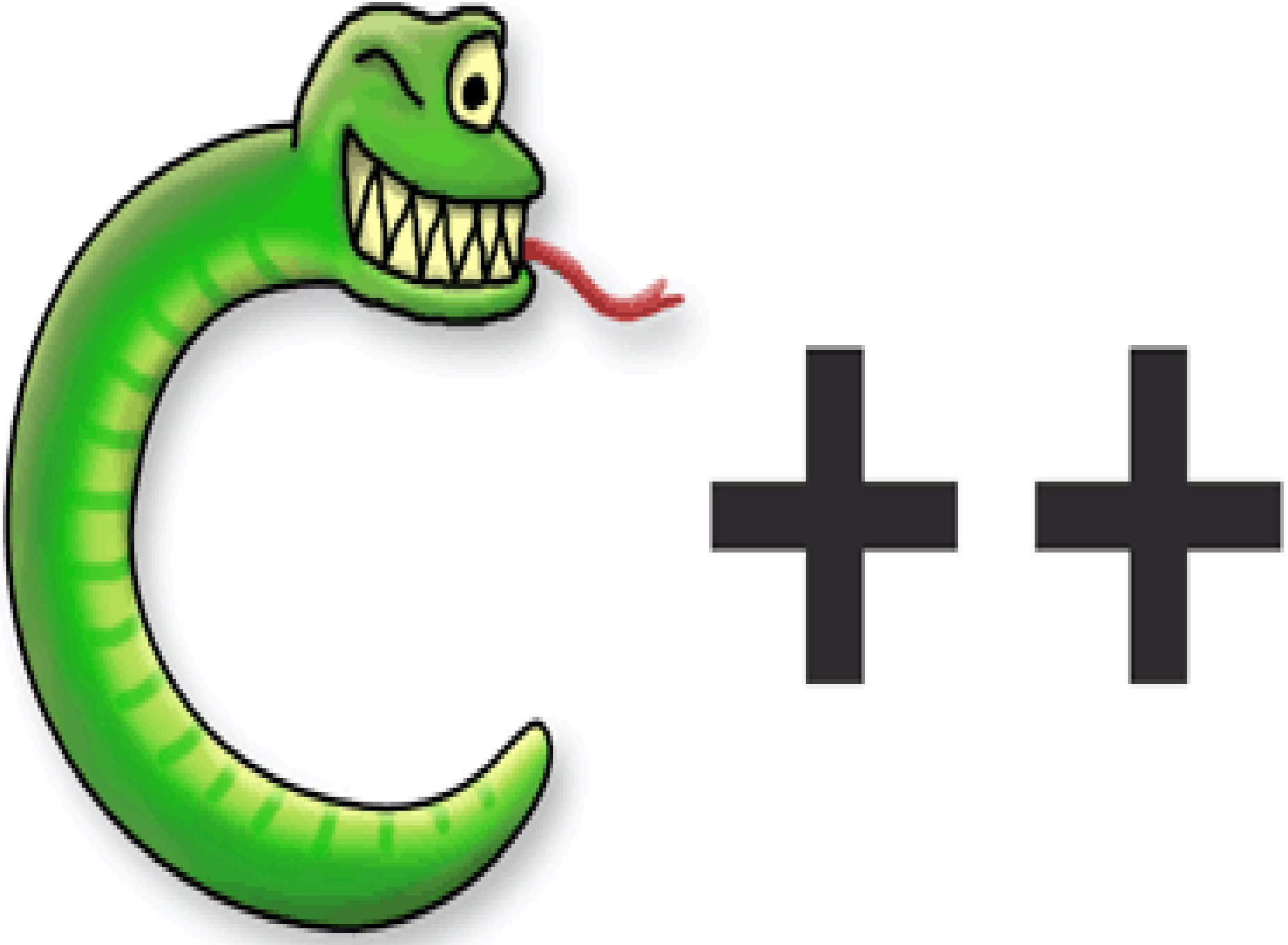


Objectives Outline- we spend lots of time here!

READ,STUDY IN CHAPTER TWO THE FOLLOWING SECTIONS 2.1,2.2,2.4,2.5,2.7,2.8,2.9 (skip 2.3,2.6)

IN THIS CHAPTER DO ALL THE "PRACTICE" EXERCISES IN ASSIGNED SECTIONS AND CHECK YOU ANSWERS IN APPENDIX E

Chapter Objectives BY SECTION

1. C++ Program Structure 2.1
2. Constant and Variables 2.2
3. C++ Operators 2.4
4. Standard Input and Output 2.5
5. Basic Functions in C++ Standard Library 2.6
6. Problem Solving Applied 2.8
7. System Limitations 2.9
8. **Be sure to run "you tube" Video(s) in the online syllabus**

Our Current Objective! Develop problem-solving solutions in C++ containing:

- Simple arithmetic computations
- Information printed on the screen
- User-supplied Information from keyboard
- THIS CHAPTER HAS THE KEY TECHNIQUES FOR SCIENTIFIC PROGRAMMING AND WE SPEND A FEW WEEKS ON IT

REMINDER: MIKE DANES CODE academy intro Lessons show you how to install CODEBLOCK on Windows and Mac computers. BE SURE to use HIS lessons as we cover the SPECIFIC topics in class or just go ahead on your own pace to learn C++!

Variables and Statements

Variables and Statements constructed UNLIKE MATLAB, C++ HAVE A LOT OF OVERHEAD THE PROGRAMMER HAS TO LEARN. THE BENEFIT OF A "C" COMPILER IS SPEED IN EXECUTION! MATLAB USES AN INTEPRETATOR TRANSLATION TO MACHINE LANGUAGE BUT IS RELATIVELY SLOW IN EXECUTION. TODAY ANOTHER MATLAB TYPE PROGRAM CALLED "PYTHON" ENOYS POPULARITY, EASY TO USE AND A BIT FASTER THAN MATLAB BUT STILL RELATIVELY SLOW..

ON AUG 26.2018 "MIT" ENGINEERS AND SCIENTIST ANNOUNCED "JULIA 1.0" A SUPER FAST LANGUAGE FOR ENGINEERS AND SCIENTIST BUT EASIER TO PROGRAM THAN A "C" LANGUAUGE!

ADVANCED HAND IN HW #2 BE SURE YOU STUDY SEC 2.1 AND 2.2 AND THESE LESSON NOTES 3rd and 4th editions 38 pts.

- part 1. #1-13 do PROBLEMS (exam practice at end of chapter) 1-13 Both editions 13 pts
- part 2 #2 Write a declaration statement to define a symbolic constant for the mass of the sun.(look it up). 3 pts
- #3 Express in scientific Notation computer style (that is with an "e") not a power of 10
- 0.000098 and -2,345,000 4 pts
- #4 Express in floating point notation the following 4 pts -3.45e-7 -4.58e8
- #5 Is the following valid identifiers legal: which are and are not legal? y_#sum Final_exam3 fx234 3 pts
- #6 Match each of the following data types with literal constants of that data type. A data type can be used more than once. Show letters answers with match.. 8 pts

A. integer	_____ 1.427E3	_____ true
B. Double	_____ "Oct"	_____ '\'
C. Character	_____ -63.29	
D. string	_____ Zipcode	
E. Boolean	_____ '+'	
F. none of the above.	_____ -85	

- **#7** What is the **exact o**utput of the following program. 3 pts

```
#include <iostream>
using namespace std;
int main ()
{  int hr, min;
    hr = 1;
    min = 50;
    cout << "The exam is over at "  <<  hr  << ":" << min << endl;
    cout << "One down\n " << "two to go!" ;
    return 0;
}
```

ADVANCED HAND IN HW #3 READ STUDY 2.4,2.5,2.7,2.8,2.9 (skip 2.3,2.6)
DO THE FOLLOWING EXAM PRACTICE 27 points
part 1. PROBLEMS AT THE END OF CHAPTER 2.
#14 TO #20 Both editions of the text. 10 PTS
part2 #2. Write the c++ code for this formula

$$T= 2\pi(L\cos(\theta)/g)^{1/2}$$ 3 PTS

#3. Write the algebraic formula for the following c++ code

W=m*((pow(v2,2)-pow(v1,2)))/2 + m*g*(y2-y1); 3 PTS

#4. What is the exact output of the following program segment (assume proper #includes)

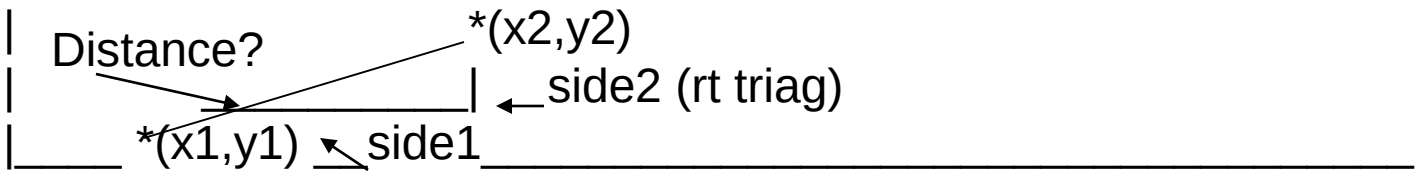
```
int WholeNumber;        4 pts
double Real1, Real2;
WholeNumber = 76;
Real1 = 3.167;
Real2 = -24.103;
cout << setw(6)  << WholeNumber << endl;
cout << setiosflags(ios::fixed);
cout << setprecision(2) << Real1 << ", " << Real2 << endl;
cout << setiosflags(ios::showpoint)  << Real2 << 8.376 << endl;
```

#5. The math function sin will compute sine when given the angle in degrees 1 pt
True or false

#6. Which of the following function invocations is an approximation of PI 1 pt
atan(-1); acos(-1); sin(-1); cos(-1);

#7. Evaluate each Evaluate the following functions 5 pts
(4 - 7) * 3 14 % 4 24 / 9 6.72 / 4.2 2 + 8 * 3 + 7

RECALL PROBLEM distance between two points in the xy plane




```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

C++ Program Structure

```
/*-------------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------------
```

Comments:

- document the program's purpose
- Help the human reader understand the program
- Are ignored by the compiler
- // comments to end-of line
- /* starts a comment block ending with */

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

Preprocessor Directives:

- Give instructions to the preprocessor before the program is compiled.
- Begin with #

#include directives ‘add’ or ‘insert’ the named files (and the functionality defined in the files)into the program

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

‘using’ Directives:

- Tell the compiler to use the library names declared in the namespace.

The ‘std’, or standard namespace contains C++ language-defined components.

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

Main function header:

- Defines the starting point (i.e. entry point) for a C++ program
- The keyword ‘int’ indicates that the function will return an integer value to the system when the function completes.

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

Code blocks:

- are zero or more C++ declarations or statements enclosed by curly brackets { }

The code that defines what a function does (in this case, the main function of the program) is often defined in a code block following the header.

```
/*-------------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------------
```

Declarations:

- Define identifiers (e.g. variables and objects) and allocate memory.
- May also provide initial values for variables.
- Must be made before actions can be performed on variables / objects.
- Note variable names must mean something
- About problem being solved
- They stand for..
- “distance”. “side1” etc

```
/*-------------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------------
```

And out put statement

Statements :

- specify the operations to be performed.

REVIEW: illustrated C++ Program

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points in a plane, x,y coordinates.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std; // all programs must have this
int main() {
// Declare and initialize objects.
   double x1(1), y1(5), x2(4), y2(7),
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
//-------------------------------------------------------
```

Important aspects of the language are; the preprocessor directives (includes), using namespace library names), int main() function and use {} to mark a block of code , comments * and // declaration types(double) and names of variables. Statements for calculations (like distance =) and output.

HAND IN LABORATORY TASK: LAB #2

SET UP A PROJECT IN MSVS, (looks familiar!)

study the illustrated program we just went over and you did as LAB#1

Be sure the first line has your name, date and LAB #2 as comment //

Then construct a similar program with lots of comments

Introduce the two points in a plane and the time in seconds each point was obtained, values assigned by you.(do not use the ones in the text or your neighbors) for time use a T1 for X1,Y1 and T2 for X2,Y2

1. Calculate the distance between them.

Since the sides and distance form a triangle also

2. Produce the area of the triangle

3. Produce the perimeter of the triangle

4. Produce the RATIO of perimeter/area.

5. Produce the velocity between the two points (need T2-T1)

Be sure to make it clear what is coming out of the program with appropriate statements. So that someone reading it understands what is going on.

For example: in the previous program example its clear we are outputting the distance between two points in a plain as follows but we are not specifying what they are.

```
cout << "The distance between the two points is "
       << distance << endl;
```

6. EXTRA CREDIT: specify the values of the two points and time on output

Constants and Variables

- Constants and variables represent memory locations that we reference in our program solutions.
- Constants are objects that store specific data that can **not** be modified.
  - `10` is an integer constant MEMORY bytes are assigned differently!
  - `4.5` is a floating point constant
  - `"The distance between the two points is"` is a string constant
  - 'a' is a character constant
- Variables are named memory locations that store values that **can be modified**. THE PROGRAMMER MUST KEEP TRACK OF WHAT IS HAPPENING IN MEMORY TO VARIABLES USED FOR CALCULATION AND LATER FOR CONTROL OF THE FLOW OF THE PROGRAM WHEN DEVELOPING A SOLUTION TO A PROBLEM
  - `double x1(1.0), x2(4.5), side1;`
  - `side1 = x2 - x1;`
  - `x1, x2` and `side1` are examples of variables since they can be modified in memory.

Initial Values(we say we initialize the variable when we do so), ie the first value assigned to the variable.

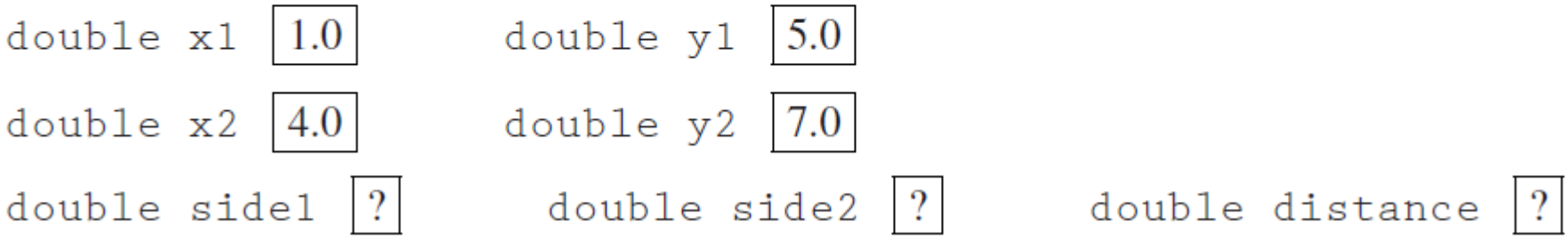
- C++ does not provide initial values for variables. And does not warn you that the value in the variable you created may be meaningless to the problem at hand.
  - Thus using the value of a variable before it is initialized may result in 'garbage'. "Garbage in …Garbage" out is very famous statement!
  - Thus one must initialize! Either in the declaration of the variable as in
  - float space (99.88);

  - Or as the first value of the variable by assignment (=)
  - float space;
  - space=99.88; or take this value 99.88 and put it in memory place for "space"

Memory Snapshots (part of understanding how our program will work).

- Memory 'snapshots' are diagrams that show the types and contents of variables at a particular point in time. The text uses boxes to illustrate the values of a variable in memory. BUT WHEN THE VALUES ARE CONSTANTLY CHANGING A TABLE IS A MUCH BETTER TOOL. MORE ON THIS LATER. SO IN OUR FIRST PROGRAM FOR 'DISTANCE".

```
double x1(1), y1(5), x2(4), y2(7),
       side1, side2, distance;
```



These last three are not initialized but are defined by computations that Follow. So we never use the original junk in the space we declared but space was allocated and address for that space is now known to the system. Numerical values Are first known and can be used (side1,side2) after the following statements are executed.

I.E. side1=x2-x1; and side2=y2-y1;

```
distance = sqrt(side1*side1 + side2*side2);
```

TRACE THE VALUE OF VARIABLES IN MEMORY for each line of code excuted sequentially

in the table, set up for each variable

Do this now in class (any questions?); We are now playing as the computer would do it and following each step in the statements below.

- note more than one statement on a line is ok
- Show values of vel and acc after each statement

Table of values

vel	**acc**

- vel=12; acc =6;
- vel =vel +4; acc =acc -2;
- vel=vel+ acc*2; acc=acc+8;
- vel=vel-24; acc= acc-12;

Valid C++ Identifiers (variable names)

- Must begin with an alphabetic character or the underscore character '_'
- Alphabetic characters may be either upper or lower case. (NOT EQUAL)
  - C++ is CASE SENSITIVE, so 'a' != 'A', etc…
- May contain digits, but not as the first character. (called alphanumeric characters A…Z,a…z, 0…9 and _)
- May be of any length, but the first 31 characters must be unique. Too long is difficult, too short makes it hard to read.
- May NOT be C++ keywords (mostly action carried out by the language.

C++ Keywords (C++ programming action).

asm	auto	bool	break
case	catch	char	class
const	const_cast	continue	default
delete	do	double	dynamic_cast
else	enum	explicit	export
extern	false	float	for
friend	goto	if	inline
int	long	mutable	namespace
new	operator	private	protected
public	register	reinterpret_cast	return
short	signed	sizeof	static
static_cast	struct	switch	template
this	throw	true	try
typedef	typeid	typename	union
unsigned	using	virtual	void
volatile	wchar_t	while	

C++ Identifiers

- Should be carefully chosen to reflect the contents of the object.
  - The name should also reflect the units of measurements when applicable. (Thus you will know right away why you did what you did!) like.. distance or
  - **typed** like integer, character, double etc) before they may be used.
  - C++ is a ***strongly typed*** programming language.

  - **REMINDER: HOMEWORK ON YOUR OWN** BE SURE TO DO THE "PRACTICE! EXERCISES in section 2.2 on valid Identifiers!
  - PRACTICE ANSWERS ARE IN APPENDIX E BE SURE TO CHECK YOURSELF

  - Eg. Now in class
  - Valid or not identifieers;
  - sec**2? , Time, x$y, last_day, 7zebra , _crazynum
  - Density Area f3 #123 Final_value degrees_C f(x)
  - R1.1 ft/s

Scientific Notation

- **FLOATING POINT** is both integer and non-integer values
- 2.5 -.0005 18.0 decimal is all over the place
- **Scientific Notation** one decimal position and uses powers of 10! For overall value
- 256.1= 2.561×10^2 -.0005 = -5.0×10^{-4} 8.0= 8.0×10^0

why is 10^0 =1??

Exponential notation recognized in the computer world for scientific notation

2.561×10^2 =2.561e2 note + is understood but – as in the next example has to be put in.

-5.0×10^{-4} = -5.0e-4

8.0×10^0 =8.0e0

- **HOMEWORK ON YOUR OWN** BE SURE TO DO THE "PRACTICE! EXERCISES ON **Scientific notation and floating point conversions IN SECTION 2.2**

- **SCIENTIFIC NOTATION IS EXPECTED FOR MOST OF THE PROBLEMS WE SOLVE IN THIS COURSE.**

IN CLASS SOLVE THE FOLLOWING

CONVERT TO SCIENTIFIC NOTATION 0.00000567 -92,000,000 45.009 18,000,098

CONVERT TO FLOATING POINT 7.34e-8 -3.45e5 -7.8e-4 6.788e5

HAND IN LABORATORY TASK: LAB #3

Design a program to compute the "area" of a rectangle of length, "L" and width "W" and the Volume and Surface area of a cube of side, "S".
You specify the values of L, W and S. use **descriptive names** not L,W and S
Figure the formulas and what variables you need and write the program
Be sure to Put in /*..*/ #1 and #2

1. Your name AND date and LAB #3 at the start
2. beginning comments that explain what the program does=the Abstract!
3. Other comments to yourself. Enough to explain what is going on. Using //
4. output to a user of the program should explain what they are getting so they know what is coming out along with the original values of L,W and S!
5. Use variable with descriptive names like "side_cube" for example.
6. Output should contain values of L.W and S used to calculate!
7. Save results to editor version (use /*…*/ and run it again for different values of L,W,S 2 more times. Copy as usual from the output screen each of the three output results to your original cpp program. Use // for output to not interfere when you run it again.

Reminder: restart new projects in VS for each new laboratory or program you do.

a. IN EVERY LAB PROJECT, LAB #, YOUR NAME and DATE AS FIRST COMMENT,
b. COMMENTS FOR THE PROGRAMMER (WHAT'S IT ALL ABOUT),Abstract
c. COMMENTS FOR USER
(ON OUTPUT WITH "cout" lots of info on what they are getting!)
(ON INPUT WE will later use "cin" to let them know what to give)

EXTRA CREDIT: USE "cin" for input of L,W and S, use math pow() like pow(x,2)=x^2

Declarations

- A type declaration statement defines new identifiers and allocates memory.
- An initial value may be assigned to a memory location at the time an identifier is defined.

"Syntax" means how its done or laid out

Syntax
[modifier] type specifier identifier [= initial value];
[modifier] type specifier identifier[(initial value)];

Examples of five data types you have to know. More on this follows!

```
double x1, y1(0);
int counter=0;
const int MIN_SIZE=0; best put before main()!
bool error(false);
char comma(',');
```

More Common C++ Data Types

- **Integer numbers**
- **short int long**
- **More memory byte from short to long were the original idea**
- **Today Short max value is 32767 and int and long are the same**
- **At 2,147,483,647**
- Integer no decimal is stored!
- **Decimal numbers**
- **float** or **double** or **long double**
- Stored as exponential numbers
- **More memory byte from float to long to double the original idea**
- **Today float max exponent 38, long and double max exponent 308**
- **and more digits of precision from 6 for float and 15 for double or long double**
- **Carried in memory.**
- **in C today most systems double =long double, could depend on system**

Common C++ Data Types

- Keyword Examples of a constant **bool and char**
  - **bool** error(false), hairy(true);
  - (Boolean true or false only)(actually 1 or 0)
  - **cout << error << endl<< hairy ;**
  - **Output would be**
  - **0**
  - **1**
  - **char chr('4') ch('n')**
  - **cout << chr << endl<<ch;**
  - **output**

  4

  n
  - Each alphanumeric value has a coded number in binary in memory (WE USE ANSI ASCII code SEE APPENDIX B)
  - So do keyboard and printer actions, like carriage return and backspace.
  - All input(ex. keyboard) and output(ex. printer) devices in the computer world (except main frames) recognize the ASCII codes.
  - ANSI ASCII EXAMPLES: WITH BINARY AND INTEGER EQUIVALENTS
  - 'B' =1000010 = 66

    'b' =1100010 = 98

    '7' =0110111 = 55

  CR(CARRIAGE RETURN) = 0001101= 13

CLASS DEMO
Prog 2_1 illustrating “char” and “int” differences

```
/*-------------------------------------------------------*/
/* Program chapter2_1                                    */
/*                                                       */
/* This program prints a char and an int                 */
#include<iostream> // Required for cout
using namespace std;
int main()
{
   // Declare and initialize objects.
   char ch ('3');
   int i(3);
   // Print both values.
   cout << "value of ch: " << ch << " value of i: " << i << endl;
   // Assign character to integer
    i = ch;
   // Print both values.
  cout << "value of ch: " << ch << " value of i: " << i << endl;
   // Exit program
  return( 0);
}
```

WE PLAY COMPUTER

OUTPUT=???

value of ch: 3 value of i: 3
value of ch: 3 value of i: 51   why?

Reminder:To stop output from flashing or disappearing on some computers, if that happens add command
cin.get(); before the return statement

String data type (lots of characters together in a memory location)

We need the string data class with an include

#include <string>

//also we need a declare a string variable

string namefun;

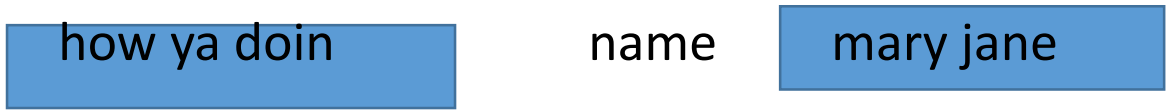
// load the variable with a string of characters..etc. these are a constant until we change the constant

namefun="I pledge alliance to the flag of the United States of America";

Consider declaration and initialization ;

String salutation ("how ya doin"), name ("mary jane");

In memory



Note stirng declaration salutation is a variable and can hold other strings

Like

salutation =" not very well";

String illustration

String constants are enclosed in quotes, “csi”, ”terminator”

But before using the type its defined in c as a **class** and you need the

Directive #include <string> see more on **class** in the text.

```
/*-----------------------------------------------*/
/* Program chapter2_2                            */
/*                                               */
/* This program prints a greeting                */
/* using the string class.                       */
#include <iostream> // Required for cout
#include <string> // Required for string
using namespace std;
int main()
{
  // Declare and initialize two string objects.
  string salutation("Hello"), name("Jane Doe");
  // Output greeting.
  cout << salutation << ' ' << name << '!' << endl;
  // Exit program.
  cin.get();
  return(0);
}
```

WHAT IS IN MEMORY!

OUTPUT?? Hello Jane Doe!

Symbolic Constants

- A symbolic constant is defined in a declaration usually before main() so all future functions can use it
- statement using the modifier `const`.
- A symbolic constant allocates memory for an
- object that can **not** be modified during execution of the program.
- **Any attempt to modify a constant will be flagged as a syntax error by the compiler**.
- A symbolic constant must be initialized in the declaration statement.
- π ? const double PI =3.14 or better PI=acos(-1.0)
- c ? const double c=2.99792e08
- **HOMEWORK ON YOUR OWN** BE SURE TO DO THE
- "PRACTICE! EXERCISES ON **SYMBOLIC CONSTANTS in section 2.2**
- **ILL: SPEED OF LIGHT =2.99792 x 10^8 m/s**
- **Unit of time =s or second**
- **const double speedc=2.998792e8;**
- **const char UnitTime='s';**
- **CLASS EXERCISE Write the declarations statement for the following engineering useful constants**
- **Charge on the electron, e=1.602177 x 10^{-19} C**
- **Acceleration of gravity, g=9.8 m/s^2**
- **Mass of Earth ME =5.98 x 10^{24} kg**
- **Unit of length UnitLength = 'm'**

ADVANCED HAND IN HW #2 BE SURE YOU STUDY SEC 2.1 AND 2.2 AND THESE LESSON NOTES:Both 3rd and 4th editions 38 pts.

- part 1. #1-13 do PRACTICE PROBLEMS 1-13 Both editions 13 pts
- part 2 #2 Write a declaration statement to define a symbolic constant for the mass of the sun.(look it up). 3 pts
- #3 Express in scientific Notation computer style (that is with an "e") not a power of 10
- 0.000098 and -2,345,000 4 pts
- #4 Express in floating point notation the following 4 pts -3.45e-7 -4.58e8
- #5 Is the following valid identifiers legal: which are and are not legal? y_#sum Final_exam3 fx234 3 pts
- #6 Match each of the following data types with literal constants of that data type. A data type can be used more than once. Show letters answers with match.. 8 pts

A. integer	_____ 1.427E3	_____ true
B. Double	_____ "Oct"	_____ '\'
C. Character	_____ -63.29	
D. string	_____ Zipcode	
E. Boolean	_____ '+'	
F. none of the above.	_____ -85	

- **#7** What is the **exact o**utput of the following program. 3 pts

```
#include <iostream>
using namespace std;
int main ()
{  int hr, min;
    hr = 1;
    min = 50;
    cout << "The exam is over at "  <<  hr  << ":" << min << endl;
    cout << "One down\n " << "two to go!" ;
    return 0;
}
```

C++ OPERATORS
Assignment Operator
(sec 2.4 3rd ed or sec 2.5 4th ed)

- The assignment operator (=) is used in C++ to assign a value to a memory location. . Is Not algebraic equal sign! What is on the right is done(calculate an expression) and placed in the what is on the left of the =

- The assignment statement:

```
x1 = 1.0;
```

  - assigns the value `1.0` to the variable `x1`.
  - Thus, the value `1.0` is stored in the memory location associated
  - with the identifier `x1`.
    (Note: `x1` must have been previously declared.)
  - Ill simple use of assignment "=" operator:
  - point: user defined data type a CLASS EXAMPLE
  - Double sum; int x1; point p1,p2(1.5,-4.7) char ch (x, y coordinates)
  - sum=10.5;
  - x1=3;
  - p1=p2; What is in MEMORY for each variable at the start and after. ?
  - ch='a'

Order of Types

High:	**long double**
	double
	float
	long integer
	integer
Low:	**short integer**

- Because different types are different representations, frequently we need to convert between types.
  - Sometimes these conversions may lose information.
  - int a; a=12.6; WHAT IS IN MEMORY?
  -
  - Conversion from lower types to higher
  - types results in no loss of information.
  - Conversion from higher types to lower types may loose information.

a has 12! Why?

  - Be careful. Best to use higher ones always to avoid problems

Assignment Statements use assignment operator , =

Syntax

```
identifier = expression;
```

Examples

```
x1 = y1;
counter = 0;
counter = counter + 1;
```

Assignment operators must not be confused with equality.

THIS IS NOT ALGEBRA

- **WHAT IS THE VALUE OF x if algebra? IN x=x+1 ?**

- **Computer Meaning of ,=,**
- **Value on the right computed is put into the memory place on the left.**
- **In the example:**
- **counter =0;**
- **counter=counter+1; The value of counter on the right is increased by 1 and the result is placed into the same memory place..ie counter**

Arithmetic Expressions

- Expressions appearing in assignment statements for numeric variables may be simple literals (e.g. `x1 = 10.4;`), reading the value stored in a variable (e.g. `x2= x1;`), or be more complex expressions involving the evaluation of one or more operators

- e.g.
- `x1 = -3.4*x2 + 10.4;`
- Areatriangle=0.5*base*height;
- v1= x2*x2*x2 + 4* x2*x2 -9; (ie. $v1=x2^3 + 4x2^2 -9$)

- x=x+1 **called a counter** when forced to repeat! Why?

Arithmetic Operators

- Addition **+**
- Subtraction **-**
- Multiplication *****
- Division **/**
- Modulus **% an important computer operator**
  - Modulus returns remainder of division
  - between **two *integers***
  - Examples

    5%2 = 1

    6%3=0

IN CLASS! Be careful here. Very tricky.

What is 7%3 ? 18%9? 21%6?

Whenever your not sure run the item to see how its handled!

Operator Basics

- The five operators (* / % + -) are ***binary operators*** - operators that require two arguments (i.e. operands). x*y 5%2
- C++ also include operators that require only a single argument – ***unary operators***.
  - For example, a plus or a minus sign preceding an expression can be unary operators: (e.g. `-x2`).

Integer Division (ALERT HERE: TAKE NOTE!)

- Division between two integers results in an integer.
- The result is truncated, not rounded
- Example:

  The expression **5/3** evaluates to 1

  The expression **3/6** evaluates to 0

  Bad use illustrated:

  int sum, count;

  double average

  average=sum/count Why is this bad. ?

Mixed Operations

- Binary operations on two values of same type yield a value of that type (e.g. dividing two integers results in an integer (AS ABOVE)).
- Binary operations between values of different types is a ***mixed operation***.
  - Value of the lower type must be converted to the higher type before performing operation.
  - Result is of the higher type. Unless we go the opposite way.. Ie. Putting higher types into lower!

Ill:given: int x; double y, z; z=y*x;? this is ok why?
Given int a(27), b(6), c; c=b%a?

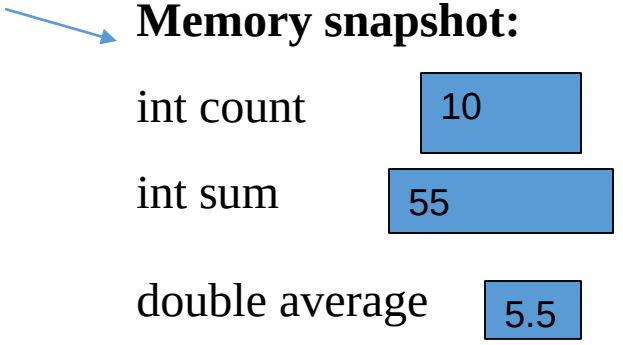
Given int a; double b(6),c(18.6); a=c/b?

Casting or changing the type!

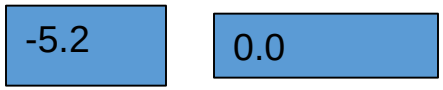
- The cast operator.
  - The cast operator is a unary operator that requests that the **value** of the operand be cast, or changed, to a new type for the next computation. **The type of the operand is not affected**.

- BY Example: Result

```
int count(10), sum(55);
double average;
average = (double)sum/count;
```



HOMEWORK ON YOUR OWN BE SURE TO DO THE "PRACTICE!
EXERCISES ON values computed in section 2.4. Memory map the various situations.
Note: the **Class Point** sets up memory for two points in the plane or two memory Areas (ie x,y): and I do not hold you responsible for this.
for example: point p2(-5.2,0.0) -> memory place called p2 with two values



In class some value examples what are the values for each of the following final values of c and d.

- Given: int a(27), b(6), c;
- double d;
- c=a%b?
- c= b%a ?
- c=a/b;
- c=b/a;
- c=a/(double)b;
- d=a/b;
- d=b/a;
- d=a/(double)b;
- d=b/(double)a;

Precedence of operators

HOW MUCH IS 4+5.0/2+1 =7.50 IF WE MEAN THIS EQUATION
BUT IF WE MEAN 4+5.0/(2+1) =5.67
OR IF WE MEAN (4+5.0)/(2+1) =3.00 SO COMPUTER SYSTEMS HAVE
A PRIORITY ORDER IN MATHEMATICAL EXPRESSIONS WHICH YOU MUST

1. () HAVE HIGHEST ORDER inside them we have
2. UNARY + - (cast type) RIGHT TO LEFT
3. * / % LEFT TO RIGHT
4. + – LEFT TO RIGHT

NOTE: some COMPUTER languages like MATLAB
Exponentiation (^) HAVE A HIGHER PRIORITY DONE AFTER () AND UNARY +-
BUT BEFORE */ EX. X=2; Y= 2*(2*X^3 -5) Y=?????
NOTE: ^ does not exist in C++!

IN CLASS PRACTICE. DO THE FOLLOWING .

Example calculations use c++ priority
Given x=2 y=3

Z= 5*(x-y)/(x+y) =?
Z= 5* (x-y)/x +y =?
Z= 5* x-y/x+y =?

Simplify complex expressions

- ON WRITING COMPLEX EXPRESSION THAT HAVE A NUMERATOR AND DENOMINATOR BREAK UP EACH PART FOR CLARITY AND THUS YOU ARE LESS LIKELY TO HAVE A CALCULATION ERROR.

- CONSIDER $$f = \frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.05005x - 3.14}$$

- WE BREAK UP THE EXPRESSION INTO NUMERATOR AND DENOMINATOR
- num = x*x*x-2*x*x+x -6.3;
- den = x*x +0.05005*x-3.14;
- And thus
- f=num/den;
- Avoiding
- f=(x*x*x-2*x*x +x-6.3)/(x*x +0.05005*x-3.14)
- Which is more likely to make problems in the program

IN CLASS ()
WRITE c++ CODE FOR THE FOLLWING algebraic formulas you might encounter in physics or engineering
USE YOUR OWN NOTATION variable names!)
NOTE: YOU CANNOT USE SUPER OR SUBSCRIPTS IN C++ CODE
So m_1 in algebra would be m1 or mass1 etc in C++
Assume the values are known. We just want the proper C++ expression

$$\text{Tension} = \frac{8 m_1 m_2}{m_1 + m_2} g$$

$$P_2 = P_3 + \frac{pv((A_2)^3 - (A_1)^3)}{2 (A_1)^3}$$

$$\text{Distance} = x_0 + v_0 t + 0.5 a t^2$$

distance=x0 +v0*t +0.5*a*t*t

Write the mathematical equation computed by the following C++ statements. PI and g are constants defined as PI=acos(-1.0) (try this) And G =6.67258e-11 and are well known

centripetal = 4*PI*PI *r/(T*T);

Potential_energy=-G*M_E*m/r;

Change= G*M_E*m*(1/R_E-1/(R_E+h));

Standard Input / Output
Sample Output - cout

- `cout` is an `ostream` object, defined in
- the header file `iostream`
- `cout` is defined to stream data to standard output (the display)
- `We use` the output operator << with `cout` to output the value of an expression.

General Form:

`cout` << expression << expression;

EG.

cout<< " THE RADIUS "<<radius<<" ft\n";

Or cout<<"THE RADIUS " <<radius<<" ft"<<endl;

\n &endl move output to next line!

Note: An *expression* is a C++ constant, identifier, formula, or function call (LATER WE WILL DEAL building FUNCTIONS).

Input - cin

- `cin` is an `istream` object defined in the header file `iostream`
- `cin` is defined to stream data from standard input (the keyboard)
- We use the input operator `>>` with `cin` to assign values to variables
  - General Form
    `cin >>` *identifier* `>>` *identifier*`>>`; **NOTE:>> NOT <<**
- **Note: Data entered from the keyboard must be compatible with the data type of the variable.**
- **EXAMPLE cin >> distance >> time >> acceleration;**
- **The computer will wait for three values to be entered at the key pad provided they are proper DATA TYPE! Assume double in this last example**

A MODEL OF SIMPLE INPUT/OUTPUT CALCULATION NO OUTPUT FORMAT…PLAIN VANILLA OUTPUT!

```
/* Results are displayed without format */
/*This program will calculate the area of circles, one time only*/
#include<iostream> //Required for cout,cin
using namespace std;
const double PI = 3.1417; // NOTE:  BEFORE MAIN SO IT AND OTHER functions can use.
int main()
{
    //Declare and initialize
   double radius, area;
   // get the  radius from the user
  cout << "This program will compute the area of circles";
  cout << "\nPlease provide the radius of  the circle in centimeters whose area is wanted ?\n";
  cin >> radius;
  area = PI*radius*radius; // calculate the area
  cout << " \nThe area is " << area << " square centimeters\n";
  //exit program
  cin.get();
  return 0            //Left an error here?  WHAT IS IT?
}
```

- Notes: to Run with PI missing and then input radius and PI to illustrate multiple inputs.
- We need the line cout<<radius<<PI; along with modified instructions and declare
- Double PI,radius,area;

HAND IN LABORATORY TASK: LAB #4 carefully cover all 8 requirements below to avoid a repeat. Save work!

- GIVEN Electrostatic problem equations: Force, F between a small charged, Q, Sphere and a point test charge q separated by a distance, r , has been measured to be 0.5 Newtons. The Voltage, V, at a distance, r, from the small . Sphere. The electric field, E, at the distance r from the small sphere. The Electric field, E_2, half way between the two charges Calculate the distance between them , r for known F! and then V, E and E_2. **USE descriptive names** for variables **not physics abbreviations as above** and
- $F= k\ Qq/r^2$ $V=kQ/r$ $E=F/q$ $E_2=kQ/(r/2)^2 + kq/(r/2)^2$
- 1.k= 9×10^9 Newtons meter ^{2} / Coulomb^2 **constant!** Put before main()!
- 2.Current values of known variables follow Q = 2×10^{-6} Coulombs, q = 1.5×10^{-6} Coulombs and F =0.5 netwons, must be input by "cin" and input in scientific notation (e).
- 3.. r=? Meters setup solution as an equation r= ? And have the computer solve it. That is solve algebraically for r in the first equation and use thx,3)e know values to get r! a MUST DO!
- **4.. V=? Unit is Newton-Meter/Coulomb or just Volt.**
- **5.. E and E_2 unit is Newton/Coulomb**
- 3. USE **e scientific notation** for constants and values.
- 4. Use **string variables** to hold all the units and output them with calculated values.
- **5. Hand Calculation on the lab exercise to check your answer. PENCIL!**
- 7. Remember for **full credit comments** to yourself (/* & //)and for the user (descriptive output) **Be sure to have an ABSTRACT (what's it all about /*)** with your name ,date and LAB#4 at start!
- 8. Be sure to ***use following math functions: pow() , sqrt() eg. pow(x,3) same as x^3***
- **WARNING: Identical copies of lab work results in a 0 for all and even the original!**

EXTRA CREDIT HW-*HAND IN LABORATORY TASK: LAB #5 start with name, date and LAB#5! READ CAREFULLY WHAT IS WANTED!* attach one hand calculation to check your computer program! EC= up to 50 points to boost lab average.

GET INPUT(cin) FROM THE USER AND OUTPUT A meaningful answer as illustrated below.

- Just use plain approach with formats like "\n and be sure to document your program and its important lines, also **with good variable names not abbreviations**. **Print up with output for each of the five inputs see below** attach each output as comments as before.
- HAND IN PROGRAMS: ONLY DO THE ONE FOR YOUR LAST NAME
- Last name BEGINNING WITH THE FOLLOWING INITIALS:NOT SURE ASK ME!
- **always output the input with the calculations**. Remember COMMENTS TO YOU AND USER FOR FULL CREDIT e.g output. The area of the triangle with base 10 and height 5 centimeters is 25 square centimeters.
- A-M –last name first initial (use both equations in your program)
- Write a program that converts degrees Celsius (TC) to degrees Rankine(TR) and Fahrenheit. Recall TF = TR-459.67. And TF=(9/5)TC+32 **Output the Fahrenheit and Rankine** equivalent to the input Celsius temperature. Run following 5 different Temps note: your hand calculation first value to check your formulas! -10,0,10,22,100 copy answers to program as before. Answers make sense?
- M-Z- last name first initial (use both equations in your program
- Write a program that converts degrees Kelvin(TK) to degrees Fahrenheit(TF) and Celsius(TC). And you also know that degrees Celsius (TC) is related to TK by TK=TC+273.15 and you Recall . TC=(5/9)(TF-32) **Output the Fahrenheit and Celsius** equivalent to the input Kelvin temperature. Run following 5 different Temps first ie. your hand calculation first value to check your formulas! 0,100,273.15,373.15, 500. answers make sense?

***Embellishments to code are now covered

Abbreviated Assignment Operators
I DO NOT RECOMMEND early, use you might get confused unless you had previous exposure!

operator	example	equivalent statement
+=	`x+=2;`	`x=x+2;`
-=	`x-=2;`	`x=x-2;`
*=	`x*=y;`	`x=x*y;`
/=	`x/=y;`	`x=x/y;`
%=	`x%=y;`	`x=x%y;`

Increment and Decrement Unary Operators
I do not recommend again these fancy operators immediately
also some problems since when the changes take place varies
system to system and its easy to get confused unless you are using them constantly

- Increment Operator **++**
  - post increment **x++;**
  - pre increment **++x;**
- Decrement Operator **- -**
  - post decrement **x- -;**
  - pre decrement **- -x;**
- Consider w=++x-y equivalent to x=x+1 and w=x-y >or bump up first then calc
- w =x++ -y is equivalent to w=x-y and x=x+1 > or calc first then bump up

- CLASS EXERCISE:1. If we start with x=5 and y =3 then what is x and w after each statement?
- w=++x-y; and w =x++ -y;

- 2. if x=2 and y=4
- z=x++*y z=++x*y; w= ++x*y; x+=y y%=x;

Figure this out now in class!

FORMATING OUTPUT: more control on our output!

"cout" is an object of type ostream and has other objects that can be called for output called **member functions that control how the output looks.**

Example: the (.) dot operator calls on other aspects of output to improve what we see.
cout.setf() sets the format (a format **flag!)** of the display like some examples

setf(ios::fixed) shows decimal notation
setf(ios::scientific) shows scientific notation
cout.precision() sets the number of significant digits we want to the right of the decimal point.
Setf **(flag**) remains until another call to setf or unsetf(what was set!) is called. Ex.
Unsetf(ios::showpoint)

WHAT DO YOU THINK THE FOLLOWING DO?
ios::right and ios::left ???

```
/*--------- DEMO of dot formatting ..I run it in class----------------------------------------*/
/* Program chapter2_4                                  */
/* This program computes area of a circle.             */
#include<iostream> //Required for cout, setf() and precision().
#include<cmath>    //Required for acos(). NEAT ACCURATE WAY FOR PI!
using namespace std;
const double PI = acos(-1.0);
int main()
{
  double radius(10), area;
  area = PI*radius*radius; //Declare and initialize objects.
   cout << "The radius of the circle is: " << radius       << " centimeters\nThe area is "
        << area << " square centimeters\n";
   //Call the setf member function using dot operator.
   cout.setf(ios::fixed); //Fixed form(xx.xx).
   //Call the precision member function using dot operator
   //cout.setf(ios::scientific); //change to scientific on 2nd run but get rid of fixed cannot be together
  cout.precision(2); //Display 2 digits to right of decimal.
  cout << "The radius of the circle is: " << radius       << " centimeters\nThe area is "
       << area << " square centimeters\n";
   cout.precision(5);  // increase precision of output!
   cout << "The radius of the circle is: " << radius       << " centimeters\nThe area is "
       << area << " square centimeters\n";
    //exit program
    cin.get();    //pauses on most systems that do not pause output immediately
  return 0;
}
```

A word on **manipulators:** making your output impressive!

- Within the cout expression we can control output with a set of commands called manipulators **related** to names of the choices in the dot command cout.setf( )
- Ex. Scientific, setprecision(n), setw(n),fixed, right, left
- We need the header **iomanip to make them work**
- cout <<scientific<<setprecision(3);
- cout << “The length of the cable is ” <<setw(12) <<length<< “meters”<<endl.
- Note **endl** is an manipulator and flushes out the place in memory where all output is stored..namely the **“output buffer”**
- **The newline character ‘/n’ does not do this. the buffer is flushed when it is full. So use endl when debugging problems to see output at each cout.**
-

```
DEMO  /* Program chapter2_5 Illustrating manipulators              */
/*                                                                 */
/* This program computes area of a circle.                        */
/* Results are displayed with two digits                           */
/* to the right of the decimal point.                             */
#include <iostream> //Required for cout, endl.
#include <iomanip>  //Required for setw() setprecision(),fixed.
#include <cmath>    //Required for acos().
using namespace std;
const double PI = acos(-1.0);
int main()
{
  //Declare and initialize objects,
  double radius(10), area;
  area = PI*radius*radius;
  cout << fixed << setprecision(2);
  //cout << scientific << setprecision(6);
    cout << "The radius of the circle is: "
       << setw(10) << radius << " centimeters" << endl;
  cout << "The area of the circle is: " << setw(10) << area
       << " square centimeters" << endl;
  //exit program
  return 0;
}
```

HOMEWORK ON YOUR OWN Manipulator PRACTICE! Section 2.5(3RD ED) 2.6 (4TH ED)

Basic Functions in C++ Standard Library

Just like your calculator has math functions so does C++ and most Computer languages.
To use them in our program in C++ we need the directive **#include<cmath> as we used to get a good value of π.**

There are standard Trignometric functions like cos(x), and sin(x) but x must be in radians so if you have a problem with degrees,d, You need a line making the conversion of degrees,d to radians,r.
Since 180 degrees = **π radians or 1 degree =π/180 radians**
So d degrees =d* π/180 radians or for x in cos(x) etc in c++ we Use x=d*PI/180
where PI=acos(-1.0)
NOTE : arc trig functions (eg.s atan(), acos(), return radians also.

IN CLASS: Use your calculator for practice. EX. Calculate in degrees sine(30) & cosine(30) now Set radians Convert 30 degrees to radians! check sine() and cosine() of your radian calculation.

Basic C++ Math Functions

fabs(x)	computes absolute value of x
sqrt(x)	computes square root of x, where x >=0
pow(x,y)	computes x^y
ceil(x)	nearest integer larger than x
floor(x)	nearest integer smaller than x
exp(x)	computes e^x
log(x)	computes ln x, where x >0
log10(x)	computes $\log_{10}x$, where x>0

Trigonometric Functions

sin(x)	sine of x, where x is in radians
cos(x)	cosine of x, where x is in radians
tan(x)	tangent of x, where x is in radians
asin(x)	This function computes the arcsine, or inverse sine, of x, where x must be in the range [−1, 1]. The function returns an angle in radians in the range [−π/2, π/2].
acos(x)	This function computes the arccosine, or inverse cosine, of x, where x must be in the range [−1, 1]. The function returns an angle in radians in the range [0, π].
atan(x)	This function computes the arctangent, or inverse tangent, of x. The function returns an angle in radians in the range [−π/2, π/2].
atan2(y,x)	This function computes the arctangent or inverse tangent of the value y/x. The function returns an angle in radians in the range [−π, π].

Important math homework

- **DO ON YOUR OWN all PRACTICE EXAMPLES IN SECTION 2.7(3rd ed) 2.8 (4th ed)**
- **AND CHECK YOUR ANSWERS.**
- **THE FOLLOWING ARC HYPERBOLIC IDENTITIES WILL**
- **BE USEFUL**

$$\operatorname{arsech} x = \operatorname{arcosh} \frac{1}{x}$$

$$\operatorname{arcsch} x = \operatorname{arsinh} \frac{1}{x}$$

$$\operatorname{arcoth} x = \operatorname{artanh} \frac{1}{x}$$

Do now in class:

Evaluate:
pow(3,2), pow (2.0, -3), log10 (113), log(94), sqrt (floor(10.9)), abs ((pow(-2,3)), log10(100)+log10(0.001),ceil(-2.6)

Write C++ code for
$d= (x^2 + y^2)^{1/2}$

fabs(x)	computes absolute value of x
sqrt(x)	computes square root of x, where x >=0
pow(x, y)	computes x^y
ceil(x)	nearest integer larger than x
floor(x)	nearest integer smaller than x
exp(x)	computes e^x
log(x)	computes ln x, where x >0
log10(x)	computes $\log_{10}x$, where x>0

Length = $(1-(v/c)^2)^{1/k}$ in text answer is wrong!
Center =$38.1972(r^3 - s^3) \sin a /(r^2 -s^2) a$

Write the algebra for the c++ code that follows
Frequency = 1/sqrt(2*pi*c/L)

Velocity=sqrt(2*g*h/(1+I/(m*pow(r,2))))

Problem Solving Applied: we study the text example

The velocity and acceleration of an unducted fan engine(UDF) which combines jet technology with propeller technology.

We want to calculate the velocity and acceleration after the plane reaches its Cruise velocity of about 180 m/s (meters/second). We know that the engine Throttle is increased to a higher power level and the plane will then Accelerate but though the velocity will increase the rate of acceleration will drop Off since the plane will experience aerodynamic drag in proportion to the square of the velocity.

Be sure when you are studying you see the curves generated on next slide

We know the equations below for velocity and acceleration at a "time" after cruise speed is reached is given by.

$$\text{Velocity} = 0.00001\text{time}^3 - 0.0048\text{time}^2 + 0.75795\text{time} + 181.3566 \ \text{m/s}$$

$$\text{Acceleration} = 3 - 0.000062\text{velocity}^2 \ \ \text{m/s}^2 \ \ \text{meters/second square}$$

The time is gotten via an input to our program. Note at time = 0 the values!

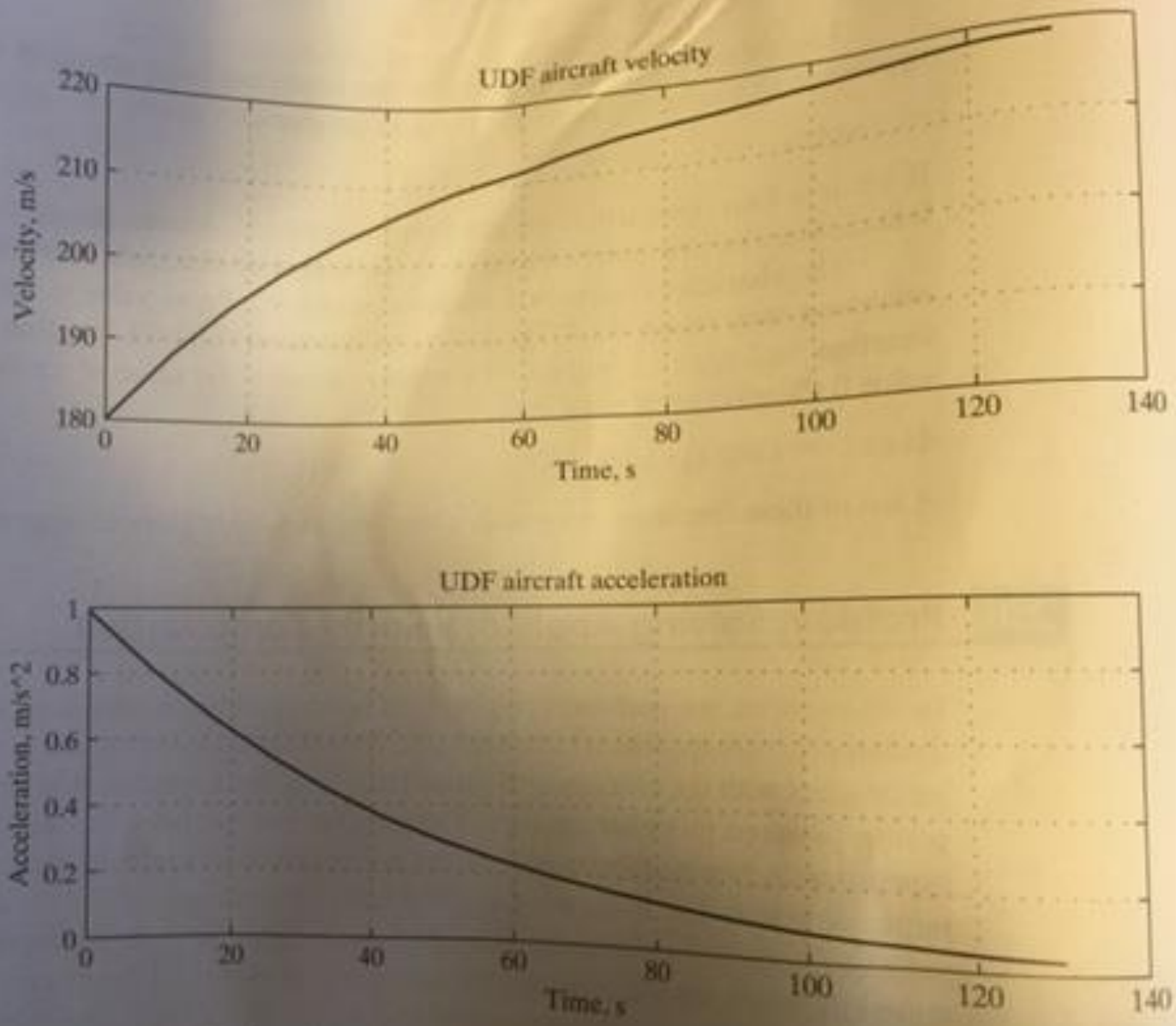


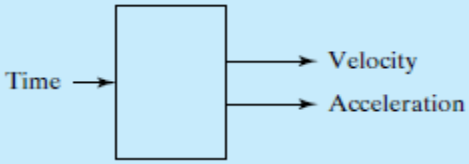
Figure 2.1 UDF aircraft velocity and acceleration.

1. PROBLEM STATEMENT

Compute the new velocity and acceleration of the aircraft after a change in power level.

2. INPUT/OUTPUT DESCRIPTION

The following diagram shows that the input to the program is a time value, and that the output of the program is the pair of new velocity and acceleration values. The built-in data type `double` can be used to represent these values.



3. HAND EXAMPLE

Suppose that the new time value is 50 seconds. Using the equations given for the velocity and accelerations, we can compute these values:

Velocity = 208.3 m/s;
Acceleration = 0.31 m/s^2.

Use hand calc
To test our
Program!!!

4. ALGORITHM DEVELOPMENT

The first step in the development of an algorithm is the decomposition of the problem solution into a set of sequentially executed steps:

Decomposition Outline

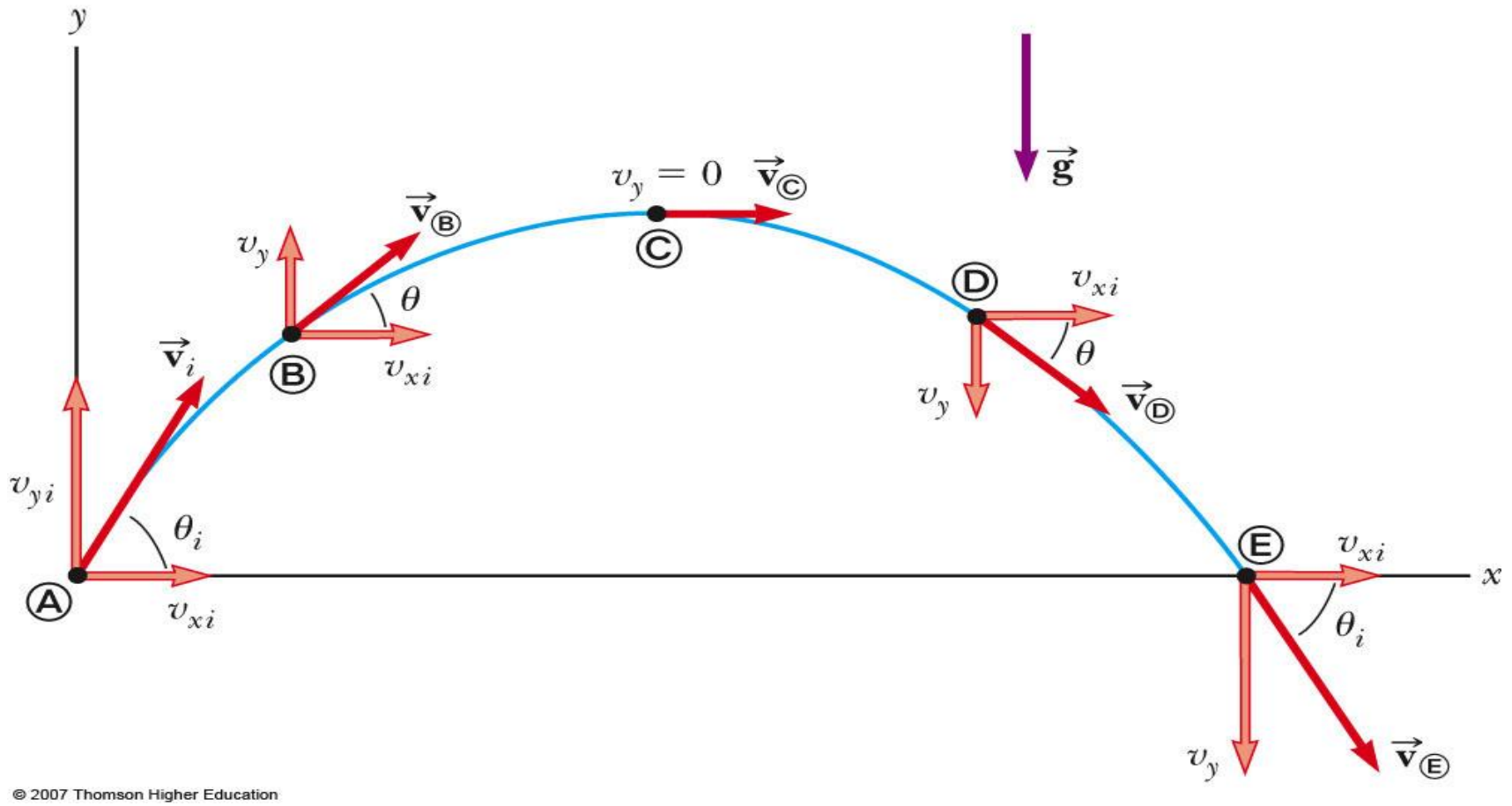
1. Read new time value.
2. Compute corresponding velocity and acceleration values.
3. Print new velocity and acceleration.

Because this program is a very simple program, we can convert the decomposition directly to C++.

DEMO FOLLOWS

```
#include<iostream> //Required for cin,cout
#include<iomanip> //Required for setprecision(), setw()
#include<cmath> //Required for pow()
using namespace std;
int main()
{
  // Declare objects.
  double time, velocity, acceleration;

  // Get time value from the keyboard.
  cout << "Enter new time value in seconds: \n";
  cin >> time;

  // Compute velocity and acceleration.
  velocity = 0.00001*pow(time,3) - 0.00488*pow(time,2)
           + 0.75795*time + 181.3566;
  acceleration = 3 - 0.000062*velocity*velocity;

  // Print velocity and acceleration.
  cout << fixed << setprecision(3);
  cout << "Velocity = " << setw(10)
       << velocity << " m/s" << endl;
  cout << "Acceleration = " << setw( 14)
       << acceleration << "m/s^2" << endl;

  // Exit program.
  return 0;
}
```

Typical vector picture of motion in two dimensions.



The initial velocity can be expressed in terms of its components The initial velocity can be expressed in terms of its components

$v_{xi} = v_i \cos\theta$ and $v_{yi} = v_i \sin\theta$

The *x*-direction has constant velocity $a_x = 0$

The y-direction is free fall $a_y = -g$

The initial velocity $v_{xi} = v_i \cos \theta_i$ and $v_{yi} = v_i \sin \theta_i$

Assume initial position is (0,0)

The *x*-direction has constant velocity $a_x = 0$

Or $x = v_{xi}\ t = v_i \cos \theta_i\ t$ solve for t gives $t = x / v_i \cos \theta_i$ *use below*

The y-direction is free fall $a_y = -g$

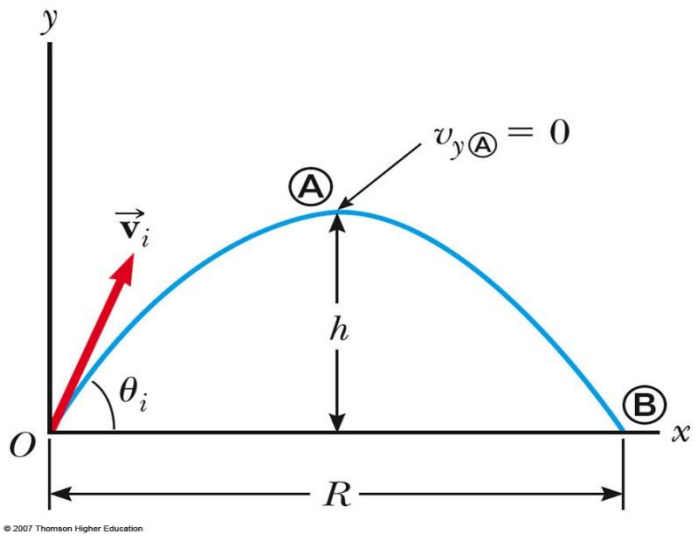
$y = v_{yi}\ t - (1/2)\ a_y\ t^2 = v_i \sin \theta_i\ t - (1/2)gt^2$
Substitute for t
$y = v_i \sin \theta_i\ x / (v_i \cos \theta_i) - (1/2)g\ (x / v_i \cos \theta_i)^2$

It can be shown from above that
Range $R = = v_i^2 \sin(2\theta_i)/g$

Max height $h = v_i^2 \sin^2(\theta_i)/(2g)$

Time cover R, $T = 2v_i \sin(\theta_i)/g$

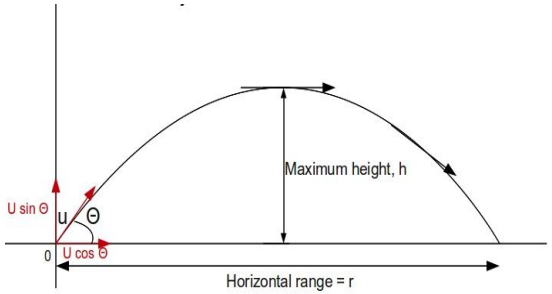
See physics text !



HAND IN LABORATORY TASK: LAB #6 name,date, LAB #6 first line

- REMINDER: comments to the user and the programmer are expected here.
- USE MANIPULATORS SETW(), SETPRECISION(3 or 4), SCIENTIFIC .
- BE SURE /* ABTRACT IS IN THE SOURCE CODE */ (IE what it is all about!)

- We are launching a projectile as part of Napoleon's army from one of our giant cannons. We need to study the canons ability to be sure we can hit the target. We have a castle at 1200 meters with a 30 meter wall, from where we are setting up our cannon. We know if we aim the cannon at some angle theta, θ, in degrees, at a initial muzzle velocity v_0 (u in image below) known to be 128 m/s for our canon, then the Range (how far it will go) in meters is given by a horizontal range = $v_0^2 \sin(2\theta)/g$ (r in image below) with g as a **constant** =9.8 m/s^2 BE SURE YOU ANSWER EACH PART! LABEL THEM!



- The maximum height(h in image below) is given by = $v_0^2 \sin^2(\theta)/(2g)$
- And the time for the complete motion = $2v_0 \sin(\theta)/g$
- (you have to convert to radians to use the sin())
- be sure output section shows the original angle in degrees
- LABEL EACH OF THE FOLLOWING PARTS IN YOUR OUTPUT SECTION

- **PART 1**. Write the program that **inputs** the Angle in degrees. THIS ANGLE SHOULD BE IN OUTPUT !
- Be sure to request input and let the user know **why** You are asking for the data with "cout statements
- **Then Output: Range, Max height and time of journey for all cases**: make it clear in **output** as to what the values mean! First run a **hand calculation** for some angle and then test it as part of the Run For two more **arbitrary angle input cases. Be sure your test angle from the hand calculation gives good answers before proceeding. SHOW HAND CALC!! Copy output screen as usual**

OUTPUT EXPECTED PART 1: FIRST TEST RUN AND TWO MORE SENSIBLE RUNS 90 AND 180 DEGREES DO NOT MAKE SENSE UNLESS YOU WANT TO KILL YOUR OWN TROOPS AND YOUR HAND CALCULATION FOR THE FIRST angle.

See NEXT PAGE FOR FURTHER INTSRUCTIONS ON LAB#6

PART 2.Run the program again for the given muzzle velocity to find the angle for the maximum Range. Don't copy these runs. **Run it a few times** for yourself to see the angle you get is the maximum range. Then **Show the output for the maximum case, as well as an *1 degree before the maximum angle and 1 degree after*** to show you really have the maximum range! . Be sure to **note the max range and what the angle is**.

OUTPUT EXPECTED PART 2: one set of data for the maximum range angle and two more runs for 1 degree before and after the maximum. Note the result shows you really have the maximum range!

PART 3. Given the position of the castle, AT 1200m. What is the angle to just reach the castle wall. TWO OPTIONS HERE. Specify the angle to two decimals.

a. solve the equations by hand to find a precise angle (use formulas to figure it out) (two digit precision). Run this PUT THIS CALCULATION ON THE LAB.

b. **Extra credit** setup in the program code to solve the angle!

in EITHER OPTION **Run the result** you get to be sure show the range is correct and what the angle is to precision of AT LEAST **two decimals**.? EG. 1145.45 Recall answer in degrees!

OUTPUT EXPECTED FOR PART 3: The run for a range of 1200 meters and hand calculation for part a on lab.

If you choose part b than only the run.

PART 4. Enemy troops are located behind the castle at 2500 meters. Can our canon reach them? Why or why not. Explain! Paly with the program to answer.

OUTPUT EXPECTED FOR PART 4: the answer as a comment with reference to the proper run above.

PART 5. The wall is 30m high what is the angle that will hit the top of a 30 m high wall and at a range of 1200m. Use previous equation for y in terms of x and with known values of v,x, ,g play with different values of θ to the equation to get y=30!. Show the angle that works closely! Does it make sense your answer? Explain.

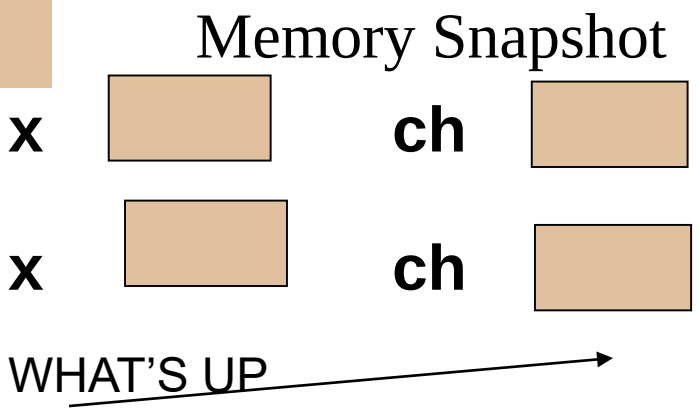
OUTPUT EXPECTED FOR PART 5: The solution of the angle with x = 1200m and angle gives y =30m.

Some last words on this chapter Characters and Input in section 2.5

- The input operator >> skips all whitespace characters.
- The get() method gets the next character.
- Example:

```
int x;
char ch;
cin >> x >> ch;
cin >> x;
cin.get(ch);
```

Input stream:
45 c
39
b



Example 2

```
#include <iostream> //Required for cout, endl.
using namespace std;
int main()
{
// Declare objects.
double r, h;
int id;
char code;
cin >> r >> h;
cin >> id;
cin >> code;

cout << r << "  " << h << "  " << id << "  " << code;
// Exit program.
return 0;
}
```

we enter 11.7 90
786
R

what’s in r, h, id and code?

Example 3

```
/*                                                    */
#include <iostream> //Required for cout, endl.
using namespace std;
int main()
{
// Declare objects.
char   ch1, ch2, ch3;
cin.get(ch1);
    cin.get(ch2);
    cin.get(ch3);

cout << ch1 << "     " << ch2 << "     " << ch3 << " \n" ;
// Exit program.
return 0;
}
```

We enter
D
6

What's in ch1,ch2, ch3?

HAND IN HW #3 READ STUDY S2.4,2.5,2.7,2.8,2.9 (skip 2.3,2.6)S**ECTIONS**

DO THE FOLLOWING EXAM PRACTICE 27 points

1. PROBLEMS AT THE END OF CHAPTER 2.

#14 TO #20 Both editions of the text. 10 PTS

2. Write the c++ code for this formula

$$T = 2\pi(L\cos(\theta)/g)^{1/2}$$ 3 PTS

3. Write the algebraic formula for the following c++ code

```
W=m*((pow(v2,2)-pow(v1,2))/2  + m*g*(y2-y1);
```
3 PTS

4. What is the exact output of the following program segment (assume proper #includes) 4 pts

```
int  WholeNumber;
double Real1, Real2;
WholeNumber = 76;
Real1 = 3.167;
Real2 = -24.103;
cout << setw(6)  << WholeNumber << endl;
cout << setiosflags(ios::fixed);
cout << setprecision(2) << Real1 << ", " << Real2 << endl;
cout << setiosflags(ios::showpoint)  << Real2 << 8.376 << endl;
```

5. The math function sin() will compute sine when given the angle in degrees 1 pt

True or false

6. Which of the following function invocations is an approximation of PI 1 pt

atan(-1); acos(-1); sin(-1); cos(-1);

7. Evaluate each Evaluate the following functions 5 pts

(4 - 7) * 3 14 % 4 24 / 9 6.72 / 4.2 2 + 8 * 3 + 7

HAND IN LABORATORY TASK: LAB #7: SEE END OF CHAPTER!

Amino Acid Molecular Weights START LINE: NAME, DATE, LAB #

ie **Molecular weight** is the **combining** of the atomic weights of the individual elements in a molecular!

The **amino acids** are composed of atoms of **oxygen, carbon, nitrogen, sulfur and hydrogen** whose atomic weights (average of isotopes) are

Oxygen(O) 15.9994; Carbon(C) 12.011; Nitrogen(N) 14.00674; Sulfur(S) 32.066
Hydrogen(H) 1.00794 in amu units

(atomic mass units- originally based on weights relative to hydrogen now they use carbon. And are constant for this program.. use **"const"!**

Program **inputs** : program asks user for the name of the amino acid and how many of each of the above atoms are in it, us 0 if none. **All on one line!!!**

Program o**utputs** the name of the acid, the number of each of the atoms above and finally the molecular weight (=sum of all the atoms atomic mass units). **with appropriate wording so the user knows what they got.**

REMEMBER

1. ABSTRACT ON THE TOP OF THE PROGRAM
2. outputs to the user so the know what this program will do and what they have to give!.
3. Hand calculation for first one only!

Use Lots of comments.

Use constants . Set up prior to main().

Output-> use formats of Scientific and setw(), continued next page

First RUN: input amino acid name "glycine" (use a string variable for all acid names), and all the number of the elements in the amino acid at a time,

ON ONE LINE **for the calculation, rather than 6 separate "cin" lines.**

Number of Atoms in glycine: O=2,C=2,N=1,S=0,H=5

Rerun it for Methionine O=2,C=5,N=1,S=1,H=11 (not a separate program!)
HINT AND REMINDER: cin>>x>>y>>z; would be a way to enter 3 values at once, either 1 at a time with a return for each or all three on the line with a space between them and a final return to load the variables.

EXTRA CREDIT: answer

1. **use the back of chapter two and do a run for two more amino acids.**
2. **State which of the amino acids is the most massive and which the least of the four you will do.**
3. **Answer this question: What are amino acids and what is their use?**
4. **Use a loop to keep the program going under your control till you decide to end it. (advanced students) extra extra!**