


Extending the Array Object

- Flexible sizing
 - Change number; track current size
- Still reference items (elements)
 - With subscript ([]) operator
- Add operations
 - E.g., sort elements


4



Standard Template Library

- A class library
 - A number of useful classes
 - Part of ANSI C++ standard
- Based on C++ templates
 - Adapt to different data types
 - More on that later (polymorphism)

5



STL **vector** Container Class

- A container
 - Like C++ array
 - Holds elements of same data type
- Adjustable
 - Can change size during execution
 - Keeps track of its current size

6

STL **vector** Definition

```
// Define a vector containing
// integer grades:
```

vector

<int>

grades;

No size specified!

↑

STL container
class type

↑

Element
data type

↑

Name of
vector
object

7

Adding **vector** Elements

```
// Add grades to a vector:

int grade_a = 95;
➡ grades.push_back (grade_a);
➡ grades.push_back (85);
➡ grades.push_back (100);
```

↑

Member function adds
element to end of vector

| | |
|---------|-----|
| grades: | 95 |
| | 85 |
| | 100 |

8

Modifying **vector** Elements

Member function returns # of elements


```
// Increase all grades by
// fifteen points:
unsigned int n = grades.size();
for (int j = 0; j < n; j++)
{
  grades[j] = grades[j] + 15;
}
```

↑

↑

Subscript operator returns reference to element
(just like C++ array)


9



Sorting **vector** Elements

```
#include <algorithm>
#include <vector>
using namespace std;
// Sort grades in ascending
// order:
    sort (grades.begin(),
          grades.end() );
```


Sort function from library Iterator objects returned by member functions (for now, it's just magic!) ¹⁰



Removing **vector** Elements

```
#include <vector>
using namespace std;
...
// Erase all grades:
    grades.clear();

// Alternative method:
    grades.erase (grades.begin(),
                  grades.end()); 11
```



Copying **vector** Objects

```
#include <vector>
using namespace std;
// Make a copy of a grades
// vector:
    vector<int> grades;
    vector<int> grades2;
...
    grades2 = grades; // Copy
```

Assignment operator replaces all elements with copies of source elements ¹²
