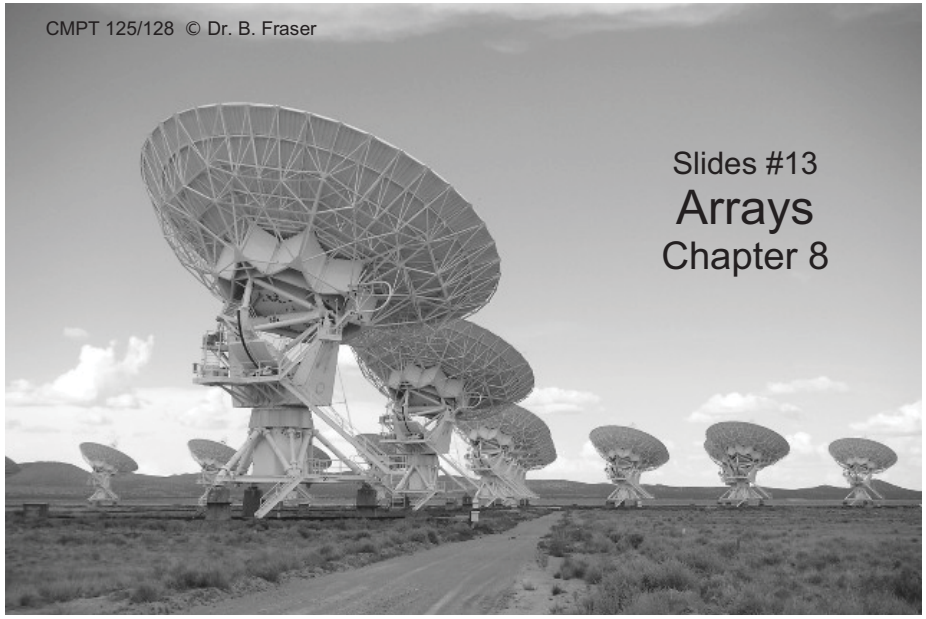


Topics

1) How can we store many values at once?
2) How can arrays (and elements) be passed to functions?
3) How can we store objects in an array?

Arrays

Arrays

- Array:
  -
- Arrays vs Vectors
  - A vector is an object with complex operations.
  - Arrays are not objects:...
- Array Declaration:
  - Specify type of elements, and # elements.
    int daysPerMonth[12];
  - Once created, the array size...
- Directly access to any element:
  - For N elements...
  - daysPerMonth[0] = 31; // January

	Idx	Val
Jan	0	31
Feb	1	28
Mar	2	31
Apr	3	30
May	4	31
Jun	5	30
Jul	6	31
Aug	7	31
Sep	8	30
Oct	9	31
Nov	10	30
Dec	11	31

Initializing an Array

- Array can be initialized when declared:
  int somePrimes[]
  - # of values in the initializer list...

- Initialized Array of strings:

```
string monthNames[] = {
    "January",
    "February",
    ....
    "December"
};
```


Array Bounds

- C++ does not do...
  on array index operations.
  - Out of bounds access...

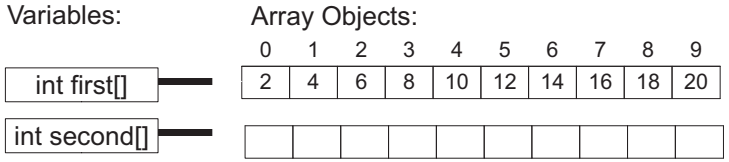
  - Could be anything else in memory!

```
int price[4];
price[0]    = 10;          //
price[4]    = 11;          //
price[-1]   = 12;          //
```

Copying an array

- To copy an array...

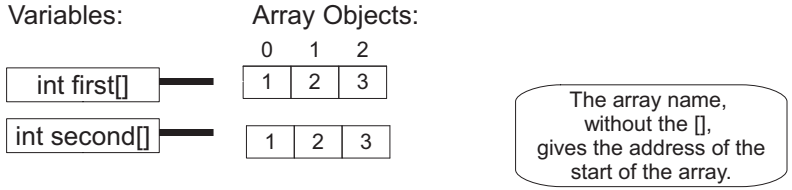
```
const int SIZE = 10;
int first[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
int second[SIZE];
for (int i = 0; i < SIZE; i++) {
    second[i] = first[i];
}
```



Comparing arrays

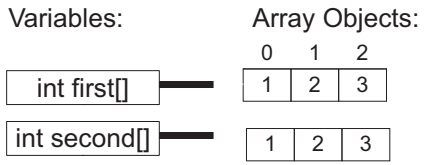
- An array variable...
  - Comparing array variables compares the memory addresses of the arrays, not their contents

```
int first[3] = {1, 2, 3}, second[3] = {1, 2, 3};
if (first == second) {          // Likely bug.
    // ...
}
```



Comparing arrays

```
const int SIZE = 3;
int first[] = {1, 2, 3}, second[] = {1, 2, 3};
bool areSame = true;
for (int i=0; i<SIZE; i++) {
     if (first[i] != second[i]) {
          areSame = false;
     }
}
if (areSame) {
     cout << "Contents of arrays are the same!";
}
```



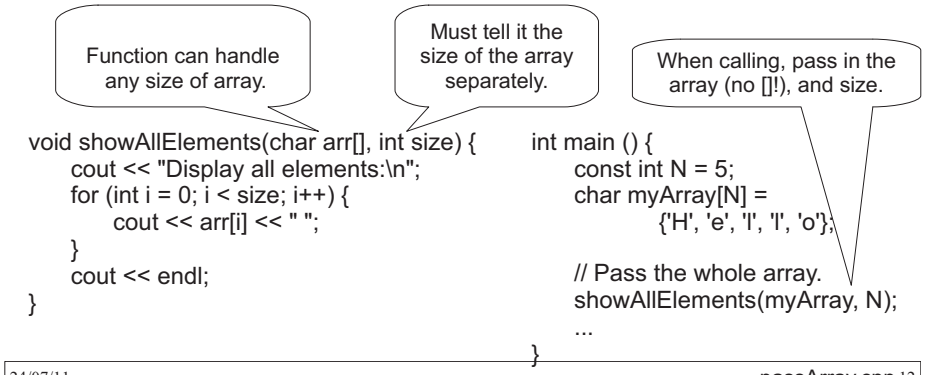
Review

- Create an array holding the values 10.1, 2.5 and 5.7; Write a loop to sum up all elements in the array; Output the total to the screen.

Arrays as function arguments

Passing a full array

- Need two things to pass an array to a function:



Pass array by ref or val?

- Passing an array to a function passes the memory address of the array.
- This behaves very much like pass-by-reference.
  - It is not a copy of the array: it is the real thing.
  -

Vectors vs Arrays

- Vectors often more flexible than arrays:
  - Can resize them while program is running.
  - Can "ask" them how many elements they have.

  (Neither of these are possible with arrays.)
- Vectors are great for storing data when...
- Arrays are built-into the core C++;
  Vectors are in the standard template library.

Review

- Write a C++ function named showOdd() which accepts an array of int and outputs all odd elements.

Arrays of objects

Array of objects

- Can create arrays of objects.

```
int main () {
    const int NUM_CARS = 2;
    // Use default constructor
    Car myCars[NUM_CARS];
    cout << myCars[0].getMake() << endl;

    // Call set method
    myCars[0].setMake("Porsche");
    cout << myCars[0].getMake() << endl;

    // Use 1 parameter constructor.
    Car myCars2[] = {"Ferrari", "Pinto"};
    cout << myCars2[0].getMake() << endl;
    cout << myCars2[1].getMake() << endl;

    return 0;
}
```

```
class Car {
public:
    Car() {make = "--none--";}
    Car(string newMake);
    void setMake(string newMake);
    string getMake();
};
```

This creates an array of two objects, using the

Able to directly access methods of the array elements with [#].method()

Initialization list calls

Vector of objects

- Vector Element Access
  - push_back()...
    and places it in the vector.
  - [ ] accesses the...

```
// Create empty vector to store cars.
vector<Car> myCars;

// Add a new car.
myCars.push_back(Car("Porsche"));

// Create another car, and add it.
Car otherCar("Ferrari");
myCars.push_back(otherCar);
otherCar.setMake("Pinto");

// Display the elements:
cout << myCars[0].getMake();
cout << myCars[1].getMake();
```


Array of Objects vs Multiple Arrays

- OOD array of class Person:
  Person myStudents[100];
- Non-OOD uses...
  string myStudentNames[100];
  int myStudentAges[100];

- OOD is better
  - Consider:
    - Adding 3 new attributes to the Person class.
    - Sorting the array(s) into alphabetical order by name.
    - Passing all the data to a method.

Summary

- Arrays store many items of the same type.
  - Must be a fixed size.
- Passing to functions:
  - Elements just like other variables (by val or by ref).
  - Actual array is passed as a memory address; so it's always affecting the original array.
- Vectors like an array, but:
  - Can add/remove elements at run-time,
  - Knows how many elements it is holding.
- Arrays of objects, or vectors of objects, are great for object-oriented programs.