- Discussion Board Questions
  Project 2 - Due Thursday At 9PM
  I'll Be Around In The Evening If Something Needs To Be Reverted To Draft Mode
  Everything Submitted By 9 PM Will Be Locked At 9 PM
  After 9 PM, If You Wish To Make Changes, Send Them To Me Via Email And They Will Be Marked Late
  Late Work Will Be Accepted On The CCLE Until Friday At 7 AM
  At That Point, The Assignment Will Be Hidden Until Grades Are Posted
  I Appreciate Your Patience While The TA's And I Finish This

- Capitalization Matters. "yes" Should Be Treated As An Error
- Use .zip Format. Please Don't Submit In .rar Format.

I Hope To Release Project 3 Wednesday
  - Due July 18th At 9 PM
  - We Need To Discuss A Bit About Strings And String Handling

Working With Functions

Working with Strings

#include <string>
using namespace std;

string s = "Hello";
string t = ""
string u = "     ";
s.size();
s.length(); // the exact same thing the number of letters in the string ==

if (s.length() != 0)
{

If (s.size() != 0)
{

}
If (s == "")
{

}
if (s == "Yes")
{

}
// code that walks a string letter - by - letter
// loop process each letter

string s = "Hello";    /// 5
for( int i = 0; i < s.size(); i++ )  /// counter is l
for( size_t i = 0; i < s.size(); i++ )
// warnings int is a signed quantity + & -
// size of a string is an unsigned quantity 0 or more
{  //I - 0, 1, 2, 3, 4
    // first letter is in position 0
    // last letter is in position .size()-1
    // grab an individual letter from a string....
    char letter = s[ i ];  /// square brackets offset to a particular
    // no checking... at runtime bad value...
    char letter = s.at( i );  /// exact same thing does bounds checking..
    if (letter == "H")
    {
        // is this letter in the string....
    }
    if (letter == 'H')
    {
        // is this letter in the string...
    }
    switch( letter )
    {
        case 'A':
        case 'E':
case 'E':
  case 'I':
  case 'O':
  case 'U':
  case 'Y':
    cout << " a vowel " << endl;
    break;
  default:
    cout << " not a vowel " << endl;
    break;

  } 
  cout << "the letter at position "  << I << " = " << letter << endl;
 }

char anA = 'A';  // numerical equivalent to int 65
anA.size( );
anA.length( );  // do not work....
string someString = "A";
someString.size( );
someString.length( );
/// does not have the value 65....

char bad = 'AA';  // fail...
string longerString = "AA";

Goals: supplying you some code that turns
"51" ----> int value of 51
"1000" ----> 1000

'5' ---->  5
char five = '5';  ----> value 5
int value = five - '0';

Convert digit sets of characters into their int equivalent
"21" ----> value 21
    loop that works letter by letter
    accumulate each digit    - '0';
### Parameter Passing Schemes

**CALLER**
The pile of code that invokes the function

**CALLEE**
The pile of code that gets called

<table>
<thead>
<tr>
<th>Pass-Declared As:</th>
<th>Called As:</th>
<th>&quot;Very Safe&quot; Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>base 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>value 2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2*10 + 1</td>
<td></td>
</tr>
<tr>
<td>&quot;210&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>By-Value</strong></td>
<td><strong>Declared As:</strong></td>
<td><strong>Called As:</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>void foo(int i);</code></td>
<td><code>void foo(int &amp; a);</code></td>
<td><code>foo(i);</code></td>
</tr>
<tr>
<td><code>void foo(int);</code></td>
<td><code>foo(j);</code></td>
<td><code>foo(d);</code></td>
</tr>
<tr>
<td><code>foo(12);</code></td>
<td><code>foo('A');</code></td>
<td><code>foo(j);</code></td>
</tr>
<tr>
<td><code>foo(j+1);</code></td>
<td></td>
<td><code>foo(d);</code></td>
</tr>
</tbody>
</table>

- Nothing The Callee Does Affects The Caller's Variable Value
- The Only Communication Between The Caller And The Callee Will Be With The Return Argument, If Not void And Captured By The Caller
- What Is Sent Is An "R-Value" Of The Desired Type
- What Arrives Is An Independent Copy Of The Caller's Original Value
- The Callee Can Change The Value But The Caller's Variable Will Not Change
- INBOUND

---

**Pass-By-Reference**

- INBOUND - OUTBOUND

**Project 1 Original Code:**
// Code for Project 1
// Ice Cream Poll results
#include <iostream>
using namespace std;

int main()
{
    int likesChocolate;
    int likesVanilla;
    int likesStrawberry;
    cout << "How many of you folks like Chocolate ice cream best? ";
    cin >> likesChocolate;
    cout << "How many of you folks like Vanilla ice cream best? ";
    cin >> likesVanilla;
    cout << "How many of you folks like Strawberry ice cream best? ";
    cin >> likesStrawberry;
    double chocolate = (likesChocolate * 100.0) / (likesChocolate + likesVanilla + likesStrawberry);
    double vanilla = (likesVanilla * 100.0) / (likesChocolate + likesVanilla + likesStrawberry);
    double strawberry = (likesStrawberry * 100.0) / (likesChocolate + likesVanilla + likesStrawberry);
    cout << endl;
    cout << chocolate << "% liked chocolate."
    << endl;
    cout << vanilla << "% liked vanilla."
    << endl;
    cout << strawberry << "% liked strawberry."
    << endl;
    return( 0 );
}

Sample Redo With Functions #1
#include <cstdlib>   /// for exit( )
void foo( );    // bit of orchestration    // not communicating with the caller
int main( )
{
    foo( );
    return( 0 );
}

void foo()
{
    int likesChocolate;
    int likesVanilla;
    int likesStrawberry;
    cout << "How many of you folks like Chocolate ice cream best? ";
    cin >> likesChocolate;
    cout << "How many of you folks like Vanilla ice cream best? ";
    cin >> likesVanilla;
    cout << "How many of you folks like Strawberry ice cream best? ";
    cin >> likesStrawberry;
    double chocolate = (likesChocolate * 100.0) /
                     (likesChocolate + likesVanilla +
                      likesStrawberry);
    double vanilla = (likesVanilla * 100.0) /
                     (likesChocolate + likesVanilla +
                      likesStrawberry);
    double strawberry = (likesStrawberry * 100.0) /
                        (likesChocolate + likesVanilla +
                         likesStrawberry);
    cout << endl;
    cout << chocolate << "% liked chocolate." << endl;
    cout << vanilla << "% liked vanilla." << endl;
    cout << strawberry << "% liked strawberry." << endl;
}

Sample Redo With Functions #2

int chocolate( );
int vanilla( );
int strawberry( );
void adjustcout( );
void output( double chocolate, double vanilla,
double strawberry );

int main( )
{
    // large task, breaking into pieces
    // code each part separately
    // orchestration calling all the pieces
    // stitching into the program
    int likesChocolate = chocolate( );
    int likesVanilla = vanilla( );
    int likesStrawberry = strawberry( );

    double c = (likesChocolate * 100.0) / (likesChocolate + likesVanilla +
likesStrawberry );
    double v = (likeVanilla * 100.0) / (likesChocolate + likesVanilla +
likesStrawberry );
    double s = (likesStrawberry * 100.0) / (likesChocolate + likesVanilla +
likesStrawberry );

    adjustcout( );
    output( c, v, s);
    return( 0 );
}

int chocolate( )
{
    int likesChocolate;
    cout << "How many of you folks like Chocolate
    ice cream best? ";
    cin >> likesChocolate;
    return( likesChocolate);
}

int vanilla( )
{
    int likesVanilla;

cout << "How many of you folks like Vanilla ice cream best? ";
    cin >> likesVanilla;
    return( likesVanilla );
}

int strawberry( )
{
    int likesStrawberry;
    cout << "How many of you folks like Strawberry ice cream best? ";
    cin >> likesStrawberry;
    return( likesStrawberry );
}

void adjustcout( )
{
    cout.setf(ios::fixed);
    cout.precision(1);
    return;
}

void output( double chocolate, double vanilla, double strawberry )
{
    cout << endl;
    cout << chocolate << " % liked chocolate." << endl;
    chocolate = 1;
    cout << vanilla << " % liked vanilla." << endl;
    cout << strawberry << " % liked strawberry." << endl;
}

Sample Redo With Functions #3 --- DOESN’T WORK! Do You See Why??

// set the value of 3 things, not just 1...
void input( int chocolate, int vanilla, int strawberry );
void adjustcout( );
void output( double chocolate, double vanilla,
double strawberry);

int main()
{
    int likesChocolate;
    int likesVanilla;
    int likesStrawberry;
    // input only variables
    input(likesChocolate, likesVanilla, likesStrawberry);

    double c = (likesChocolate * 100.0) /
               (likesChocolate + likesVanilla +
               likesStrawberry);
    double v = (likeVanilla * 100.0) /
               (likesChocolate + likesVanilla +
               likesStrawberry);
    double s = (likesStrawberry * 100.0) /
               (likesChocolate + likesVanilla +
               likesStrawberry);

    adjustcout();
    output( c, v, s);
    return( 0 );
}

void input( int chocolate,
               int vanilla,
               int strawberry )
{
    cout << "How many of you folks like Chocolate
          ice cream best? ";
    cin >> chocolate;
    cout << "How many of you folks like Vanilla
          ice cream best? ";
    cin >> vanilla;
    cout << "How many of you folks like Strawberry
          ice cream best? ";
    cin >> strawberry;
}

void adjustcout()
{

cout.setf(ios::fixed);
cout.precision(1);
}

void output( double chocolate, double vanilla,
double strawberry )
{
    cout << endl;
    cout << chocolate << "% liked chocolate." << endl;
    cout << vanilla << "% liked vanilla." << endl;
    cout << strawberry << "% liked strawberry." << endl;
}

Sample Redo With Functions #4 --- MUCH BETTER! Do You See Why??

    // set the value of 3 things, not just 1...
    void input( int& chocolate, int& vanilla, int& strawberry );
    void adjustcout( );
    void output( double chocolate, double vanilla,
                 double strawberry );

int main( )
{
    int likesChocolate;
    int likesVanilla;
    int likesStrawberry;
    // input only variables
    input(likesChocolate, likesVanilla,
    likesStrawberry);

    double c = (likesChocolate * 100.0) /  
              (likesChocolate + likesVanilla +  
              likesStrawberry );
    double v = (likeVanilla * 100.0) /  
              (likesChocolate + likesVanilla +  
              likesStrawberry );
    double s = (likesStrawberry * 100.0) /  
              (likesChocolate + likesVanilla +  
              likesStrawberry );
void input( int& chocolate,
          int& vanilla,
          int& strawberry )
{
    cout << "How many of you folks like Chocolate ice cream best? ";
    cin >> chocolate;
    cout << "How many of you folks like Vanilla ice cream best? ";
    cin >> vanilla;
    cout << "How many of you folks like Strawberry ice cream best? ";
    cin >> strawberry;
}

void adjustcout( )
{
    cout.setf(ios::fixed);
    cout.precision(1);
}

void output( double chocolate, double vanilla,
             double strawberry )
{
    cout << endl;
    cout << chocolate << "% liked chocolate." << endl;
    cout << vanilla << "% liked vanilla." << endl;
    cout << strawberry << "% liked strawberry." << endl;
}

A Bit More About String
- Each string Is A Set Of Characters. For Machines Using The ASCII Code, Each Character Is One Of Those Listed In The Table Shown At:
string orderstring;
getline( cin, orderstring);
// C++ starts counting with 0
// spot 0 will be the first letter of the string
char firstLetter = orderstring[ 0 ];
// single letter
// unsigned int not exactly correct...
size_t length = orderstring.size( );
size_t length = orderstring.length( );

// loop to walk the entire string...
for ( size_t i = 0; i < orderstring.size( ); i++ )
{
    char c = orderstring[ i ];
    cout << c << endl;
}

- Turning The String Data Into int Equivalency
"123"  ---->  123

#include <cassert>
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

int number( string s, int startingposition );

int main ( )
{
    cout << number( "1900 Pico Boulevard", 0 ) << endl;
    // prints: 1900

    assert( number("1912 Pico Boulevard", 0)==1912 ); // ----> 1912
    assert( number( "1912 Pico Boulevard", 0 )==

assert( number( "1912 Pico Boulevard", 2 ) == 12 );  // ---->  12
assert( number( "1912 Pico Boulevard", 3 ) == 2 );  // ---->  2
assert( number( "1912 Pico Boulevard", 10 ) == 0 );  // ---->  0

return( 0 );

int number( string s, int startingposition )
{
    int result = 0;  /// answer
    int length = s.length();
    if (length > startingposition &&
        s[ startingposition ] >= '0' &&
        s[ startingposition ] <= '9')
    {
        while( s[ startingposition ] >= '0' &&
               s[ startingposition ] <= '9')
        {
            /// digit converted from the letter '1'
            /// into the int = 1
            int value = s[ startingposition ] - '0';
            //   '5' ->   53-48 --> 5
            result = ( result * 10 ) + value;
            // move on down the road....
            startingposition = startingposition + 1;
            if (startingposition == length)
                break;
        }
    }

    return( result );
}

"123"       result = 123       '3'  ---->  3

bool isWellFormedThermostatString(string commands);
```cpp
bool isWellFormedThermostatString(string commands)
{
    return( true );
}

#include <cassert>

int number( string s, size_t position );

int main()
{
    string s = "456    789";
    int result = number( s, 0 );    /// 456
    assert( result == 456 );    /// code that tests
    assert( result != 0 );
    assert( result != -1 );

    s:"456    789"
    number( s, 0 ) ---> 456
    number( s, 1 ) ---> 56
    number( s, 2 ) ---> 6
    number( s, 3 ) ---> -1

    int number( string s, size_t position )
    {
        // walking off the edge of the string
        //   -1
        // go look at letter at position value
        // 0-9 , not return -1
        // continue looking at the letters after this position
        // 0-9 , return value
        //
        int value = 0;
    }
```
if (s.length() < position)  /// walk off the edge
    return( -1 );
char c = s.at( position );  /// first letter  0-9
switch( c )
{
    '0':
    '1':
    '2':
    '3':
    '4':
    '5':
    '6':
    '7':
    '8':
    '9':
        value = c - '0';
    break;
    default:  
        /// letter that is not a digit character
        return( -1 );
    break;
}

// 1. declare
// reuse the same name over again
// overloading...
// as long as the arguments number or type are different
int foo( );  /// legal...
int foo( string s );  /// legal...
int foo( int I );  /// legal...
int foo( int i, double d, string s );