Programming

Algorithm: Is a sequence of steps to accomplish a specific task.

Procedural Programming: It is focused on *actions*, also called procedures or functions.

Object Oriented Programming: Focused on *modeling* things: both their characteristics and actions. OOP is the focus of this class.

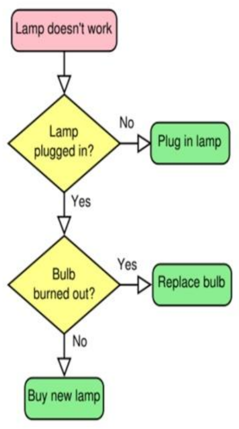
PROCEDRAL PROGRAMMING

Take CIS 118 – 'Introduction to Computer Science' to Study C++ procedural Programming.

In Procedural Programming you *focus on* ***sequence of actions***…

A Simple Process Flowchart

to understand basic idea of "Process"



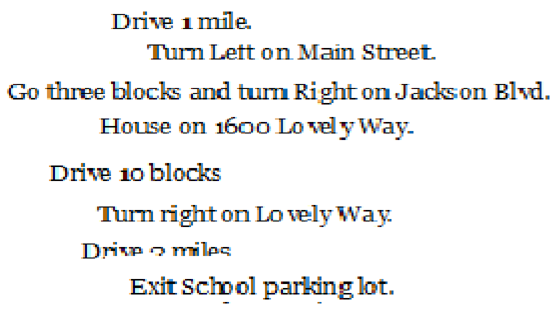
Definition: **Procedural Program** – A *complete* and *correct* 'sequence' of instructions, *actions* or events. A Process.

The key element of a program is the characteristic 'sequence', not a group, not a set, not a bunch, but a sequence.

Imagine you were to give directions from one location to another.

1) Exit School parking lot.
2) Turn Left on Main Street.
3) Go three blocks and turn Right on Jackson Blvd.
4) Drive 2 miles.
5) Turn left on Farms Road.
6) Drive 10 blocks
7) Turn right on Lovely Way.
8) Drive 1 mile.
9) House on 1600 Lovely Way.

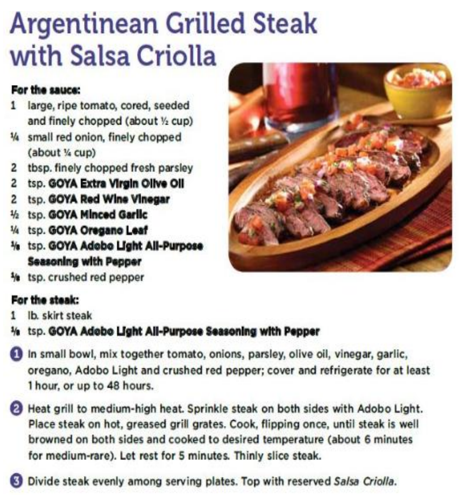
Imagine you printed out the directions… then took a pair of scissors, cut each line of directions into its own strip of paper with no number, then tossed them into the air… and then randomly pick up one at a time and followed that particular instruction. How far would the random sequence take you ?



Drive 1 mile.
Turn Left on Main Street.
Go three blocks and turn Right on Jackson Blvd.
House on 1600 Lovely Way.
Drive 10 blocks
Turn right on Lovely Way.
Drive 2 miles
Exit School parking lot.

Examples of Non Computer <u>Procedural</u> Programs:

1) **CookBook**… Recipes list the ingredients and the *order* of preparation to make a dish.
2) **Shoe Laces**… *Sequence* of steps to tie a shoe lace
3) **Ikea Furniture** Assembly *Directions* to assemble furniture you purchases.



Argentinean Grilled Steak with Salsa Criolla

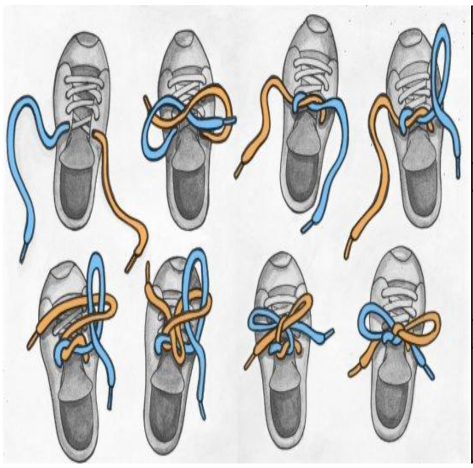
For the sauce:

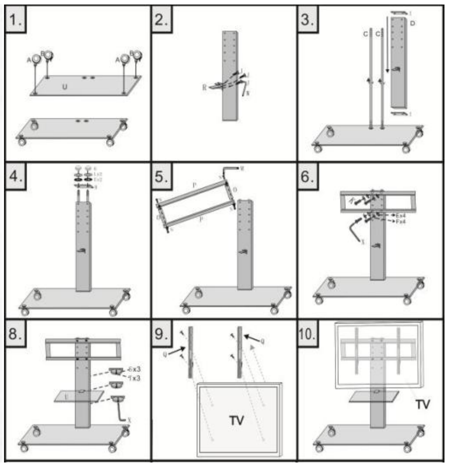
- 1 large, ripe tomato, cored, seeded and finely chopped (about ½ cup)
- ¼ small red onion, finely chopped (about ¼ cup)
- 2 tbsp. finely chopped fresh parsley
- 2 tsp. **GOYA Extra Virgin Olive Oil**
- 2 tsp. **GOYA Red Wine Vinegar**
- ½ tsp. **GOYA Minced Garlic**
- ¼ tsp. **GOYA Oregano Leaf**
- ⅛ tsp. **GOYA Adobo Light All-Purpose Seasoning with Pepper**
- ⅛ tsp. crushed red pepper

For the steak:

- 1 lb. skirt steak
- ⅛ tsp. **GOYA Adobo Light All-Purpose Seasoning with Pepper**

1. In small bowl, mix together tomato, onions, parsley, olive oil, vinegar, garlic, oregano, Adobo Light and crushed red pepper; cover and refrigerate for at least 1 hour, or up to 48 hours.
2. Heat grill to medium-high heat. Sprinkle steak on both sides with Adobo Light. Place steak on hot, greased grill grates. Cook, flipping once, until steak is well browned on both sides and cooked to desired temperature (about 6 minutes for medium-rare). Let rest for 5 minutes. Thinly slice steak.
3. Divide steak evenly among serving plates. Top with reserved *Salsa Criolla*.





Procedural Programming is concerned mainly with the ***Function****ality/actions. The code is a sequence of instructions.*

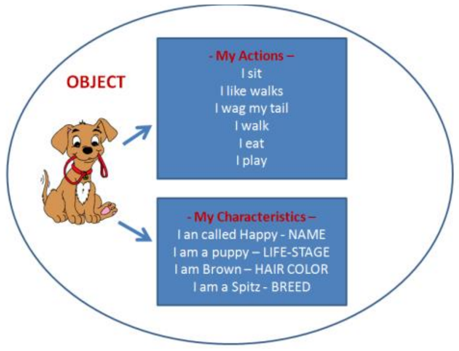
To be focused on just actions/functions, is limiting. It can make programs more difficult to write. So…

In comparison, there is another way to approach programming. You can be focused **THINGS**. This called **Object (Thing) Oriented Programming**.

Object Oriented Programming (OOP)

Take CIS 250 to study C++ OOP.

In OOP, your code is focused on Things – **THINGS have** attributes/characteristics/**properties** AND **actions**.



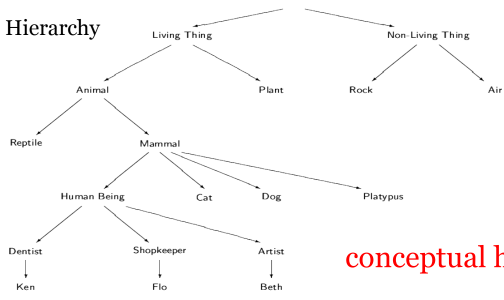
Working with and using OOP is creating a description of a category of things. The thing could be a physical entity or a mental entity. An OOP model is a class of things.

The Thing/class description/definition contains variables and functions.

- The **variables** would hold data about characteristics/attributes/features/properties of a thing.
- The **functions** would be the actions a thing could perform. Also called methods or behaviors of a class.

The idea of a '**concept**' that we discussed in the first lecture, is called a '**class**' in OOP. The methods of concept formation are used to create classes. Be sure to review the first lesson – The CROW and HOW TO THINK (Make and use concepts/classes) – It is absolutely essential you master concept formation for this OOP class.

Hierarchy



Each class/category of things is not independent, it is part of a vast set of related classes/categories. See diagram above.

The **'conceptual hierarchy'** in we discussed in the CROW lecture is called an '**Inheritance Hierarchy**' in OOP.

The CIS 250 and CIS 252 classes will study concepts (classes) and conceptual hierarchies (Inheritance) in depth. Take those classes.

Programming Languages

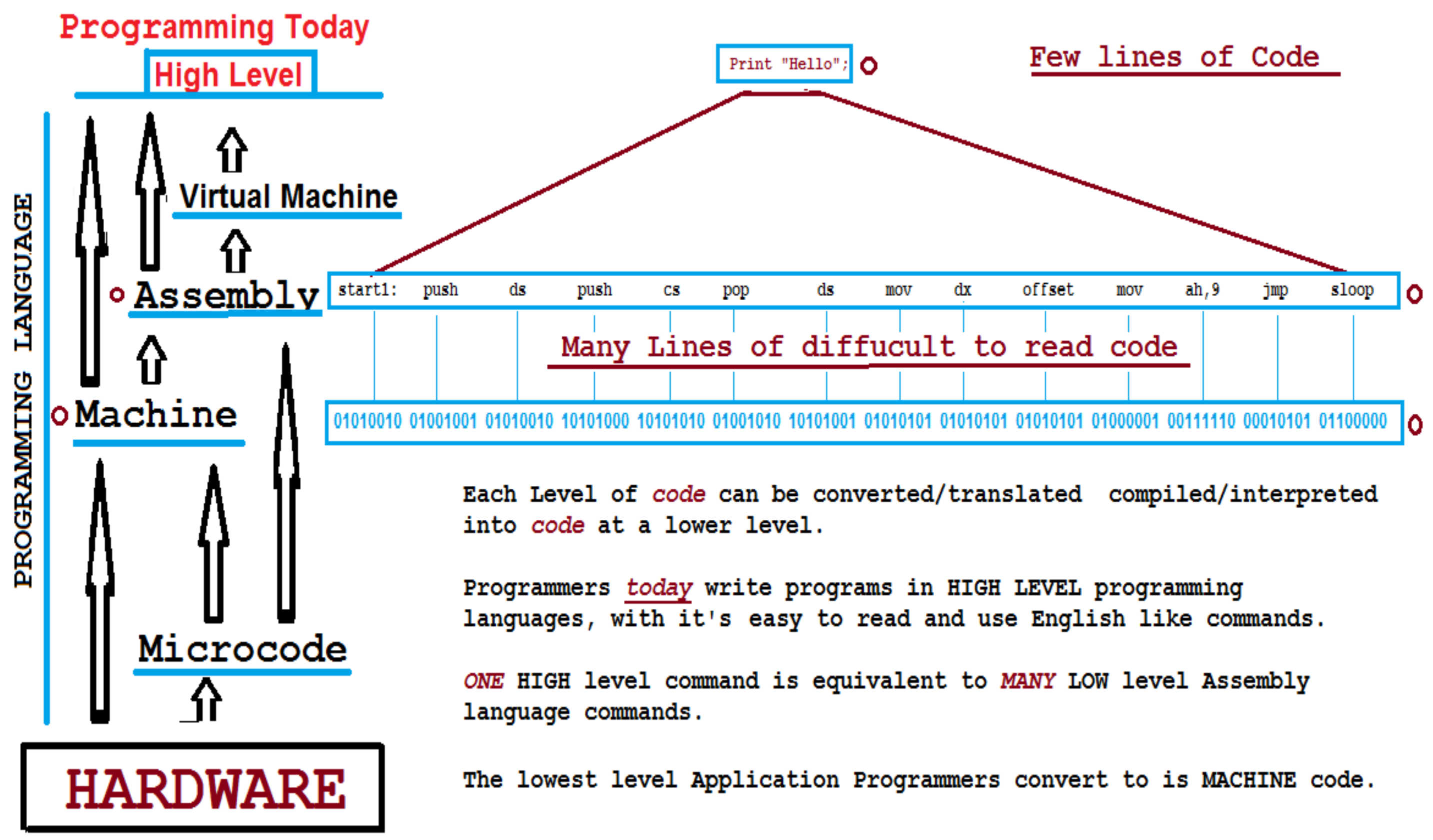
Definition: Programming **Language -** A language is used to write a sequence of instructions for the computer. It lets the programmer write data processing instructions in a symbolic/abstract manner ***without*** regard to hardware/machine-specific details. (Hierarchy of concepts)

There are many *Levels of Programming languages.* The level is associated with how close it is to a sequence of 1's and 0's, which would tell the transistors/switches, in the CPU, when to turn on or off.

Levels (Hierarchy) of Programming Languages.

What does 'levels of programs' mean to a programmer ?

- One command in a higher level, is many commands in its lower level equivalent
  - High Levels – Less line of code for you to write
  - Low levels – More lines of code for you to write.
- Writing programs with a Higher Level programming requires you to write less code



Each Level of *code* can be converted/translated compiled/interpreted into *code* at a lower level.

Programmers *today* write programs in HIGH LEVEL programming languages, with it's easy to read and use English like commands.

ONE HIGH level command is equivalent to *MANY* LOW level Assembly language commands.

The lowest level Application Programmers convert to is MACHINE code.

In C++ You will be writing source code and converting/compiling it to Machine Code.

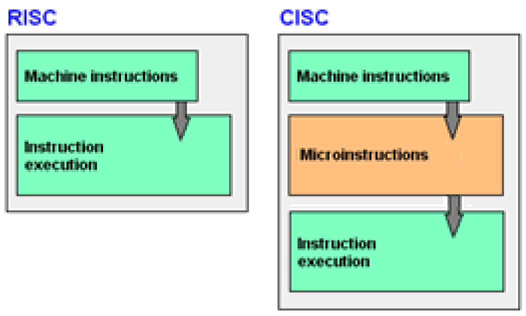
The computer read 1's and 0's of machine/microcode.

* Microcode or No Microcode Programming

You are not writing Microcode in this class

The commands the CPU can execute are called the Instruction Set Architecture (ISA).

There are two design Philosophies with CPUs'.



CISC Complex Instruction Set Computer
Only essential basic actions are hardwired. A special programming language called **microcode** is used to combine basic actions so to handle additional needed hardwired actions. Easier to design hardware.

RISC Reduced Instruction Set Computer
ALL actions hardwired, built into the hardware.
Much more difficult to design hardware.

The Application Software Programmer (you) NEVER sees microcode nor does any Microcode Programming. Microcode resides on ROM built into the CPU. It is unpublished and proprietary.

Microcode is also called Firmware.

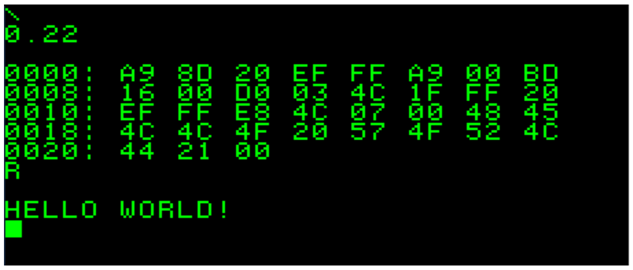
Watch This Video – CPU and Instructions - http://www.youtube.com/watch?v=N3pLvGMTeHw

Machine Language –

You are *not* doing Machine Language coding in this class

Most programmers never heard of microcode. They think that the lowest level of programming is Machine code. Machine code is the 1's and 0's for switch settings that the CPU hardware uses to cause different actions to occur. Machine code is the lowest level of programming if the CPU architecture is RISC.

If the Computers CPU architecture is CISC, then machine code may be converted into the CPU's microcode, which then throws the 1's and 0's switch setting that the CPU hardware uses to cause different actions to occur.



```
\
0.22

0000: A9 8D 20 EF FF A9 00 BD
0008: 16 00 D0 03 4C 1F FF 20
0010: EF FF E8 4C 07 00 48 45
0018: 4C 4C 4F 20 57 4F 52 4C
0020: 44 21 00
R

HELLO WORLD!
█
```

Hex code printout of Machine ***code*** for Hello World Program. Not fun to write.

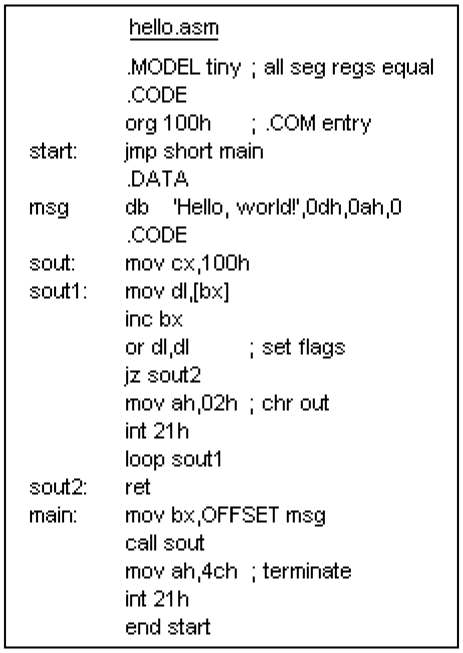
Remember HEX is shorter to print out. What really is stored is a binary version.
Binary-coded machine instructions that are specific to a CPU model or family, such as X86.
If you want to see what machine language programming is like using the program HEX Editor NEO:

Watch this video: http://www.youtube.com/watch?v=h-tubAlE9o8

Assembly Language

You are *not* doing Assembly Language coding in this class (Take CIS 242 to study Assembly Language)

Because the 1's and 1'0 of machine code are extremely time consuming and error prone to have people directly program, it was found that there was a need to make programming simpler. Thus was developed Assembly Language. Assembly Language uses words. Each word corresponds to a sequence of 1's and 0's for a machine instruction. So instead of writing all in binary code, you could write the assembly language English like words/commands instead. This makes writing and reading programs vastly easier. Easier to read and write.



```
            hello.asm
            .MODEL tiny ; all seg regs equal
            .CODE
            org 100h     ; .COM entry
start:      jmp short main
            .DATA
msg         db   'Hello, world!',0dh,0ah,0
            .CODE
sout:       mov cx,100h
sout1:      mov dl,[bx]
            inc bx
            or dl,dl       ; set flags
            jz sout2
            mov ah,02h ; chr out
            int 21h
            loop sout1
sout2:      ret
main:       mov bx,OFFSET msg
            call sout
            mov ah,4ch  ; terminate
            int 21h
            end start
```

The file in which you write your **SOURCE CODE** is called a **SOURCE FILE**. The problem now is that the computer cannot understand the ASSEMBLY language 'Words' in your source file. The computer only understands Machine Language.

What must be done is to convert/translate the Key 'words' in an Assembly program into Machine Language 1's and 0's.

This conversion process is called **Compiling** or **Interpreting**.

The resulting file containing the Machine Language code is called a **Binary File** or **Executable file**.

Above: Assembly Language source file for "Hello World" program.

Programming in Assembly use to be very popular into the 1970's. It must be noted, some programmers still use assembly because it offers greater efficiency and speed to their programs.

Assembly Language Program: "Hello World".

Video Watch Me: http://www.youtube.com/watch?v=x1znUIZ3ABE

Accompanied by song "You can do anything you want too"...

Recall, when they created an Assembly Language instruction, they took the ***MANY*** 1's and 0' of a machine instruction, and simplified it into ***ONE*** assembly instruction. (Concept)

Now, if we use the same process, we can take ***MANY*** assembly commands and combine them into ***ONE*** High Level language Instruction.

The CROW rules… makes EVERY one's life easy …if you convert many to one. The crow can handle one thing easy !

High Level Language

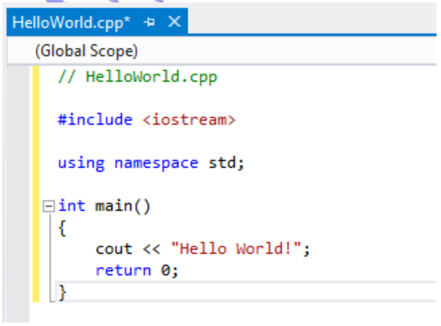
Your will be writing High Level Language programs in this class (C++)

Programming Languages are used for controlling the behavior of a machine (often a computer). Like natural languages (English, Spanish, German, etc.), programming languages conform to rules for **syntax** (grammar).

Syntax: - Using key commands and the order in which to place commands.

Each command in a high level language is very powerful. It can be translated in too many individual Assembly language commands. Likewise each Assembly language command can be translated in many 1's and 0's of an individual machine instruction set command (ISA). (See 'Levels of programming Languages' diagram above)

Today, the most versatile and popular programming languages are High Level programming languages. The language we will be using for the class is C++.


HelloWorld.cpp*
(Global Scope)

```
// HelloWorld.cpp

#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World!";
    return 0;
}
```

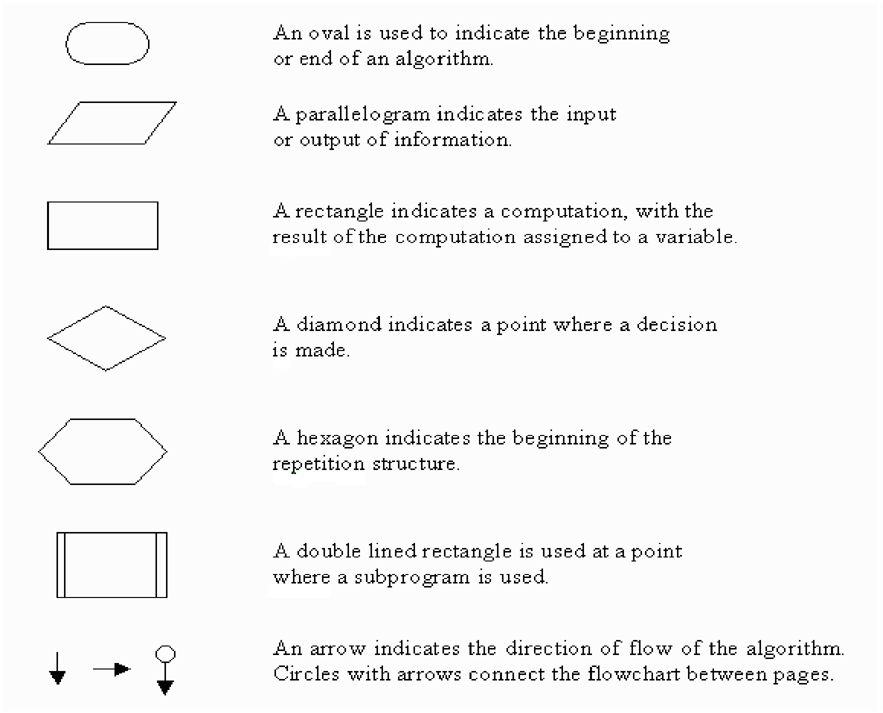
The creator of C++ is **Bjarne Stroustrup**.

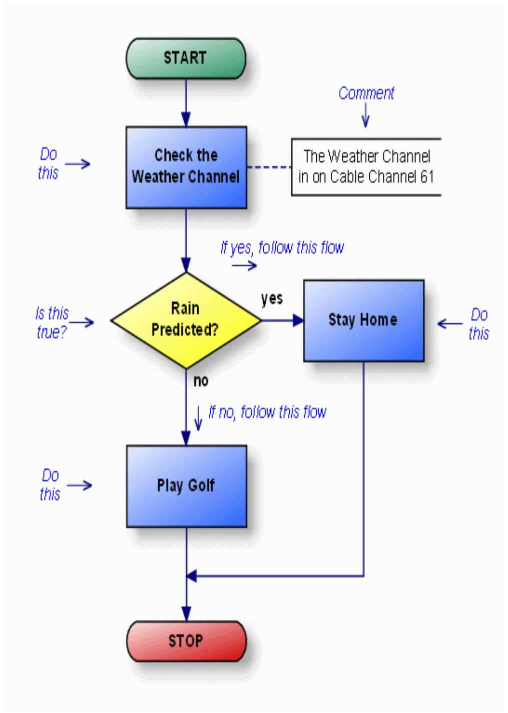
Watch This Video: Why I created C++ -
http://www.youtube.com/watch?v=JBjjnqG0BP8
'Hello World' Program for C++: Video Watch Me
http://www.youtube.com/watch?v=2sSMzRj7RTs

Flow Charts

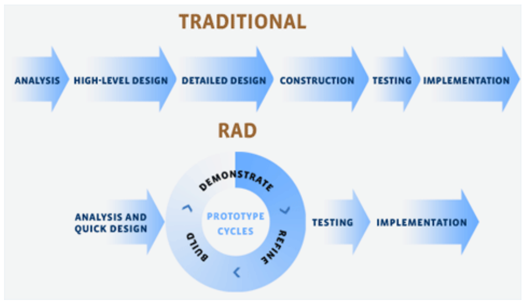
Video Watch Me - VISUAL flowcharts in MS Word: http://www.youtube.com/watch?v=WDUwqYLlAA8

Recall that a program is a sequence of instructions. There are several methods to represent the sequence. A visual or graphical method is called Flow charting. You can represent a program's action sequence with these symbols and arrows. Creating a flow chart helps you insure your code sequence is correct.


An oval is used to indicate the beginning or end of an algorithm.
A parallelogram indicates the input or output of information.
A rectangle indicates a computation, with the result of the computation assigned to a variable.
A diamond indicates a point where a decision is made.
A hexagon indicates the beginning of the repetition structure.
A double lined rectangle is used at a point where a subprogram is used.
An arrow indicates the direction of flow of the algorithm. Circles with arrows connect the flowchart between pages.



Software Development Life Cycle



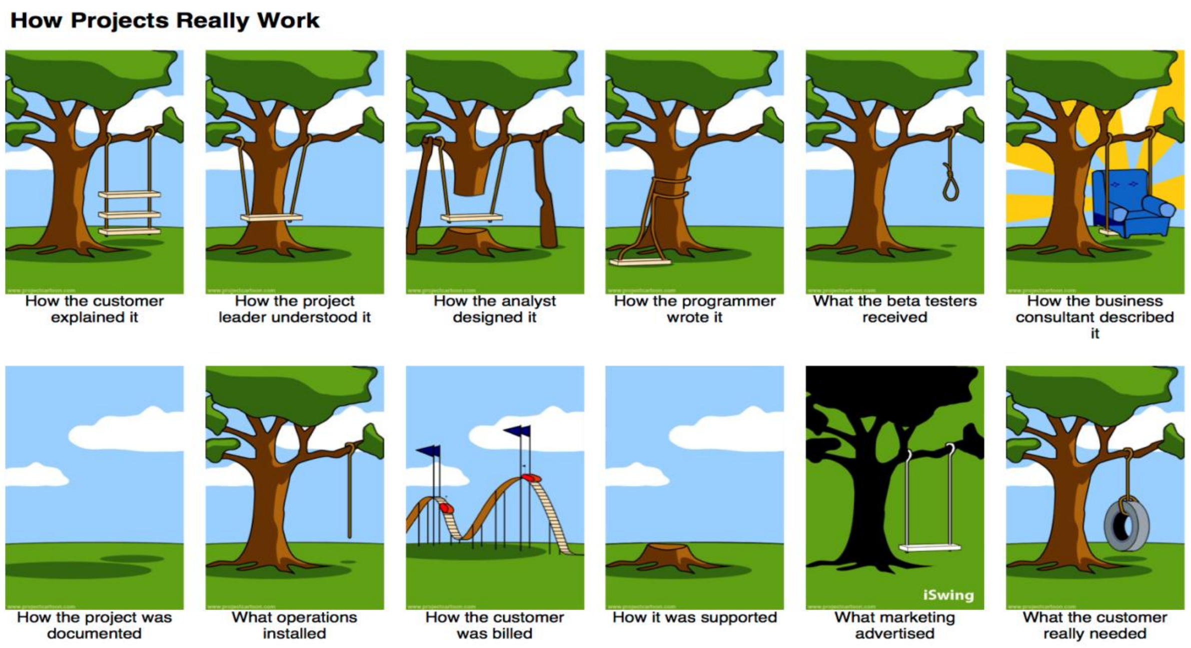
Best Practices – Millions of people have done it before you. They have noticed that certain ways of doing things significantly increase the likelihood of success. These are called best practices.

Recall, a process is sequence of steps used to achieve a specific result. There are processes for software development. The Software Development Life Cycle (**SDLC**) is a process developed to increase the odds of producing a computer program. ***SDLC is 'Best Practices'***.

Each step of the SDLC needs to be **Documented** and Seriously **Reviewed, with signature sign off**. The reason for documentation and reviews is to catch mistakes EARLY. *The earlier you catch a mistake, the quicker and less expensive it is to fix it*. Even with an SDLC, over 50% of all software projects fail. Your business can actually be rated on how well you implement an SDLC in your company, and your company may or may not receive business depending on your level of rating. Using an SDLC fully is actually considered a sign of MATURITY.

The **Traditional** SDLC process spends significant time at each step of the process. Note – The number of steps and names of each step of the SDLC can significantly vary, but they are basically all the same.

The **Rapid Application Development** (RAD) speeds up the development process. It can be more error prone.

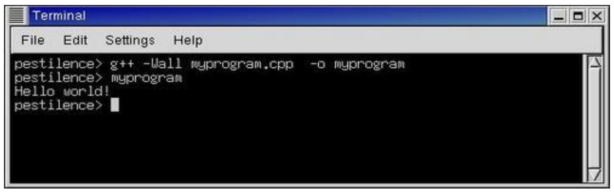

How Projects Really Work
How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

Video Watch Me - SDLC http://www.youtube.com/watch?v=TWWTE8gCC9k

Programmer Alert: When you apply for a job, part of your interview questions is to determine if the prospective company utilizes a SDLC. If so, then +1 to work for them. If not, then -10 for working for them.

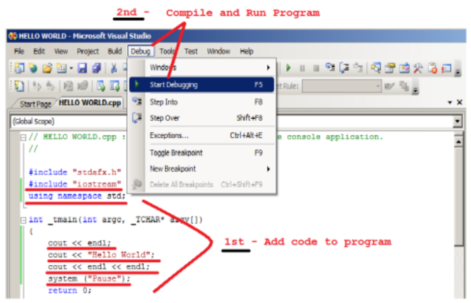
Computer Programming and Integrated Development Environments (IDE):

Today, you use a special category of programs to help you write code. It is called an Integrated Development Environment. (IDE)



```
pestilence> g++ -Wall myprogram.cpp  -o myprogram
pestilence> myprogram
Hello world!
pestilence>
```

In the old days, a programs source code was typed into a simple text file, using a Text editor. The Source code was then compiled, by using a command on a **COMMAND Line** (CLI). Some programmers still live in the old days…


2nd - Compile and Run Program
HELLO WORLD - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Windows
Start Debugging F5
Step Into F8
Step Over Shift+F8
Exceptions... Ctrl+Alt+E
Toggle Breakpoint F9
New Breakpoint
Delete All Breakpoints Ctrl+Shift+F9
Start Page
HELLO WORLD.cpp
(Global Scope)
1st - Add code to program

```
// HELLO WORLD.cpp :                    console application.
//

#include "stdafx.h"
#include "iostream"
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    cout << endl;
    cout << "Hello World";
    cout << endl << endl;
    system ("Pause");
    return 0;
```

We will use the modern tools/programs, to write the code and debug and compile your C++ programs.

In this class you will be using a Modern **IDE** program. It will save considerable time and money when we write and compile C++ programs with an IDE.

See a following lecture for details about how use the MS Visual Studios IDE.

In this course you will write computer programs as part of many lab assignments.
You must down load and use a C++ IDE. An IDE is a program is your friend !.

PC Users - Window OS: Visual Studios

Method 1: Free Microsoft software for students: https://www.dreamspark.com/
You must use an email account ending in .edu (This proves you are a student).
Use your Cañada student email, or any others school email.
Create an account, and down load MS Visual Studios for free.
Note: Other nifty soft is available for free, so be sure to sign up.

Method 2: Down a free 90 free trial version of MS Visual Studios
http://www.microsoft.com/visualstudio/eng/downloads

MAC Users - Mac OS: xCode

Go to your Apple Store and download it for free.

List of Alternative IDE's:
http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

Please Install MS Visual Studios or xCode IDE on your PC or MAC.

You can use another C++ IDE if you chose, but your are responsible for training yourself on its use.

Next lesson:

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

"All things have parts.

Know what the parts are and know how they fit together,

and make it so.

Only then you will be successful"

Quote Professor Schwarz

ASSIGNMENT Questions:

1) What is a Compiled language ?
2) What is an Interpreted language ?
3) What are the different levels of programming Languages ?
4) What is a CLI, How is it still used to write and compile programs ?
5) What is the Purpose of an IDE ?
6) What are the top 10 SDLC's, what percentage is each used ?
7) What are 5 essential window in an IDE ?
8) Which Level of programming language would you rather code in ? Why ?
9) What is the latest release of C++, what is it called and when was it released ?
10) What is the Hierarchy of Programming Languages ?

Read Chapter 6 Low Level Programming – Whole chapter

Read Chapter 7 Problem Solving and Algorithms - ONLY section 7.1 How to solve problem