4 string

1. Structs. The data type struct. Structs as parameters.
2. Arrays. array<>. Multidimensional arrays. array as parameter
3. **Text, strings of chars. string. string as parameter.**
4. Examples

string is for text

- char is for saving single ASCII letters

  ```
  char c = 'w';
  ```

- arrays<char, 256> astring;
  has many inconveniences, one of them is strings of chars have quite variable lengths but array<> are static-fixed size

string

- strings are specialised arrays for handling sequences of chars (no limit, flexible, size)

```
string s = "Juan";
```

```
string name;
string address = "Larios, 6, Malaga";
name = "Juan";
```

string

- string is a kind of *object* and has methods:
  - knowing it size:
    `s.size() or s.length()`
  - `s.substr(pos, len)` …
  - (Go here for many others)

input

- Arrays can't be read from keyboard (as a whole)
- … (nor printed on screen)

```
array<float,10> a;
cin >> a;
```

- You need to use for loops to iterate over each element to read/print them

input printing string

- You CAN

```
string s;
cin >> s;          // read only next word
cout << s;

getline(cin, s); // reads until \n
getline(cin, s, '\t'); // reads until \t
```

enlarging

- You can't write in at positions not already occupied of the string.
- To enlarge them you must use specific operators that enlarge them

adding chars

```
string toLowercase(string s)
{
    string r;
    for (int i = 0; i < s.size(); ++i)
        r += tolower(s[i]);
        // r.append(1,tolower(s[i]));
    return r;
}
```

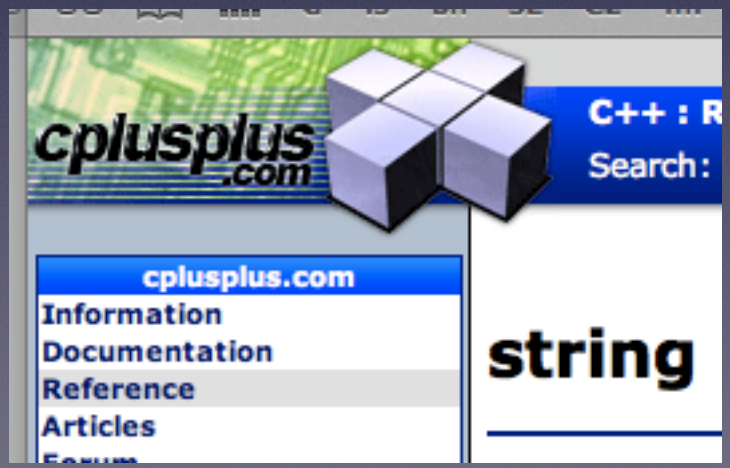
string is an object!

- Objects, as cin/cout, have methods (different syntax), and string has many useful methods. The most important are: **string s;**
  - `s = s1 + s2; s += s3;`
  - `s.length() s.size()`
  - `s.find(…)`
  - `s.substr(…)`
  - `s.replace(…)`
  - etc

ref

- To know the many properties the string object has, visit:

  http://www.cplusplus.com/reference/string/string/



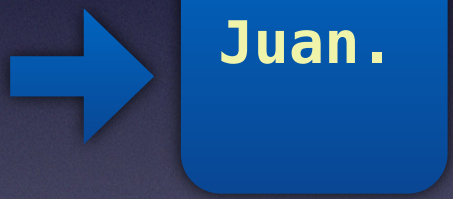
function returning a string

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

string reverse(const string s);

int main() {
    string s = "Juan";
    cout << reverse(s) << endl;
    s.append(".");
    cout << s << endl;
    return 0;
}

string reverse(const string s) {
    string r;
    for ( int i = s.size()-1; i >= 0; --i )
        r += s[i];
    return r;
}
```



Exercise

- Build a function that returns the file name from its complete path

/Users/me/Desktop/fname.cpp  fname.cpp

use
- s.rfind('/')
- s.substr(pos, count)

```
string fileName(string path)
{
    …
    return …;
}
```

efficiency issues

- Passing parameters string or structs by copying them, by value, or returning them as result of functions is, in general, convenient, but we must know it has a cost in time.
- For our exercises and problems this cost is irrelevant, but it were, consider passing structs and objects (like string) as parameter by reference& (perhaps const)

```
void toUpper(string& s);
int countWords(const string& s);
```