

STL vector Container Class

- **C++ has array type**
  - Aggregate of individual data objects
- **But array has limitations**
  - Must allocate size at definition
- **Standard Template Library**
  - Class library provides an alternative

---

---

---

---

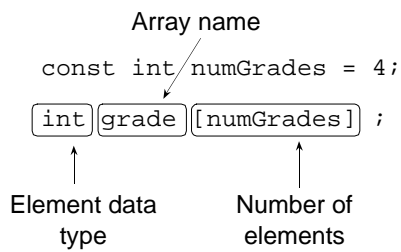
---

---

---

---

Defining a C++ Array Object




---

---

---

---

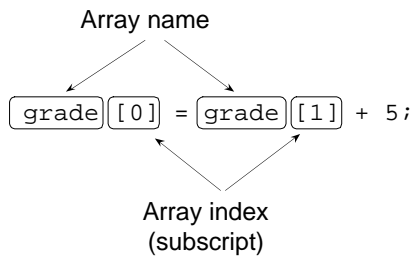
---

---

---

---

Accessing C++ Array Elements




---

---

---

---

---

---

---

---

Extending the Array Object

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

---

---

---

---

---

---

---

---

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)

---

---

---

---

---

---

---

---

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

---

---

---

---

---

---

---

---

### STL vector Definition

```
// Define a vector containing
// integer grades:
vector<int> grades;
```

Annotations:  
 - `vector`: STL container class type  
 - `<int>`: Element data type  
 - `grades`: Name of vector object  
 - Note: No size specified!

---

---

---

---

---

---

---

---

---

---

### 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);
```

grades: 

95
85
100

Member function adds element to end of vector

---

---

---

---

---

---

---

---

---

---

### 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)

---

---

---

---

---

---

---

---

---

---

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!)

---

---

---

---

---

---

---

---

Removing vector Elements

```

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

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

```

Assignment operator replaces all elements with copies of source elements

---

---

---

---

---

---

---

---

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

---

---

---

---

---

---

---

---

Character Strings

- **Very common data type**
- **Represent with char array?**
  - Holdover from C language
  - Text, page 307
- **But lots of problems**
  - Similar to C++ array limitations

---

---

---

---

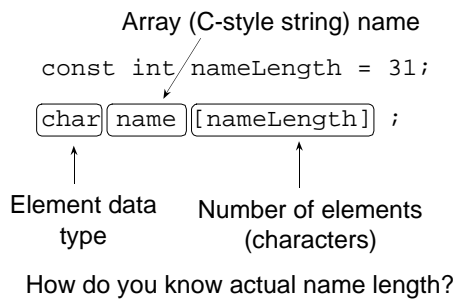
---

---

---

---

Defining a char Array Object




---

---

---

---

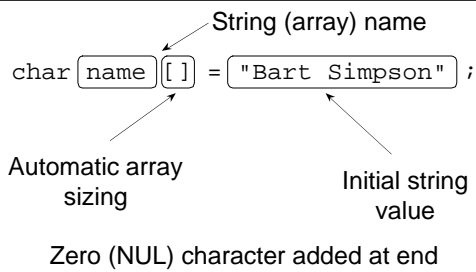
---

---

---

---

Another char Array Object




---

---

---

---

---

---

---

---

Limitations of char Array

- **Sizing inflexibility**
  - Like other C++ arrays
- **Must have special terminator**
  - Zero (NUL) character at end
  - Need room for it
    - Even though not part of "string"

---

---

---

---

---

---

---

---

The string Class

- **Class library**
  - Part of ANSI C++ standard
- **Preferred string data type**
  - Avoids C-style string problems
  - Works with character string literals
  - Internal implementation hidden
    - May use C-style strings inside

---

---

---

---

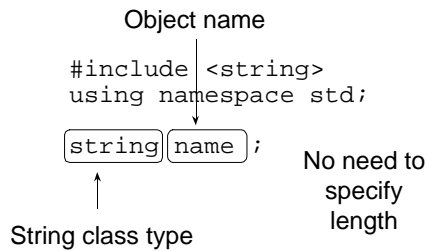
---

---

---

---

Defining a string Object




---

---

---

---

---

---

---

---

Using the string Class

```

string FirstName = "Bjarne";
string LastName;           Concatenation
LastName = "Stroustrup";
string names = FirstName +
Assignment " " + LastName;
cout << names << endl;
Insertion

```

---

---

---

---

---

---

---

---

string Member Functions

- **Initialization**
  - Uses special member function known as a "constructor" (more later)
- **Length**
- **Conversion to C-style string**
- **Others (won't discuss now)**

---

---

---

---

---

---

---

---

string Initialization (1)

```

string str1; // Zero length
No initial value (empty)
string str2 = str1;
string str3 = str1 + str2;
string str4 (str2);
Alternate form
String expressions

```

---

---

---

---

---

---

---

---

string Initialization (2)

```

string str4 = "Hello";
string str5 ("Goodbye");

string str6 = 'A';
string str7 ('z');

```

Character string literals

Single character (char)

---

---

---

---

---

---

---

---

string Initialization (3)

```

string str8 = "ABCDEFGHIJKL";

string str9 (str8, 2, 5);

```

Substring of existing string value

Starts at character 2 with a length of 5 (or less)

"CDEFG"

---

---

---

---

---

---

---

---

String Length

```

unsigned int len;

string str9 = "Hello";
len = str9.length();

```

Use unsigned int for length (more complex answer later)

Get length of string

5

---

---

---

---

---

---

---

---

Conversion to C-Style String

```

string filnam;
cout << "Enter file name: ";
cin >> filnam;
ofstream outf( filnam.c_str() );
outf << "Data";
outf << endl;

```

Annotations: "Get file name" points to `cin >> filnam;`; "Convert string to C-style string" points to `filnam.c_str()`.

---

---

---

---

---

---

---

---

string Operators

- Assignment (=)
- Concatenation (+)
- Assign/concatenate (+=)
- Relational (comparison)
- Insertion (<<)
- Extraction (>>)

---

---

---

---

---

---

---

---

string Assignment

```

string string_one = "Hello";
string string_two;
string_two = string_one;
string_two = "Goodbye";
char ch = 'a';
string_two = ch;

```

Annotations: "string" points to `string_one`; "Character string literal" points to `"Goodbye"`; "Character" points to `ch`.

---

---

---

---

---

---

---

---

string Concatenation

```

string str1 = "Hello ";  string
string str2 = "there";
string str3 = str1 + str2;
    Character
    string literal
string str4 = str1 + "there";
string str5 = "The End";
str4 = str5 + '!'; ← Character

```

---

---

---

---

---

---

---

---

string Assign/Concatenate

```

string str1 = "Hello ";
str1 += "there";
Concatenate and assign
cout << str1;
    Hello there

```

---

---

---

---

---

---

---

---

string Comparison

- **Use relational operators**
  - E.g., >, >=, ==
  - Condition for if, while, etc.
- **Each has two operands**
  - Two string objects
  - Character string literal and string

---

---

---

---

---

---

---

---

⋮

string Relational Operators

```

==    equality
!=    not equal
>     greater than
<     less than
<=   less than/equal
>=   greater than/equal

```

---

---

---

---

---

---

---

⋮

string Stream Operators

```

string str1 = "Hello there";
cout << str1 << endl;

string str2;
cin >> str2;

```

Insertion

Extraction

In some versions, some stream operations (e.g., width()) may not work as you would expect.

---

---

---

---

---

---

---

⋮

Design Example

- **Read values from user**
  - A sequence of words
  - Terminated with "END"
- **Display (print) all values**
- **Sort values** Using vector and string
- **Display (print) again**

---

---

---

---

---

---

---