Today - Lecture #2

1. C# Syntax
2. Getting Started with unix
3. Writing a Sample Program

Reminders:

Rules for Programs in CS162

1. NO Global variables (which means all variables must be defined INSIDE your functions
2. Learn about and use Arrays of Characters instead of the String class
3. All I/O should be done with the iostream library which means at the beginning of every program you will have:

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

Difference between #include <iostream> using namespace std;

vs. #include <iostream.h> ← No Longer used

1. The older .h version put all identifiers in the global namespace, making for global namespace "pollution."
2. Without the .h, all identifiers are placed in a grouping. You can leave them in this grouping, make them global, or bring them in locally into a function.
3. Saying using namespace std; brings the identifiers in globally (Which is OK!)
4. If you keep the identifiers in the grouping every time you need to read or write, the "cout" and "cin" (input) identifiers will have to be qualified by the namespace name (std in this case)

```
std::cout << "bluck" << endl;
```

std is the namespace
:: is the Scope Resolution Operator

Basic Syntax

1. output:

```
cout << variable ;
cout << "literal string";
cout << endl;
```

insertion operator

lower case L

pronounced "see out"

2. Input:

```
cin >> variable;
```

pronounced "see in"

>> is the extraction operator

3. The extraction operator skips leading whitespace, reads in the appropriate information <u>Next</u> in the input stream (also known as an input buffer)

4. If the WRONG data type is in the input buffer no input will happen and the data will need to be removed from the input buffer

Removing data from Input Buffer

1. cin.ignore();

   removes 1 character (1 byte)

2. cin.ignore(100, '\n');

   removes UP TO 100 characters or until newline is removed

3. cin.get(); char ch; ch=cin.get();

   removes one character <u>and</u> returns it (in case you want to know what it is)

4. cin.ignore(100);

   <u>DON'T</u> <u>USE</u> (ignores 100 characters or until end of file (CONTROL D) at the keyboard on unix systems. Essentially, it will remove 100 characters always.

5. cin.peek();

   Returns the next character in the input buffer. ch=cin.peek();

Data Types

1. Whole #'s: int, short, long, unsigned int
2. Real #'s: float, double
3. Characters: char
4. Boolean: bool

Variables

1. MUST be defined before they can be used
2. Local variable are garbage unless initialized
3. Identifiers must start with a letter (a-z, A-Z) and be any sequence of LETTERS, DIGITS, or UNDERSCORES (_) no dollar signs

Examples:

```
int count;
int i, j;
char initial = 'K';
char response('Y');
int Total_Length;
int total_length;
int TotalLength;
float cost = 0.0;
```

?

You can create a comma separated list

Notice two different ways to initialize!

pick a consistent naming convention!

Avoid unnecessary type conversions

Local vs. Global

```
#include <iostream>
using namespace std;
// Any variables defined outside of functions
// are called Globals
int variable; //Global
const int SIZE=42;
int main()
{
    int number=0;    //Local Variable
```

AVOID!

Constants are Great

Arithmetic Operations

1. Operators: + − * / % ← mod operator

   $\underbrace{+= \quad -= \quad /= \quad \%=}_{\text{compound arithmetic}}$

2. Form: variable = mathematic expression;

   ↑ assignment operator

3. Examples:

   a = a + 10; a+=10;

   temp

   a = a + b* 10; a += (b*10);

4. ILLEGAL: $\underbrace{a+10}$=b;

   rvalue ... a temp. Can't be on the left hand side of an = op.

5. Integer Division:

   a = b/c; //gives the quotient

   c⟌b

   a = b % c; // gives the remainder

Other Important Operators

1. Increment: ++ ← Adds 1

   can be prefix: ++i

   or, postfix: i++

2. Use Prefix unless the problem needs the postfix behavior

3. ++i ← Adds 1

4. i++ ← first grabs the value of i, stores it in a temporary, then adds 1 to the variable

   Finally, the residual value is the value of the temporary.

```
int i=10;
cout << ++i; ← outputs 11
cout << i++; ← outputs 11
cout << i; ← outputs 12
```

```
++i;
i++;
```

Conditionals (if)

1. Form: if ( ) ← parens are required!

   ↑ Put a conditional expression here

   ↑ must be lower case

2. Relational Operators:

   < > <= >= == !=

3. Logical Operators:

   && || !

   (and) (or) (not)

4. Examples:

```
if('y' == response)
        cout << "yes!";

else if('Y' == response)
        cout << "capitalized";
```

5. `if('y'==response || 'Y'==response)`

```
// single statement or
// block statement using {}
```

More Examples

Important: ϕ is false, NON-Zero is true

```
char response = 'n';
cout << "Ready to begin? Y or N : ";
cin >> response;
if ('y' == response || 'Y' == response)
{
    cout << "Let's Begin!" << endl;

    //etc.
}
```

← notice the use of compound blocks

what if I forgot a = sign?

```
else
{
    cout << "Maybe next time!"
         << endl;
}
```

this is one statement on two lines!

Common Errors

1. = vs. ==

```
if (response = 'y')
```

this assigns the value of 'y' to variable response. Then, it takes the ascii value of a 'y' (which is <u>not</u> <u>zero</u>) and checks to see if it is true or false. It will be <u>true</u> 100% of the time!!!!

Common TYPO!!

2. There are no short cuts!

```
if (response == 'y' || 'Y')
```

T or F

Always T

Always True!

3. Really.... no short cuts!

```
if ( 10 < a < 100 )
```

T or F

(1) (Ø)

Always true!

Corrected "Answers"

1. `if (response == 'y')`

   `if ('y' == response)` ← This is safer!

2. `if (response == 'y' || response == 'Y')`

   `if ('y' == response || 'Y' == response)`

3. `if (10 < a && a < 100)`

Quick Review of Logicals

1. || (OR) is true when either operand is true (or both)
2. && (and) is true ONLY when <u>both</u> operands are true
3. Usually || is used when we compare for equality (==)
4. Usually && is used when we compare !=

Preview: Loops

1. 3 loops in C++
2. while, do while, for
3. while (conditional)

```
{  //body
}
```

break;

4. do

```
{
    //body
}while (conditional);
```

5. for (initialize; test condition; increment)

```
{  //body
}
```

1 2, 5 4 3

Examples

char response = 'n';

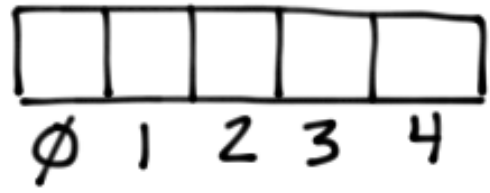
```
while ('n' == response)
{   cout << "Enter a number ";
    cin >> num ;  cin.ignore();
    cout << "You entered: " << num << endl
         << "is this correct?";
    cin >> response ; cin.ignore();
}
```

```
do
{
   cout << "Please enter a number: ";
   cin >> num; cin.ignore();
   cout << "You entered " << num
        << "is this correct? ";
   cin >> response; cin.ignore();
} while ( response != 'y' );
```

tolower(response) != 'y'
(from the cctype library)

Preview: Arrays of characters!

1. char name[5];



The size must be specified and MUST be a constant or a literal

We must allow for ONE extra element to hold the '\Ø' which is the terminating nul character, indicating the end of the used part of the array versus the unused

This array can hold 4 characters for the name and 1 for the '\Ø'

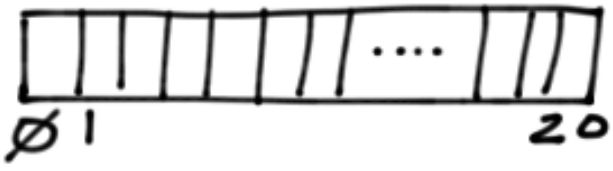
Reading in Arrays of characters

Assume: `char name[21];`

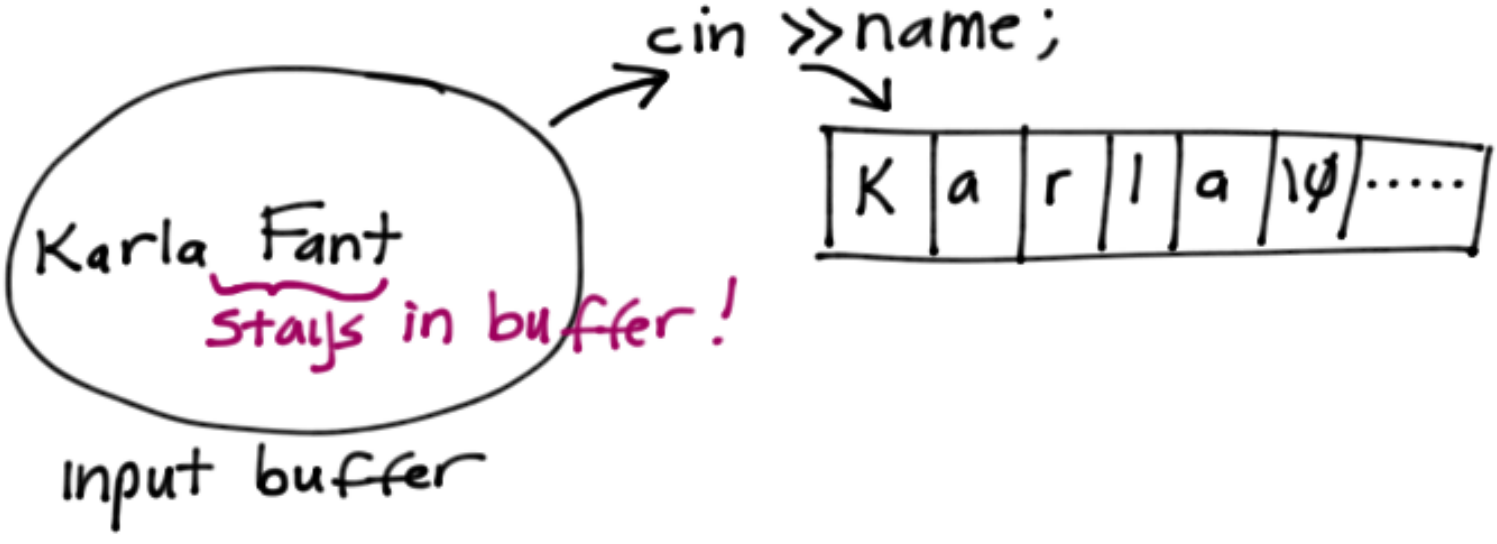
1. `cin >> name;`

   skips leading whitespace and reads until whitespace is encountered but not read

   issues: what if the user types too many characters?



   issues: what if the user types in more than one word?



Fixing these issues

1. Read 1 word without risk of a seg fault:

```
cin.width(21);
cin >> name;
```

← send in the size of the array!

2. Read in multiple words:

```
cin.get(name, 21, '\n');
```

the name of the array

the size of the array

a single character used to stop the input.

issues: Does not skip whitespace.

```
cin >> response;
//then later
cin.get(name, 21, '\n');
```

NOTHING IS READ!

Solution: REMOVE newline after Every Single Input Operation

Solution

```
cin >> response;
cin.ignore();  //or, cin.ignore(100,'\n');

//sometime later
cin.get(name, 21, '\n');
cin.ignore(100,'\n');
```

OR// `while(cin.get() != '\n');`

null bodied while loop!

Operations on Arrays

1. Only [ ] (subscript)

2. To compare, copy, count the number of characters used in an array use the cstring library

```
if ( strcmp(name,"Karla Fant") == Ø)
```

Same

Returns < Ø if name is before alphabetically

Returns > Ø if name comes after alphabetically

```
strcpy (name, "Karla Fant");
```

```
int length = strlen(name);
```

counts the number of characters up until '\Ø' ("hello\Ø" is 5)