CHAPTER 9/10
CLASSES

CMPS 148

Lab 6

- Create a BankAccount Class
  - Properties (Member Variables):
    - Balance
    - Interest Rate (yearly)
  - Actions (Member Functions):
    - Withdrawl(double amountToWithdraw)
    - Deposit(double amountToDeposit)
    - ApplyYearlyInterest() *use Deposit function*
- Write a main program that uses your class and lets the user perform the three actions.

“Example” main

```
int main() {
    BankAccount account;

    account.Deposit(400);
    cout << "Balance:  $" << account.getBalance() << endl;

    account.ApplyInterest();
    cout << "Balance:  $" << account.getBalance() << endl;

    account.Withdraw(50);
    cout << "Balance:  $" << account.getBalance() << endl;

    system("pause");
}
```

Prints 400

Prints 420

Prints 370

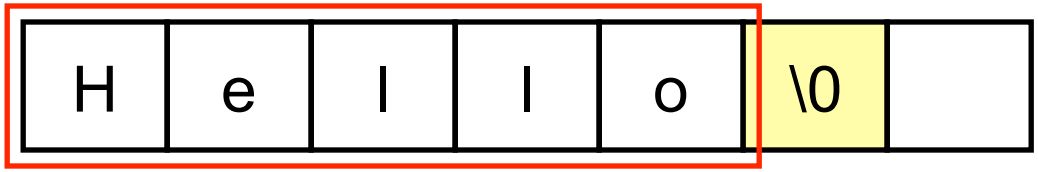
Using Classes

- You can use classes as if they were normal data types
  - Passing to functions…
  - Arrays of objects…

- Write a program that contains an array of up to 5 rectangles
- Ask user to enter height and width
- Print a table containing the dimensions/area of each + total area

String Class

- We have seen C-Strings already
  - Working with them requires knowledge of their underlying representation
  - Can be cumbersome to work with



C++ provides a built-in class for us to use instead

String Class

- Like all *built-in* C++ classes, the string class is lower case.
- The class defines several useful constructors and append methods to build strings.

```
int main() {
  string s1, s2;
  string s3 ("Welcome to C++");
  s1.append("Hello World");
  s2 = "Hello CMPS 147";
  cout << s1 << endl;
  cout << s2 << endl;
  cout << s3 << endl;
}
```

```
> Hello World
> Hello CMPS 147
> Welcome to C++
```

String class: Append

- Adding to existing strings is quite easy

```
int main() {
  string s1, s2;
  string s3 ("Welcome to
  s1.append("Hello World"
  s2 = "Hello CMPS 147";
  cout << s1 << endl;
  cout << s2 << endl;
  cout << s3 << endl;
  s1.append(" - great to be here!");
  cout << s1 << endl;
}
```

```
> Hello World
> Hello CMPS 147
> Welcome to C++
> Hello World - great to be here!
```

String class: Assign

- Re-Assigning the contents of the string can be performed quickly

```
int main() {
  string s1("Hello");
  cout << s1 << endl;
  s1.assign("Goodbye");
  cout << s1 << endl;
}
```

```
> Hello
> Goodbye
```

String manipulation

```
int main() {
  string s1("Welcome to C++");
  cout << s1.at(5) << endl;
  s1.erase(7, 3);
  cout << s1 << endl;
  cout << s1.empty() << endl;
  s1.clear();
  cout << s1.empty() << endl;
}
```

```
> m
> Welcome C++
> 0
> 1
```

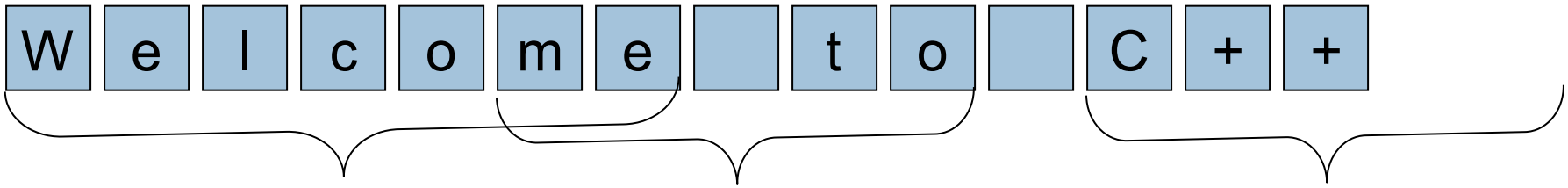
String manipulation

```
int main() {
  string s1("Welcome to C++");
  cout << s1.length() << endl;
  cout << s1.size() << endl;
  cout << s1.c_str() << endl;
  string s2("Welcome to C");
  if ( s1.compare(s2) != 0 )  {
     cout << "C and C++ are NOT the same!" << endl;
  }
}
```

```
> 14
> 14
> Welcome to C++
> C and C++ are not the same!
```

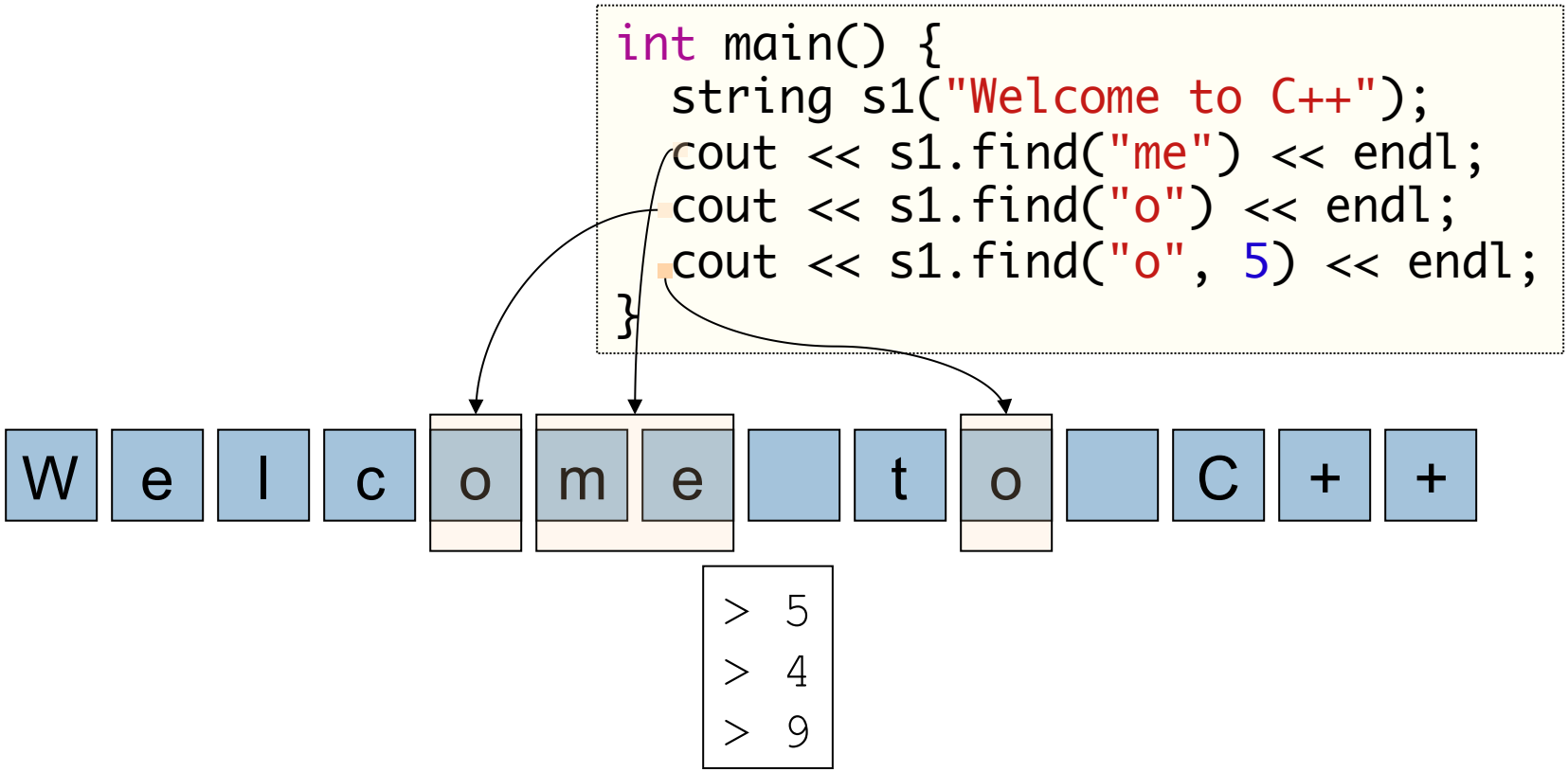
Substrings

```
int main() {
  string s1("Welcome to C++");
  cout << s1.substr(0, 7) << endl;
  cout << s1.substr(5, 5) << endl;
  cout << s1.substr(11, 12) << endl;
}
```




```
> Welcome
> me to
> C++
```

Finding substrings



Insert and Replace

```
int main() {
   string s1("Welcome to C++");
   s1.insert(11, "ANSI ");
   cout << s1 << endl;
   s1.replace(11, 8, "Java");
   cout << s1 << endl;
}
```

```
> Welcome to ANSI C++
> Welcome to Java
```

Numeric Conversion

```
#include <iostream>
#include <sstream>
using namespace std;

int main() {
  string s1("14.5");
  cout << s1 << endl;
  double n1;
  stringstream ss;
  ss << s1;
  ss >> n1;
  n1 += 5;
  stringstream ss2;
  ss2 << n1;
  string s2 = ss2.str();
  cout << s2 << endl;
}
```

Print string to stream

Read number from stream

```
> 14.5
> 19.5
```

Reading a string from cin

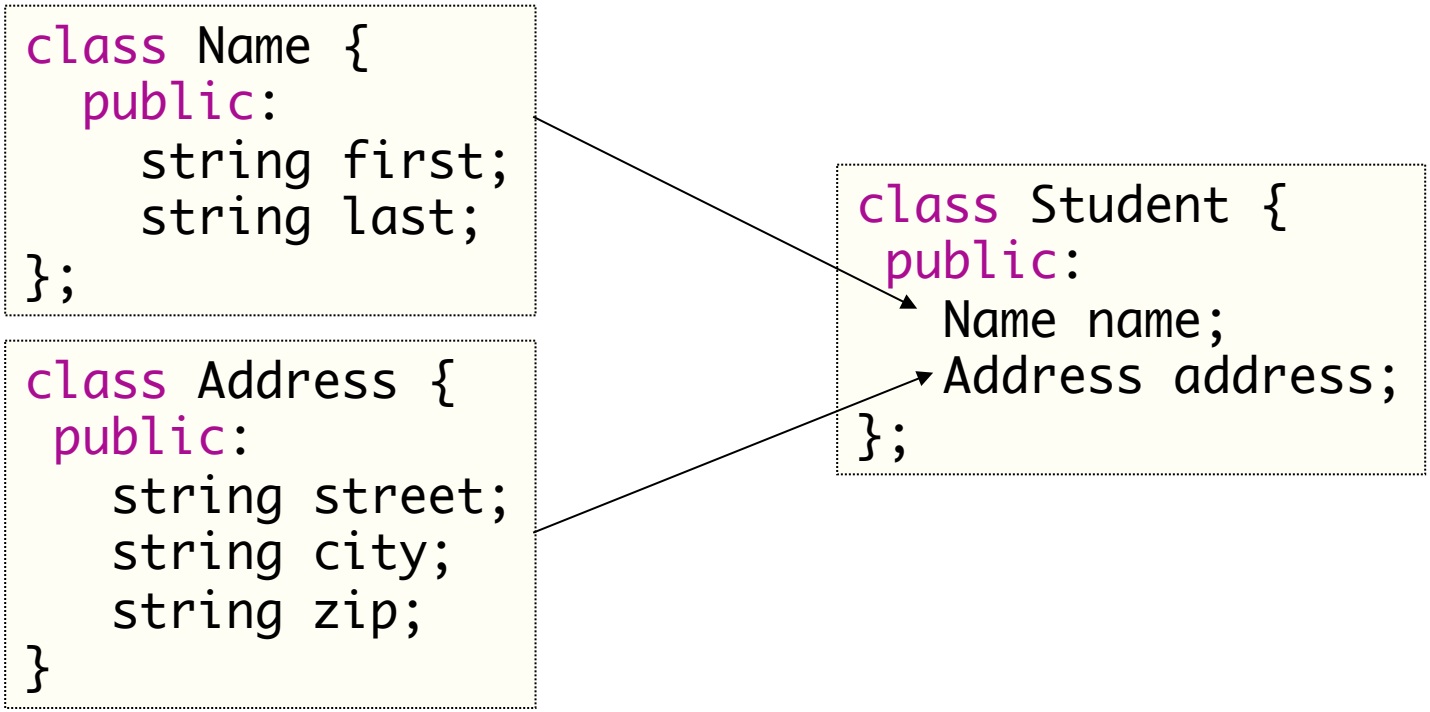
```
#include <iostream>
#include <sstream>
using namespace std;

int main() {
  string s1;
  cout << "Enter a string:  ";
  getline(cin, s1);
  cout << s1 << endl;
}
```

Don't use cin.getline!!!
That is only for C-Strings

Composition

- Objects can have objects within them:



Example: Student

- Enhance the Name class to provide:
  - Encapsulation
  - getFullName() : string
  - getLastFirst() : string
  - getEmail() : string
- Enhance Address
  - getSingleLine() : string
  - getMultiLine() : string
- Enhance Student
  - isRoomate(Student other) : bool

Elements of good design

- Cohesion:
  - Classes should describe **single** entity
  - *i.e. don't combine students and staff info in the same class….*
- Consistency:
  - Naming conventions
- Encapsulation:
  - Expose as little as possible as public
- Clarity:
  - "Contract" is easy to understand.
  - No ordering between method calls

Lab 7

- Write a program that reads two strings from the user (string class)
- Determine if the two strings (A & B) are *anagrams*
  - Anagrams are words that contain the same letters, in any order

Recommended Solution:

- Write a function (countChars) that accepts a string and a character as parameters. Returns the number of times the character appears in the string
- Check for anagram using the following steps
  - Assume you have two strings – A and B
  1. If A is not the same length as B, not anagrams
  2. Convert A and B to all lower case.
  3. For **each** letter in A, call your countChar on A and B. If the results do not match, stop

If each letter in A returns the same number in B, then they are anagrams.

Note – you probably won't need to create your own classes for this lab….