Selections

Kornchawal Chaipah, PhD

Computer Engineering Department,
Khon Kaen University

Agenda

- Selection and conditions
- Single and compound conditions
- Statement blocks
- Nested conditions

Selection and Condition

- Select what to do based on a condition
  - If (condition is true), then do A, B, …
  - If (condition is not true), then don't do A, B, …
    - May have to do something else instead
- Condition
  - Must have a value of true or false
  - By numerical value
    - True: a non-zero value, e.g. -1, -5.678, 0.01, 3, 56
    - False: 0
  - By comparison: <, ≤, >, ≥, ==, !=
  - Single condition vs. compound condition

Value Comparison

x < y	x is less than y
x <= y	x is less than or equal to y
x > y	x is greater than y
x >= y	x is greater than or equal to y
x == y	x is equal to y
x != y	x is not equal to y

- Numerical value: true = 1, false = 0
- Be careful!
  - Value assignment: =
  - Equality operator: ==

Also notice that we use

>= not ≥

<= not ≤

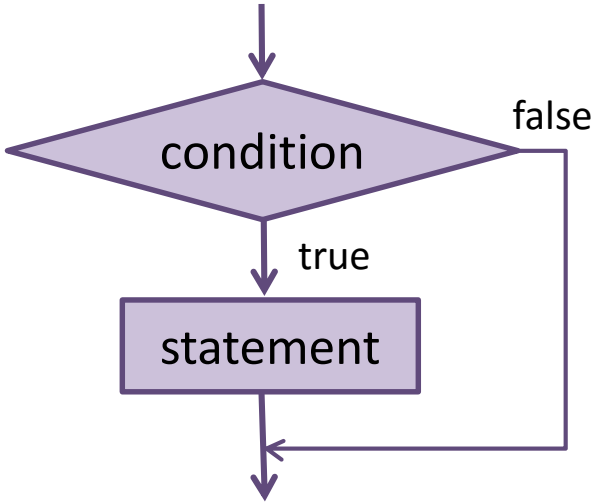
!= not ≠

Selection Statements in C++

- In C++, we can use
  - if ..
  - if .. else ..
  - if .. else if .. else ..
  - switch
- Let's see one by one

The if Statement

```
if(condition){
   statement;
}
```



- If condition is true (non-zero), then execute the statement
- If not, just do nothing and skip

if Example

- Problem: read 2 integers n and d, then show output if n is not divisible by d

Code:

```
int main(){
   int n, d;
   cout << "Enter 2 positive integers: ";
   cin >> n >> d;
   if (n % d) {
        cout << n << " is not divisible
        by " << d << endl;
   }
   return 0;
}
```

- condition: if n%d == not-zero (true), n is not divisible by d.
- Can also use if(n%d != 0)

Output:

Statement Block

```
if(condition)
    statement1;
```

```
if(condition)
{
    statement2;
    statement3;
    statement4;
}
```

- A sequence of statements enclosed by { }
  - Act as if all of statements are a single statement
  - Can be put anyway a single statement can
  - E.g. after if(condition)

What's Different?

Code1:

```
if(condition)
{
    statement2;
    statement3;
    statement4;
}
```

Code2:

```
if(condition)
    statement2;
    statement3;
    statement4;
```

- Code1:
  - If condition = true, statement2-4 will be executed
  - If false, NONE will be executed
- Code2:
  - If condition = true, statement2-4 will be executed
  - If false, statement3-4 will be executed!!! – Why?

Variable Scope

- A scope of a name
  - Begins where the name is declared
  - Ends at the end of the innermost block in which it is declared
- A program can have the same variable name *if* their scopes are nested or disjoint
  - But is it a good practice? Probably not.

Statement Block Example

Code:

```
int main(){
   int x, y;
   cout << "Enter 2 integers: ";
   cin >> x >> y;
   if(x > y) {
        int temp = x;
        x = y;
        y = temp;
   }
   cout << x << " <= " << y <<
   endl;
   return 0;
}
```

Temp is only valid within this block!

What does this do?

Output:

Compound Conditions

- Checking > 1 condition to select what to do
- Need logical operator(s)

p && q	AND: true if both p and q are true
p || q	OR: false if both p and q are false
!p	NOT: negate p

p	q	p && q
0	0	0
0	1	0
1	0	0
1	1	1

p	q	p || q
0	0	0
0	1	1
1	0	1
1	1	1

p	!p
0	1
1	0

Compound Condition Example

- Problem: check validity of inputs
  - We need both inputs to be positive

Remember, -5 0 are what we put into a keyboard (input)?

Code:

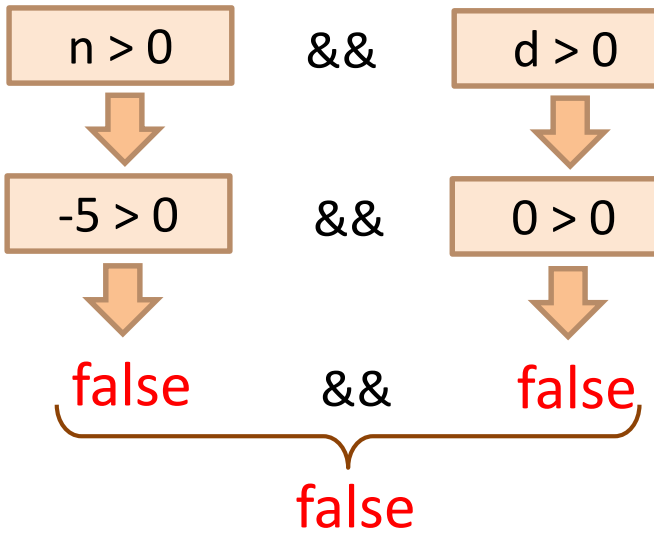
```
int main(){
   int n, d;
   cout << "Enter two positive
   integers: ";
   cin >> n >> d;
   if(n > 0 && d > 0){
        cout << "inputs ok";
   }
   return 0;
}
```

Compound condition:
1) n > 0
2) d > 0

Output:

```
Enter two positive integers: -5 0
```

n d



Compound Condition Example

- Problem: check validity of inputs
  - We need both inputs to be positive

4 -3 are what we put into a keyboard (input)

Code:

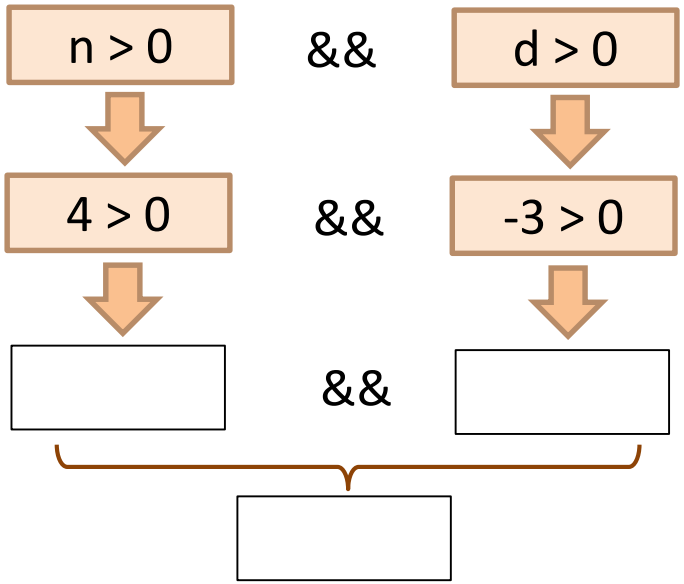
```
int main(){
   int n, d;
   cout << "Enter two positive
   integers: ";
   cin >> n >> d;
   if(n > 0 && d > 0){
        cout << "inputs ok";
   }
   return 0;
}
```

Compound condition:
1) n > 0
2) d > 0

Output:

```
Enter two positive integers: 4 -3
```

n d



Compound Condition Example

- Problem: check validity of inputs
  - We need both inputs to be positive

Code:

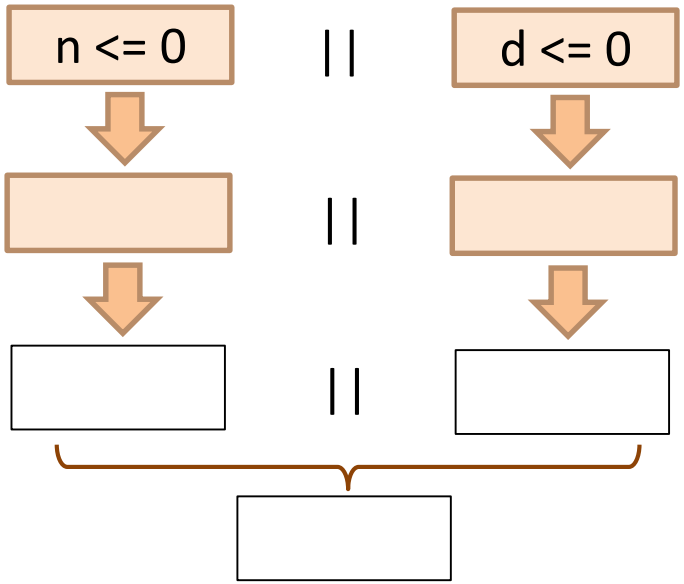
```
int main(){
   int n, d;
   cout << "Enter two positive
   integers: ";
   cin >> n >> d;
   if(n <= 0 || d <= 0){
        return 0;
   }
   cout << "inputs ok";
   return 0;
}
```

Compound condition:
1) n <= 0
2) d <= 0

Output:

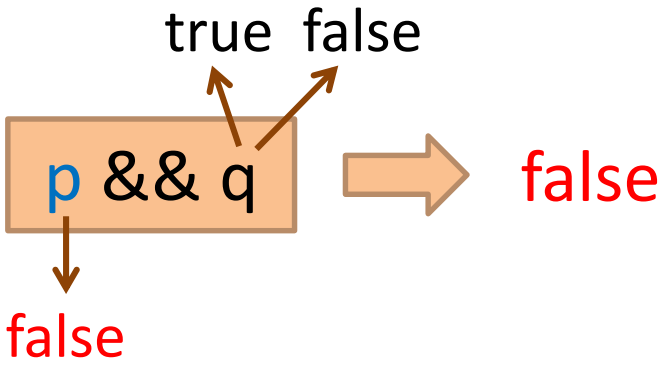
```
Enter two positive integers: 4 -3
```

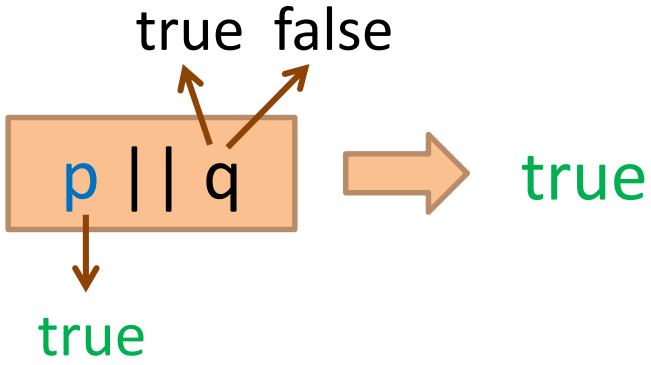
n d



Short Circuiting

- May not need to evaluate the second operand of a compound condition
  - p && q : if p is false, the condition is false regardless of q
  - p || q : if p is true, the condition is true regardless of q





Short Circuiting Example: &&

Code:

```
int main(){
    int n, d;
    cout << "Enter two positive
    integers: ";
    cin >> n >> d;
    if(d != 0 && n%d == 0){
        cout << d << " divides " <<
        n << endl;
    }
    else{
        cout << d << " does not
        divide " << n << endl;
    }
    return 0;
}
```

Compound condition:
1) d!= 0
2) n%d == 0

Remember, 2 0 are what we put into a keyboard (input)?

Output:

```
Enter two positive integers: 2 0
```

How is this a short circuit?

Short Circuiting Example: ||

Code:

```
int main(){
    int n, d;
    cout << "Enter two positive
    integers: ";
    cin >> n >> d;
    if(n < 0 || d < 0){
        return 0;
    }
    if(d != 0 && n%d == 0){
        cout << d << " divides " <<
        n << endl;
    }
    else{
        cout << d << " does not
        divide " << n << endl;
    }
    return 0;
}
```

Compound condition:
1) n < 0
2) d < 0

Remember, -5 0 are what we put into a keyboard (input)?

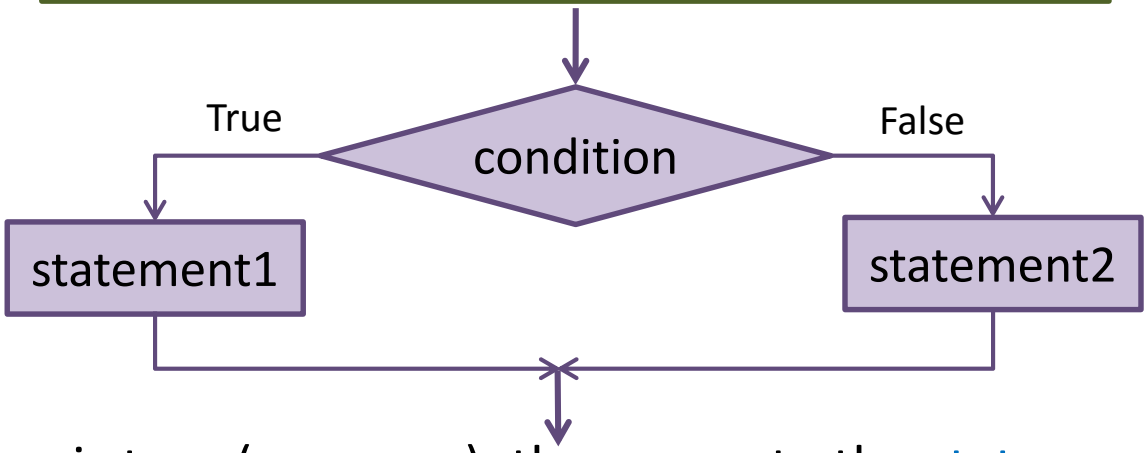
Output:

```
Enter two positive integers: -5 0
```

How is this a short circuit?

The if .. else Statement

```
if(condition){
    statement1;
}
else{
    statement2;
}
```



- If condition is true (non-zero), then execute the statement1
- If not, execute statement2
  - What's different between if and if .. else?

if .. else Example

- Problem: read 2 integers n and d, then show output if n is not divisible by d

Code:

```
int main(){
    int n, d;
    cout << "Enter 2 positive integers: ";
    cin >> n >> d;
    if (n % d) {
          cout << n << " is not divisible by " << d
           << endl;
    }
    else{
          cout << n << " is divisible by " << d <<
           endl;
    }
    return 0; }
```

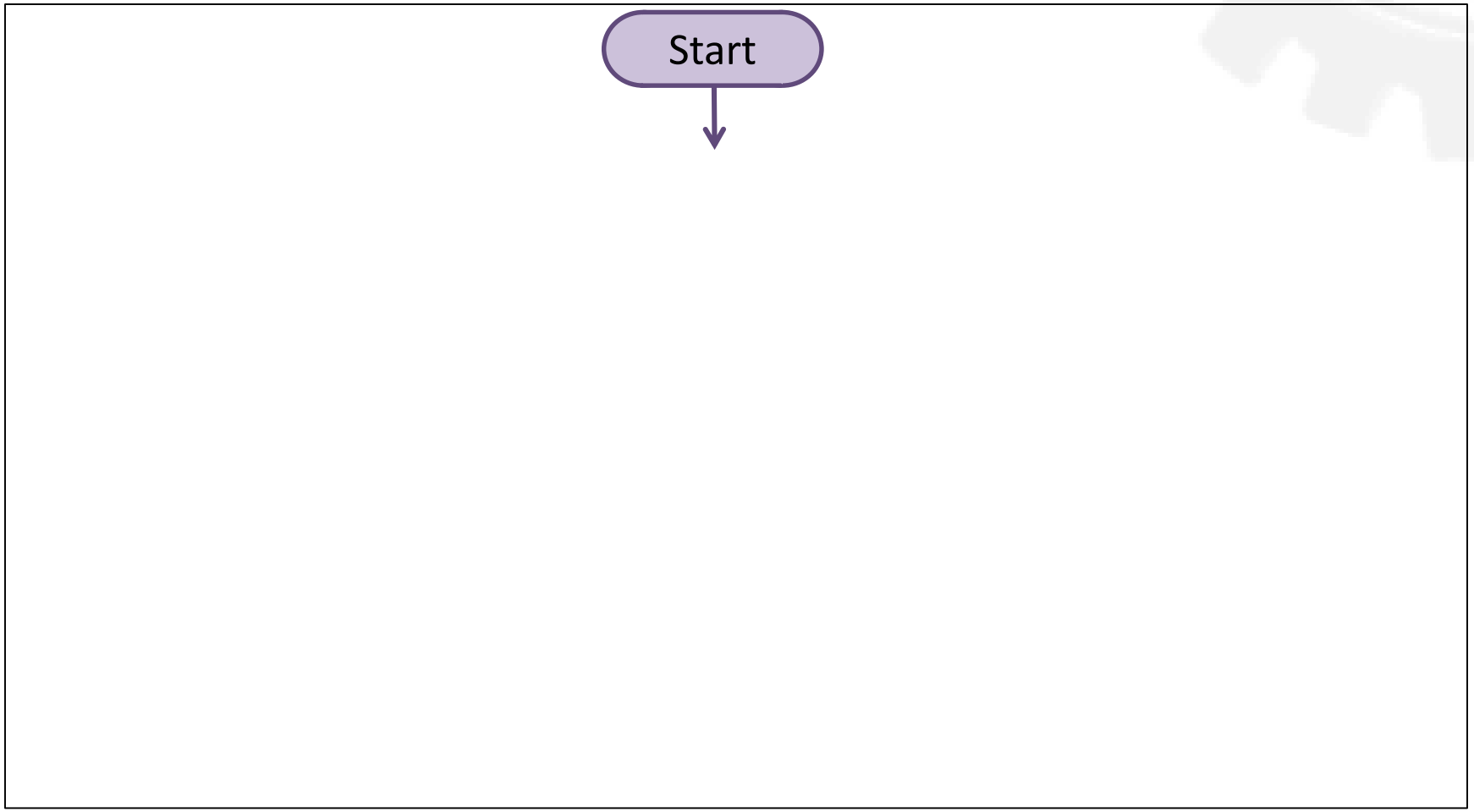
- Condition by numerical value (0 to d-1)
- What about condition by comparison?

Output:

Example: min(a,b)

- Problem: find a minimum of two integers
- What you know:
  - How do you know which one is smaller/larger?
- What are inputs, outputs, processes, and/or decision points?
- Let's write a flowchart first

min(a,b) Flowchart



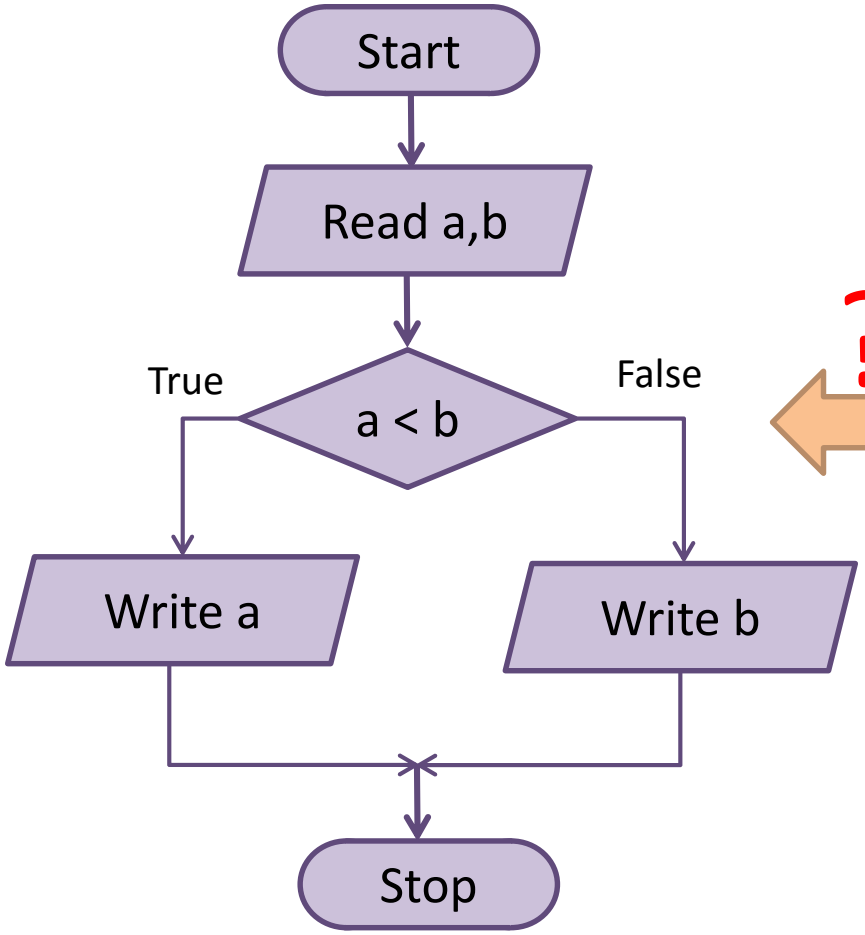
min(a,b) Code and Output

Code:

```
int main(){
   int a, b;
   cout << "Enter 2 positive integers: ";
   cin >> a >> b;
   if(          ){
        cout <<      << " is the minimum." << endl;
   }
   else{
        cout <<      << " is the minimum." << endl;
   }
   return 0;
}
```

Output:

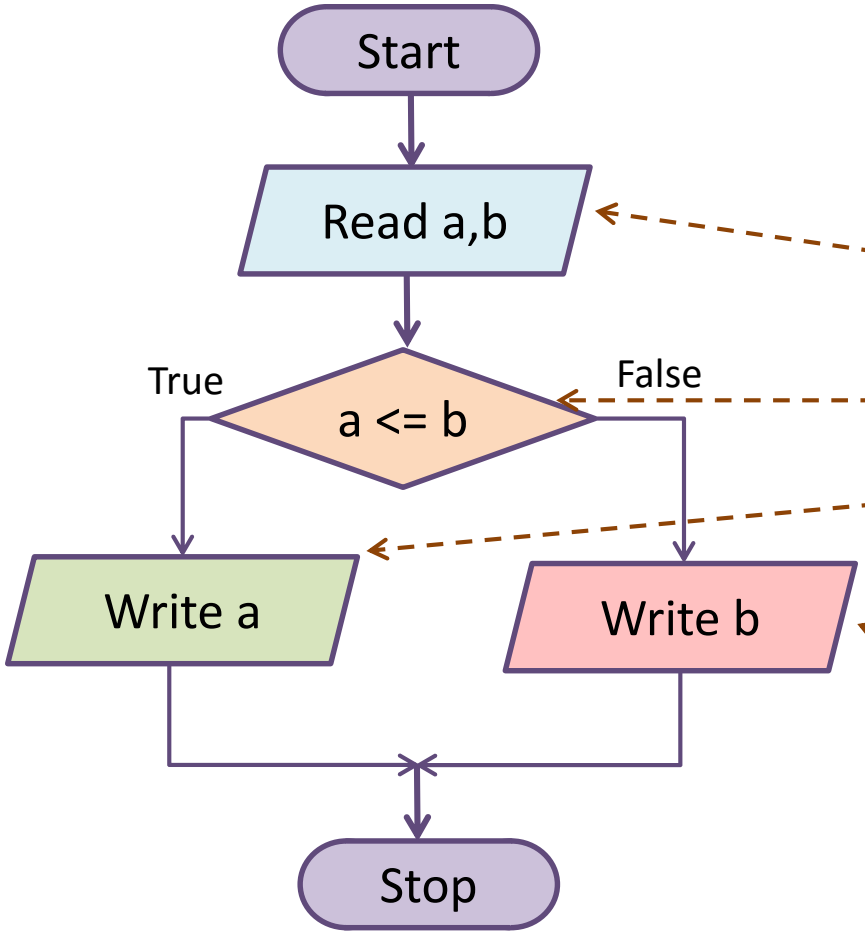
Flowchart VS. Code



?

```
int main(){
   int a, b;
   cout << "Enter 2 positive
   integers: ";
   cin >> a >> b;
   if(a < b){
        cout << a << " is the
   minimum." << endl;
   }
   else{
        cout << b << " is the
   minimum." << endl;
   }
   return 0;
}
```

Flowchart VS. Code




```
int main(){
    int a, b;
    cout << "Enter 2 positive
    integers: ";
    cin >> a >> b;
    if(a <= b){
        cout << a << " is the
    minimum." << endl;
    }
    else{
        cout << b << " is the
    minimum." << endl;
    }
    return 0;
}
```

Example: IsEnrolled?

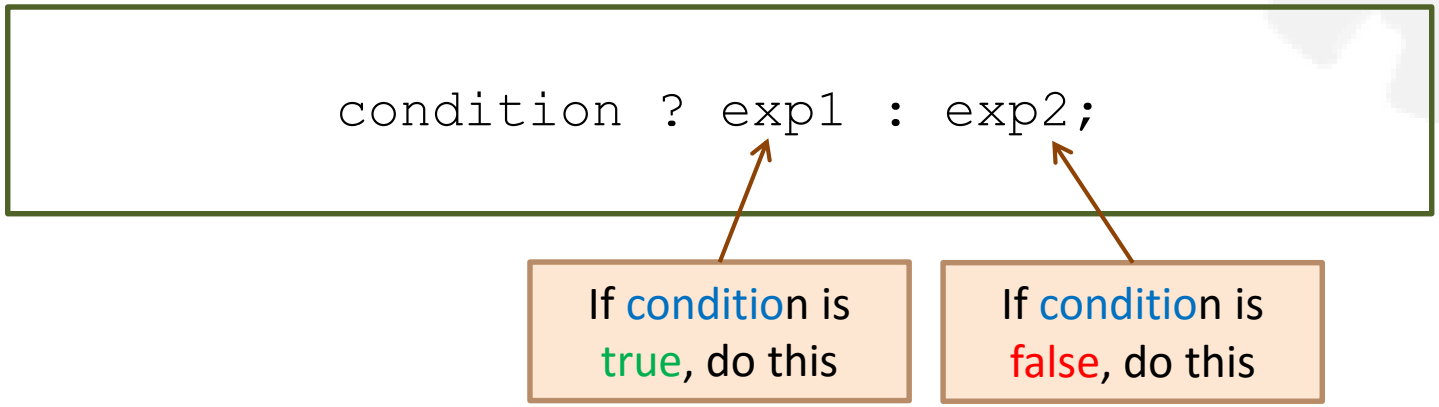
- Problem: ask a user whether s/he is enrolled
  - if y or Y, print enrolled
  - if n or N, print not enrolled

Code:

```
int main(){
   char ans;
   cout << "Are you enrolled? (y/n) ";
   return 0; }
```

Output:

The Conditional Expression Operator



- Shorter way to represent a simple if..else statement
- E.g.
  - cout << (x>y) ? 100 : 200;
  - a = (x>y) ? 100 : 200;
- Can be nested too
  - (x>y) ? 100 : ((a>b) ? 200 : 300);

min(a,b) Revisited

Code#1:

```
int main(){
   int a, b;
   cout << "Enter 2 integers: ";
   cin >> a >> b;
   cout << (a < b ? a : b) <<
   " is the minimum." << endl;
   return 0; }
```

Output#1:

Output#2:

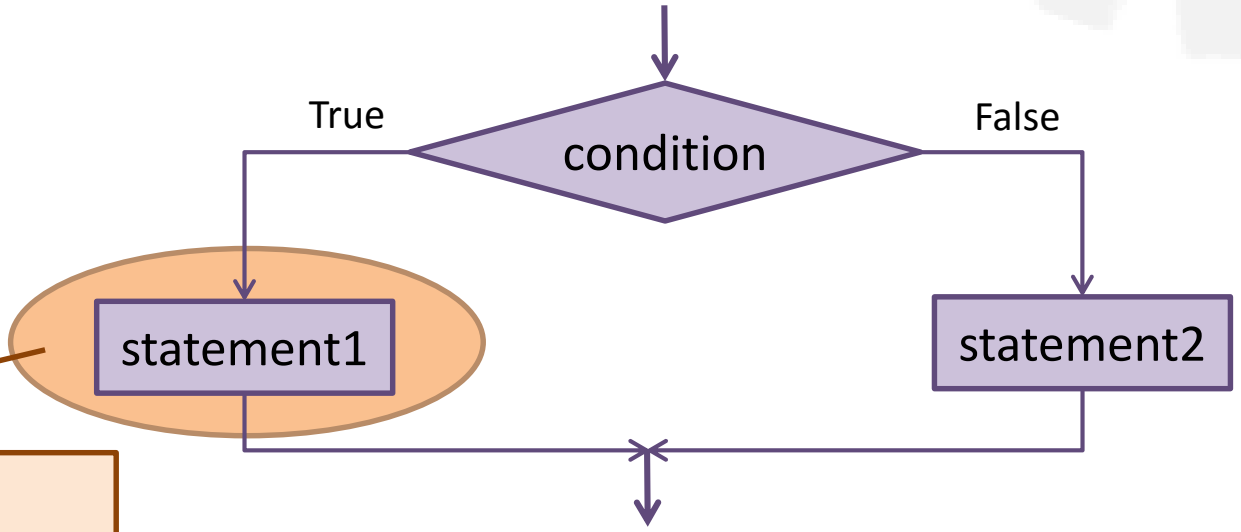
Code#2:

```
int main(){
   int a, b;
   cout << "Enter 2 integers: ";
   cin >> a >> b;
   if(a < b){
        cout << a;
   }
   else{
        cout << b;
   }
   cout << " is the minimum." <<
   endl;
   return 0;
}
```

What's different? Why?

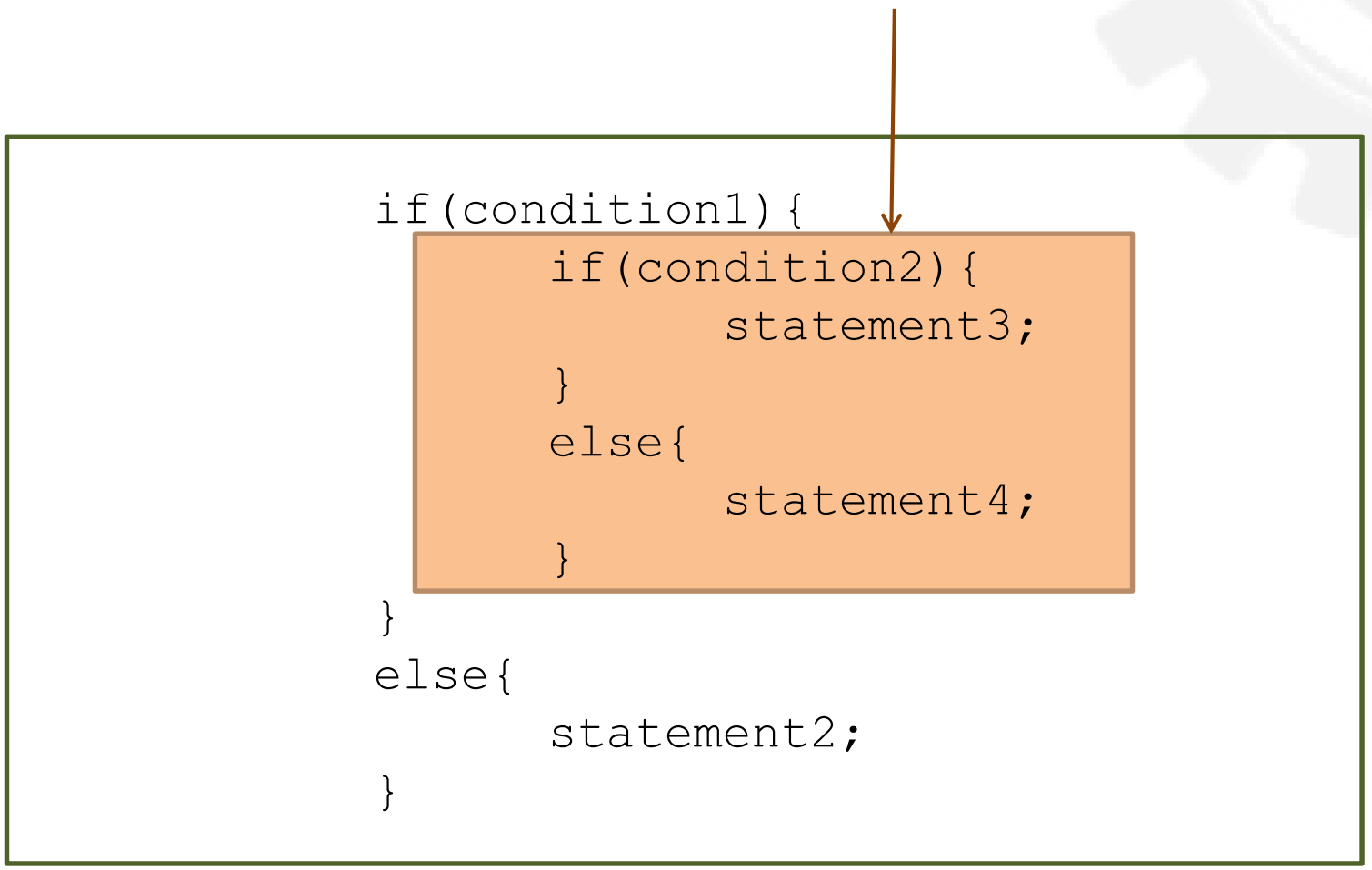
Nested Selection

- Consider a simple if .. else command



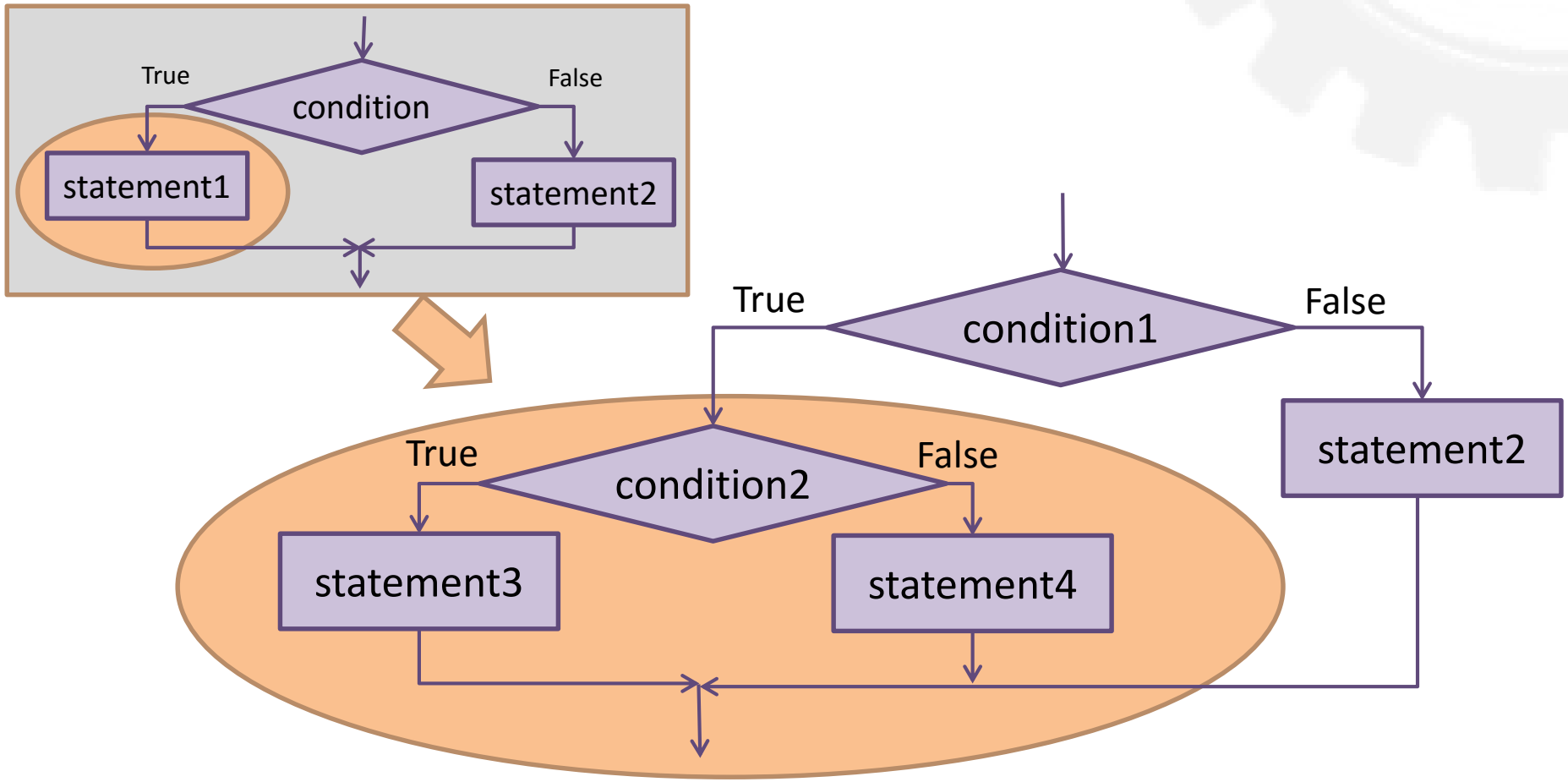
- This statement can be another if .. else command set
- We call this to "nest" a selection set within another selection set

Nested Selection

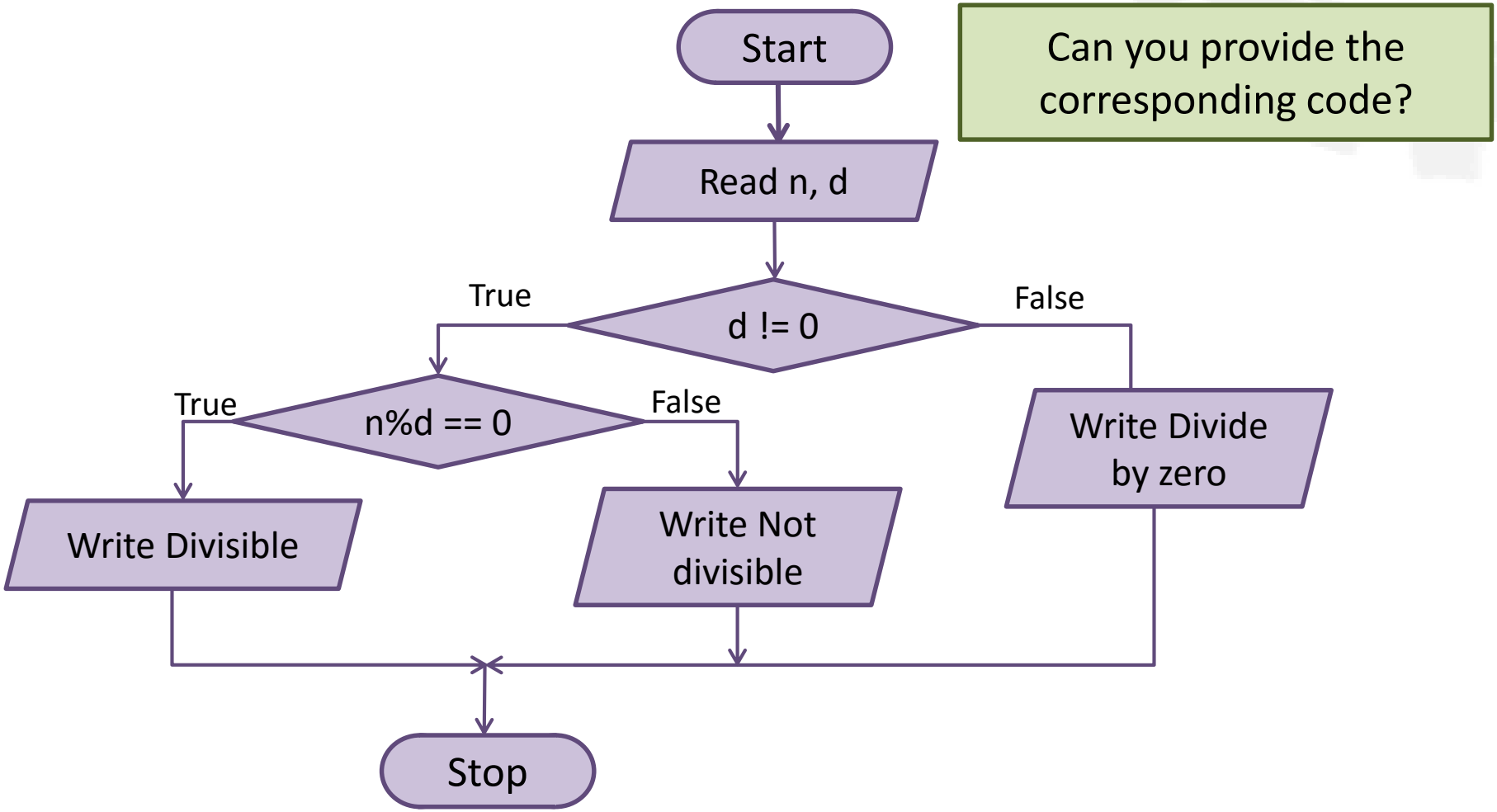


```
if(condition1){
        if(condition2){
                statement3;
        }
        else{
                statement4;
        }
}
else{
        statement2;
}
```

Nested Selection



Nested Selection Example Flowchart



Nested Selection Example Code and Output

Code:

```
int main(){
   int n, d;
   return 0;
}
```

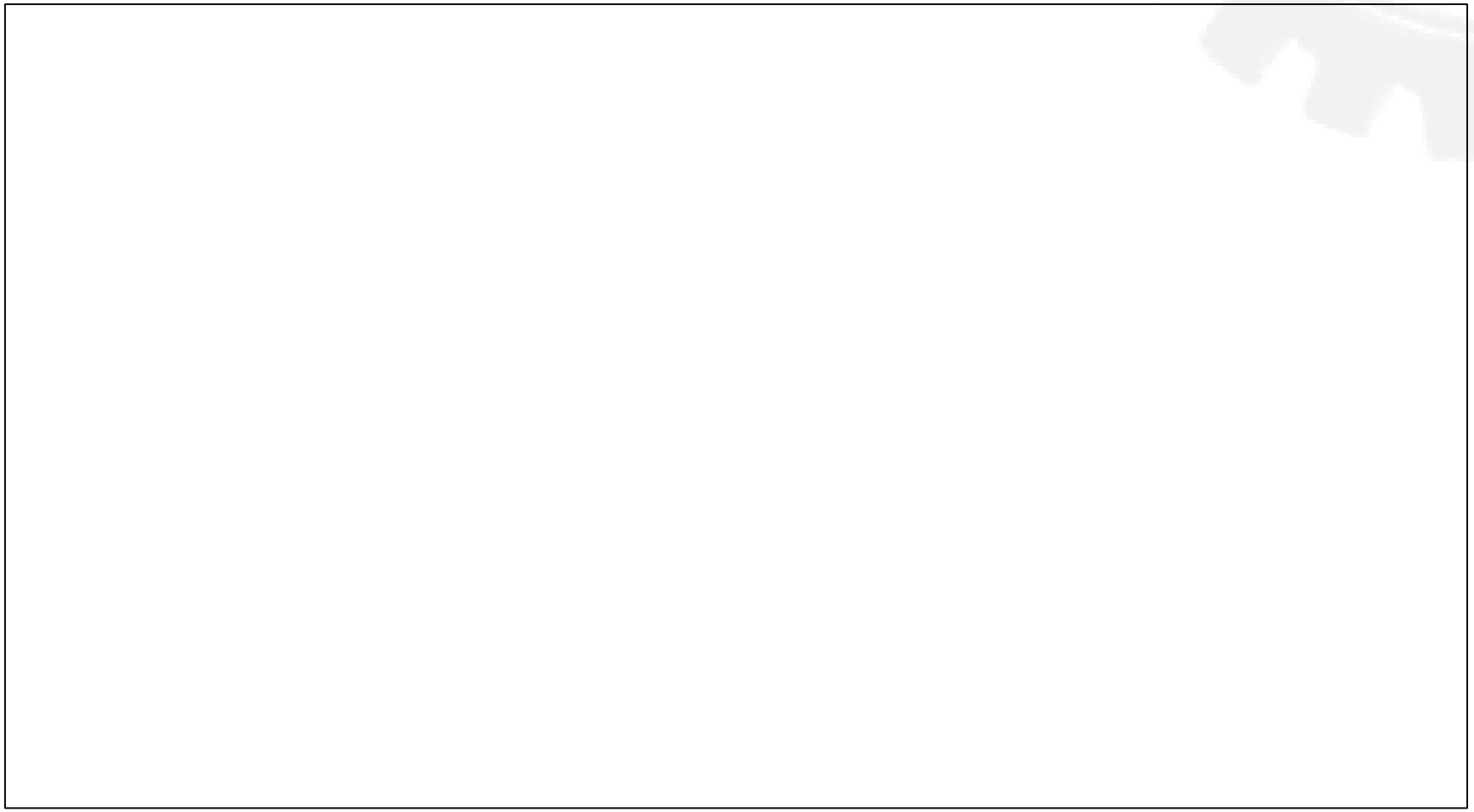
Output:

```
Enter 2 positive integers: 5 6
```

Example: min(n1,n2,n3)

- Problem: find a minimum of three integers
- What you know:
  - How do you know which one is smaller/larger?
- What are inputs, outputs, processes, and/or decision points?
- Let's write a flowchart first

min(n1,n2,n3) Flowchart



min(n1,n2,n3) Code and Output

Code:

```
int main(){
   int n1, n2, n3;
   cout << "Enter 3 positive
   integers: ";
   cin >> n1 >> n2 >> n3;
   return 0;
}
```

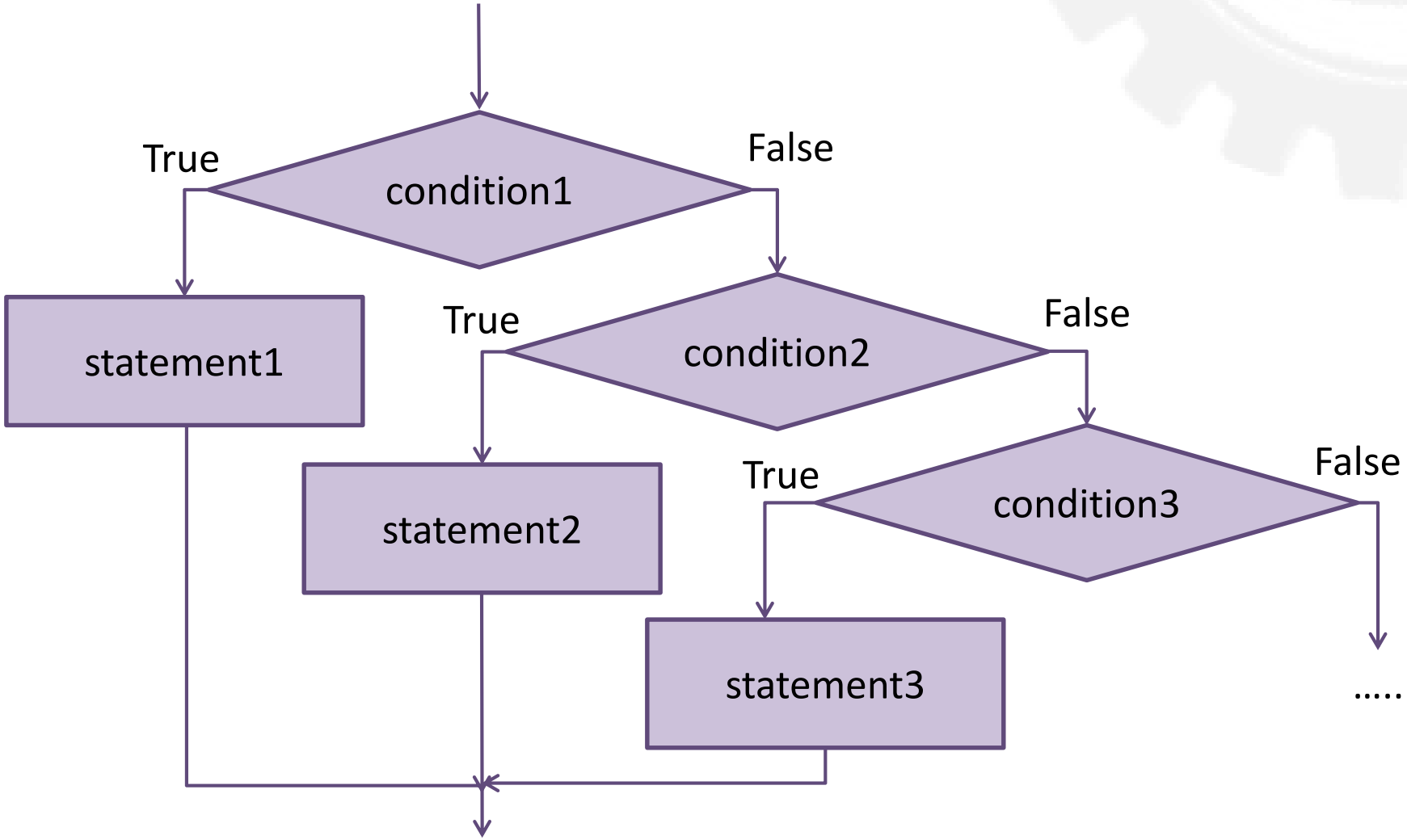
Output:

The else if Statement

```
if(condition1){
    statement1;
}
else if(condition2){
    statement2;
}
else{                       // optional
    statement3;
}
```

As many else if as you need

The else if Statement



else if Example

- What do these programs do?
- What are different?

Code#1:

```
int main(){
    char lang;
    cout << "En Fr De It Ru?
        (e|f|d|i|r) ";
    cin >> lang;
    if(lang == 'e')
        cout << "Welcome";
    else
        if(lang == 'f')
            cout << "Bon Jour";
        else
            if(lang == 'g')
                cout << "Guten Tag";
            else
                …
    cout << endl;
   return 0; }
```

Shorter?
Cleaner?

Code#2:

```
int main(){
    char lang;
    cout << "En Fr De It Ru?
        (e|f|d|i|r) ";
    cin >> lang;
    if(lang == 'e')
        cout << "Welcome";
    else if(lang == 'f')
        cout << "Bon Jour";
    else if(lang == 'g')
        cout << "Guten Tag";
    else if(lang == 'i')
        cout << "Bon Giorno";
    else if(lang == 'r')
        cout << "Dobre Utre";
    else cout << "We don't speak your
        language.";
    cout << endl;
   return 0; }
```

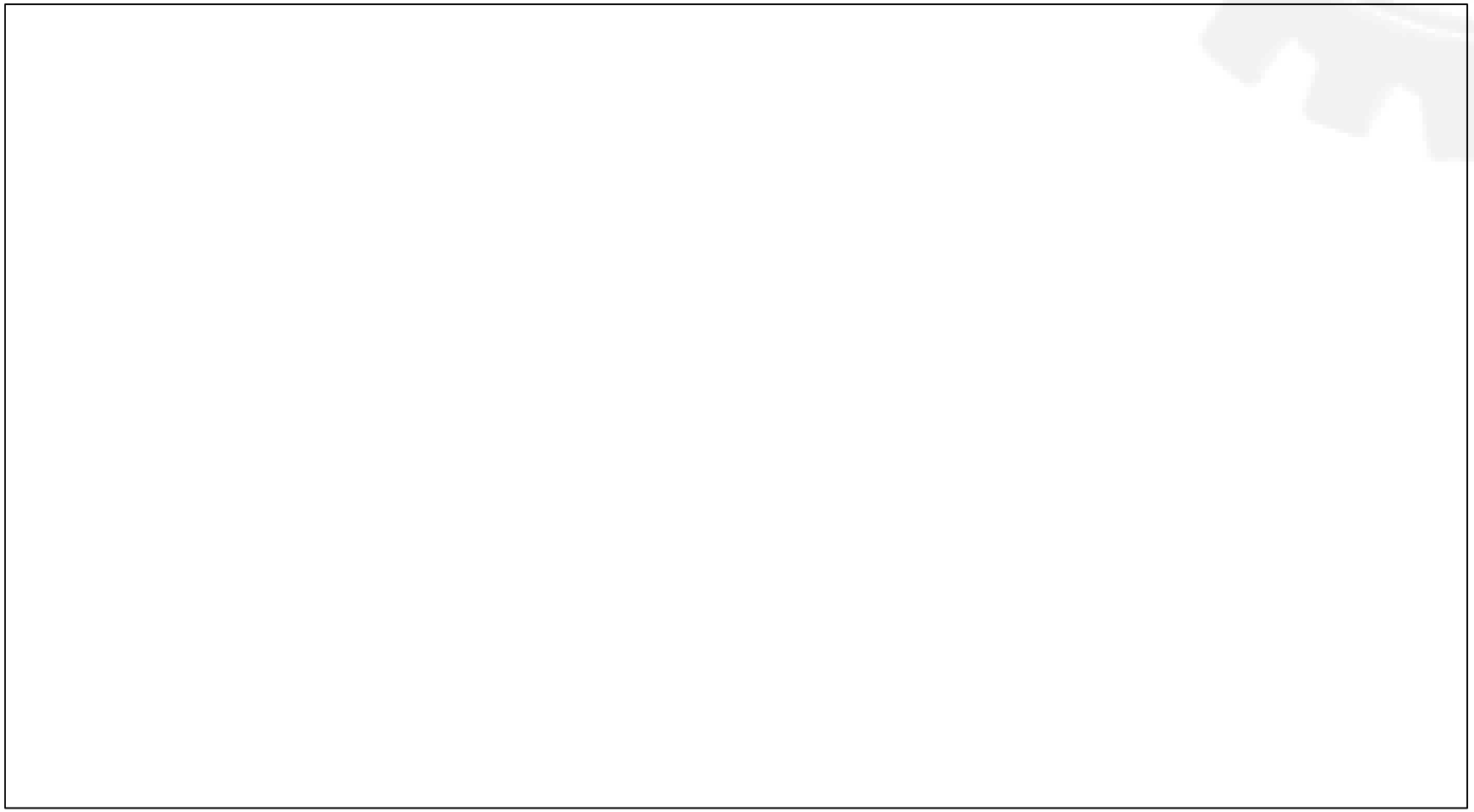
Grading Program

- Problem: read a score from a user and assign a grade corresponding to the score
  - Check for a valid score too

Score	Grade
≥ 80	A
≥ 70 && < 80	B
≥ 60 && < 70	C
≥ 50 && < 60	D
< 50	F

- You've seen this before
  - Let's do the flowchart again

Grading Program Flowchart



Grading Program Code and Output

Code:

```
int main(){
   float score;
```

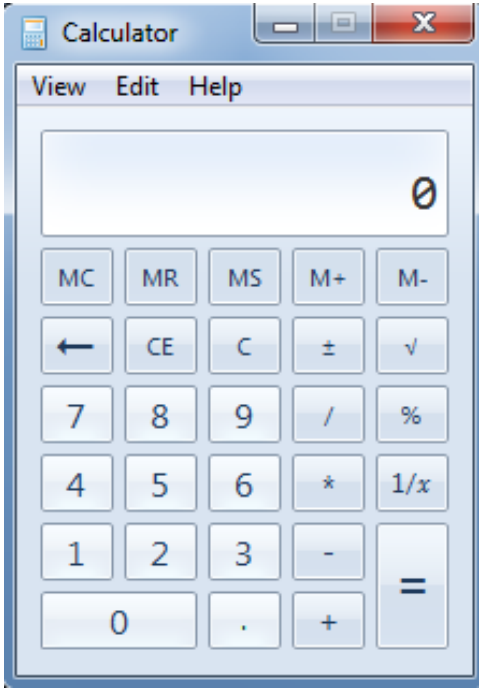
```
   return 0; }
```

Output:

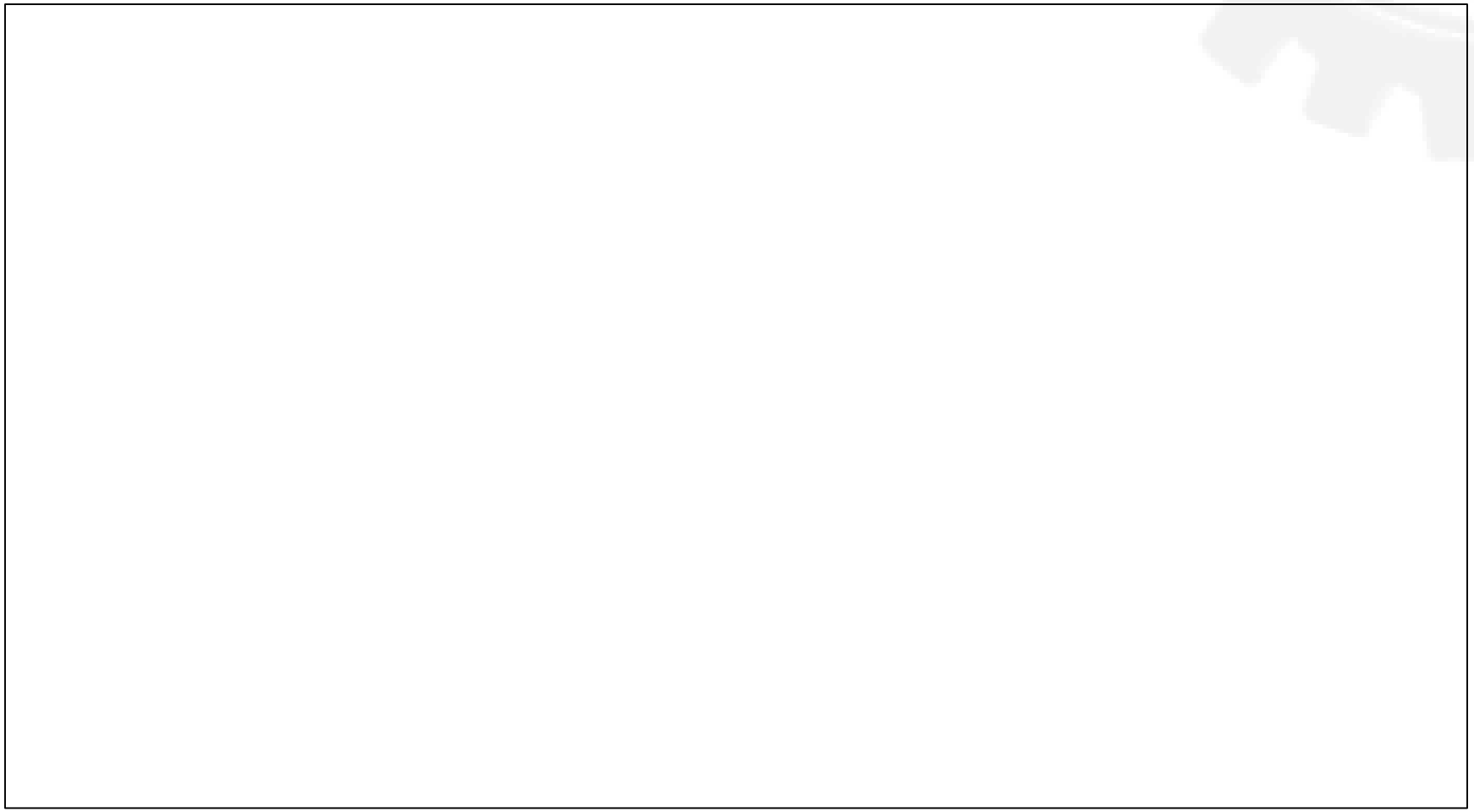
```
Enter a score (0-100): 58
Score = 58, grade = D
```

Calculator

- Problem: take 3 inputs (2 int and 1 char) and do math operations according to the char value
  - Can do only + - * / %
- What do you know?
  - How do we know which math operation to execute?
    Ans: look at the char!
- What are inputs, decisions, processes, and output?
- Let's write a flowchart first



A Simple Calculator Flowchart



A Simple Calculator Code and Output

Code:

```
int main(){
   int x, y;
   char m;
```

```
   return 0; }
```

Output:

```
Enter 2 integers: 5 6
Enter + - * / or %: *
5 * 6 = 30
```

Example: min(n1,n2,n3) revisited

- Problem: find the minimum of three integers using else if

Code:

```
int main(){
   int n1, n2, n3;
   // read input
   cout << "Their minimum is: ";
   // compare integers
   return 0; }
```

Output:

if .. else if VS. if .. if

- Both may check several conditions
- But if..else..if will terminate the checking after a condition is met

Given condition2 and condition4 are true

Code #1:

```
if(condition1) st1;
else if(condition2) st2;
else if(condition3) st3;
else if(condition4) st4;
```

if..else if will stop after executing st2 and will not check condition3 or condition4

Code #2:

```
if(condition1) st1;
if(condition2) st2;
if(condition3) st3;
if(condition4) st4;
```

if will check every single condition, thus executing st2 and st4

if .. else if VS. if .. if Example

Code #1:

```
int main(){
    int score;
    cout << "Score (0-100): ";
    cin >> score;
    if(score >= 80) cout << 'A';
    else if(score >= 70) cout << 'B';
    else if(score >= 60) cout << 'C';
    else if(score >= 50) cout << 'D';
    else cout << 'F';
    return 0;
}
```

Code #2:

```
int main(){
    int score;
    cout << "Score (0-100): ";
    cin >> score;
    if(score >= 80) cout << 'A';
    if(score >= 70) cout << 'B';
    if(score >= 60) cout << 'C';
    if(score >= 50) cout << 'D';
    if(score < 50) cout << 'F';
    return 0;
}
```

What's different?
Why?

Output #1:

```
Score (0-100): 58
```

Output #2:

```
Score (0-100): 58
```

switch Statement

```
switch(expression) {
case constant1: statementList1; break;
case constant2: statementList2; break;
case constant3: statementList3; break;
..
case constantN: statementListN; break;
default: statementList0;
```

Optional

- Used instead of else..if
- Compare expression with each constant
- Optional: default and break

Example: How Many Stars?

Code#1:

```
int main(){
   int star;
   cout << "Enter number of
   star(s): ";
   cin >> star;
   switch(star) {
        case 5: cout << "* ";
        case 4: cout << "* ";
        case 3: cout << "* ";
        case 2: cout << "* ";
        case 1: cout << "* ";
   }
   return 0; }
```

Code#2:

```
int main(){
   int star;
   cout << "Enter number of star(s):
   ";
   cin >> star;
   switch(star) {
        case 5: cout << "* "; break;
        case 4: cout << "* "; break;
        case 3: cout << "* "; break;
        case 2: cout << "* "; break;
        case 1: cout << "* "; break;
   }
   return 0; }
```

What's different?
Why?

Output#1:

Output#2:

switch vs. else..if

- switch <-> else..if are always interchangeable?
  - No!

Code #1:

```
if(score >= 80) grade = 'A';
else if(score >= 70) grade = 'B';
else if(score >= 60) grade = 'C';
else if(score >= 50) grade = 'D';
else grade = 'F';
```

Why?
Because case is followed by a constant only!
(No range allowed)

Code #2:

```
switch(score){
    case >= 80: grade = 'A'; break;
    case >= 70: grade = 'B'; break;
    case >= 60: grade = 'C'; break;
    case >= 50: grade = 'D'; break;
    default: grade = 'F';
}
```

WRONG!

Code #3:

```
switch(score){
    case 80-100: grade = 'A'; break;
    case 70-80: grade = 'B'; break;
    case 60-70: grade = 'C'; break;
    case 50-60: grade = 'D'; break;
    default: grade = 'F';
}
```

WRONG!

Grading Program Revisited

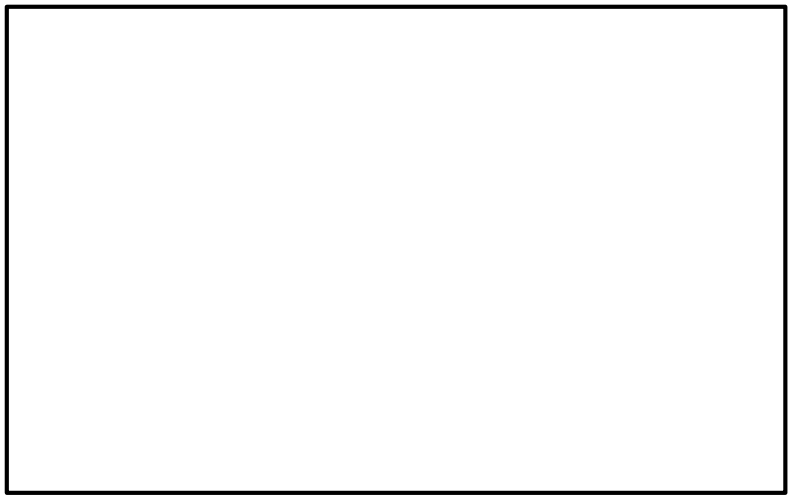
- Use switch instead of if..else..if
- Remember we can't use ranges with switch
  - How to do this?
  - Hint: use the integer division property!

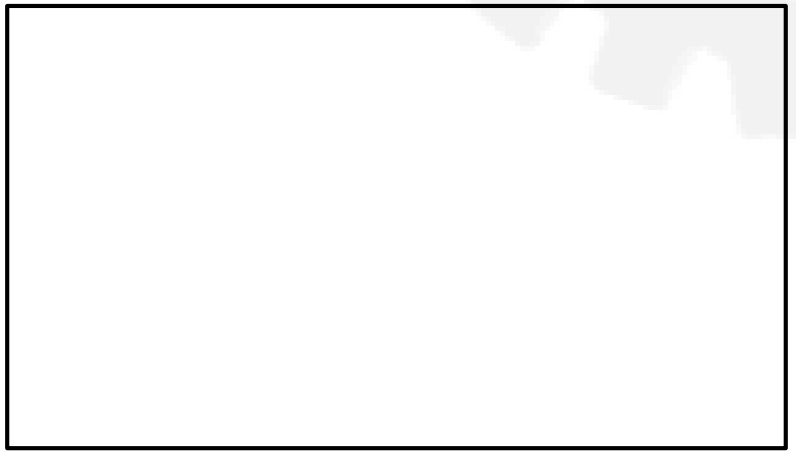
Score	Grade
≥ 80	A
≥ 70 && < 80	B
≥ 60 && < 70	C
≥ 50 && < 60	D
< 50	F

Grading Program Code and Output

Code:

```
int main(){
   int score;
   char grade;
   cout << "Score (0-100): ";
   cin >> score;
   switch(                ){
        case     : grade = 'A';
```





```
return 0; }
```

Output:

```
Enter a score (0-100): 58
Score = 58, grade = D
```

Take Home Messages

- A selection code selects to do statement(s) given a condition is true
- Single condition vs. compound condition
  - Use logical operators: && || !
- In C++, we can use
  - if .. (else ..)
  - if .. else if .. (else ..)
  - switch
- Be careful when comparing

  >= not ≥ <= not ≤

  != not ≠ == not =

Take Home Messages (2)

- if .. else if will do only one set of statement(s) of the first condition that is true
  - if .. if will do all statements if conditions are met
- switch checks a value of a variable against constants
  - Ranges are not allowed
- switch needs break; to stop doing the statements of following cases

References

- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารชุดนำเสนอภาพและบรรยายวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารประกอบการสอนวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- รศ. วิโรจน์ ทวีปวรเดช. (2554). **การเขียนโปรแกรมคอมพิวเตอร์ Computer Programming.** พิมพ์ครั้งที่ 2 โรงพิมพ์มหาวิทยาลัยขอนแก่น
- Cplusplus.com. (n.d.). **C++ Documentation**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- Cplusplus.com. (n.d.). **Library Reference**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- ISO/IEC 14882 Programming Language – C++

EXTRA

min(n1,n2,n3) Revisited

Code:

```
int main(){
   int n1, n2, n3;
   cout << "Enter 3 integers: ";
   cin >> n1 >> n2 >> n3;
   cout <<
          
   << " is the minimum." << endl;
   return 0; }
```

Can you try?
Hint: use nested selection and the conditional expression operator

Output: