Introduction to Looping

Larry Caretto
Computer Science 106

Computing in Engineering and Science

March 7, 2006

California State University Northridge

Outline

- Review last topic
  - Choice (if) statements
- Looping
  - Basic idea for looping
  - while and do while statements
  - Practice writing loops

California State University Northridge

Review Choice (if statements)

- Three structures if, if-else, and if-else-if
- Based on statement if (***<condition>***)
- Condition used relational operators (<, >, <=, >=, ==, !=) and logical operators not(!) and(&&) or(||)
- Condition evaluates to true or false
- In if-else and if-else-if only one block of code is executed
- Nested if blocks

California State University Northridge

Review Type bool Variables

- Type bool variables have two possible values: true and false
- Can be used to hold result of expressions that give these values
- leapYear = year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)
- Test bool variables in if statements and use with logical operators

if (leapYear && month == 2) days = 29;

California State University Northridge

Review DeMorgan’s Laws

- Have two bool expressions, a and b, that can have values of true or false
- Combinations of conditions for a and b satisfy both of the following
- !(a && b) = !a || !b
- !(a || b) = !a && !b
- Proved these using truth table
- Application to data validation follows

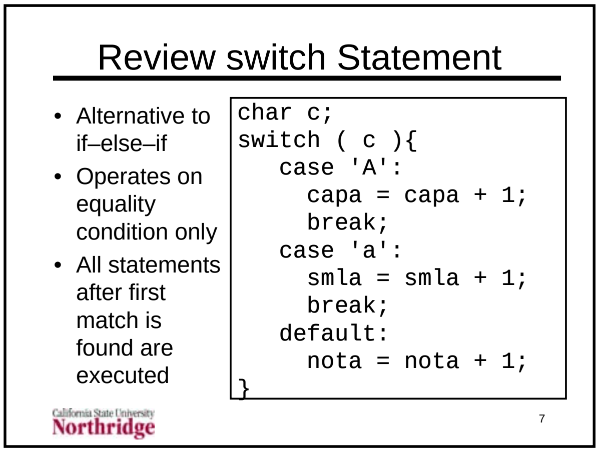
California State University Northridge

Review Data Validation

- Apply DeMorgan’s Law to Validation

```
badData = x < Min || x > Max;
goodData = x >= Min && x <= Max;
goodData = !badData;
goodData = !(x < Min || x > Max);
!(a || b ) = !a  && !b
goodData = !(x < Min) && !(x>Max)
goodData = x >= Min && x <= Max
```

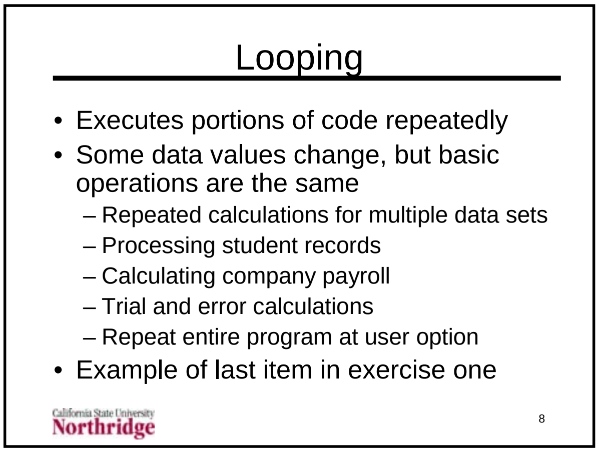
California State University Northridge



Review switch Statement

- Alternative to if–else–if
- Operates on equality condition only
- All statements after first match is found are executed

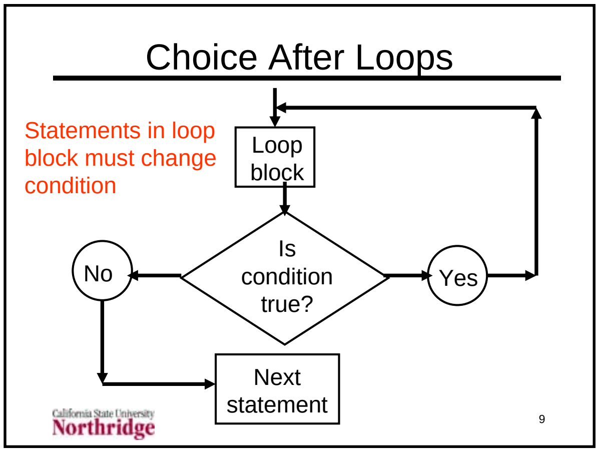
```
char c;
switch ( c ){
   case 'A':
     capa = capa + 1;
     break;
   case 'a':
     smla = smla + 1;
     break;
   default:
     nota = nota + 1;
}
```



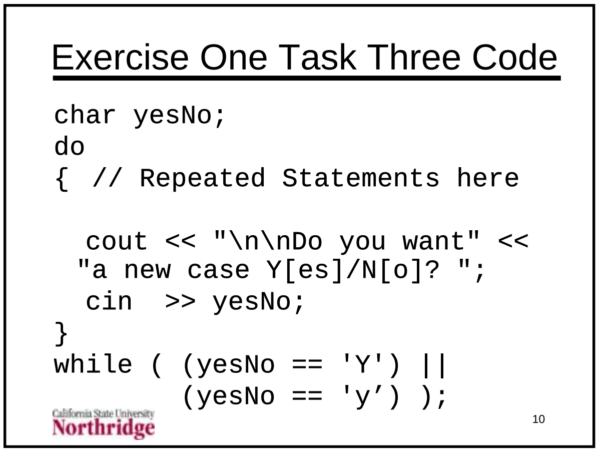
Looping

- Executes portions of code repeatedly
- Some data values change, but basic operations are the same
  - Repeated calculations for multiple data sets
  - Processing student records
  - Calculating company payroll
  - Trial and error calculations
  - Repeat entire program at user option
- Example of last item in exercise one

Choice After Loops



Statements in loop block must change condition

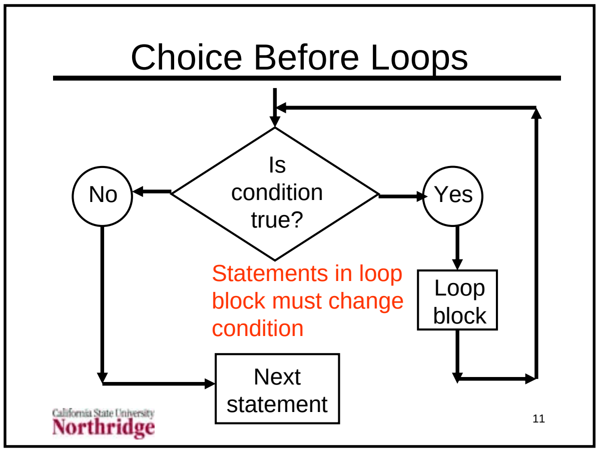


Exercise One Task Three Code

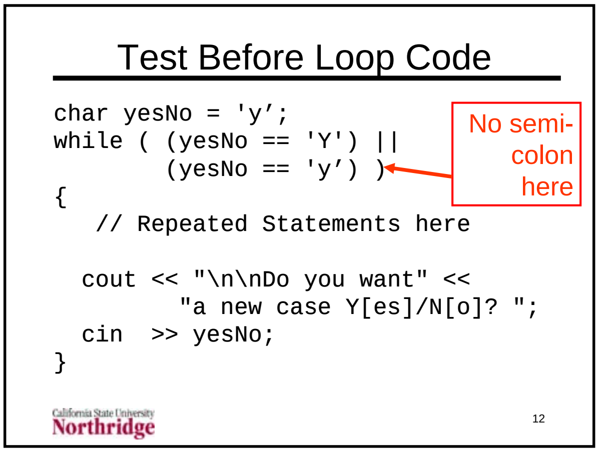
```
char yesNo;
do
{  // Repeated Statements here

   cout << "\n\nDo you want" <<
  "a new case Y[es]/N[o]? ";
   cin  >> yesNo;
}
while ( (yesNo == 'Y') ||
        (yesNo == 'y’) );
```

Choice Before Loops



Statements in loop block must change condition

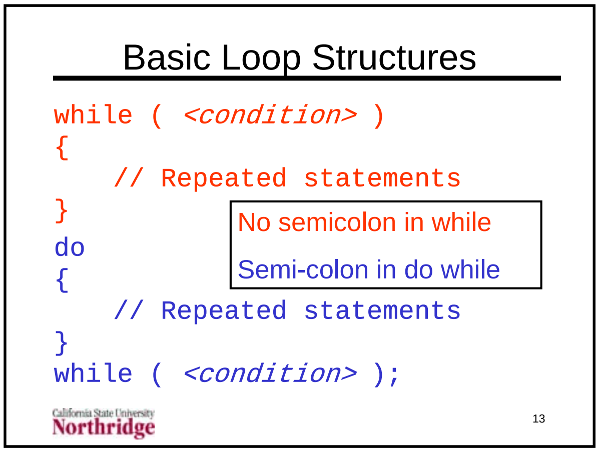


Test Before Loop Code

```
char yesNo = 'y’;
while ( (yesNo == 'Y') ||
        (yesNo == 'y’) )
{
    // Repeated Statements here

   cout << "\n\nDo you want" <<
          "a new case Y[es]/N[o]? ";
   cin  >> yesNo;
}
```

No semi-colon here

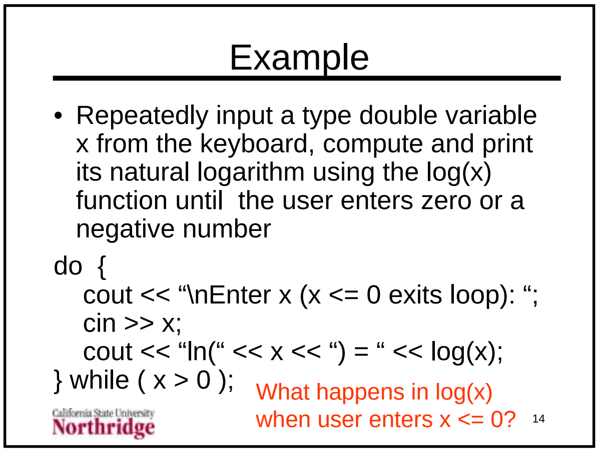


Basic Loop Structures

```
while ( <condition> )
{
    // Repeated statements
}
do
{
    // Repeated statements
}
while ( <condition> );
```

No semicolon in while

Semi-colon in do while

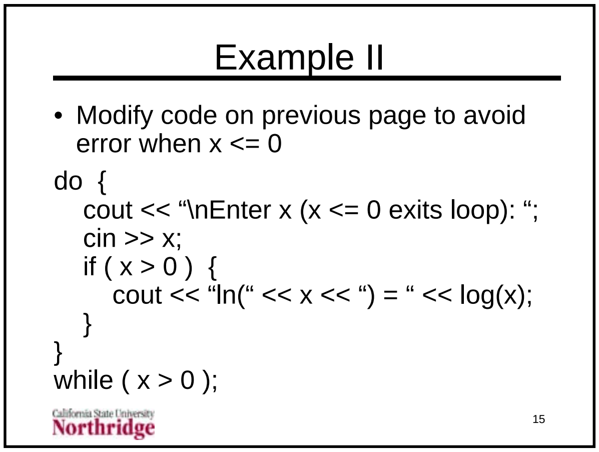


Example

- Repeatedly input a type double variable x from the keyboard, compute and print its natural logarithm using the log(x) function until the user enters zero or a negative number

```
do {
   cout << “\nEnter x (x <= 0 exits loop): “;
   cin >> x;
   cout << “ln(“ << x << “) = “ << log(x);
} while ( x > 0 );
```

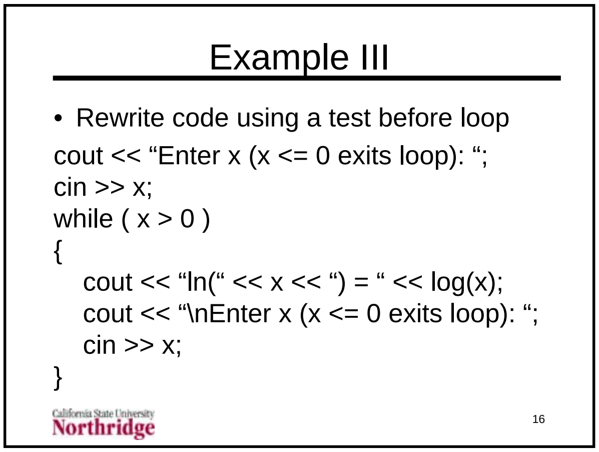
What happens in log(x) when user enters x <= 0?



Example II

- Modify code on previous page to avoid error when x <= 0

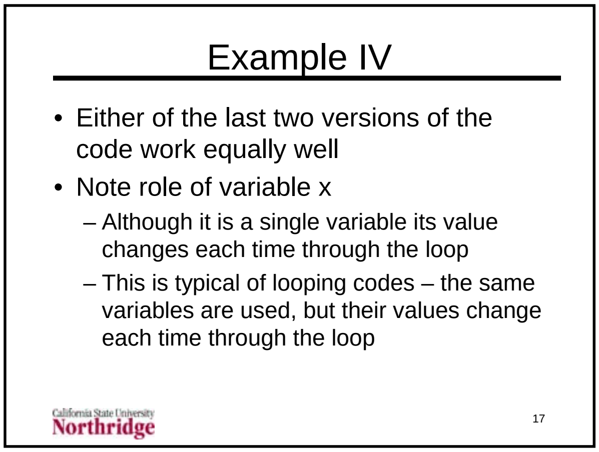
```
do {
   cout << “\nEnter x (x <= 0 exits loop): “;
   cin >> x;
   if ( x > 0 ) {
      cout << “ln(“ << x << “) = “ << log(x);
   }
}
while ( x > 0 );
```



Example III

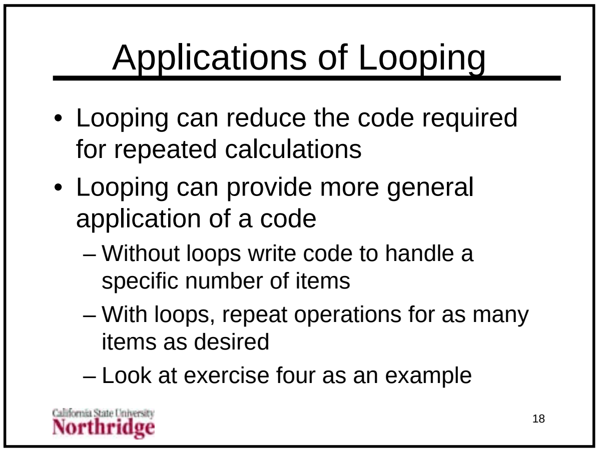
- Rewrite code using a test before loop

```
cout << “Enter x (x <= 0 exits loop): “;
cin >> x;
while ( x > 0 )
{
   cout << “ln(“ << x << “) = “ << log(x);
   cout << “\nEnter x (x <= 0 exits loop): “;
   cin >> x;
}
```



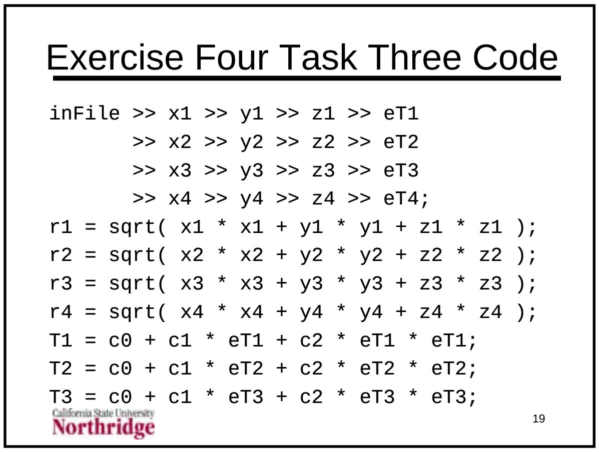
Example IV

- Either of the last two versions of the code work equally well
- Note role of variable x
  - Although it is a single variable its value changes each time through the loop
  - This is typical of looping codes – the same variables are used, but their values change each time through the loop



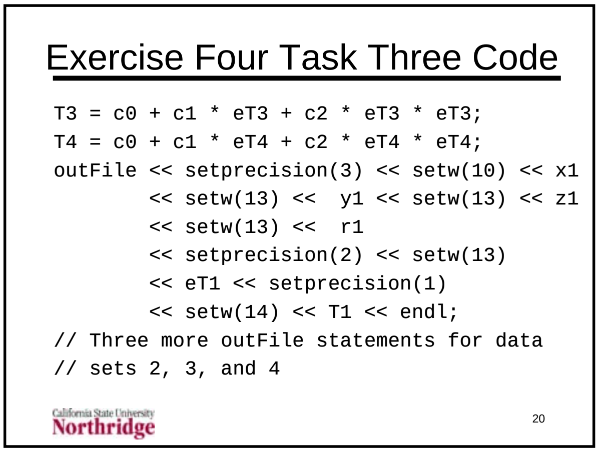
Applications of Looping

- Looping can reduce the code required for repeated calculations
- Looping can provide more general application of a code
  - Without loops write code to handle a specific number of items
  - With loops, repeat operations for as many items as desired
  - Look at exercise four as an example



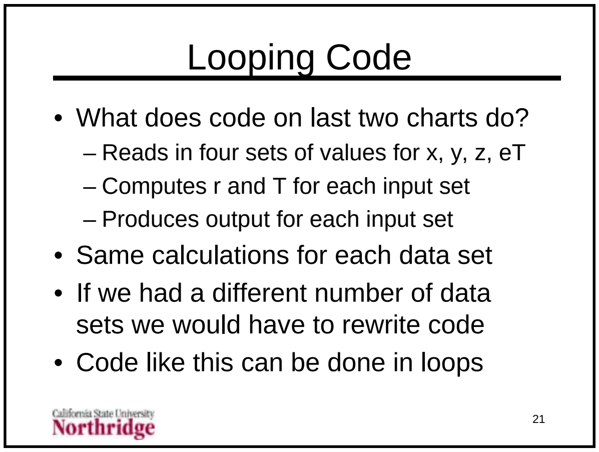
Exercise Four Task Three Code

```
inFile >> x1 >> y1 >> z1 >> eT1
       >> x2 >> y2 >> z2 >> eT2
       >> x3 >> y3 >> z3 >> eT3
       >> x4 >> y4 >> z4 >> eT4;
r1 = sqrt( x1 * x1 + y1 * y1 + z1 * z1 );
r2 = sqrt( x2 * x2 + y2 * y2 + z2 * z2 );
r3 = sqrt( x3 * x3 + y3 * y3 + z3 * z3 );
r4 = sqrt( x4 * x4 + y4 * y4 + z4 * z4 );
T1 = c0 + c1 * eT1 + c2 * eT1 * eT1;
T2 = c0 + c1 * eT2 + c2 * eT2 * eT2;
T3 = c0 + c1 * eT3 + c2 * eT3 * eT3;
```



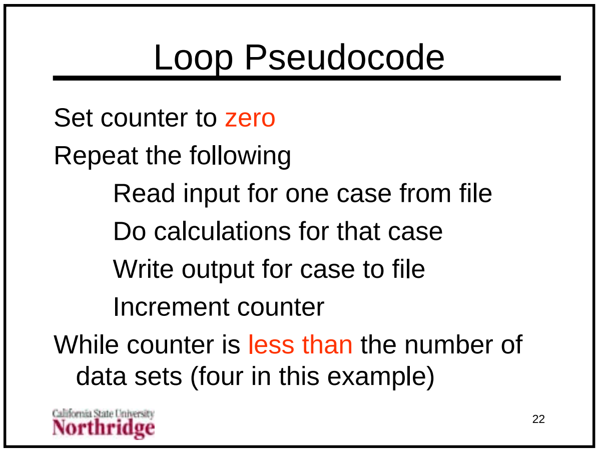
Exercise Four Task Three Code

```
T3 = c0 + c1 * eT3 + c2 * eT3 * eT3;
T4 = c0 + c1 * eT4 + c2 * eT4 * eT4;
outFile << setprecision(3) << setw(10) << x1
        << setw(13) <<  y1 << setw(13) << z1
        << setw(13) <<  r1
        << setprecision(2) << setw(13)
        << eT1 << setprecision(1)
        << setw(14) << T1 << endl;
// Three more outFile statements for data
// sets 2, 3, and 4
```



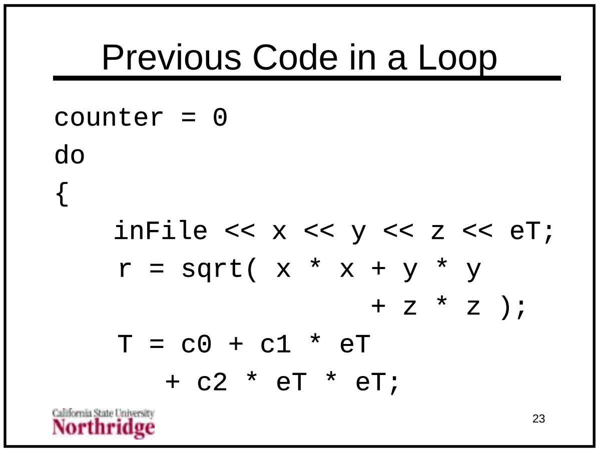
Looping Code

- What does code on last two charts do?
  - Reads in four sets of values for x, y, z, eT
  - Computes r and T for each input set
  - Produces output for each input set
- Same calculations for each data set
- If we had a different number of data sets we would have to rewrite code
- Code like this can be done in loops



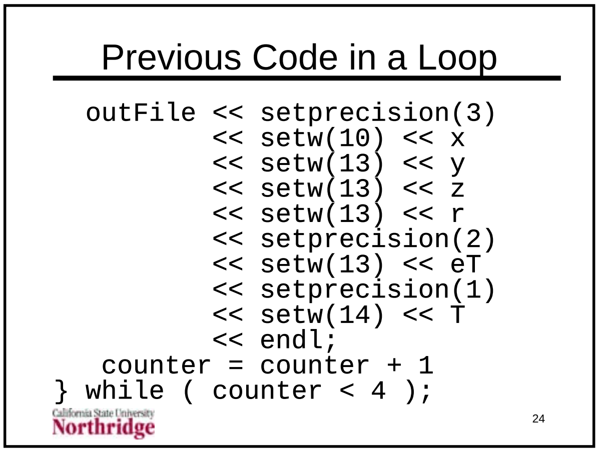
Loop Pseudocode

```
Set counter to zero
Repeat the following
     Read input for one case from file
     Do calculations for that case
     Write output for case to file
     Increment counter
While counter is less than the number of
  data sets (four in this example)
```



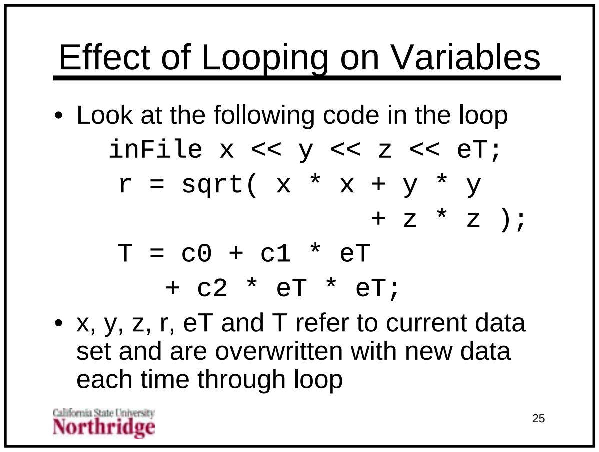
Previous Code in a Loop

```
counter = 0
do
{
   inFile << x << y << z << eT;
   r = sqrt( x * x + y * y
                + z * z );
   T = c0 + c1 * eT
      + c2 * eT * eT;
```



Previous Code in a Loop

```
  outFile << setprecision(3)
          << setw(10) << x
          << setw(13) << y
          << setw(13) << z
          << setw(13) << r
          << setprecision(2)
          << setw(13) << eT
          << setprecision(1)
          << setw(14) << T
          << endl;
   counter = counter + 1
} while ( counter < 4 );
```



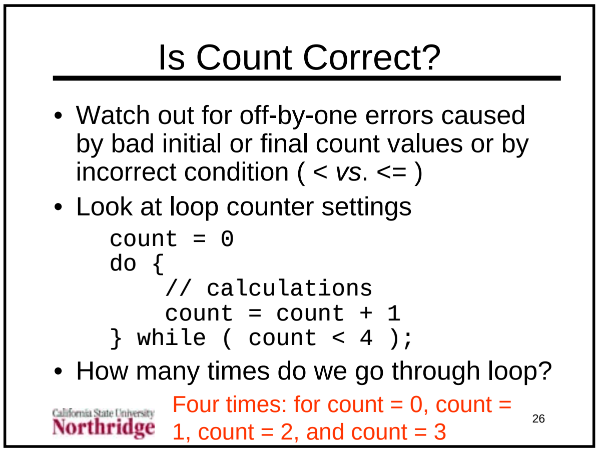
Effect of Looping on Variables

- Look at the following code in the loop

```
  inFile x << y << z << eT;
   r = sqrt( x * x + y * y
                  + z * z );
   T = c0 + c1 * eT
      + c2 * eT * eT;
```

- x, y, z, r, eT and T refer to current data set and are overwritten with new data each time through loop

25



Is Count Correct?

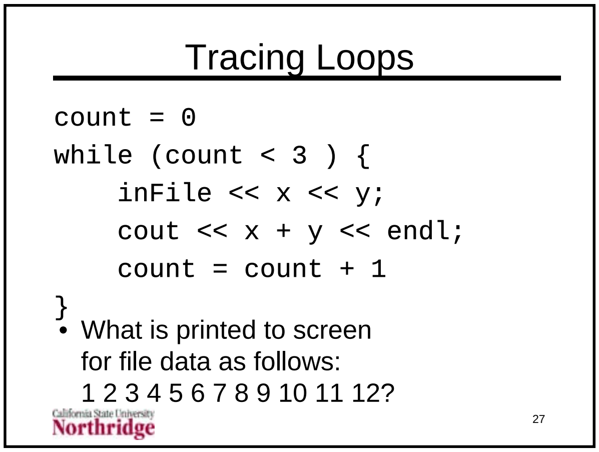
- Watch out for off-by-one errors caused by bad initial or final count values or by incorrect condition (< *vs.* <=)
- Look at loop counter settings

```
count = 0
do {
    // calculations
    count = count + 1
} while ( count < 4 );
```

- How many times do we go through loop?

Four times: for count = 0, count = 1, count = 2, and count = 3

26

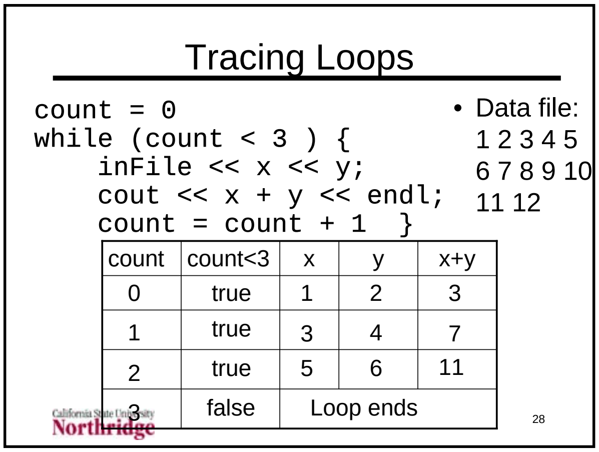


Tracing Loops

```
count = 0
while (count < 3 ) {
    inFile << x << y;
    cout << x + y << endl;
    count = count + 1
}
```

- What is printed to screen for file data as follows: 1 2 3 4 5 6 7 8 9 10 11 12?

27



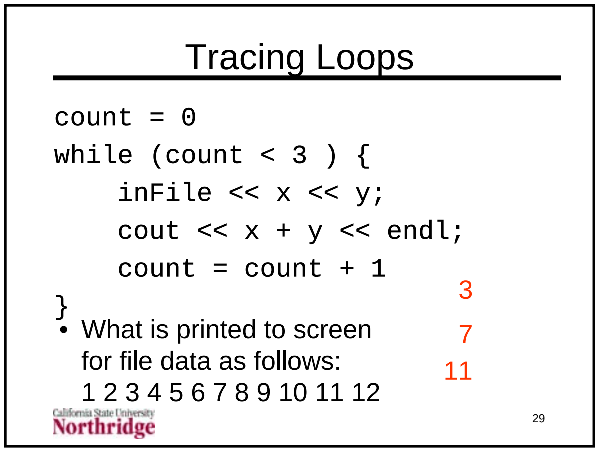
Tracing Loops

```
count = 0
while (count < 3 ) {
    inFile << x << y;
    cout << x + y << endl;
    count = count + 1   }
```

- Data file:
  1 2 3 4 5
  6 7 8 9 10
  11 12

count	count<3	x	y	x+y
0	true	1	2	3
1	true	3	4	7
2	true	5	6	11
3	false	Loop ends		

28



Tracing Loops

```
count = 0
while (count < 3 ) {
    inFile << x << y;
    cout << x + y << endl;
    count = count + 1
}
```

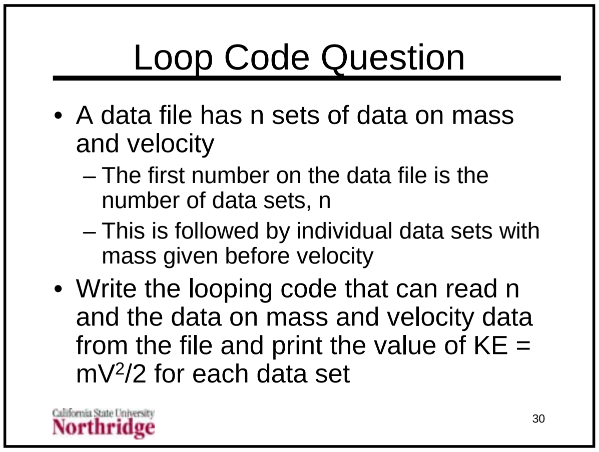
- What is printed to screen for file data as follows: 1 2 3 4 5 6 7 8 9 10 11 12

3

7

11

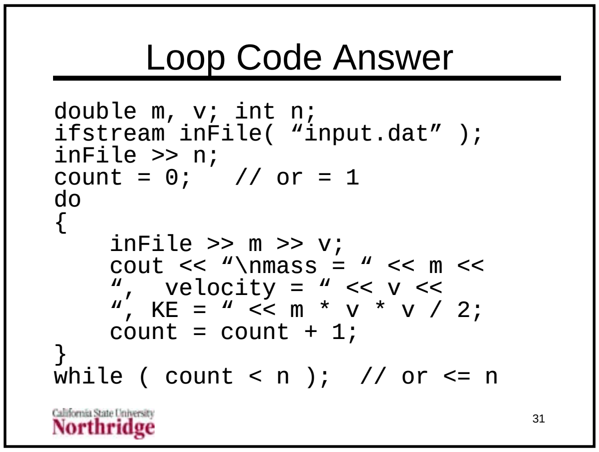
29



Loop Code Question

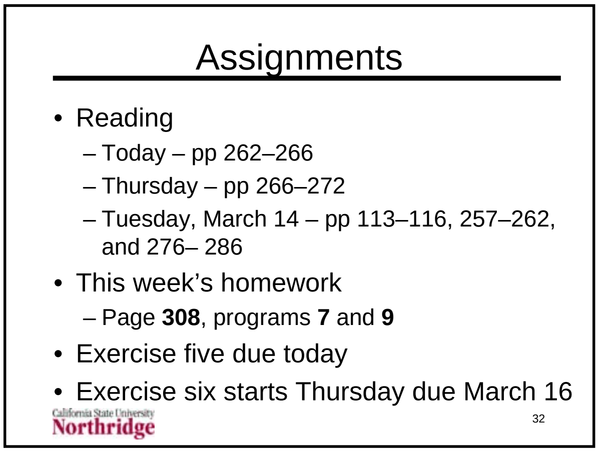
- A data file has n sets of data on mass and velocity
  - The first number on the data file is the number of data sets, n
  - This is followed by individual data sets with mass given before velocity
- Write the looping code that can read n and the data on mass and velocity data from the file and print the value of KE = $mV^2/2$ for each data set

30



Loop Code Answer

```
double m, v; int n;
ifstream inFile( “input.dat” );
inFile >> n;
count = 0;   // or = 1
do
{
    inFile >> m >> v;
    cout << “\nmass = “ << m <<
    “,  velocity = “ << v <<
    “, KE = “ << m * v * v / 2;
    count = count + 1;
}
while ( count < n );  // or <= n
```



Assignments

- Reading
  - – Today – pp 262–266
  - – Thursday – pp 266–272
  - – Tuesday, March 14 – pp 113–116, 257–262, and 276– 286
- This week’s homework
  - – Page **308**, programs **7** and **9**
- Exercise five due today
- Exercise six starts Thursday due March 16