

num	0	1	2	3	4	5	6	7	8	9
value	15	10	12							7

int

str	0	1	2	3	4	5	6	7	8	9
value	today					aha				

String

C	0	1	2	3	4	5	6	7	8	9
value	a			m			z			!

character

str(variable)

```
string str;  
str = "today";
```

today

$i = 0, 1, 2, 3, 4$

length - 1

C
character

0	1	2	3	4	5	6	7	8	9
t	o	d	a	y	'\0'				

end of char array

Strings

```
char char_array[10];
```

```
char_array[0] = 'a';
```

```
.....
```

```
char_array[5] = '\0'; // \0 means that it's the end of an array character.
```

```
string str_var;
```

```
str_var = "aegioufzxn";
```

values

a	e	g	i	o	'\0'				
---	---	---	---	---	------	--	--	--	--

index

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Strings

- C-style strings are null-terminated arrays of `chars`

- e.g. `char my_string[100];`

- To get the length of a c-style string:

```
int length = 0;
while( my_string[length] != '\0' )
{
    length++;
}
```

- To copy a c-style string:

```
int i = 0;
while( my_string[i] != '\0' )
{
    other_string[i] = my_string[i];
}
other_string[i] = '\0';
```

The string class

- These operations are wrapped up in standard functions
 - Used all the time, no sense rewriting the code
 - `strlen`, `strcpy`, `strcmp`, etc.
- But c-style strings are still inconvenient
 - Arrays are fixed size, so you have to always make sure the array is bigger than whatever text is coming in
- That's why C++ introduced the class `string`

The string class

- String variables don't have to specify a size

```
string s = "This is my string";
```

- A string variable can take values of different sizes

```
string s = "short";
```

```
s = "loooooooooooooooooooooooooooooooooooooooooong";
```

The string class

- Strings have built-in functionality

- Length of a string

```
string s = "Jujufruit";  
cout << s.length();           // prints 9
```

- Copy a string

```
s2 = s1;
```

- Append to a string

```
string s = "You win";  
s.append( " a boot to the head" );
```

- Find a sub-string, get a sub-string

```
string s = "Rapunzel";  
int pos = s.find( "zel" );    // returns 5  
cout << s.substr( 2, 5 );    // prints "punze"
```

0 1 2 3 4 5 6 7 8 9 10 11 12
this semester is almost over
↑
next 12 elements

How Can `string` Do All This?

- Variables can't hold different size things
 - Even arrays are fixed sized
- Variables don't have functions, they just store stuff
- Any ideas?

Object-Oriented Programming

- In C++ classes provide the functionality necessary to use *object-oriented programming*
 - OOP is a particular way of organizing computer programs
 - It doesn't allow you to do anything you couldn't already do, but it makes it arguably more efficient
 - OOP is by far the dominant software engineering practice in the last two decades
- Classes combine data and functionality
 - Class members can store *structured* data, as we've seen
 - Class members can also be functions
 - Class-specific functions are called *methods*

The string class

- The string class has *private* data members to store the characters that make up a string
 - It probably uses an array, although it doesn't have to
 - It probably has ints to keep track of the size of the array and the number of characters

- The string class has *public* methods to do stuff
 - Return the number of characters in the internal storage

```
int length();
```

- Append the characters in s to the internal storage

```
void append( string s );
```

- returns the position of s within the internal storage

```
int find( string s );
```

class Poi

};

int main(

- `i++`
- `++i`

`i = 0;`

`j = 5;`

`++i; // updated value of i is 1; the stored value is 1;`

`j = j + i; // j values would be only depends on the store i
value => j = 5 + 1 = 6;`