Nested Control Structures

Nested Selection structures && Nested Repetition structures

Jessica Sizemore SIP 2015 , 17 JUN 2015

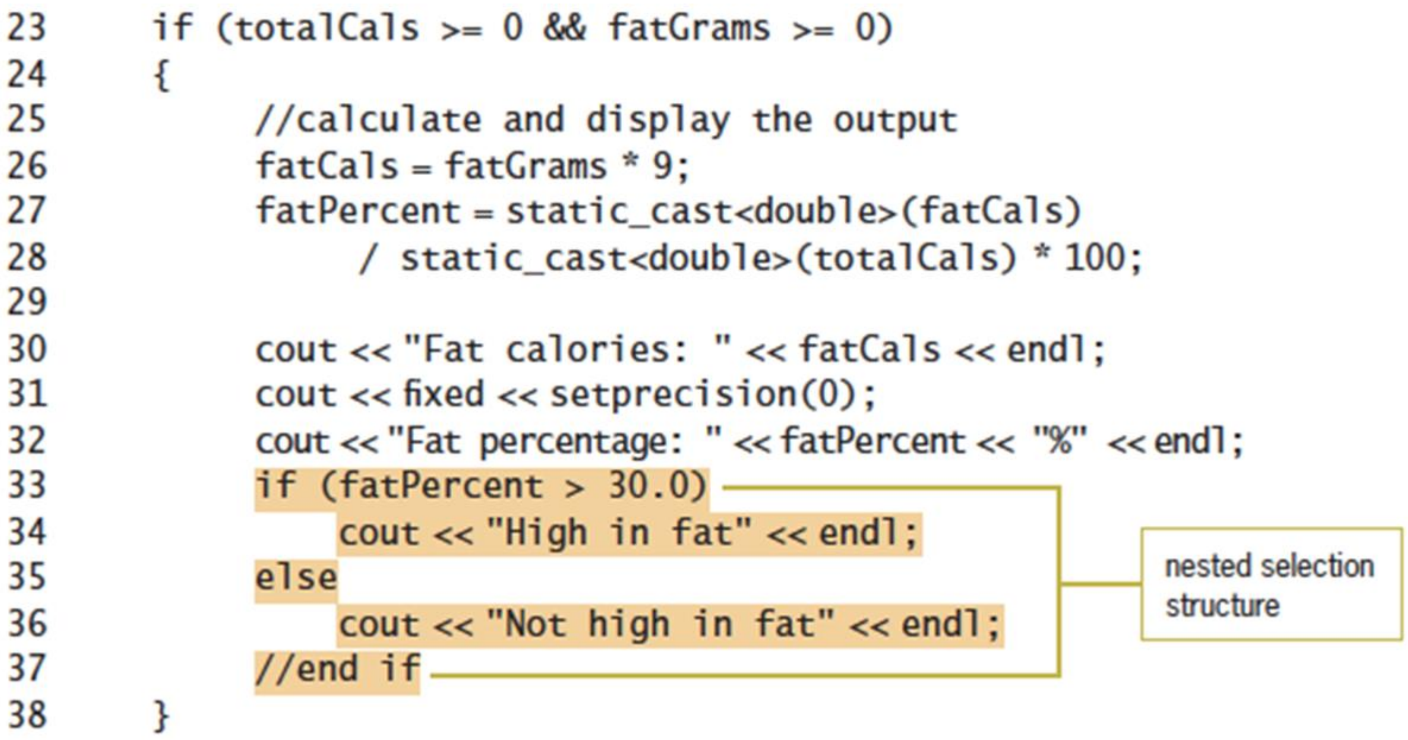
Nested Selection Structures

Nested Selection Structures

- Contained within an outer selection structure
- Nested selection structures are used when more than one decision needs to be made before choosing an instruction
- Inner selection structures are indented within their outer selection structures
- Outer and inner selection structures can be thought of as making primary and secondary decisions, respectively

Coding nested selection structures

- Nested selection structures uses the `if` and `else` statements
- Can be placed in either `if` or `else` statement blocks



```
    if (totalCals >= 0 && fatGrams >= 0)
    {
        //calculate and display the output
        fatCals = fatGrams * 9;
        fatPercent = static_cast<double>(fatCals)
            / static_cast<double>(totalCals) * 100;

        cout << "Fat calories:  " << fatCals << endl;
        cout << fixed << setprecision(0);
        cout << "Fat percentage:  " << fatPercent << "%" << endl;
        if (fatPercent > 30.0)
            cout << "High in fat" << endl;
        else
            cout << "Not high in fat" << endl;
        //end if
    }
```

Figure 6-7 Modified program for the health club problem from Chapter 5's Lab 5-2 (cont'd.)

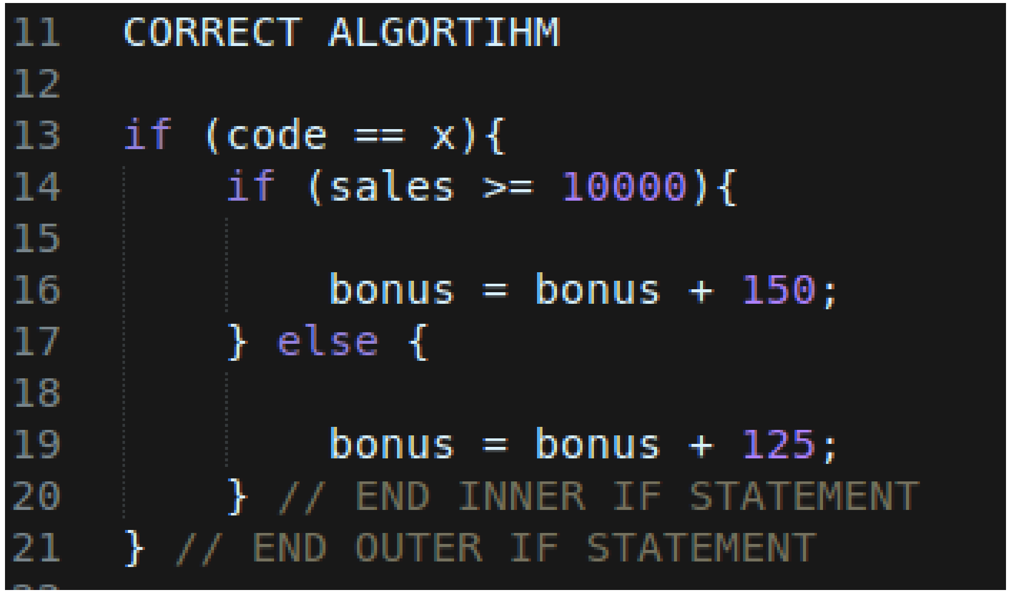
Logic Errors in Selection Structures

- *Three common logic errors made when writing selection structures*
  1. Using a compound condition rather than a nested selection structure
  2. Reversing the outer and nested decisions
  3. Using an unnecessary nested selection structure

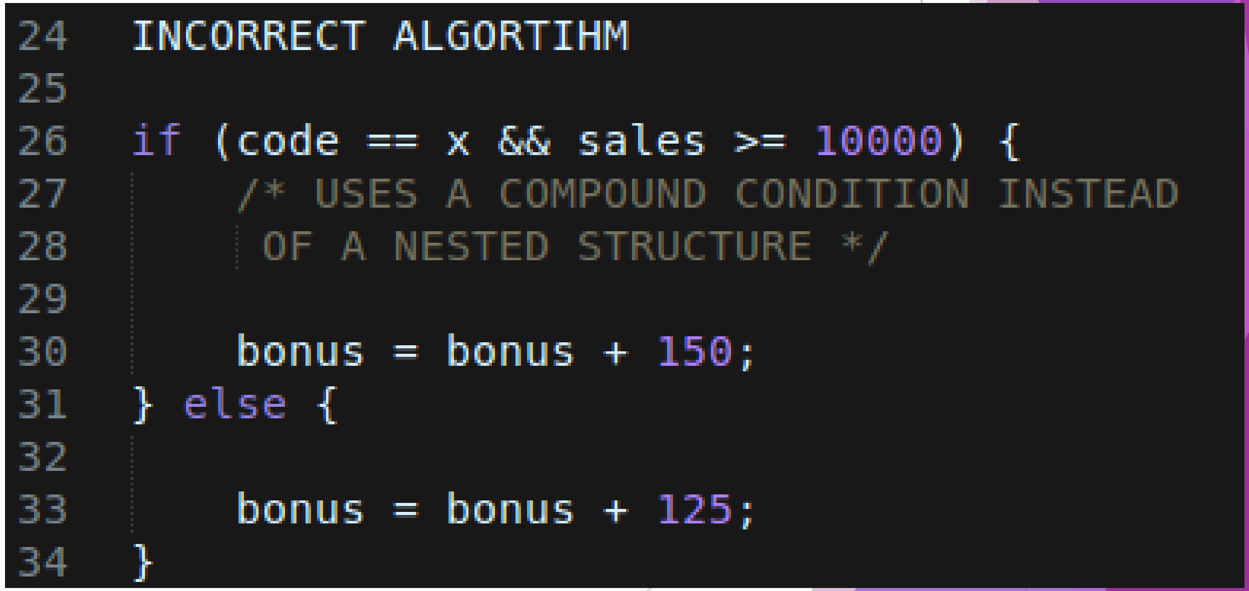
First Logic Error

- Using a compound condition rather than a nested selection structure
- Ignores the hierarchy between two sub-conditions

  - One applies only if the other is a certain value



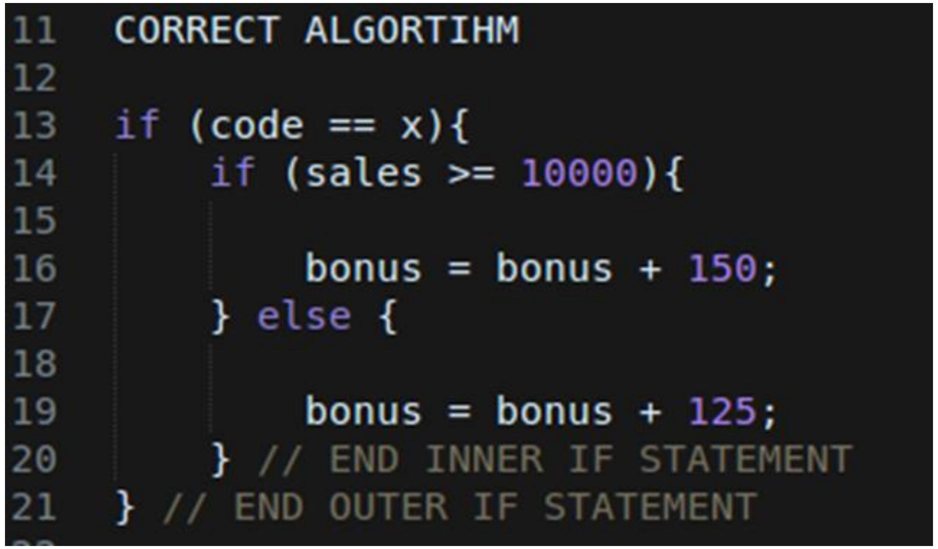
```
CORRECT ALGORTIHM

if (code == x){
    if (sales >= 10000){

        bonus = bonus + 150;
    } else {

        bonus = bonus + 125;
    } // END INNER IF STATEMENT
} // END OUTER IF STATEMENT
```



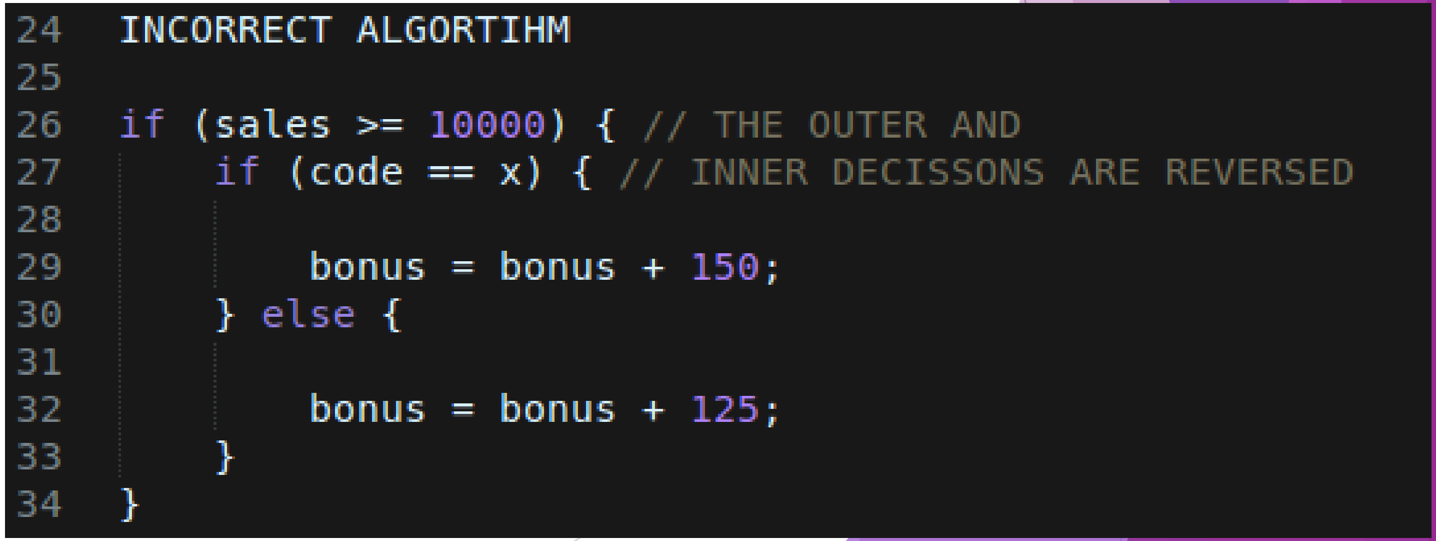
```
INCORRECT ALGORTIHM

if (code == x && sales >= 10000) {
    /* USES A COMPOUND CONDITION INSTEAD
     OF A NESTED STRUCTURE */

    bonus = bonus + 150;
} else {

    bonus = bonus + 125;
}
```

Second Logic Error

- Reversing the outer and nested selection structures



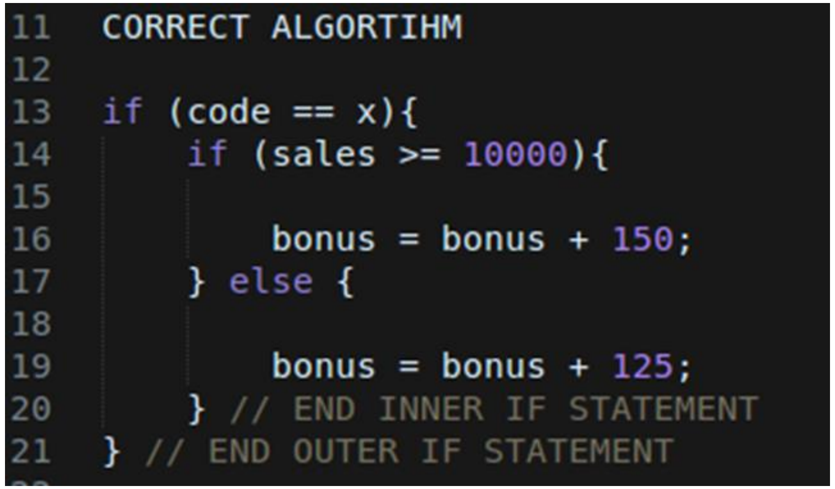
```
CORRECT ALGORTIHM

if (code == x){
    if (sales >= 10000){

        bonus = bonus + 150;
    } else {

        bonus = bonus + 125;
    } // END INNER IF STATEMENT
} // END OUTER IF STATEMENT
```



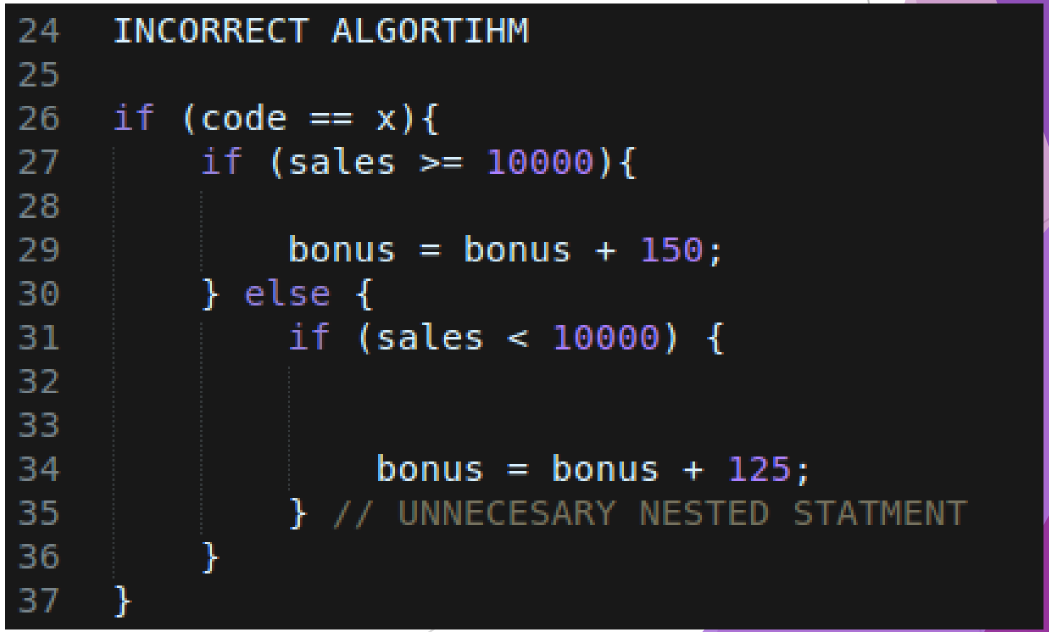
```
INCORRECT ALGORTIHM

if (sales >= 10000) { // THE OUTER AND
    if (code == x) { // INNER DECISSONS ARE REVERSED

        bonus = bonus + 150;
    } else {

        bonus = bonus + 125;
    }
}
```

Third Logic Error

- Using an unnecessary nested selection structure
- Often will produce the correct result, but will be inefficient



```
CORRECT ALGORTIHM

if (code == x){
    if (sales >= 10000){

        bonus = bonus + 150;
    } else {

        bonus = bonus + 125;
    } // END INNER IF STATEMENT
} // END OUTER IF STATEMENT
```



```
INCORRECT ALGORTIHM

if (code == x){
    if (sales >= 10000){

        bonus = bonus + 150;
    } else {
        if (sales < 10000) {


            bonus = bonus + 125;
        } // UNNECESARY NESTED STATMENT
    }
}
```

Multiple-Alternative Selection Structures

- Sometimes problems require a selection structure that chooses between several alternatives
- Called *multiple-alternative selection structures* or extended selection structures

Grading Calculator

Processing

Algorithm:

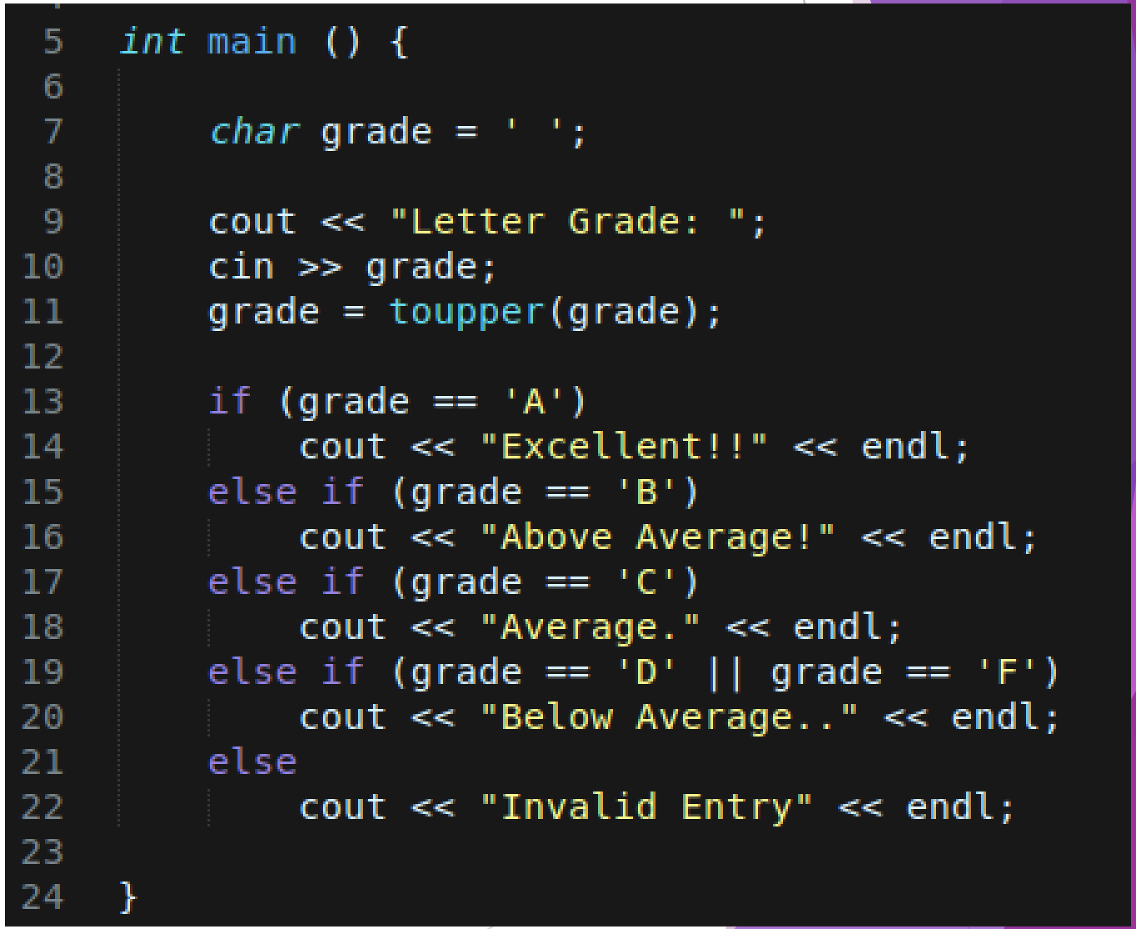
1. Enter Grade
2. If (the grade is one of the following:)

A	Display “Excellent”
B	Display “Above Average”
C	Display “Average”
D of F	Display “Below Average”

3. Else

Display “Invalid Grade”

Figure 6-21 IPO chart for the Kindlon High School problem



```
int main () {

    char grade = ' ';

    cout << "Letter Grade: ";
    cin >> grade;
    grade = toupper(grade);

    if (grade == 'A')
        cout << "Excellent!!" << endl;
    else if (grade == 'B')
        cout << "Above Average!" << endl;
    else if (grade == 'C')
        cout << "Average." << endl;
    else if (grade == 'D' || grade == 'F')
        cout << "Below Average.." << endl;
    else
        cout << "Invalid Entry" << endl;

}
```

The `Switch` statement

- The **`switch` statement** to code a multiple-alternative selection structure
- Begins with `switch` keyword followed by a selector expression in parentheses
- Selector expression can be anything
- Must result in a data type that is `bool`, `char`, or `int`
- After selector expression, there are one or more `case` clauses inside brackets

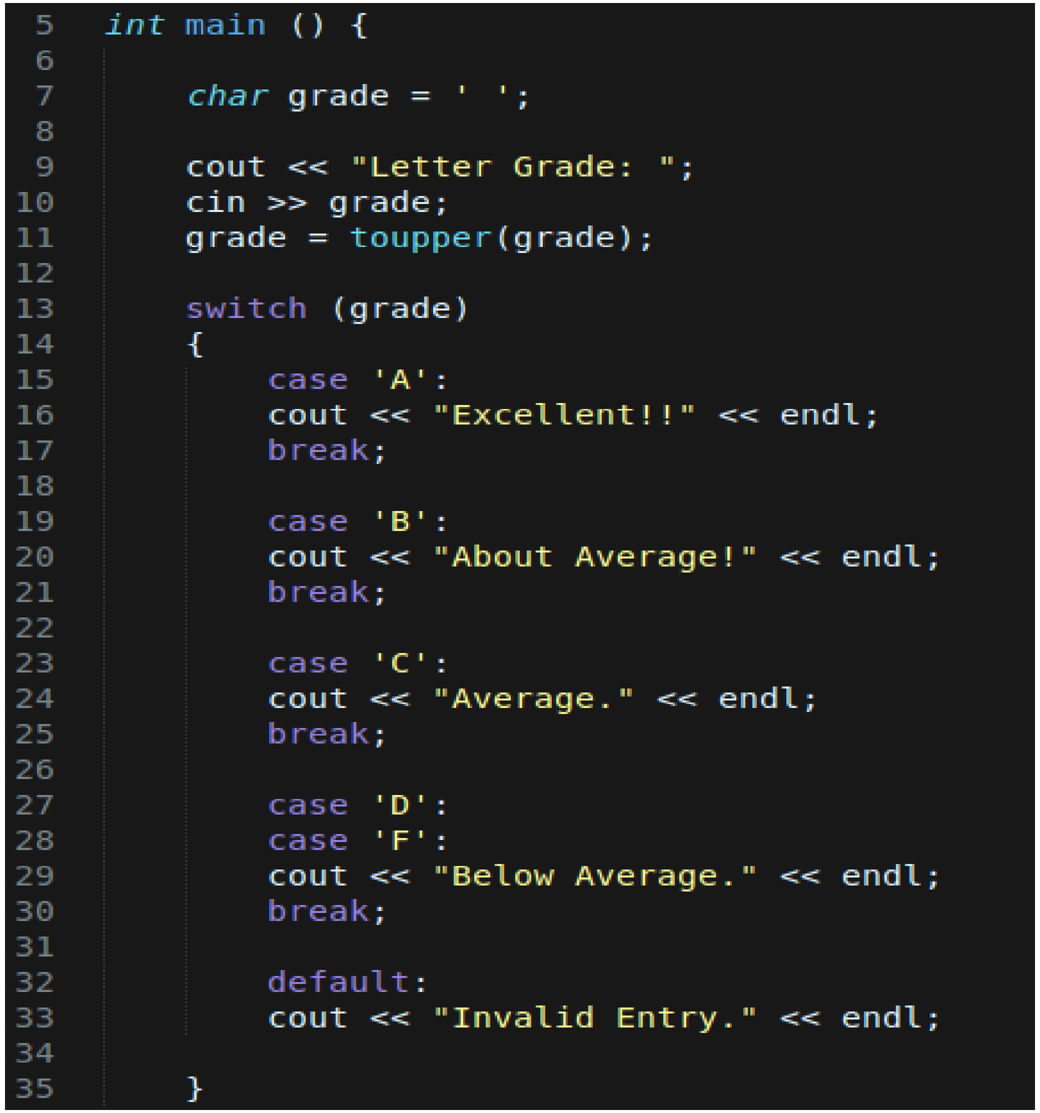
The `switch` Statement

- Each `case` clause represents a different alternative
- Each case clause contains one or more statements processed when selector expression matches that `case`'s value

The `switch` Statement

- **`break` statement** tells computer to break out of `switch` at that point; must be the last statement of a case clause
- Without a break statement, computer continues to process instructions in later case clauses
- Can also include one `default` clause; processed if selector expression does not match any values in `case` clauses
- `default` clause can appear anywhere, but usually entered as last clause
  - If it is the last clause, a `break` statement is not needed at its end
  - Otherwise, a `break` statement is needed to prevent computer from processing later case clauses

The `switch` Statement



```
int main () {

    char grade = ' ';

    cout << "Letter Grade: ";
    cin >> grade;
    grade = toupper(grade);

    switch (grade)
    {
        case 'A':
        cout << "Excellent!!" << endl;
        break;

        case 'B':
        cout << "About Average!" << endl;
        break;

        case 'C':
        cout << "Average." << endl;
        break;

        case 'D':
        case 'F':
        cout << "Below Average." << endl;
        break;

        default:
        cout << "Invalid Entry." << endl;

    }
```

- This is the same Grading program that we wrote with `else if` statements.

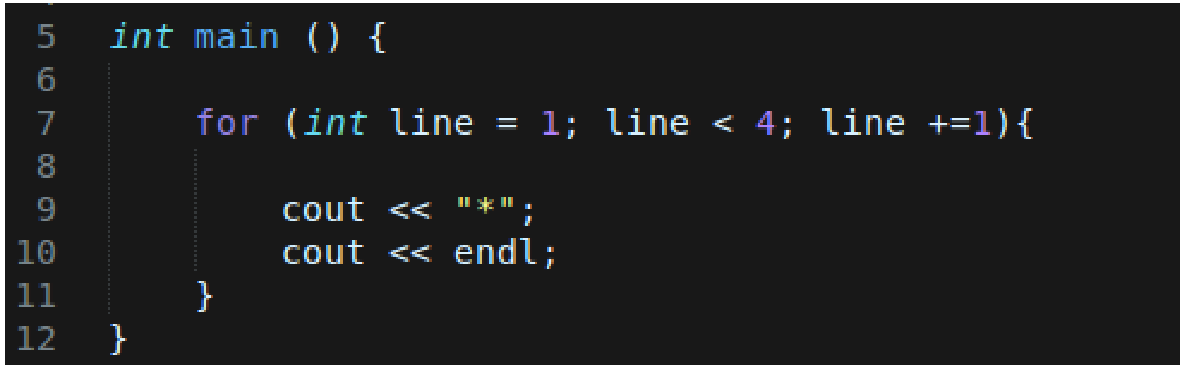
Nested Repetition Structures

Nested Repetition Structures

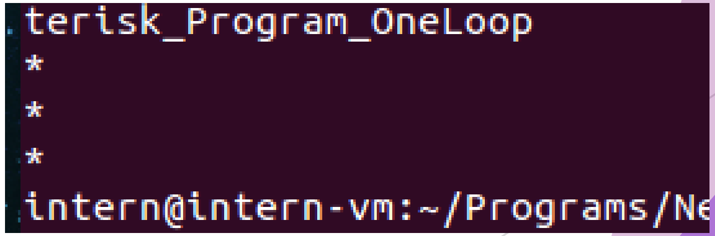
- You can place one loop (inner loop) inside another loop (the outer loop)
- Both loops can be pretest loops or posttest loops, or different types

The Asterisks Program

Simple program that prints asterisks to the screen (Without Nested Loops):



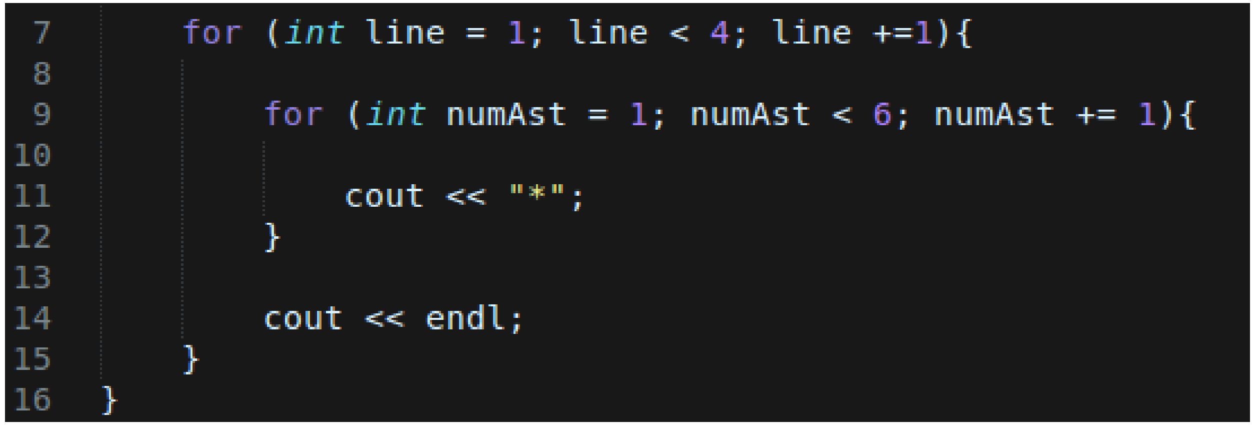
```
int main () {

    for (int line = 1; line < 4; line +=1){

        cout << "*";
        cout << endl;
    }
}
```



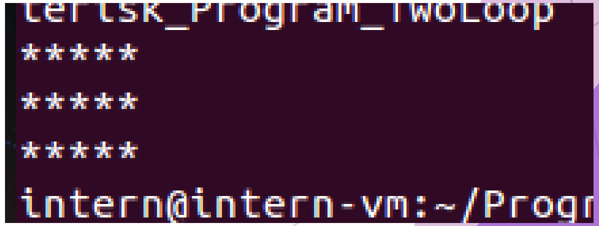
```
terisk_Program_OneLoop
*
*
*
intern@intern-vm:~/Programs/Ne
```

The Asterisks Program

Simple program that prints asterisks to the screen (With Nested Loop):



```
    for (int line = 1; line < 4; line +=1){

        for (int numAst = 1; numAst < 6; numAst += 1){

            cout << "*";
        }

        cout << endl;
    }
}
```



```
terisk_Program_TwoLoop
*****
*****
*****
intern@intern-vm:~/Progr
```

The `pow` Function

- convenient tool to raise a number to a power (**exponentiation)**
- The `pow` function raises a number to a power and returns the result as a `double` number
- *Syntax:*

  `pow (x,y);`   Where `x` is the base and `y` is the exponent.

- At least one of the two arguments must be a `double`
- Program must contain the `#include <cmath>` directive to use the `pow` function

Work Cited

- Diane Zax, “An Introduction to Programming with C++, Sixth Edition”,
  - Chapter 6 – More on the Selection Structure.
  - Chapter 8 – More on the Repetition Structure.
- Towson University, Professor Robert Eyer, COSC 175,
  - Chapter 6 Lecture Slides.
  - Chapter 8 Lecture Slides.