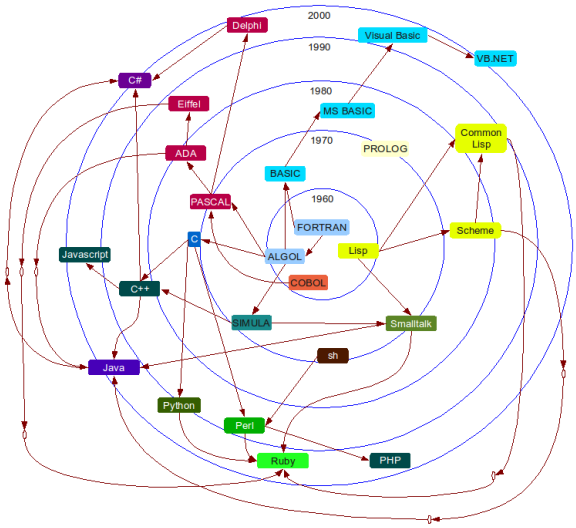
***Language Words** and **Grammar**, **Libraries**, iostream, **Variables** (Reserving RAM), **Storing (Assigning)** values in to RAM memory variables, **Manipulation** of values in RAM, Math Operations*

What is a Language



A natural/spoken language has '**words'** and a '**grammar'**.

The natural languages' **words** have a specific meaning (connected down the conceptual hierarchy to a physical reality) and it has a **grammar**, a set rules on how to combine the NOUN and VERB word types into sentences.

A computer language too has '**words'** and a '**grammar'**, but …

The **words** are referred to as '**Predefined Words'** or '**Commands'** in a programming language

The **grammar** is referred to as '**SYNTAX'** in a programming language

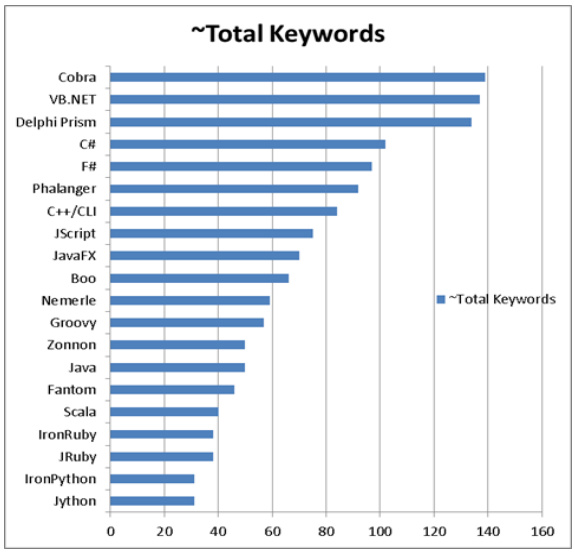
*The **Predefined Words, also called Key Words**, each as a specific meaning.*

The connection of **each key word**, down the hierarchy levels of computer languages (High –> Assembly –> Machine) to the 1's and 0's of a **specific combination** of CPU machine code instructions, is the hierarchy of links you need to grasp.

Video Watch Me: What is a programming Language: http://www.youtube.com/watch?v=58RABlprr4s
What is a computer program : http://www.youtube.com/watch?v=YCDTYIUWkQE

Here is the **list of key words** we will be learning in Procedural C++. Notice it is a rather SHORT list … sweet.

and	double	new	true
bool	else	not	try
break	enum	or	unsigned
case	false	return	using
catch	float	short	void
char	for	signed	while
class	if	sizeof	
const	#include	static	
continue	int	struct	
delete	long	switch	
do	namespace	throw	



Note: Take the CIS 250 Object Oriented Programming class to learn an additional 20+ key words.

We will also learn a few more commands that are found in **special libraries**.

Libraries of code

"Good programmers are lazy programmers, they reuse code"
Quote Professor Schwarz

All Compilers (IDEs) today come with a lot of extra prewritten code, in addition to the commands listed above. This prewritten code is stored in **libraries**. These libraries of code are there for you to use. The types of code you will find there are for common tasks. Use of the code libraries does *save* considerable *time*. These libraries are **standard** to any C++ IDE/Compiler. Here is a list of the **Standard C++ Libraries** today…

Code Library List: http://www.cplusplus.com/reference/

Libraries we will be using in this class:

•	<**cctype**>	Character handling Library
•	<**cmath**>	Math Functions library
•	<**iostream**>	Keyboard input/monitor output Library
•	<**fstream**>	Read and write a file
•	<**string**>	Strings of characters

Using a library

- First, write the name of the code library at the top of your source code
- Next, you then can use the key words that are found in that library.

Syntax: **#include <libraryName>**
Example: **#include <iostream>**

iostream Library example

The iostream library provides commands to easily write output to a monitor, and accept input to the keyboard. What does iostream mean:

i (input of character from keyboard)
o (output a character to monitor)
stream (A sequential stream of individual characters)

The ***keywords/commands*** found in the 'iostream' code library, that we will use, are:

- **cout** example cout << variableName; // used with << - 'variable content' **output** to **monitor**
- **cout** example cout << " Message "; // used with << - "message" **output** to **monitor**
- **cin** example cin >> variableName; // used with >> - **Keyboard input** into **variable**
- **endl** example cout << endl; // only used with cout - **End Line**

We will look at the other libraries mention above in later lectures. In each we will find a few more commands.
Note: All lines of code in C++ end in a ';' symbol.

Programmer Alert: Look at the direction of the >> or << symbols for cin and cout out commands.
It is a common programmer mistake to have them in wrong direction.

Symbols

In addition to the **key words** listed above, there are a handful of common **symbols** used in a language. Each symbol is an instruction to perform a specific action.

For example the + symbol means add.

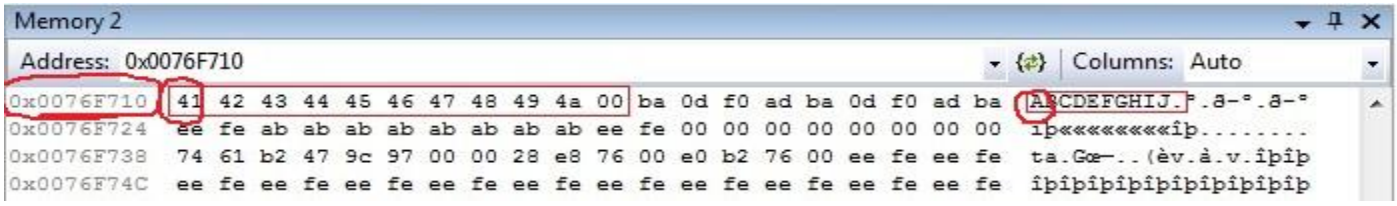
Look at the list of symbols below. Notice it is a rather SHORT list … sweet.
You should already recognize many of them.

Operator	Symbol	
Arithmetic operator	+ - * / %	Add, Subtract, Multiply, Divide, Modulus
Assignment operator	= += -= *= /= %=	Move/Assign and combinatins
Relational operator	< <= > >= !=	Your every day math operaors
Logical operator	! && !!	Not, And, Or
Increment/decrement operator	++ --	Add 1, Subtract 1

RAM / Memory – and Declaring Variable

The Random Access Memory of your PC is temporary. It can only hold information if the PC has power.

Today, there is Billions of Bytes of RAM memory available to use. ***Each byte has a Unique ADDRESS***. An address of memory is commonly referenced as a HEX number. You can use a program to inspect memory.



The memory address, hex 0x0076f710, has a hex value/content of 41, which is Decimal 65, for the ASCII char capital 'A'.

I wanna to use some RAM… yeah !!

A major command, line of code, in programs is to *request the OS Memory Manager to set aside some number of bytes of memory* to use to temporally store a value.

The memory set-aside is called a **variable**. Allocation of memory for your programs use is called **Declaring a Variable**.

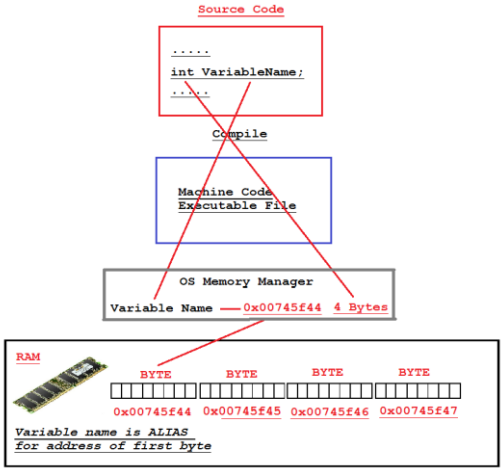
Declaring a variable - How to Reserve some memory bytes.

1) **Information programmer provides:**

   * Number of Bytes (Be sure to make it large enough to hold the amount of data)
   * Type of Data to be stored in memory location
   * Unique Name/Identifier for variable.

2) **What the OS Memory Manager does :**

   * Automatically assigns/allocates to you the use of available free memory bytes **at a certain address**.



A declaration statement has **TWO** parts: The left side and the right side.

Syntax:

DataType VariableName;
Left Side Right Side
#Bytes ALIAS to first address

Left side is always ***Data type***,
Right side is always a ***Variable name***.

Example: **int age**; // Allocates 4 bytes…

Left Side: The '**DataType**' fulfills two purposes.
It tells the computer the **number of bytes** needs and tell the computer the type of data.

Right Side: The '**VariableName**' fulfills two purposes.
Serves as an ALIAS to the **byte's address**. You do **not** have to keep track of the actual addresses allocate/used in the RAM. The OS Memory manager does that. All you have to do is use the VARIABLE NAME… NOT the Address behind the variable.

Each 'Data type' has a SIZE – a fixed number of BYTES.

Each **Data Type** has a specific number of bytes associated with it. When you declare a variable, you are requesting the OS Memory Manager to allocate that number of bytes. *NOTE: The size of the variable (number of bytes) determines the MAXIMUM amount of data that can be put into it.*

Look at the **SIZE** column below: 1 byte = 8 bits, 2 bytes = 16 bits, 4 bytes = 32 bits, 8 bytes = 64 bits.

The 'char' type only needs **1** byte to store a Character – SEE ASSCII TABLE
The 'int' type only stores positive or negative whole numbers. There is a maximum and minimum due to the **4** byte size
The 'double' type stores decimal numbers. There is a maximum and minimum value due to the **8** byte size.

Programmer ALERT..
What if you want to store a value that is BIGGER than the type size. What would happen..??
The value would be chopped off.. and thus your values would be ABSOLUTELY MESSED UP… and your program would produce garbage results. SO PAY ATTENTION to the variable data-type/size and the size of the value you want to put in it.

Data Type	Description	Size*	Min-Max Value Range*
char	Character or small integer.	**1** byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	**4** bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1 byte	true or false
float	Floating point number.	4 bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	**8** bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8 bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 *or* 4 bytes	1 wide character

Looking at the table above:

INTEGER variables – Only 4 bytes, which is 32 bits. Since it is base 2 we get 2^{32}. This is 4,294,967,292 decimal.
Since we need to store positive and negative whole numbers,
we get a range between -2,147,483,648 and 2,147,483,647.

On the ***prior page***, the hierarchy of concepts illustrates and describes what happens when you write a simple Declaration statement.

DECLARING a Variable…

As a programmer, in your source code **ALL you do** is write a simple Declaration Statement. ALL the rest is handled by the OS.

```
int age;          // Create an integer variable called age.
char letter;      // Create a single char variable called letter.
double wage;      // Create a decimal variable called wage.
```

Key Notions:

- The OS Memory manager handles the details
- A Variable is a piece of MEMORY/RAM...a bucket, a place, an area to store a 'value'.
- A Variable is allocated, reserved, set aside by a special statement called a 'DECLARATION' statement.
- Variables are reserved bytes of RAM that your program can access(read and write too)
- You can 'declare/allocate' variables for different uses...
    - Number -- integers and decimal numbers
    - character - for just one character
    - string - words or sentences...

Rules for Variable names

C++ is case sensitive. This means the lower case is different from upper case letters,
for keywords and variable names

Example:

```
int age;
int AGE;
```

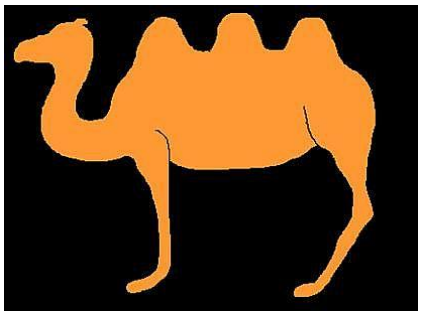
'**age**' and '**AGE**' are different variable.

Programmer Alter… A common mistake in writing your coding is to MIX up CASE.
When you compile your program, you will get an ERROR - '**Variable Undefined**'.
This commonly means you make a mistake in spelling or the CASE of letter.
This is important for the Assignment below…

Legal letters, digits in a variable name:

* In variable name you can use the letters a-z, A-Z, digits 0-9, and the underscore '_'.
* DO NOT USE SPACES.

Camel Case Naming Convention



If your variable name contains one word – use ALL lower case letters
If your variable name contains two or more words – first word all lower case, each following word's first letter Uppercase, rest of word lower case.

Example:
int age;
string first**N**ame;
char first**I**nitial;
double dollar**A**mount**R**eturned;

Create ALL your variables using the Camel Case Convention… Looks like the bumps on a camel's back.
Video Watch Me: http://www.youtube.com/watch?v=4Jta6w5OrMc

Assignment – Placing a 'VALUE' in a variable

The next statement type is the **ASSIGNMENT** Statement. (The **Container** and the **Contents** of the container)

The **assignment** statement is used to load-move-place a **value** into a variable you created with a declaration statement.

Syntax: Assignment Statement

Left Side = **Right Side**
Variable Name = Expression;

Left Side: Variable Name is of a certain type.
= Assignment Operator – move value from right side into variable on left side.
Right Side: Value, evaluate expression – single value result

Note: The '=' symbol is used in MATH CLASS as the equal operation (This is not a math class)
Note: The '=' symbol is used in PROGRAMMING as the MOVE/COPY/ASSIGN operator.

Examples:

```
int  price;              // declaration  syntax    datatype  Identifier/Name;
price = 3;               // assignment  syntax    identifier/Name =  Expression;

int discount;            // declare variable
discount =  price – 1;   //  first evaluate right side:  price – 1, result is 2.  Place '2' in variable discount.
```

Short Cut: Declare and Assign on same line.

```
Int price = 3                        // declare and assign on same line
Int discount = price – 1;            // declare and assign on same line
```

As a programmer, your code will instruct different parts of the hardware to do things.

Declaration statement	**Reserve some bytes of RAM, create a variable.**
Assignment statement	**Puts a 'value' into a variable.**

A variable can be both input and output.
Example.

```
Int age = 0;
// input
cout << "Input your age: ";        // print out message to monitor
cin >> age.                        // pause and wait for user to enter integer
// output.
cout << endl << endl << "Your age is: " << age << endl;  // print out message and value
```

In the above program, you used the variable to store a value from the user, and output the value to the monitor.

Expressions and Order of Operations

In a programming language, it is quite common to what have two variables and you want to perform an action using values in two variables.

There are many different SYMBOLS called OPERATORS that instruct the program to perform specific types of actions.

You are familiar with the 'Arithmetic Operators': +, -, / , and *.

In math, the order in which operators are executed is important. Do you remember ***PEMDAS*** ?

For example:

3 + 4 * 2 - if you performed the addition first, you would get 7 * 2, which would equal 14. Of course this is **WRONG**.

3 + 4 * 2 - if you performed the multiplication first, you would get 3 + 8, which would equal 11. Of course this is **RIGHT**.

So you can see you MUST have a fixed, consistent predetermined/set/consistent **'Order of Operations'**, in which to perform actions, so everyone gets the same results.

Using Parentheses:

You can use ( ) to change the DEFAULT order of operations:

(3 + 4) * 2 = 14. This is OK/RIGHT.

Operator Precedence Table for C++

When given an expression, the order in which the different operator symbols are executed is important.

Here is a table showing the Precedence/Order of the C++ Operators:

Do 1st	() [] -> . ::	Grouping, scope, array/member access
2	! ~ - + * & sizeof *type cast* ++x --x	(most) unary operations, sizeof and type casts
3	* / %	Multiplication, division, modulus
4	+ -	Addition and subtraction
5	<< >>	Bitwise shift left and right
6	< <= > >=	Comparisons: less-than, ...
7	== !=	Comparisons: equal and not equal
11	&&	Logical AND
12	||	Logical OR
13	?: = += -= *= /= %= &= |= ^= <<= >>=	Conditional expression (ternary) and assignment operators
14 **Do last**	,	Comma operator

Good Suggestion: Print out the above table, and have it handy when you are programming.

ASSIGNMENT Part 1 – Watch and comment on Videos

Watch these videos – and comment on what you learned for each video

a. Your first program - http://www.youtube.com/watch?v=M3jeozLUr8w

b. Data Types and Variables - http://www.youtube.com/watch?v=6_2OaaTi-Lw

c. Naming Convention - http://www.youtube.com/watch?v=4Jta6w5OrMc

d. Assignment – http://www.youtube.com/watch?v=3Iq_uFbc4L4

f. Math Operators - http://www.youtube.com/watch?v=L1z2dpCosXU

g. iostream library – http://www.youtube.com/watch?v=FJx9oZjC_2M

ASSIGNMENT Part 2

1) This code has errors. Do not delete lines, Do not add lines. Only fix lines if needed. // add comment to end of line describing fix. Compile good code, Put Good code in MS Word document with Screen print of good output.
   Note: MAC users may have to delete a few lines.
   Copy and paste the code into your main function, and run to get error list.
   Click on the error message .. it will take you to the bad line of code. Fix the code.

```
#  iostream>
using namespace std;

Int main ( )  {  / start program

    // Declare and Initialize Variables
    int height = 0;
    int feet = 0
    int inches = 0;

    // Prompt for height
    cout >> “Enter your height in inches: ”;
    cin >> Height;

    // Calculate Height in feet and inches
    feet = height / 12;
    inches = height % 12;

    /Print out height in feet and inches
    cout << “You are “ < FEET << “ feet and “ <> inches << “inches” ;

    system(pause);   // Mac user can delete this line of code
    return 0;

} // end of program
```

ASSIGNMENT Part 3

Write a program from scratch, that asks a person how tall they are.
"How many feet and how many inches are you"
"How tall are you: "

Feet: 5 ← they enter 5
Inches: 4 ← they enter 4

Calculate how many total inches high they are.
Output the results.

Example output: "You are 64 inches high"
Be sure to comment your program.

ASSIGNMENT Part 4

Write a program that inputs the values of height and width of a rectangle.
Use them to calculate area of a rectangle.

Output the Values of the height and width and area of a rectangle.
Be sure to comment your program.

ASSIGNMENT Part 5

– Answer these questions.

1) What happens when you Declare a variable ?
2) What Happens when you Assign a value to a variable ?
3) What is variable Initialization, why is it necessary ?
4) Why use comments // ?
5) Code Reuse, Why reuse code ?