CS 180: Intro to C++

Note Title 8/29/2011

Announcements

- Syllabus
- Lab tomorrow
- HW 1 soon

Resources for this class

- Text book
- Transition guide (look for pdf on webpage)
- cplusplus.com
- Tutoring & office hours

This course: data structures in C++

First, C++. (More on that next.)

But – what is a data structure?

Container for data

plus constrained way to interact

- trees (sorted)
- list
- dictionary
- array
- set
- tuple
- heaps
- graphs

Why you should care about them:

- Many ways to solve a problem

<u>Goals</u>: ① Correct

② Fast

③ Efficient ⇝ space

↳ Data structure choice is key!

(And you will use them!)

C++ versus Python

High level versus low level.

readable

closer to machine code

Interpreted versus compiled.

compile + then run executable

Dynamic versus static typing

int x;
x = 5;
~~x = "Hello";~~

Why learn C++?

- faster
- ubiquitous
- understand low level details
- control

Comparison

Python

```python
def gcd(u, v):
    # we will use Euclid's algorithm
    # for computing the GCD
    while v != 0:
        r = u % v      # compute remainder
        u = v
        v = r
    return u

if __name__ == '__main__':
    a = int(raw_input('First value: '))
    b = int(raw_input('Second value: '))
    print 'gcd:', gcd(a,b)
```

import libraries

C++

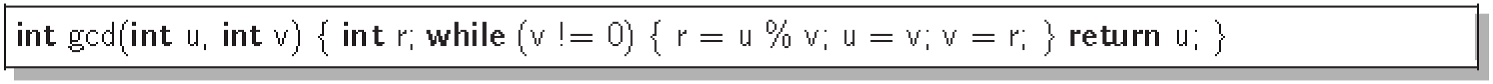
```cpp
#include <iostream>
using namespace std;

int gcd(int u, int v) {
    /* We will use Euclid's algorithm
       for computing the GCD */
    int r;
    while (v != 0) {
        r = u % v;    // compute remainder
        u = v;
        v = r;
    }
    return u;
}

int main( ) {
    int a, b;
    cout << "First value: ";
    cin >> a;
    cout << "Second value: ";
    cin >> b;
    cout << "gcd: " << gcd(a,b) << endl;
    return 0;
}
```

White space

- returns, tabs, etc. are ignored in C++



```
int gcd(int u, int v) { int r; while (v != 0) { r = u % v; u = v; v = r; } return u; }
```

(Recall that these were very important in python)

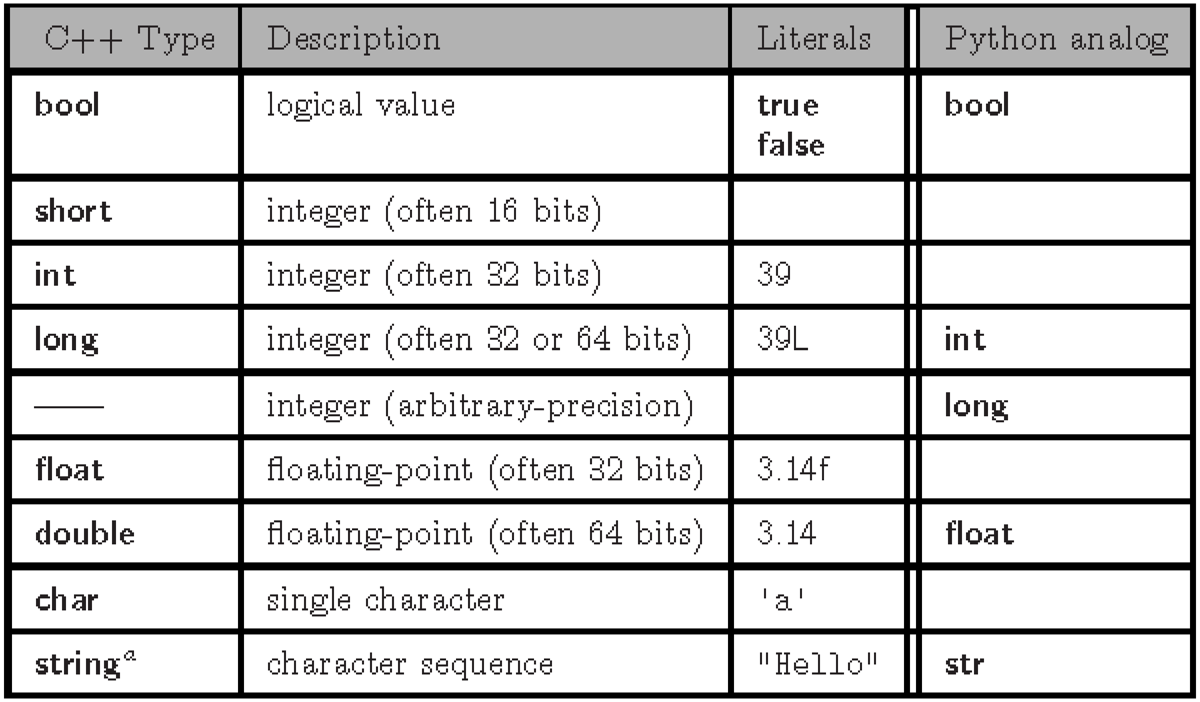
Here, we use () and {} to mark loops, booleans, etc.

Compiling

- In Python, you save code as gcd.py & then type "python gcd.py" to run it.

- In C++:
  - Save as gcd.cpp
  - type "g++ -o gcd gcd.cpp"
  - type "./gcd"

Data Types



C++ Type	Description	Literals	Python analog
bool	logical value	**true** **false**	**bool**
short	integer (often 16 bits)		
int	integer (often 32 bits)	39	
long	integer (often 32 or 64 bits)	39L	**int**
——	integer (arbitrary-precision)		**long**
float	floating-point (often 32 bits)	3.14f	
double	floating-point (often 64 bits)	3.14	**float**
char	single character	'a'	
string[a]	character sequence	"Hello"	**str**

Data Types (cont)

- Ints can also be unsigned:
  instead of ranging from $-(2^{b-1})$ to $(2^{b-1}-1)$,
  go from 0 to $2^{(b-1)}$.

- Strings and chars are very different.

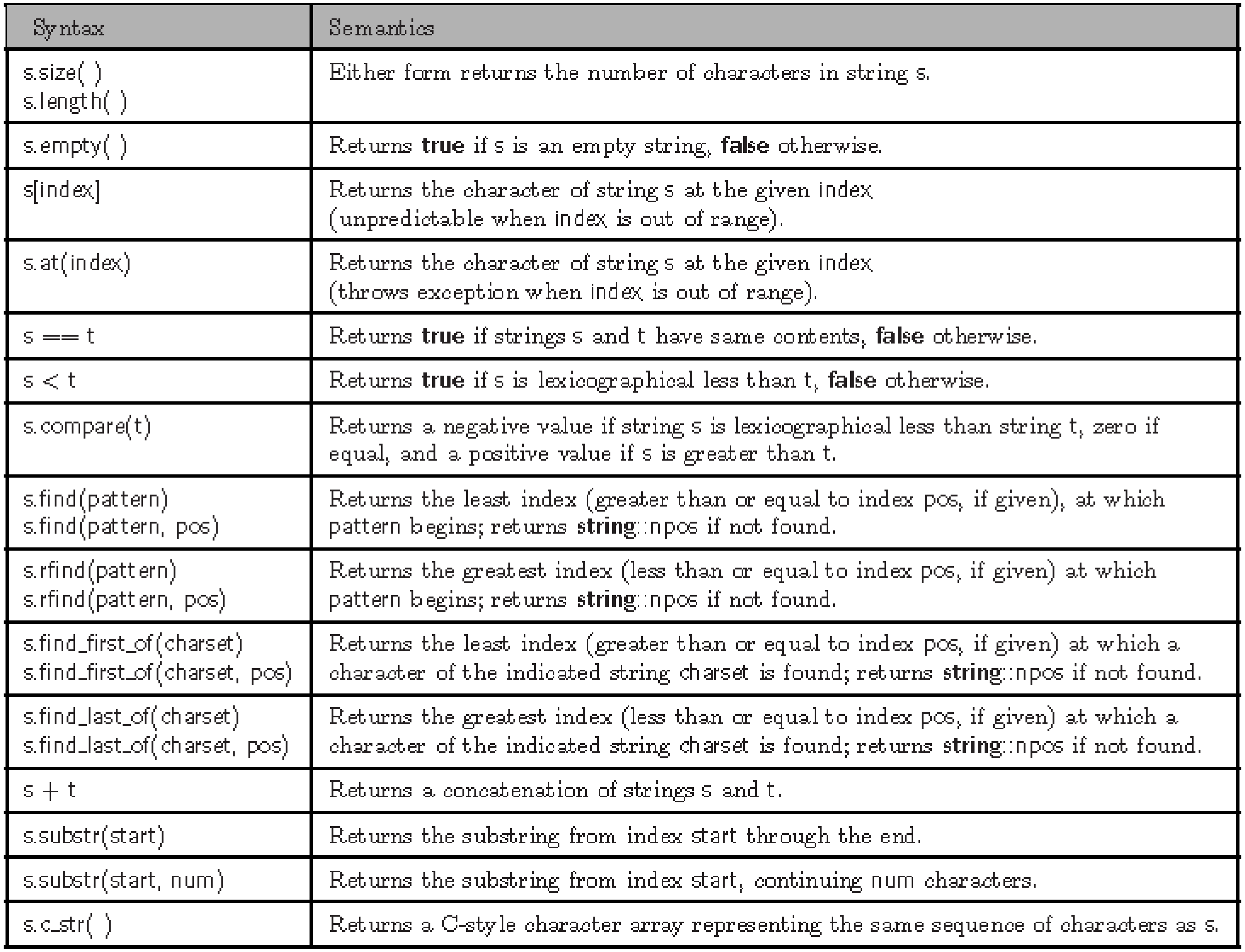
Char versus string

```
#include <string>
 char a;
  a = 'a';
  a = 'h';


 string word;
 word = "CS 180";
```

Strings are not automatically included. Standard in most libraries, but need to import.

Strings



Syntax	Semantics

s.size( )
s.length( )
	Either form returns the number of characters in string s.

s.empty( )	Returns **true** if s is an empty string, **false** otherwise.
s[index]	Returns the character of string s at the given index (unpredictable when index is out of range).
s.at(index)	Returns the character of string s at the given index (throws exception when index is out of range).
s == t	Returns **true** if strings s and t have same contents, **false** otherwise.
s < t	Returns **true** if s is lexicographical less than t, **false** otherwise.
s.compare(t)	Returns a negative value if string s is lexicographical less than string t, zero if equal, and a positive value if s is greater than t.

s.find(pattern)
s.find(pattern, pos)
	Returns the least index (greater than or equal to index pos, if given), at which pattern begins; returns **string**::npos if not found.

s.rfind(pattern)
s.rfind(pattern, pos)
	Returns the greatest index (less than or equal to index pos, if given) at which pattern begins; returns **string**::npos if not found.

s.find_first_of(charset)
s.find_first_of(charset, pos)
	Returns the least index (greater than or equal to index pos, if given) at which a character of the indicated string charset is found; returns **string**::npos if not found.

s.find_last_of(charset)
s.find_last_of(charset, pos)
	Returns the greatest index (less than or equal to index pos, if given) at which a character of the indicated string charset is found; returns **string**::npos if not found.

s + t	Returns a concatenation of strings s and t.
s.substr(start)	Returns the substring from index start through the end.
s.substr(start, num)	Returns the substring from index start, continuing num characters.
s.c_str( )	Returns a C-style character array representing the same sequence of characters as s.