# Structured data

- Parallel arrays aren't a natural fit for *heterogeneous* rows of data
  - One set of names, one set of positions, one set of scores
- What we have is structured data
  - Name, position, score for each employee
  - One set of employees
- For a single employee we could do:

```
string name;
string position;
int review_score;
```

  - Allocates memory space for 2 strings and 1 int
    - Names each location

# Using classes

- C++ provides *classes* to group structured data together

```
class employee
{
 public:
    string name;
    string position;
    int review_score;
};
```

- This is a *class definition*
  - Give the class a name
  - Tell the compiler what the parts of the class are
    - Each part has a type and a name (looks just like a variable)
    - The parts of a class are called *members*

# Using classes

- Defining the class creates a blueprint
  - No memory is allocated yet
  - The class is used as a data type in a variable declaration:
    - Variable declaration is always:

```
type name;
```

    - So in this example:

```
employee emp;
```

- This variable declaration:
  - Allocates memory space for an *instance* of the class
    - 2 strings, 1 int
  - Names that memory space
  - A class instance is also called an *object*

# Using class objects

- With arrays, you always have to indicate which element in the array you want to use
  - Using the array subscript operator []
  - E.g. `this_array[15]`
- With class objects, you have to indicate which part of the class you want to use
  - The *member access operator* (.) indicates part of an object
  - The parts are used like any other variable:

```
emp.name = "peter";
cin >> emp.position;
emp.review_score = emp.review_score + 1;
```

# Arrays of objects

- Now that we've defined a class for employee
  - We can have a set of employees using an array

    ```
    employee emps[10];
    ```

  - Allocates space for 10 employee objects
    - Each one has 2 strings and 1 int

- Combine array and class access operators
  - The 5th employee's name:
    - emps[5].name
  - the first employee's review score:
    - emps[0].review_score

# Exercise: arrays of objects

- Define a class to hold a point (x, y)
  - Like you would use to specify points on the screen
- Write a statement to declare an array of 100 points
- Write statements to set the first point to (1, 4)
  - That is, x is 1, y is 4
- Write statements to set the second point to (5, 3)
- Assuming there is an integer n, and there are n valid points in your array:
  - Write statements to print the values of all n points to the screen

# Example: lookup a record

- Given the arrays of employee objects and the following code:

```
string lookup_name;
cout << "Enter a name to look up: ";
cin >> lookup_name;
```

- Write a function to return the requested employee
- Write a function to print an employee object
  - E.g. “samir (developer) received a review of 75”

# Exercise: lookup the highest score

- Given the array of employee objects
- Write a function to return the employee object with the highest review score