PIC 10A

Lecture 5: Using Classes

More about the string class

We have already seen the `length` and `substr` member function of the string class. Let us look at some others.

insert

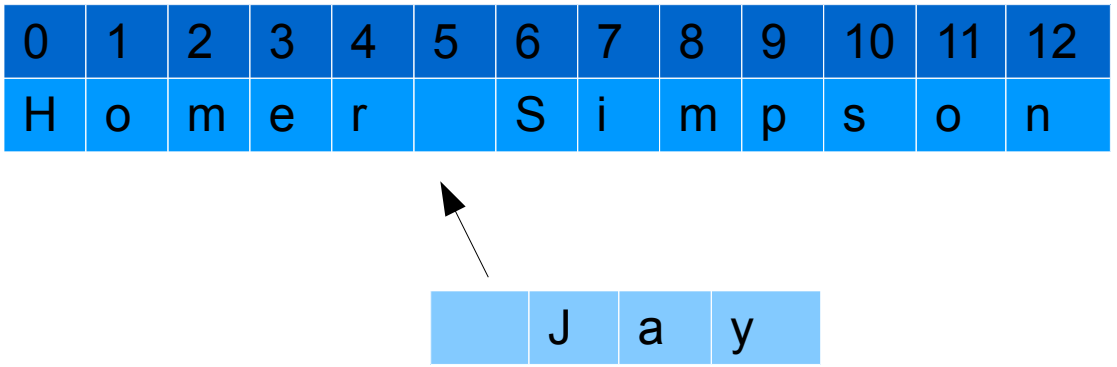
```
my_string.insert(location,some_string);
```

Inserts string some_string into string my_string starting at `location` slot.

insert

```
string s1="Homer Simpson";
string s2=" Jay";

s1.insert(5,s2);
```



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
H	o	m	e	r		J	a	y		S	i	m	p	s	o	n

erase

Erase

```
s.erase(i,n);
```

Erases a substring starting at slot i and of length n.

```
string s="Homer Simpson";
cout << "The length of s is: " << s.length();
```

0	1	2	3	4	5	6	7	8	9	10	11	12
H	o	m	e	r		S	i	m	p	s	o	n

```
s.erase(5,8); // s is now "Homer"
```

0	1	2	3	4
H	o	m	e	r

```
cout << "The length of s is now: " << s.length();
```

find

Syntax:
`s1.find(s2, i)`

Returns the index of the slot where substring `s2` starts inside `s1`. The search starts at slot `i`.

Example:

```
int location;
string s1="Homer Simpson";
string s2 = "S";
location = s1.find(s2,0);
```

Returns starting position of substring “S” in “Homer Simpson”, starting search from position start of s.

Example

Our goal is to write a program that reads an entire line of input from the user. Then it finds the first occurance of the phrase “The Simpsons” and then replaces it by “Futurama”.

See example in the examples section of our class web page.

User defined classes

string class is a standard built in class for C++.

We can also use classes that have been created by other programmers.

The authors of our book have written some classes for us to practice with.

- The `ccc_time` timekeeping class
- The `ccc_employee` employee data class
- The `ccc_win` graphics class

Later we'll learn how to write our own classes.

ccc libraries

To use the textbooks classes you have to download the libraries the authors have written. You can download the libraries from:

http://horstmann.com/bigcpp.html

You can also get all of the books examples in code form.

All the books files start with ccc which stands for Computing Concepts in C++.

User defined libraries

User defined libraries are called header files and end in .h

In general, you can name your own header files whatever you want as long as it ends .h

To use a built-in class, include the library with brackets < >

```
#include <string>
```

To use a user-defined class, use quotes “ “

```
#include "ccc_time.h"
```

Using user created classes

To declare a variable and initialize it we used a syntax like:

```
int my_number = 17;
```

For classes we use a slightly different syntax.

To initialize an instance of a class we use a constructor.

Syntax:

```
class_name object_name(initial values);
```

`Time` class

The books Time class keeps track of a time in military time. There are 2 constructors:

```
Time our_time; // Default: Sets to current time.
Time our_time (hours,min,sec);  //Sets to specified time
```

Example:

```
Time my_time(11,0,0);
```

`Time` class is the user defined data type. `my_time` is an instance of the `Time` class. Our declaration uses a constructor defined in the `Time` class to initialize `my_time`.

another way to initialize is:

```
Time my_time = Time(11,0,0);
```

Member functions

Objects usually have member functions. We have already seen how to use member functions with the string class.

The syntax is:

```
object_name.function_name(parameters if any);
```

You should be able to use a class in your program if one is given to you with a description of its member functions.

`Time` class

`Time T`	Constrcuts object T using current time
`Time T(h, m, s)`	Constructs object T with hours h, minutes m, and seconds s.
`T.get_seconds( )`	Returns the seconds value of T.
`T.get_minutes( )`	Returns the secinds value of T.
`T.get_hours( )`	Returns the hours value of T
`T.add_seconds(n)`	add n seconds to time T.
`T1.seconds_from(T2)`	Computes the number of seconds between times T1 and T2.

`Time` class example

Compute the number of seconds between now and one second before midnight.

```
Time now;
Time day_end (23,59,59);

int seconds_left = day_end.seconds_from(now);

cout << "There are " << seconds_left << " seconds  left.\n";
```

The employee class

`Employee e (n, s)`	Constructor for Employee with name n and salary s (name is a string, salary a double)
`e.get_name()`	returns the name of an Employee e
`e.get_salary()`	returns the salary of an Employee e
`e.set_salary(s)`	sets the salary of an Employee e to s

The employee class

Two methods of creating an instance of a class

Method 1

```
Employee e = Employee("Homer Simpson", 35000.0);
```

Method 2

```
Employee e("Homer Simpson", 35000.0);
```

What is the difference? Both result in same object e. In method 1 the object on the right of the = sign is bit strange...

The employee class

```
Employee e("Homer Simpson", 35000.0);

cout << e.get_name() << " has salary "
     << e.get_salary() << "\n";

cout << "After his demotion " << e.get_name()
    << "'s salary is: ";

e.set_salary(25000.0);

cout << e.get_salary() << "\n";
```