Arrays: Introduction: Lecture Outline

- **Array indexed** from 0 to some upper limit

  *example*: List of dog breeds is indexed collection of elements of same type

  0. Beagle
  1. Cocker Spaniel
  2. Labrador
  3. Dalmatian
  4. Golden Retriever
  5. Siberian Husky
## Arrays: Introduction: Lecture Outline

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
</tr>
</tbody>
</table>

- Temperature at 0 is 34
- Temperature at 1 is 36
- Temperature at 2 is 39
- Temperature at 3 is 35
- Temperature at 4 is 29
Arrays: Introduction: Lecture Outline

temperature at 0 is 34 → $t_0 = 34$

temperature at 1 is 36 → $t_1 = 36$

temperature at 2 is 39 → $t_2 = 39$

temperature at 3 is 35 → $t_3 = 35$

temperature at 4 is 29 → $t_4 = 29$
Arrays: Introduction: Lecture Outline

ARRAY structure

data values or instances

<table>
<thead>
<tr>
<th>index</th>
<th>data name</th>
<th>base type is INTEGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$t_0 = 34$</td>
<td>34 $t[0]$</td>
</tr>
<tr>
<td>1</td>
<td>$t_1 = 36$</td>
<td>36 $t[1]$</td>
</tr>
<tr>
<td>2</td>
<td>$t_2 = 39$</td>
<td>39 $t[2]$</td>
</tr>
<tr>
<td>3</td>
<td>$t_3 = 35$</td>
<td>35 $t[3]$</td>
</tr>
<tr>
<td>4</td>
<td>$t_4 = 29$</td>
<td>29 $t[4]$</td>
</tr>
</tbody>
</table>

temperature at 0 is 34

temperature at 1 is 36

temperature at 2 is 39

temperature at 3 is 35

temperature at 4 is 29
Arrays: Introduction: Lecture Outline

- **Array structures** can be defined as part of declaration statement:

```c
// define an integer array
int temperature[5];
```

- **Number** in square brackets indicates **how many** elements are in array

- **Indexes** are **zero-based**. First available position is index ← 0.

- When using array, all spaces do not have to be used; must have enough.

- Usual practice: establish extra
int main () {
    // define array of integer temperatures:
    int temperature[5];   // up to 5 temperatures
    temperature[0] = 45;
    temperature[1] = 62;
    cout << "0: " << temperature[0] << endl;
    cout << "1: " << temperature[1] << endl;
}
Arrays: Introduction: Lecture Outline

• If reading in data, could read data in for each line, one by one.

```cpp
// purpose read in five values
#include <iostream.h>
main () {
int temperature[5];
    int temperatureIndex;
    int inputTemperatureValue;
    temperatureIndex = 0;
    cin >> inputTemperatureValue;
    temperature [temperatureIndex] = inputTemperatureValue;
    temperatureIndex = 1;
    cin >> inputTemperatureValue;
    temperature [temperatureIndex] = inputTemperatureValue;
    temperatureIndex = 2;
    cin >> inputTemperatureValue;
    temperature [temperatureIndex] = inputTemperatureValue;
    temperatureIndex = 3;
    cin >> inputTemperatureValue;
    temperature [temperatureIndex] = inputTemperatureValue;
    temperatureIndex = 4;
    cin >> inputTemperatureValue;
    temperature [temperatureIndex] = inputTemperatureValue;
    cout << endl << "All temperatures read in";
}
```
Arrays: Introduction: Lecture Outline

- Iteration statement `for` best used with arrays.

- Example: Store and then display two temperatures.

```cpp
void main () {
   // define array of integer temperatures:
   int temperature[5];
   int temperatureIndex;
   temperature[0] = 45;
   temperature[1] = 62;
   for (temperatureIndex = 0;
        temperatureIndex < 2;
        ++temperatureIndex)
      cout << temperatureIndex << " : " <<
           temperature[temperatureIndex] << endl;
}
```

iterative control

controlled statement
If reading in data AND unknown count of data AND can read to sentinel value, read using while, for example:

```cpp
// test data: 3 5 9 4 -1 8 4
#include <iostream.h>
int main () {
    const int sentinelValue = -1;
    // define array of integer temperatures:
    int temperature[5];
    int temperatureIndex;
    int inputTemperatureValue;
    int temperatureCount;
    temperatureCount = 0;
    cin >> inputTemperatureValue;
    while (inputTemperatureValue > sentinelValue) {
        temperatureIndex = temperatureCount;
        temperature [temperatureIndex] =
            inputTemperatureValue;
        cin >> inputTemperatureValue;
        temperatureCount = temperatureCount + 1;
    }
    cout << endl << temperatureCount
        << " temperatures read in";
```
Arrays: Introduction: Lecture Abstract

// Define two dimensioned array
int temperatures [5][3];
Arrays: Introduction: Lecture Abstract

- Example, subscripted `temperature_{3,1}` would be array reference `temperature[3][1]`.
- Example,
  
  `temperature[3][1] = 26;`
Arrays of Class Objects

- Arrays of simple classes can have multiple data types. Arrays of simple classes are effectively same as files.
  
  data items ⊆ records ⊆ file

<table>
<thead>
<tr>
<th>Name</th>
<th>Major</th>
<th>GPA</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pat</td>
<td>Computer Science</td>
<td>3.9</td>
<td>Cincinnati</td>
</tr>
<tr>
<td>Kim</td>
<td>Electrical Engineering</td>
<td>3.8</td>
<td>Newport</td>
</tr>
<tr>
<td>Jo</td>
<td>Music</td>
<td>3.9</td>
<td>New York</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Arrays of Class Objects

• Defining an array of class objects is a straightforward extension of defining arrays.

```
class TrainCar
   public:
      char type; // type of train car
      int age;   // age of train car
};

TrainCar batchCars[100];
```

Separate objects records
Arrays of Class Objects

class TrainCar
    public:
        char type; // type of train car
        int age;   // age of train car
};

TrainCar batchCars[100];

• Storage space is referenced by pattern of:
  
  <array name>
  <array index>
  <member variable name>

  for example,

  batchCars[carCount].type = 'b'
Arrays of Class Objects

• Example program:

#include <iostream.h>
//@-------------------------------------------------------------------------------
//@ define train car class:
class TrainCar {
  public:
    char type; // type of train car
    int age;   // age of train car
};
int main () {
//@-------------------------------------------------------------------------------
//@ define collection of train cars:
TrainCar batchCars[100]; //less than 100 need
//@-------------------------------------------------------------------------------
int carCount = 0; // count of cars stored
//@ insert sample test data
  carCount = carCount + 1;
  batchCars[carCount].type = 'b';
  batchCars[carCount].age = 2;
  cout << batchCars[carCount].age;
}
Arrays of Class Objects

• As with other array structures, for statement can be used in iteration.

// define train car class:
class TrainCar {
    public:
        char type; // type of train car
        int age;   // age of train car
};
int main () {
    // define collection of train cars:
    TrainCar batchCars[100]; // less than 100 needed
    // insert sample test data
    for (carCount = 1; carCount <= 2; ++carCount)
        {cout << carCount << endl;
         cin >> batchCars[carCount].type;
         cin >> batchCars[carCount].age;}
    // display sample output
    for (outputCounter = 1; outputCounter <= 2;
         ++outputCounter)
        {cout << batchCars[outputCounter].type << " "
           << batchCars[outputCounter].age << endl;}
}