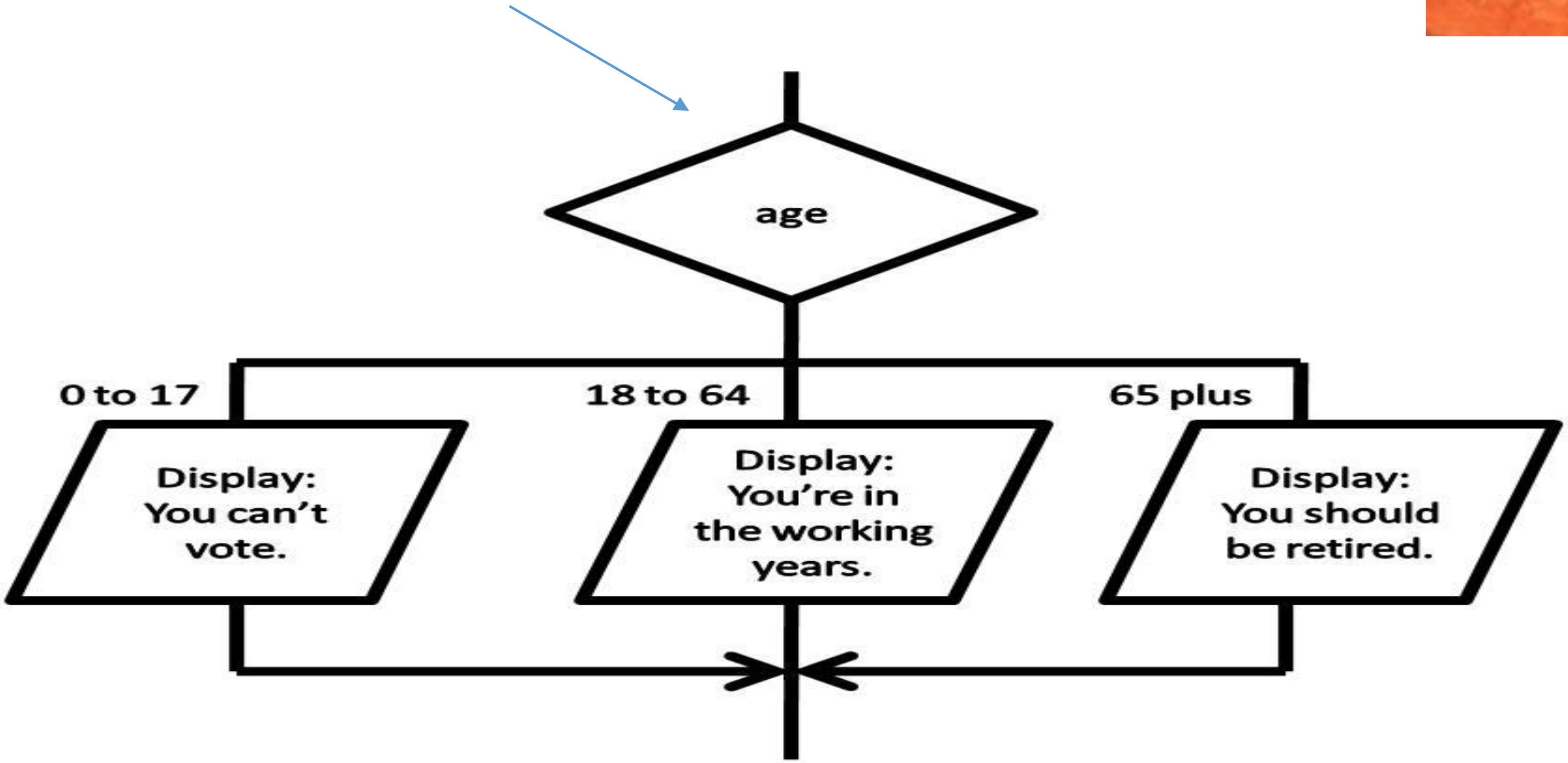
SELECTION:"Flow" chart example

Decision point IF Age = ?


age
0 to 17
18 to 64
65 plus
Display: You can't vote.
Display: You're in the working years.
Display: You should be retired.

Chapter 3 Outline:
Read study sections 1 to7

Objectives

1. Algorithm Development
2. Structured Programming (avoiding spaghetti code).
3. Conditional Expressions
4. Selection Statements: `if` Statement
5. Numerical Technique: Linear Interpolation
6. Problem Solving Applied: Freezing Temperature of Seawater
7. Selection Statements: `switch` Statement

View and study the following videos in this order as we cover the topics

- 8 Variables and Data Types Part 2
- 10 IF statements Part 1
- 11 IF statements Part 2
- 16 SWITCH statement

C++ CONTROL STRUCTURES

The real power of a computer program resides in the programmer tools that can cause different statements to be executed by various conditions. This is called SELECTION and this chapter covers that.

The other major ability is for a program to repeat itself based on conditions which is called REPETITION which we cover in detail the next chapter. But some of it covered now.

Objectives clarified.

Develop problem-solving solutions in C++ containing:

- Conditional expressions that evaluate to **true of false**.
- Selection structures that allow us to **provide alternative paths** in a program.
- Algorithm development and descriptions of algorithms using **flowcharts** and pseudocode.

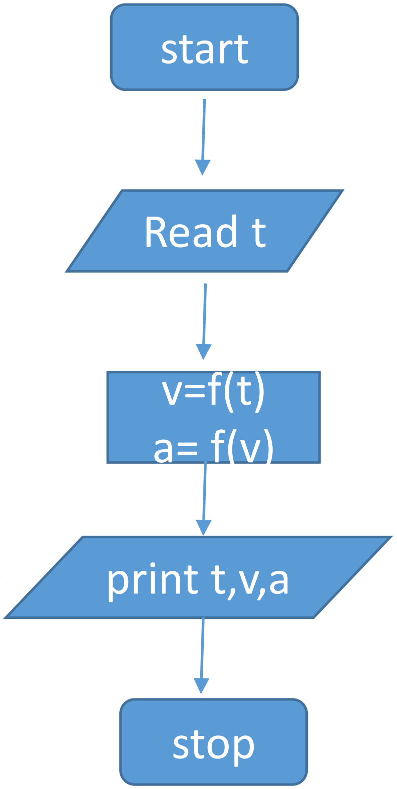
Algorithm Development

- An algorithm is a sequence of steps for solving a problem.
- Which can be expressed as a **flow chart (emphasized in this course!)**
- Engineering problem solutions to real world problems require complex algorithms.
- **Development of a good algorithm increases the quality and maintainability of a solution, and reduces the overall time required to implement a correct solution.**

Structured Programming: Top-Down Design we avoid jumping around anywhere in the program called spaghetti code which becomes extremely difficult to proof read and correct errors

- Top-down design begins with a "big picture" description of a problem solution in sequential steps. Decomposition!
- Example
- Decomposition Outline of UDF airplane example we did previously (pseudo code solution)
- Read time (or say input or get)
- Compute Velocity and acceleration (computation)
- Print Velocity and acceleration (or say output or write)
- The sequential steps are refined until the steps are detailed enough to translate to language statements.
- The refined steps, or algorithm, can be described **using pseudo code or flowcharts.**

FLOWCHARTS! Permit you to see the control almost immediately needed for programing and will be emphasized in this course.



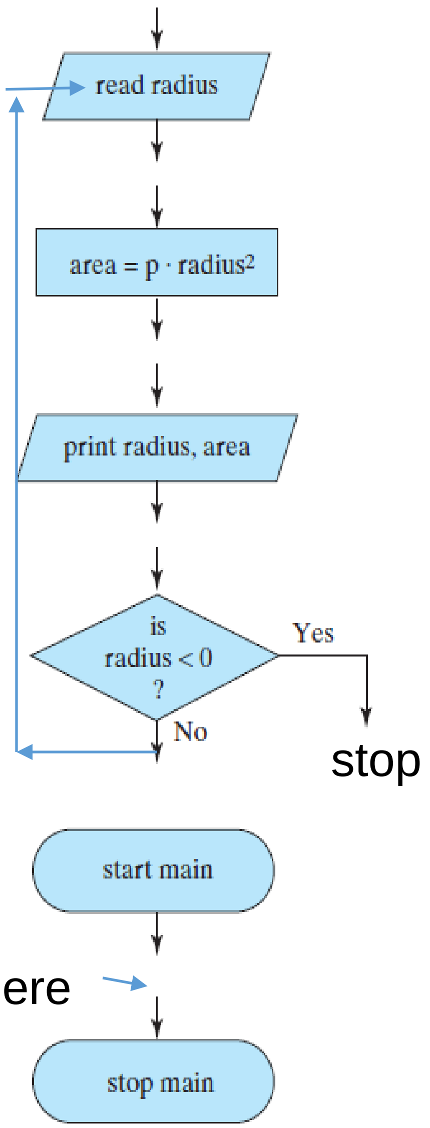
Decomposition of a problem to its key elements needed for a solution **another example-a little fancy –got repetition(next chapter) in it!**

- Area of circles –not just one but any circle (lots of them)
- (pseudo code)
- Read a radius, r
- Calculate the area of a circle, A
- Output radius and area   r,A
- **Go back and get another radius u**nless its <0 end program
- (This is basic its just to understand the next figure)

Pseudocode Notation and Flowchart Symbols

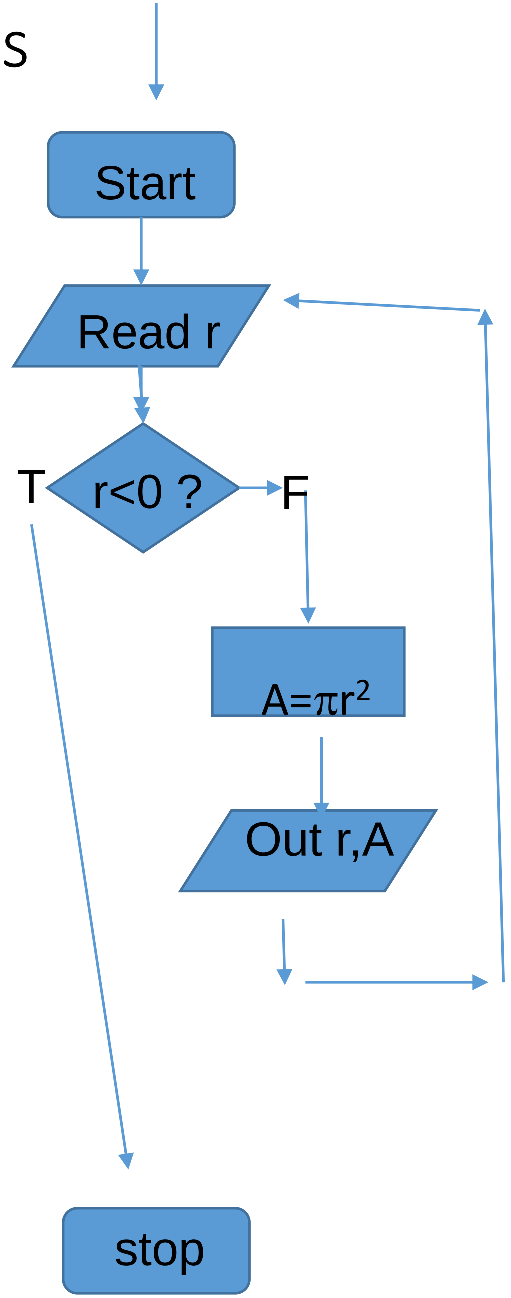
Bad design! Why?

Basic Operation	Pseudocode Notation	Flowchart Symbol
Input	read radius	read radius
Computation	Set area to p · radius^2	area = p · radius^2
Output	print radius, area	print radius, area
Comparisons	if radius < 0 then…	is radius < 0 ?
Beginning of algorithm	main:	start main
End of algorithm		stop main

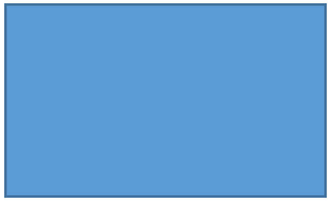


Overview All above is here

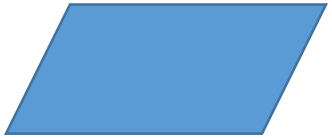
Better flowchart



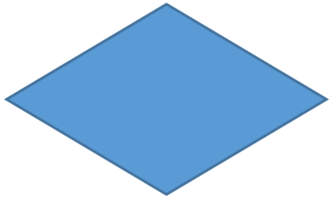
Flowcharts algorithm symbols some basics



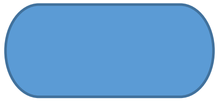
Assignments (calculations) example y = 2*pow(r,3);
Also called Statements!



Input or output example cin cout



Decision or selection (Logical) program has 2 possible paths to Travel example we will study IF statements



Start or end main program

Structured Programming: The design idea

A structured program is written using
simple control structures, including:

- **Sequence –** steps are performed one after another.
- **Sequencel flow (not spaghetti code or jumping around anywhere in the original programming with control code like GOTO anywhere! Which languages support!**
- **Selection –** one set of statements is executed if a given condition is true, a different set of statements, or no statements at all, is executed if the condition is false.
- **Repetition –**A set of statements is executed repeatedly as long as a given condition is true. **Also called a loop (we cover in detail the next chapter)**

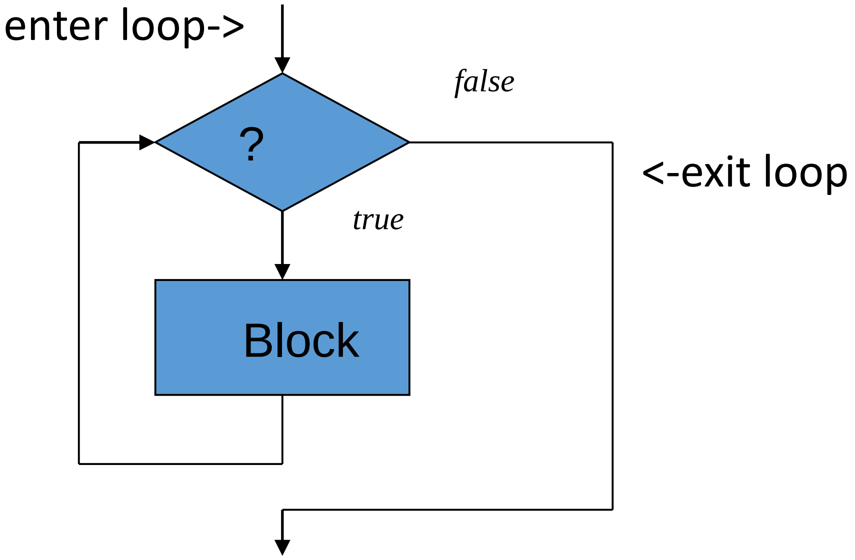
Quick look at repetition (a “while” loop!

-

//;Repetition (loop)

-   x=7;
- while (x<9)
- {
-       cout <<x<<endl;
-       x=x+1;
- }
- what is the output here?
-
-
-

Block of code is defined
With { }
Here while() command
Controls the block



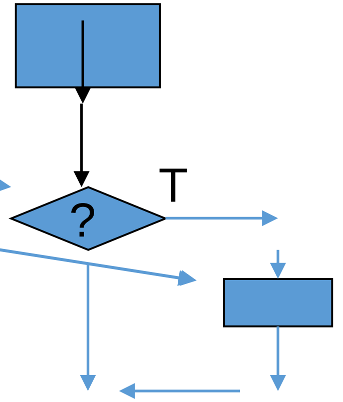
Initial introduction to the **if and If-else** selection
Control of what is executed next depends upon a
“conditional expression” like (x>=u) or (y=x+z) being true or false.
Lots more on this to follow

1. Single statement single outcome

Lone IF STATEMENT

```
x=5;  r=8;
    If( x < r)
       cout << “How are You”;
```

? = conditional expression



2.Sequence

```
c=9;  x =9;   y= 4;
```

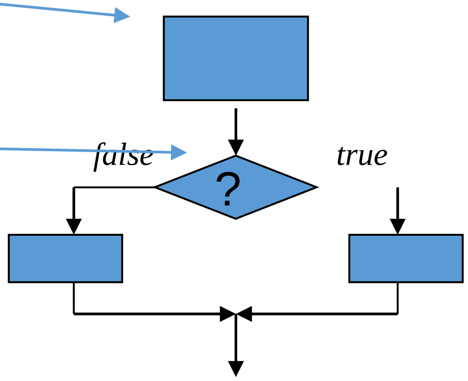
Lone IF –ELSE STATEMENT

Selection

```
if(c>y)
     v=0.5*x;
 else
     v=  sqrt(x);
```

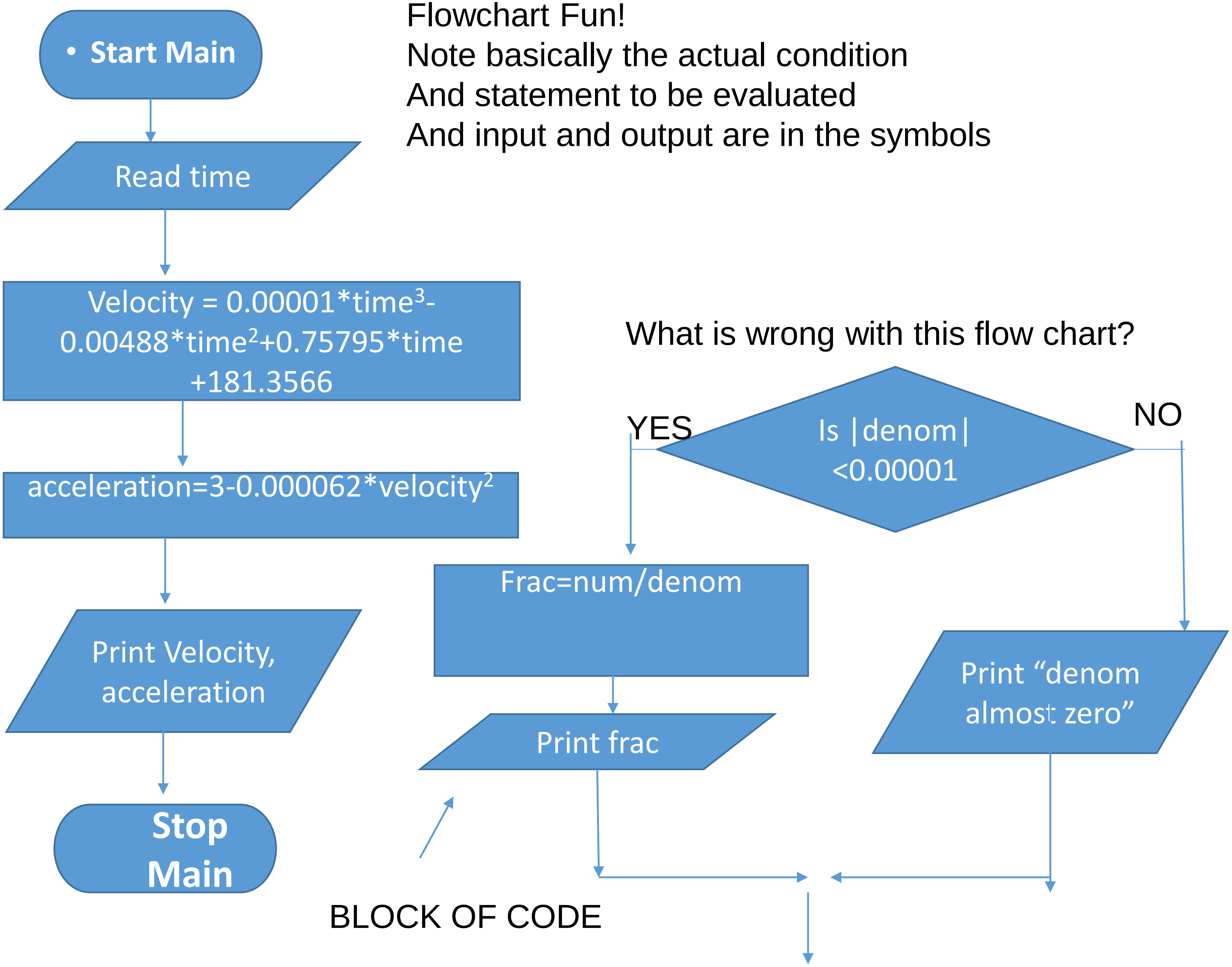
//What is the value of v?

If c= 3 what is the value of v?



Flowchart Fun!
Note basically the actual condition
And statement to be evaluated
And input and output are in the symbols

What is wrong with this flow chart?



• Start Main
Read time
Velocity = 0.00001*time3-0.00488*time2+0.75795*time +181.3566
acceleration=3-0.000062*velocity2
Print Velocity, acceleration
Stop Main
YES
Is |denom| <0.00001
NO
Frac=num/denom
Print frac
Print "denom almost zero"

BLOCK OF CODE

We now examine the types of Conditional Expressions that can be constructed. They can be very complex! Maybe even *drive you crazy*! Take them apart one step at a time when they are involved.

- A conditional expression is a Boolean expression that evaluates to true or false. (actually have the value of 1 for true and 0 for false)
- Selection structures and repetition structures rely on conditional expressions.
- **Relational operators** and **logical operators**
- are used to form conditional expressions.

Relational Operators with (conditional expression)

== equality (a==b)

NOTE NEVER USE only “=“ alone in conditions

Since it is an assignment statement and will cause logical errors, very difficult to detect!

!= non equality (a!=b)

< less than (a<b)

> greater than (a>b)

<= less than equal to (a<=b)

>= greater than equal to(a>=b)

e.g. (sin(x) <0.5) or (z+y == u) or (sqrt(x)>25)

Important Logical Operators

! Not **&&** and **||** or

The Truth Table (false =0 and true =1)

A	B	A&&B	A||B	!A	!B
false	false	false	false	true	true
false	true	false	true	true	false
true	false	false	true	false	true
true	true	true	true	false	false

MATLAB ALSO USES SIMILAR LOGICAL AND RELATIONAL OPERATORS

For efficient programming:

C++ only evaluates as much of an expression as necessary to evaluate it.

E.g. if A is false, A && B is always false, regardless of the value of B.

E.g. if A is true, A || B is always true, regardless of the value of B.

In class, complete the following truth tables

A	B	C	A&&B || B&&C	A&&B&&C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

A	B	C	A||B||C	A&&!B (ignore C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

- We generate the table **(A && B || B && C)** as an example of neat output using program 3_1 from the text that follows next :
- Note \t in the next program and how it is used to create data columns. What does it mean?

```
?*DEMO
/*  Program chapter3_1 generates a truth table for the condition:   */
/*     A && B || B && C                                             */
#include <iostream>
using namespace std;
int main()
{ //Declare and intialize objects
  bool A(false), B(false), C(false);
 //Print table header condition is (A && B || B && C)
  cout << "TABLE 3.2\n A\tB\tC\t\tA && B || B && C" << endl;
  cout << "_________________________________________________" << endl;
  cout << A << '\t' << B << '\t' << C << "\t\t\t"<<(A && B || B && C) << endl;
 //Toggle C
  C = !C;
    cout << A << '\t' << B << '\t' << C << "\t\t\t" << (A && B || B && C) << endl;
    //Toggle B and C
  B = !B;    C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t" <<(A && B || B && C) << endl;
  //Toggle C again
  C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t" << (A && B || B && C) << endl;
  //Toggle A, B and C
  A = !A;    B = !B;    C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t" << (A && B || B && C) << endl;
  //Repeat the pattern for B and C..
  //Toggle C again
  C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t" << (A && B || B && C) << endl;
  //Toggle B and C
  B = !B;    C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t"<<(A && B || B && C) << endl;
  //Toggle C again
  C = !C;
  cout << A << '\t' << B << '\t' << C << "\t\t\t" << (A && B || B && C) << endl;
  return 0;
}
```

Precedence	Operator	Associativity
1	()	Innermost First
2	Unary operators: + - **(++ --)** ! (type)	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	< <= > =>	Left to right
6	== !=	Left to right
7	&&	Left to right
8	||	Left to right
9	= **(+= -= *= /= %=)**	Right to left

CLASS EXERCISE Evaluate these conditional expression: note precedence!.

b= 3; c=5;

!(b==c || b==5.5) precedence order? Why? What is the final Boolean answer?

Given a=5.5 k=3 b=1.5

Evaluate the Boolean answer for each of the following conditional expressions

k!=a-b b-k>a b-a<k !(a==3*b) fabs(k)<3|| k<b-a

HAND IN LABORATORY TASK: LAB #8

Truth Table and modification Prog 3_1
+ EC 0.5 pt on Final average for last condition

- Run the program as found online 3_1 to generate truth Tables
- Found online in source code directory
- First run it as is. Attach output as comments as before.
- Then edit it and Run it for the following new conditions and attached output tables generated. For the original and the next two conditions
- Table for the condition A&&B&&C
- Table for the condition A||B||C

Be sure to put all three Boolean expression in one table with headings

- Ie. Original 3_1 Boolean and A&&B&&C and A||B||C in a table with 3 columns

Hand in HW #4 36 pts

Read study SECs 3.1-3.3

1. Evaluate the following conditional expressions (T or F). 4 pts

a. 13 < 6 (7 > 9) && (6 != 5) (4 < 3) || (4 >= 0) !('F' < 'M')

2. FLOWCHART THE FOLLOWING (condensed) CODE in detail 8 PTS

```
int main()
 {   int number;
  cout << "Enter an integer: ";
  cin >> number;
  if ( number > 0)     cout << "You entered a positive integer: " << number << endl;
   else
        if (number < 0)      cout<<"You entered a negative integer: " << number << endl;
        else                    cout << "You entered 0." << endl;
cout << "When is this line printed?.";
return 0; }
```

•Complete the true table for C++ logical operators.

3. Setup at table like below and Complete the truth table 16 pts

A	B	A&&!B	A||!B	!A & B	!B & !B
false	false				
false	true				
true	false				
true	true				

4. In C++, the = and == operators perform identical operations and may be used interchangeably. True or False 1 pt

5.Expalin what type of operator for Each. (logical, arithmetic, relational, unary, binary)

A. ! B. == C. % D. != E. (F. ++ G. && 7 pts

HAND IN LABORATORY TASK: LAB #9 v.3/2/18

Write a program that inputs the values of three Boolean variables, a, b, and c, from a “cin” operator (**user gives the values be sure to prompt user for what they have to give!**). Then the program determines the value of the conditions that follow as true or false.

1. !(a&&b&&c) && ! (a||b||c)
2. !(a||b)&&c

Output should include the values of a,b,c ie 0 or 1 in the patterns that follow reflecting the Boolean logic being tested.

Where “True” or “False” are in string variables.

Study this example!

for input of a=1 and b =0 and c=1 results in output looking like

!(1&&0&&1) && !(1|| 0||1) is False

!(1||0)&&1 is False

Run for the eight cases of a,b,c in Table 3.2 (both editions of text)

Warning follow this output example format else you will have to redo it!

Hint: your output statements will be large!

Selection in detail
The basic `if` Statement

Simplified Syntax

```
if (boolean_expression)
        statement;
```

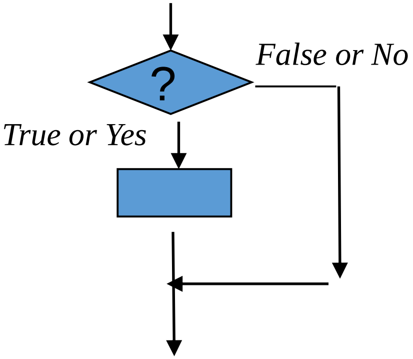
Statement Block { } execution!

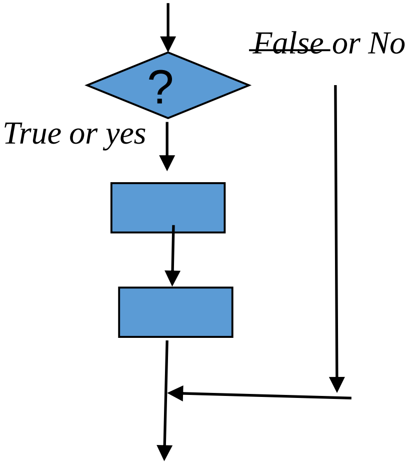
```
if (boolean_expression) {
          statement1;
               …
          statement_n;
} // style 1

if (boolean_expression)
 {
          statement1;
               …
          statement_n;
}  // style 2
```

Prof recommends style 2

Examples of writing Basic IF statements

```
//single statement executed if x>0
if (x>0)
     ++k;
or


if (x>0)  ++k;


// block(multi statements) executed if x>0

//Style 1 indented statements, note braces{}
//NOT RECOMMENDED BY PROF!

if (x>0) {
      x = sqrt(x);
     ++k;
}

//Style 2 Easier to read, better organized

if (x>0)
 {
      x = sqrt(x);
     ++k;
 }
Note we use braces to show block
 and indent to show what belongs in a block!
```





Another example

```
If (!error && x>0)
 {
    sum = sum+x;  or    sum +=x
    count = count+1  or  ++count
 }
```

IN CLASS PROBLEM Nesting of if statements

What will happen here???

USE table to follow variables in memory   CALLED PLAY COMPUTER

DO THIS NOW IN CLASS? On Paper!

output for input of a =100,35, 10

```
#include<iostream>  //io
Using namespace std;
Int main()
{
   int a, count(1), sum(100);
   cin>>a;
   if(a<40)
       {
           ++a
            if (a<20)
              {
                   ++count;
                   sum=sum+a;
               }
           }  // first if end
    cout <<(double) sum/count<<“    “<<a;
    return 0;
{
```

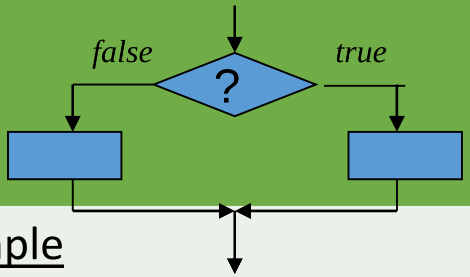
VARIABLE TABLE			OUTPUT
a	count	sum	

Selection The if-else Statement

Syntax

```
if (boolean_expression)
        statement;
[else
        statement;]
```

```
if (boolean_expression)
  {
                block ...
  }
[else
  {
                block
  }
```



Example

```
if (x>0)
 {  //statement block executed if x>0

 }
else
 {    //statement block executed if x<=0

 }
```

NOTE: Using multiple simple if statements when if-else logic is called for will not
Be acceptable!

More Examples If-else

```
if (d<30)
   velocity = 0.425 + 0078*pow(d,2);
else
   velocity = 0.625 +0.08*d +0.25*pow(d,3);

if (fabs(denominator) <.0001)
    cout << “Forget dividing denominator is almost zero” << endl;
else
{
  fun =numerator/denominator;
  cout << “fun = “ << fun<< endl;
}
```

NOT ACCEPTABLE SOLUTION TO THE FIRST EXAMPLE

```
if (d<30)      velocity = 0.425 + 0078*pow(d,2);

If (d>=30)   velocity = 0.625 +0.08*d +0.25*pow(d,3);
```

Would the latter work? Why is this an inefficient use of a program?

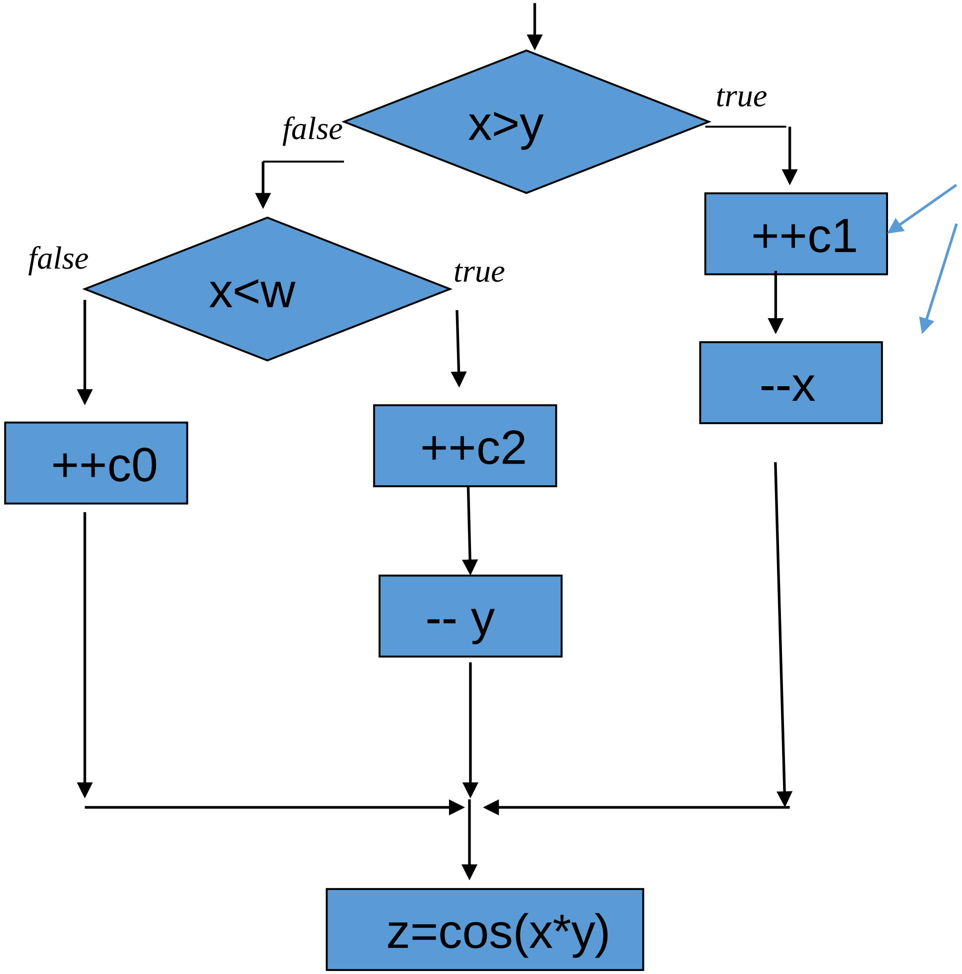
More examples

WHY SOME SOFTWARE CRASHES
Choices we make for a
hot dog as an example?

Nesting if ELSE

```
If (x>y)
   {   ++c1
        -- x
    }
 else if (x<w)
          { ++c2
               --y
           }
          else
           {
              ++c0
            }
 z= cos (x*y)
```

Note: Exact conditions and statements
Should be in the symbols



Each statement
Has separate
rectangle

HAND IN (HW #5) read study section 3.4   part I &II (next) 32 points

Do EXAM PRACTICE PROBLEMS **1 to 4   and 7 to10   8PTS**

11. show the output for the following program segment

```
 int A, B; A = 6;  B = 1;                                          3 pts
if (A > B)
   {      A = A * B + 2;
          B ++;    }
else
    {     A = A / 2;
           B = B + 4;}
cout << “A = “ << A << “ B = “ << B;
```

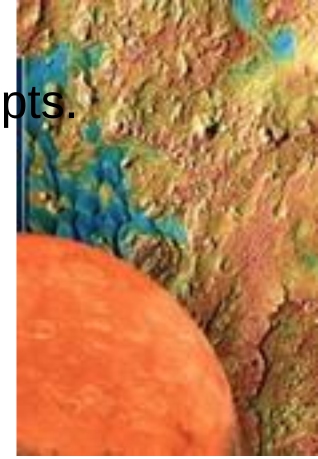
12. show output for each case   X= 3;   2.   X= 8;   3. X= 15;   4. X=30;   8pts   (2 each)

```
           int X, Y;
          if (X <= 5)
            {    Y = X * X;
                 X = X + 5;}
          else
             {        if (X < 10)
                         Y = X - 2;
                      else
                           if (X < 25)
                           Y = X + 10;
                         else
                              Y = X / 10;
                       X++;
             }
           cout << “Y = “ << Y << “; X = “ << X;
```

HW#5 continued next slide

Homework # 5 continued part II

13.   Draw a flow chart! show statements in proper Flow chart symbols. 8 pts.



```
if (x>y)
   if (y<z)
      {
        ++k;
        j=y+9;
      }
    else
       {
           y=log(x);
           --k;
        }
 else
     sum =sum+y+x;
cout << y<< x<<sum;
```

14. Which of the following are characteristics of a good algorithm: 3 pts

A. Improved quality of the solution.

B. Improved maintainability of a solution.

C. Reduced implementation time.

D. All of these are characteristics of a good algorithm.

15Structured programming requires the use of complex control structures.

True or False   2 pts

HAND IN LABORATORY TASK: LAB #10

Write a C++ program to calculate the wind chill index and report on frost bite times. In other words how long till a person is frost bitten! With a given temperature and wind Velocity you will determine the time for frostbite by evaluating the wind chill index. T_{wc} !

Use the Following formula.

T_{wc}=35.74 +0.6215T_a-35.75$V^{0.16}$+0.4275 $T_a V^{0.16}$

Where T_{wc} is known as the “wind” chill index, based on the Fahrenheit scale ,

T_a is the air Temperature, measured in °F and V is the wind speed, in mph.

Prompt the user for the required input (to solve formula for $\mathbf{T_{wc}}$)

If the index value of T_{wc} is below -19 report a frostbite time of 30 minutes or less.

else If the index value is below -48, report the Frostbite time of 5 minutes or less.

Be sure to add the case when there is *no Frostbite*, i.e. Person is Safe!

note: use only one **IF/else** structure to cover the 3 cases

not 3 **IF’s in a row(bad programming).**

Find on your own by trial and error input data to Run for each of the three possible outcomes. Output all values, $\mathbf{T_{wc}}$ **,$\mathbf{T_a}$& V** and appropriate statement!

Run one **hand check** for the value of T_{wc} for one of your inputs of T_a and V

Be sure to show your hand check and value of the output that confirms formula is working properly by pencil on the lab exercise..

Avoiding Bugs

- To avoid confusion and possible
  errors when using if/else statements, you should use {} to clearly define statement blocks.
- As I mentioned == works fine for intergers but what to do with other types. Answer is below.
- **Do not use == with real values**
  - **Instead of `x==3.14`, use `fabs(x-3.14)<0.0001`**
  - **`Why does this work?`**

Some notes to avoid problems Do not confuse relational equality(==) and assignment operators(=)!

- What will output here?

```
#include <iostream>
using namespace std;
int main()
{
int x(4), y(5);
  if(x = y)
   {
     cout << x << "is equal to " << y << endl;
   }
return 0;
}

//MAJOR ERROR OF ALL PROGRAMMERS! CHECK OUTPUT CAREFULLY!
//Also using equality with integer is ok but not double!!!
//double   x, y
  //if ( x==y)    do not use this for double variables equality can be illusive!
//If (abs(x-y)< 0.000001)    then maybe then they are essentially equal!
```

In class: exercise

Draw flow chart and write C++ statement(s) for each of the following

1. When the square root of ‘quad’ is less than 0.4 print the value of quad

2. If the absolute difference of Voltage1 and Voltage2 is larger than 20.0 print the values of each voltage.

3. if the natural logarithm of y is equal to 4.23 set time to zero and decrement the count. Recall equality in using decimal numbers!

Our first **Numerical Technique**!
Linear Interpolation followed
Why do we use it?

- Answer!

- Collecting data in a computer usually results in a collection of samples.
- Often, we are interested in what happens 'in between' the sample points.
  - i.e. we need to interpolate between the samples.

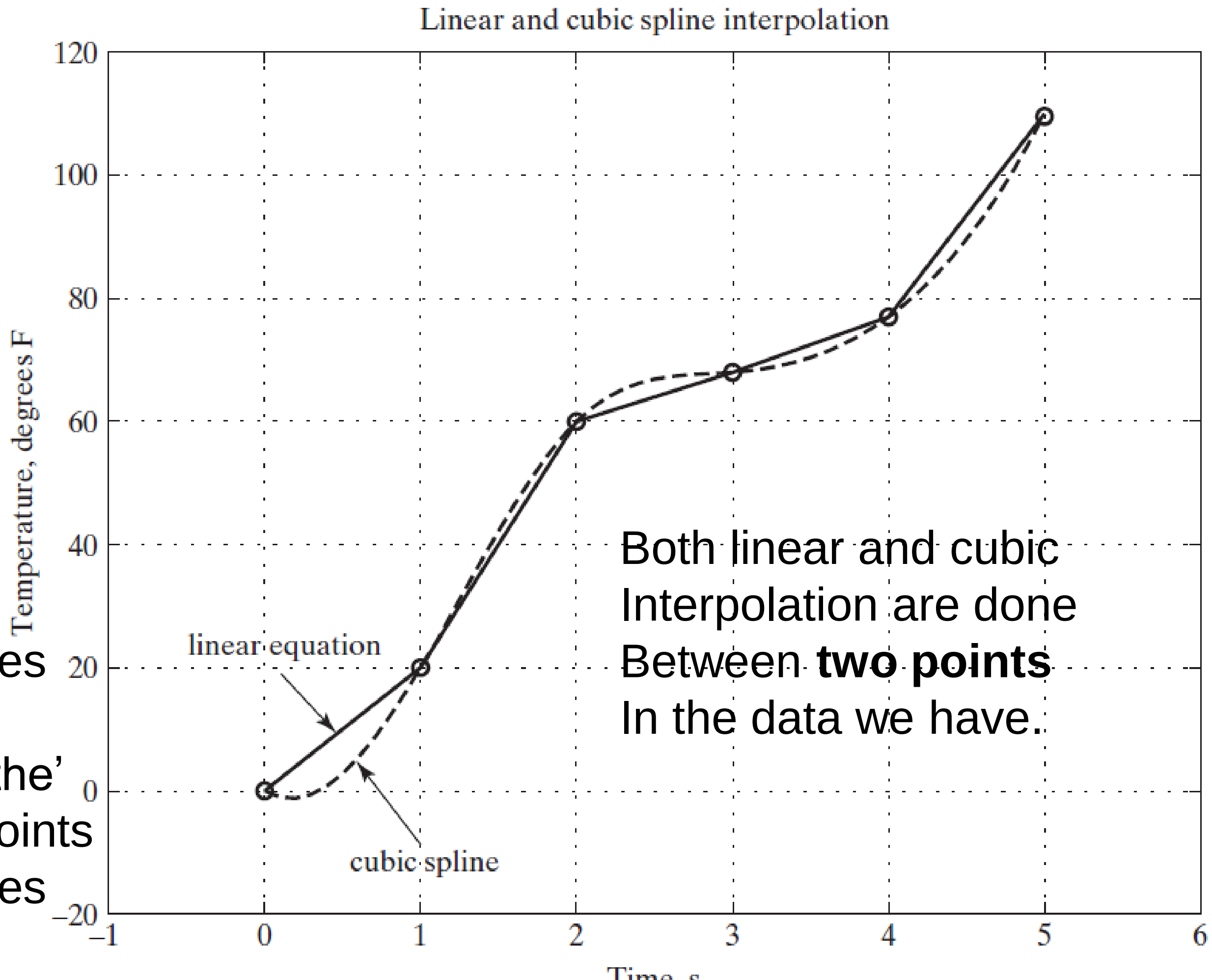
- Two ways to find the intermediate value
- 1.Linear interpolation.
- 2. Cubic spline interpolation we will cover this in MATLAB OR LATER IN THE TERM WITH C++
- The latter are both common techniques for interpolation .

Interpolation Example
Temperature as a function of time of a cylinder head in the engine of a racing car.
We only have a table of certain point of(T,t)

Time,s	Temperature, F
0.0	0.0
1.0	20.0
2.0	60.0
3.0	68.0
4.0	77.0
5.0	110.0

Points are marked on
Graph on the right->

Note the linear and cubic
Spline equations give us values
Between the points we have!
We only know the points not the'
Data values in between the points
Interpolations lets us get values
For in between the points.



Both linear and cubic
Interpolation are done
Between **two points**
In the data we have.

Given two points how do we estimate the values between them.
The linear interpolation generates a straight line between the points
From which we get value within the two points.
The last graph showed the straight lines between points that are
Assumed.

Time,s	Temperature, F	
0.0	0.0	
1.0	20.0	
2.0	60.0	
3.0	68.0	
3.5	F?	->ANSWER IS 72.5 FOR Time = 3.5 sec SEE BELOW.
4.0	77.0	
5.0	110.0	

We want F for time=3.5 from the data set.
We assume a linear relationship between values which means we can
Set up simple ratios illustrated with braces above to get matching values.
So we have options for ratios but as long as we are consistent we get values
We can do the ratio in various ways for here we pick

(F-68)/(77-68) = (3.5-3)/(4-3) -> (F-68)/9 =(.5/1) or

F = 68 + 9*.5 =68+4.5=72.5

Linear Interpolation ->developing a fancy formula

f(a), a, f(c), and c are known. We wish to estimate f(b) where a < b < c. **See equation below.** In our last example f(b) =F and b=3.5 a=3.0 c=4.0 f(a)=77 f(c) =68 generates the F =72.5 solution. See graph that puts our last table values in perspective

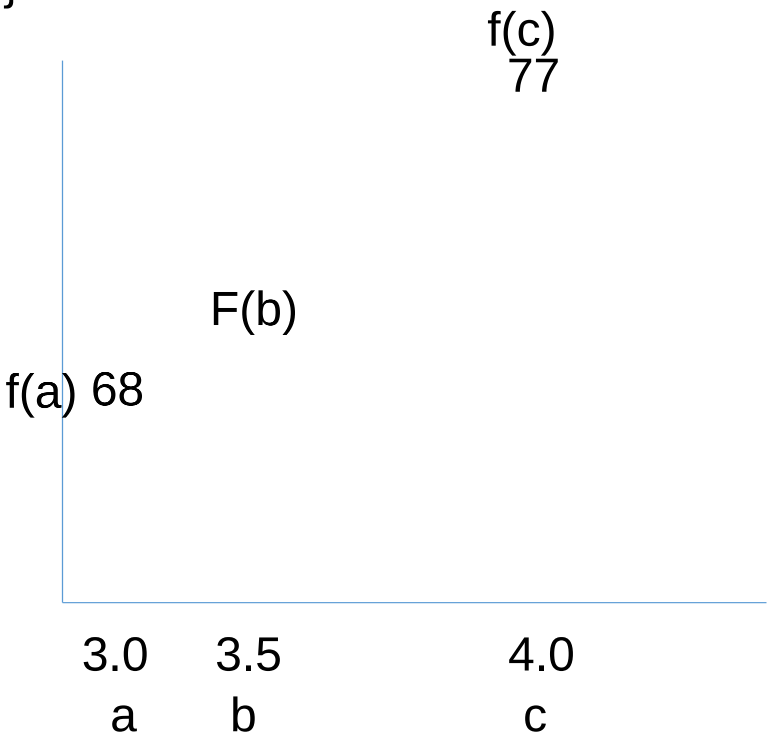
Linear interpolation is a straight-line approximation between the known points. The formula below is used with the previous data on the right.

{ f(b)-f(a) } / { f(c)-f(a) } = {b-a} /{c-a }
(f(b)-68)/(77-68) = (3.5-3)/((4-3)
-> (f(b)-68)/9 =(.5/1)
f(b) = 68 + 9*.5 =68+4.5=72.5

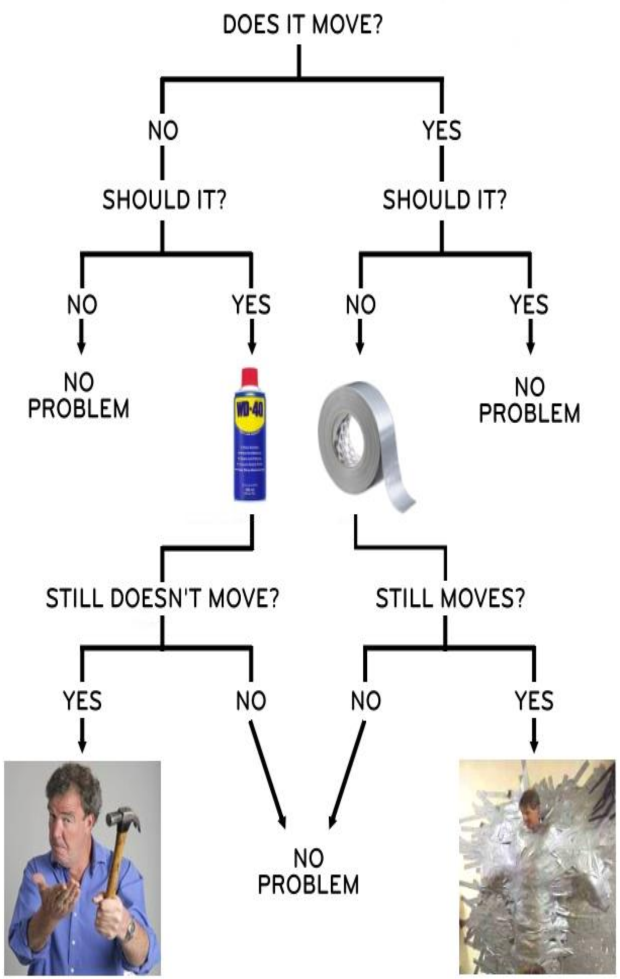
Or Finally we can state for f(b)!

$$f(b)=f(a) + \frac{(b-a)}{(c-a)} [f(c) – f(a)]$$

Applied to our previous case
We have
f(b)= 68 +(3.5-3)/(4-3) [77-68]
=68+4.5=72.5 as before.



ENGINEERING FLOW CHART



DOES IT MOVE?

NO

SHOULD IT?

NO

NO PROBLEM

YES

STILL DOESN'T MOVE?

YES

NO

NO PROBLEM

YES

SHOULD IT?

NO

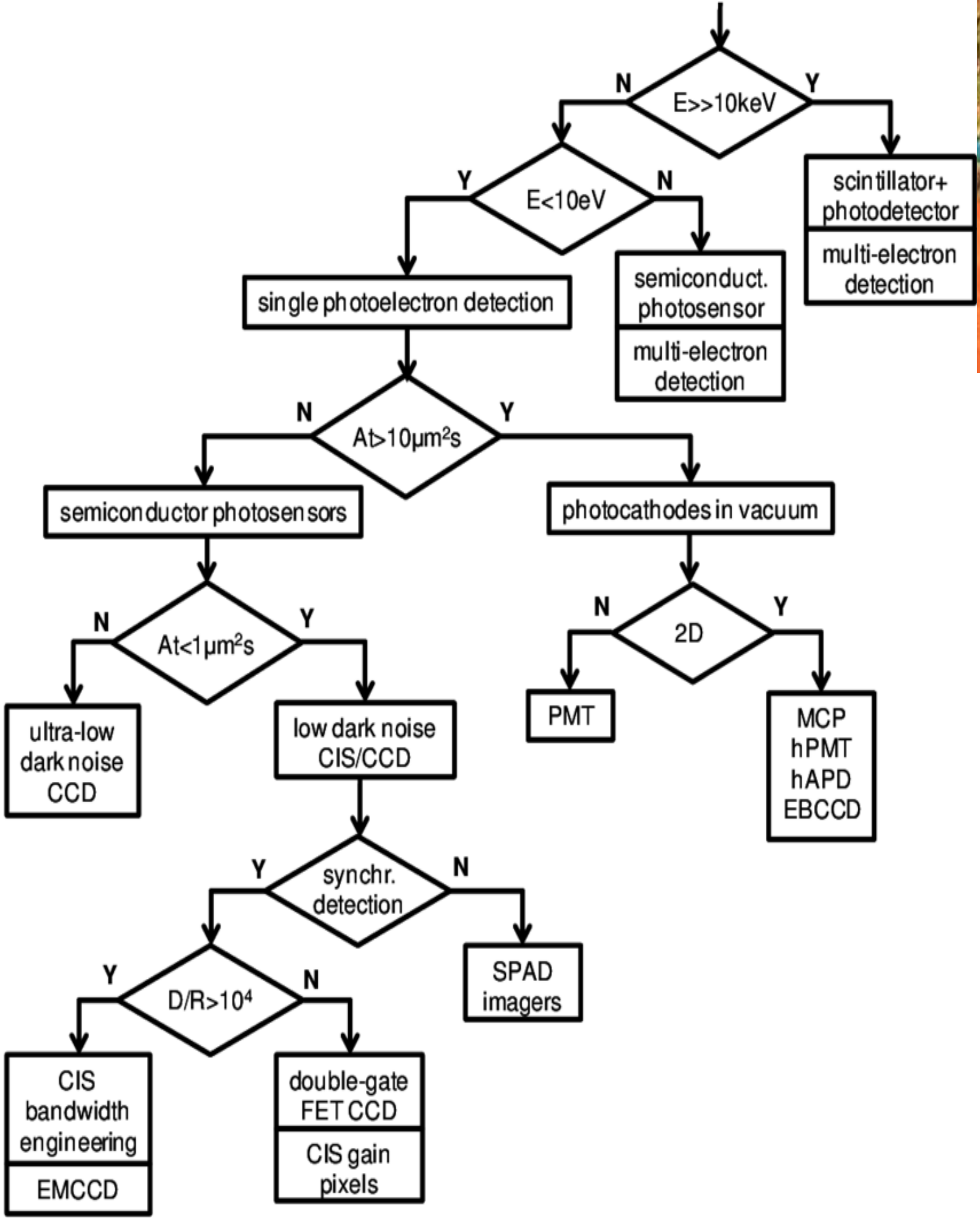
STILL MOVES?

NO

YES

YES

NO PROBLEM



E>>10keV

Y

scintillator+ photodetector

multi-electron detection

N

E<10eV

N

semiconduct. photosensor

multi-electron detection

Y

single photoelectron detection

At>10μm²s

Y

photocathodes in vacuum

2D

N

PMT

Y

MCP

hPMT

hAPD

EBCCD

N

semiconductor photosensors

At<1μm²s

N

ultra-low dark noise CCD

Y

low dark noise CIS/CCD

synchr. detection

N

SPAD imagers

Y

D/R>10⁴

Y

CIS bandwidth engineering

EMCCD

N

double-gate FET CCD

CIS gain pixels

Class problem to solve now:

Time (s)	Temp (degrees F)
0.0	72.5
0.5	78.1
1.0	86.4
1.5	92.3
2.0	110.6
2.5	111.5

$$f(b)=f(a) + \frac{b-a}{c-a}\,[f(c) - f(a)]$$

Solve for the Temperature after 1.3 seconds but setup the last formula with the Appropriate values ? .

HW #6 read/study sections 3.5-3.6 HAND IN: DO THIS HW BY HAND 19 pts

Given the Table of Time vs Temperature below

1. 5 pts Use the linear interpolation formula developed. To find The temperature at a time of 1.7 seconds Show the formula and What values are being assigned to each part. See example at the End of the section 3.5.
2. 5 pts Use the same formula or inverse formula but now find the time when the temperature I at 79.6 degrees .

Time (s)	Temp (degrees F)
0.0	72.5
0.5	78.1
1.0	86.4
1.5	92.3
2.0	110.6
2.5	111.5

3. Do Exam practice problems 5, 11-13 4 pts

34. 3 pts. Linear interpolation is used to …

a. approximate a value outside of the range of known data values by determining a line which best fits the entire data set.

b. approximate a value outside of the range of known data values by extending the line which connects the last two known data points.

c. approximate a value inside of the range of known data values by determining a line which best fits the entire data set.

d. approximate a value inside of the range of know data values by connecting adjacent points in the data set with a line.

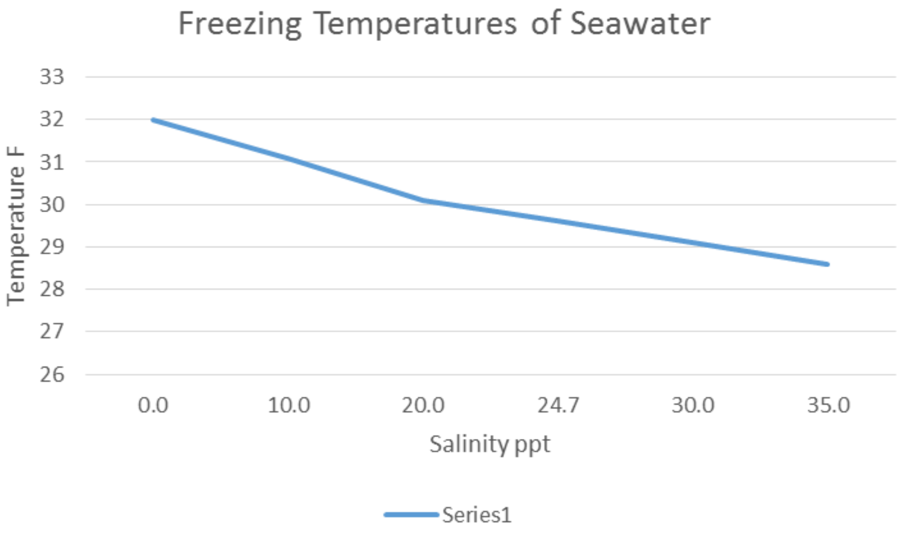
Problem Solving Applied: Freezing Temperature of Seawater
USING C++ for linear interpolation

We examine salinity of sea water which
is the amount of salt dissolved and varies
from 33 to 38 ppt (parts per thousand)
= 3.3 to 3.8%
The higher the salinity the lower the freezing point.

Very important in Science and Engineering:
3.3%-> means-> fractional part is =33/1000 =.033
percent x 100 -> .033x100 =3.3%
Definition of % of A to B =A/B *100
%error or % difference of a measurement M to a known value
K = (M-K)/K*100 (M-K) is the error or difference
(M-K)/K is the fractional part.

Data obtained with electrical conductivity apparatus. More salt more conductivity
-> higher Freezing point
can be established!

Salinity(ppt)	Freezing Temp F°
0 .0	32.0
10.0	31.1
20.0	30.1
24.7	29.6.
30.0	29.1
35.0	28.6

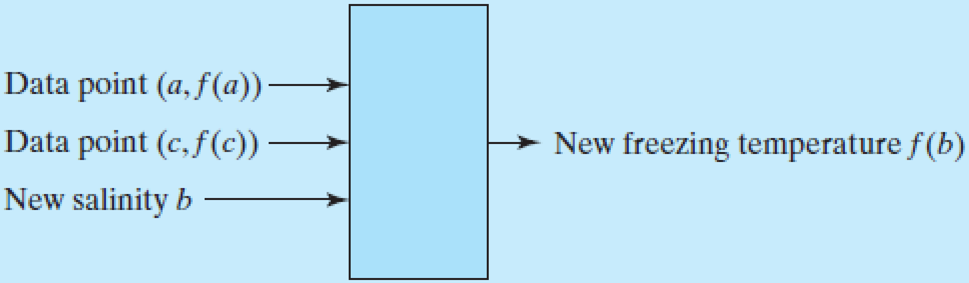


1. PROBLEM STATEMENT

Use linear interpolation to compute a new freezing temperature for water with a specified salinity.

2. INPUT/OUTPUT DESCRIPTION

The following diagram shows that the input to the program includes two consecutive points $(a, f(a))$ and $(c, f(c))$ and the new salinity measurement b. The output is the new freezing temperature.



Data point $(a, f(a))$

Data point $(c, f(c))$

New salinity b

New freezing temperature $f(b)$

3. HAND EXAMPLE

Suppose that we want to determine the freezing temperature for water with a salinity measurement of 33 ppt. From the data, we see that this point falls between 30 and 35 ppt:

a	30	29.1	f(a)
b	33	?	f(b)
c	35	28.6	f(c)

Using the linear equation formula, we can compute $f(b)$:

```
    f(b) = f(a) + (b-a)/(c-a) . (f(c)-f(a))

         = 29.1 + 3/5 . (28.6 -29.1)

         = 28.8
```

As expected, this value falls between $f(a)$ and $f(c)$.

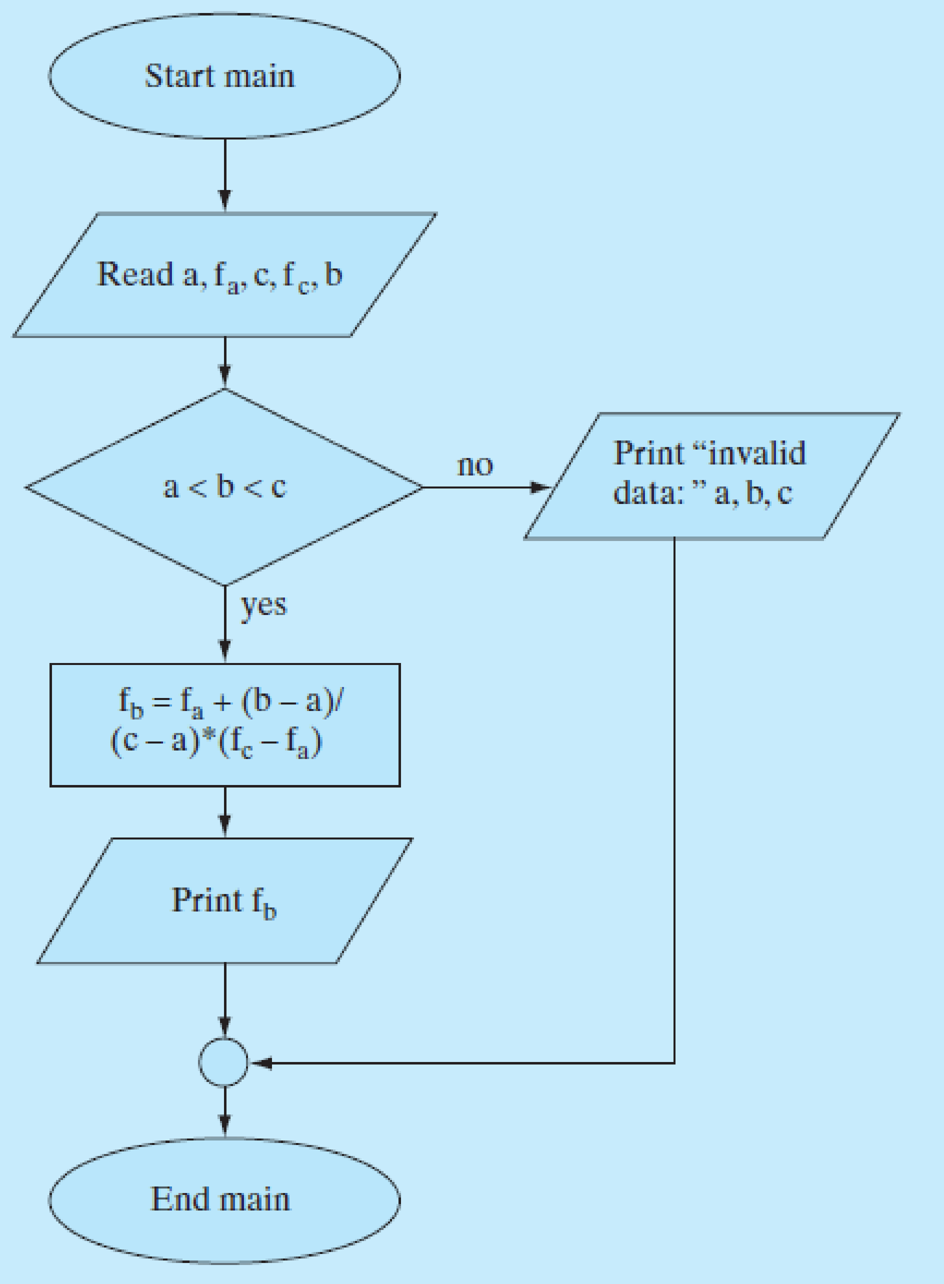
4. ALGORITHM DEVELOPMENT

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 the coordinates of the adjacent points and the new salinity value.
2. Validate input.
3. Compute the new freezing temperature.
4. Print the new freezing temperature.

HAND IN LABORATORY TASK: LAB #11`


Start main
Read a, f_a, c, f_c, b
a < b < c
no
Print "invalid data: " a, b, c
yes
$f_b = f_a + (b - a)/(c - a)*(f_c - f_a)$
Print f_b
End main

Write the program for this flow
chart IN YOUR OWN STYLE
NOT THE TEXT BOOK!
COMMENTS FOR YOU
AND A USER IS A MUST!

1. Use it to get the freezing
Temperature for a
salinity of 33 gives the
Previous calculated freezing
Point. Check the text answer against yours

2. Using the program what is the freezing
Temperature of water for a salinity of 23.6
Does your answer make sense? Why or why
Not?
DO THIS last one BY A HAND
CALCULATION TO
CHECK YOUR ANSWER..SHOW THIS ON
THE LAB EXERCISES YOU HAND IN.

HAND IN LABORATORY TASK: LAB #12

- Part 1. Modify the previous program to determine the salinity for a temperature of 31.0 by reversing the technique formulas (either write a new formula or interpret the current one in reverse. i.e. a,b,c is now temperatures and solve for f(b) now salinity). Be careful of the check in the input data since temperatures drop in the table while salinity were going higher)
- Do a hand calculations on the paper to show your answer makes sense using the temperature of 31.0
- Part 2. Modify again the program using a **while loop** to keep asking for a temperature and produce Salinities for 4 additional temperature cases of your choices (see text table for reference) be sure the provided input temperature number makes sense. When would it not. The loop should end on the entry of a temperature value for water(your choice)!
- Hand in the final version and hand calculation on it and answers from each part.
- Ie. Include the temperature of 31.0 and four other cases you did.

SKIP the rest of these notes FALL 18 Selection Statements:`switch` Statement

If you have a problem that requires lots of selection and you find you are nesting A bunch of IF-ELSE statements than the “switch “ statement may be an easier Solution for your problem.

EXAMPLE: We have a large piece of machinery and a sensor inside is picking up the Temperature, we want a status code which comes from the Sensor depending on the temperature to alert the operator of the machine. (sound like modern automobiles!)

Code	Meaning
10	Warning Too hot, turn the system off
11	Warning check temperature every 5 minutes
13	Warning, better turn on the circulating fan
16	Emergency, Abandon the building explosion imminent
Any other number	Normal operating range, no problems

The nested If-Else solution follows: Assume control variable “**code”** has been assigned as an integer and value above come from the Temperature Sensor into our program

```
If (code==10)
{      cout << “Warning Too hot, turn the system off”<< endl;
}
Else
{
    if (code== 11)
     {
        cout << “Warning check temperature every 5 minutes ”<< endl;
     }
    else
     {
       if (code ==13)
         {
           cout << “Warning, better turn on the circulating fan ” << endl;
         }
       else
          {
            if (code==16)
               {
                 cout << “Emergency, Abandon the building explosion imminent” << endl;
               }
            else
                {
                  cout << “Normal operating range, no problems” << endl;
               {
           {
       {            NOTE ONCE ANY CONDITION IS TRUE THE PROGRAM MOVES ON
{      THIS IS EFFICIENT CODING . THE SYSTEM DOES NOT HAVE TO TEST ALL.
```

THE FOLLOWING WILL HAVE THE SAME BASIC LOGIC BUT
IS VERY BAD PROGRAMMING!
WHY?   Such coding is not acceptable in this class.

```
If (code==10)
{         cout << “Warning Too hot, turn the system off”<< endl;
}
 if (code== 11)
 {
          cout << “Warning check temperature every 5 minutes ”<< endl;
  }
if (code ==13)
  {
           cout << “Warning, better turn on the circulating fan ” << endl;
   }
 if (code==16)
   {
            cout << “Emergency, Abandon the building explosion imminent” << endl;
    }
  else
     {
            cout << “Normal operating range, no problems” << endl;
     {
```

The `switch` Statement

Syntax

```
switch (control_expression) {
      case constant:
             statement(s);
             break;
[     case constant:
             statement(s);
             break;
[…] ]
[ default:
                 statement(s); ]
}
```

- Control expression must be an integral type (e.g. char, int, bool…)
- ie
- **A single value!**
- **NEEDS #include <stdlib.h>**
- **Special library.**
- Break statements are not required; without a break statement execution ‘runs through’ other case labels.

The switch statement to do the previous Temperature warning system would look as follows.

The break statements just are efficient ways to pass control immediately the statement after the switch statement. Thus saving computation time, especially if the cases have large blocks of code. WARNING NO BREAK STATEMENT MEANS THE NEXT CASE WILL BE EXECUTED! Not using means unstable situations can Arise!.

```
switch (code)
{
      case 10:
           cout << “Warning Too hot, turn the system off”<< endl;
            break;
        case 11:
            cout << “Warning check temperature every 5 minutes ”<< endl;
            break;
        case 13:
            cout << “Warning, better turn on the circulating fan ” << endl;
             break;
         case 16:
              cout << “Emergency, Abandon the building explosion imminent” << endl;
               break;
          default:
               cout << “Normal operating range, no problems” << endl;
                break;
{
NEXT STATEMENT  (EXECUTION OF THE BREAK STARTS HERE IMMEDIATELY
/* Program money converter  Robbins version of page 126  */
#include<iostream> //Required for cin, cout, endl.
#include<stdlib.h> //NEW calling on use of toupper() function
/* toupper() converts entered character to capital so switch
*below will work whether you enter for example: an 'E' or 'e‘*/
using namespace std;
/* To get currency dollars to country letters are entered for currency name
* E=> Euros of Italy P => Pesos OF Mexicao S= British pound */
int main()
{          double dollars, equivalentCurr;
           char currencyCode;
           const double ECONVERSION(0.8795), PCONVERSION(18.95), SCONVERSION(0.6529);
           // Prompt and get user input from the keyboard.
           cout << "Enter dollar amount to be converted" << endl;
           cin >> dollars;
cout<< "Enter code for the currency\n" << "E =>for Italian Euros\nP=>Mexican Pesos\nS=British pound
sterling\n";
           cin >> currencyCode;
           switch (toupper(currencyCode))
           {
           case 'E':
                      cout << "Converting dollars to Italian Euros..\n";
                      equivalentCurr = dollars*ECONVERSION;
                      break;

           case 'P':
                      cout << "Converting dollars to Mexican Pesos..\n";
                      equivalentCurr = dollars*PCONVERSION;
                      break;
           case 'S':
                      cout << "Converting dollars to British pound sterlings..\n";
                      equivalentCurr = dollars*SCONVERSION;
                      break;

           default:
                      cout << currencyCode << " not supported this week\n ";
                                 equivalentCurr = dollars;
           }//end switch
           cout << " Equivalent amount is \n"; cout << equivalentCurr << endl; return (0);
}// end main
```

HAND IN LABORATORY TASK: LAB #13

- Run last Program on currency conversion found online called
- **money.cpp** which is my version of programs 3_3 in the text which uses the switch statement.
- Make the modification below!
- Program converts dollars to different currencies. You input dollar amount and get the value of a countries conversion when you enter a code for the country. Ie. You have the value for a 1 dollar conversion!
- 1. So modify this program for Hand in which will loop continuously until the dollar amount entered is zero or less . Dollar value controls the loop!
- 2. Look up on the web the current currency conversion for dollars for Kenya, Peru, China, India, Papua New Guinea. Add these choices to your program
- Then run for all cases to convert $500 to the local currency with the output naming the input curreny and output currency unit converted to fully.
- Example: $500 is equal to 200 pounds sterling
- Hand in outputs for each case with the final version of the program as usual.

Hand in HW # 7. read/study 3.7 15 pts

- Be sure to read and study the **Summary section** of each
- chapter to understand what you should know with the exception of the section(s) we skipped

1. 4 pts HAND IN the Exam Practice 14,15 only of Chapter 3rd ED

   For the 4th edition Exam Practice 19,20

- Note for the memory snapshot 14,15 (3rd ed and questions.) and 19,20 4th ed. Show the value of variables after the statements have been executed and any output that might occur in the end

2. Also hand in Practice problem page page 122 3rd ed and page 133 4th ed. 5 pts
ie. Convert the if/else statements to an efficient switch statement

3. Additional Memory snapshots for these program segments, A,B,C 6 pts 2 ea

```
A. int n1(0); count(0);
     If (n1>0)  ++count;
     else if (n1<0)  --count;
     count =count+2;
```

```
B. int n1(-1),count(0);
     if (n1>0)  ++count;
     else if (n1<0)   -- count;
      count=count+2;
```

```
C.  Int mon(1);
     switch (mon)  {
          case 1:
                ch="J";
              case 2:
                 ch="F";
                  break;
```