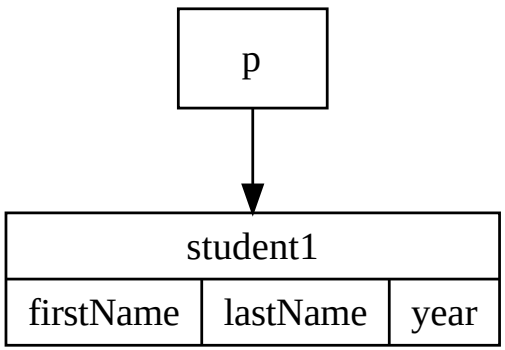
We want to store one student's first name, last name, age, and marks they earned in 100 courses. What should we use?

- A. An array, each of whose elements is a structure with four components
- B. A structure with four components, one of which is an array of ints
- C. An array, each of whose components is a string
- D. A structure, each of whose components is a string

Pointers to Structs

```
struct student student1;
struct student *p;
p = &student1;
```

- Now, `student1` is a `student struct`, and `p` is a pointer to `student1`



Pointers to Structs...

```
struct student student1;
struct student *p;
p = &student1;
```

- We have three ways to access or modify the members of `student1`

```
student1.year = 3;
(*p).year = 3;
p->year = 3;
*p.year = 3; /*Wrong!*/
```

Pointers to Structs...

We can use pointers to structs to write a function that modifies a struct's members.

```
void changeFirst (struct student *s, char *newFirst) {
  s->firstName = newFirst;
}
```

ConcepTest

We would like to write a function that takes two `struct` parameters `from` and `to`, and copies the values in `from` to the values in `to`. What types of parameters should the function take?

- A. Two `structs`
- B. Two pointers to `structs`
- C. One pointer to `struct`, and one `struct`