logical operators:

and: && or || not !(=)

addresses:

| Name | Type | Address | Value |
| --- | --- | --- | --- |
| my\_long | long | 0x7fff519b7a8c | 456 |
| p\_long | long\* | 0x7fff519b7a80 | 0x7fff519b7a8c |
| r\_long | long& | 0x7fff519b7a8c | 456 |

long &ref\_long = my\_long; // & in decl, a ref// one ref, one long (& goes with var)

| Name | Type | Address | Value | |
| --- | --- | --- | --- | --- |
| my\_long | long | 0x7fff519b7a8c | 123 | |
| ref\_long | long& | 0x7fff519b7a8c | 123 | |
| my\_long | long | 0x7fff519b7a8c | | 456 | |
| ref\_long | long& | 0x7fff519b7a8c | | 456 | |

The :: is the scope resolution operator. It gives you access to f(x)s and variables that are defined as part of a class. So string::npos is the name of a variable within the string class.

It stands for "no position", a position not found in the string.

str.length()**==**str.size(); string str= ”tiger”;

[str.at](http://str.at)(2)=> ‘g’ character; str[2]=> ‘g’ character;

[str.at](http://str.at)(2)=‘m’ => “timer”; str.substr(0,2)=“ti”;

str.push\_back(’s’)=> “timers”; temp=str;

temp.insert(str.size(),” are loud”)=> “timers are loud”

pos=str.find(‘x’,pos); => string::npos;

pos=str.find(‘e’,pos)=> 3;

E.g. "aardvark" < "ant" since the second characters 'a'<'n' because 97<110

my\_str.substr(0,4)

•  s.rfind(arg): find last of arg in s   
• s.find\_first\_of(arg) : first of any of the args in s

* s.find\_last\_of(arg): find last of any of the args
* s.find\_first\_not\_of(args): find first of any char in s that is not in arg s
* s.find\_last\_not\_of(args): find last of any char in s that is not in arg

Useful boolean member f(x)s: cin.good(): all is well in the istream cin.bad(): something is wrong with istream

cin.fail(): last op could not be completed

cin.eof(): last op encountered end-of-file cin.ignore(num\_chars\_to\_skip,stop\_char)

cin.clear()

cin >> 18.123 cin my\_int=> 18, still in a good state.

if there are two int vars for 18.123 the first one will be 18 and the second one will fail because of the ‘.’

stod(s,&pos); pointer into size type **.** stod, stol, etc. (read "string to double" or "string to long")

•  To read a single character, not skipping:   
• cin.get()•  To put that character back into the buffer • cin.putback() • To peek without removing it: cin.peek()

Single character f(x): cout.put() puts a single character into the ostream.

A *reference* is a variable declaration that is a name alias for another variable.

• it is indicated by the &(ampersand) • but it has different meanings, context!

• it **requires** initialization •  **when** you declare a reference, you have to say what it refers to •  **it** **cannot** refer to a literal, must be another object   
string dash\_str(10,’\_’) ; makes a variable dash\_str of 10 dashes

-to reverse a string you use str.reverse() =“remit”

example:

#include<fstream> // automatically open in\_file

ifstream in\_file(“my\_file.txt"); ofstream out\_file;

string file\_name;

cin >> file\_name; // out\_file created and now opened

out\_file.open(file\_name);

open(); is\_open() true if file is open; close() terminates the connection between a program and a file (flushes buffer);

Include  
#include <cassert>

• Check for successful opening of stream. If assertion is false, halt.

in\_file.open(“file.txt"); assert( in\_file.is\_open( ) ); if result is true, continue. false will halt.

fstream in\_out\_file ("file.txt",

fstream::in | fstream::out | fstream::ate);

vertical bars are bitwise or operator,  
look it up. We combine all aspects this way

ostringstream oss;

oss << fixed << setprecision(4)

<< boolalpha;

oss << 3.14159 << " is great == "

<< true <<endl; cout << oss.str(); Output: 3.1416 is great = true

seekg : input streams (g = get) seekp : output streams (p = put)

stream::beg from the file beginning

stream::end from the file end   
stream::cur from the current file position

seekg(0, ifstream::beg)

• 0 offset from the beginning. move the marker to the beginning (of the stream)

seekg(0, ifstream::end)

• 0 offset from the end. move the marker to the end (of the stream)

seekg(-7, ifstream::cur)

• -7 (7 to the left) of the current marker position.   
old exam:

-With regards to f(x)s, the const citify would be a parameter, so that a programmer know that the f(x) will not alter the given parameter

-With regards to f(x)s we would “pass by the reference,” so that the argument is not copied to the f(x) and that the f(x) can alter the past argument

-decltype(1.0+2.0) my\_var; is a double type

-In C++11, auto keyword means that the variable type is determined when you comply the program

-‘cin’ is returned when cin>>my\_int is called

-The reason to use multiple files to write C++ code is to create libraries that can easily be shared, make code easier to read and contain.

-The “FAIL\_FLAG” for cin is set when the user inputs the letter ‘a’ when we execute int my\_int; cin>> my\_int;

-When you attempt to extract more data from a stream that is in the fail state, the extraction continues to fail and the buffer is not altered.

-You can define template f(x)s in the same cpp in which they are called and in a header file.

-The advantage of the STL with regard to streams is that you interact with all streams using a similar interface.

-The \*symbol has multiple purposes-

-int a=3; a/4 will return an int

-the pointed size of C++11 depends on the os

-You should only use unsigned when you need a higher and you know that the values will not be negative (unsigned cant be negative).

-An int, a long & a double can all use the long type mod

-Two f(x)s can exists with the same name as long as at min, they have different number of arguments and there is at least one argument with a different type

-When an input streams in the ‘fail-state’ we should clear the fail flag with the .clear member f(x) and clear the input buffer wit the .ignore member f(x).

-endl; at the end of a steam flushes the buffer and adds a new line

-The proper watch to create a f(x) that does return anything is to declare it with a return type of void

-The [ ] operator for strings is different from using the

at member f(x) in that the at member f(x) treats a negative index as number of characters from the end of a string.

-C++’s default\_random \_engine eventually repeats itself and requires an initial seed.

-uniform\_int\_distribution<long> dist(0, 10); all number within its range are equally likely to be returned (also10)

-In order to make a f(x) accessible from another file you must specify all f(x) types in the declaration

setprecision(x): set # of dec poinst w/ rounding (#include<iomanip>); fixed: fixed points for floats/ scientific: use scientific notation; showpoint, noshowpoint: always use a dec point on output or only with fractions; setw(space\_cnt): min width the output occupies - must be set for every output ; #include<iomanip>) setfill(char) in a wider field, fill with char (#include<iomanip>); noskipws or skipws: do you count whitespace as a char; cin.eof() is true is the end of file character is reached; cin.clear() clears that fail state to keep going; cin.eof() is true is the end of file character is reached; use getline(cin, my\_str)

turn off skipping white space w/ cin >> x >> y >>z; with cin >> noskipws or opp with cin >> skipws

or use an input until that read exact one character: cin.get(); put it back in buffer with cin.putback() and peek without removing it: cin.peek() - can do cout.put() to put ch in ostream

s.insert(pos, “str”). s.push\_back(char). s. erase (pos, length) erases that part of the string. pos=str.find(‘x’,pos). s.append (args) adds args to end. s.replace(range, args) range is index and a length or two iterators into s. s.rfind(arg): find last of arg in s   
sdigit, isgraph(not a space but pritnable), islower, isprint (printable), ispunct, isspace, isupper, isxdigit (hexadecimal digit). tolower returns a lowercase equivalent, toupper does opp; **streams** are objects with names such as cin, cout, and cerr; #include <cassert> check for successful opening of a stream: in\_file.open(“file.txt”); asser(in\_file.is\_open());

stream::beg from the file beginning. stream::end from the file end   
stream::cur from the current file position