We are 183

L06: Week 4 - Wednesday

Engineering Career Fair: January 26th-27th 1 – 6pm

North Campus

Hundreds of Companies

Go talk to them

Discover what they are looking for

Last Time… on EECS 183

Requires, Modifies, Effects (RMEs)
Global and local variables
if-else statements
Boolean operators

Variable and function scope (visibility)

- Starts at declaration point
- Ends at
  - Variables: the closing brace of the enclosing block
  - Functions: the end of the file

```
int foo(int x) {
    cout << x;
    int y = x + 1;
    cout << y;
    return y;
}
```

Scope of x

Variable and function scope (visibility)

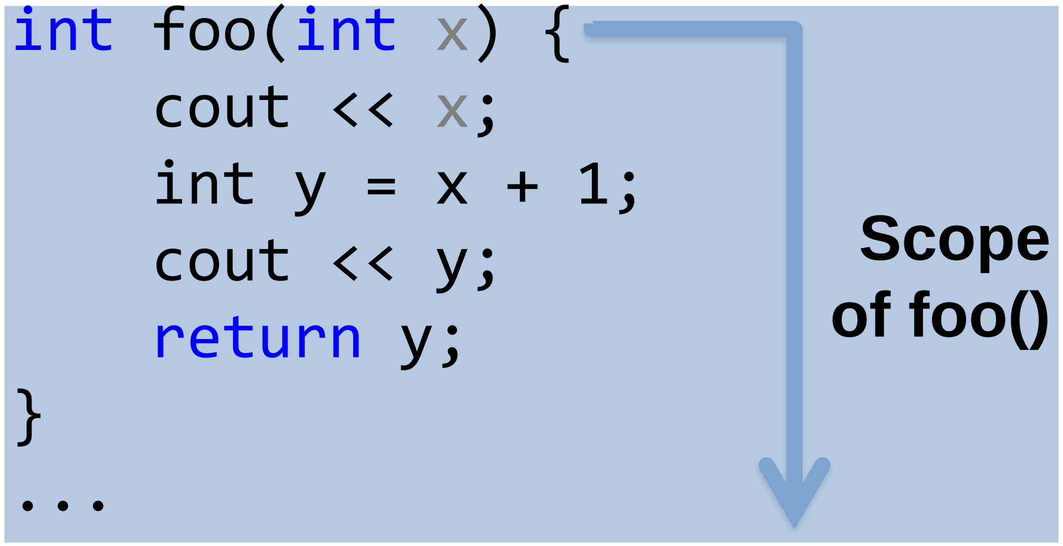
- Starts at declaration point
- Ends at
  - Variables: the closing brace of the enclosing block
  - Functions: the end of the file

```
int foo(int x) {
    cout << x;
    int y = x + 1;
    cout << y;
    return y;
}
```

Scope of y

Variable and function scope (visibility)

- Starts at declaration point
- Ends at
  - Variables: the closing brace of the enclosing block
  - Functions: the end of the file



```
int foo(int x) {
    cout << x;
    int y = x + 1;
    cout << y;
    return y;
}
...
```

Variable and function scope (visibility)

- Starts at declaration point
- Ends at
  - Variables: the closing brace of the enclosing block
  - Functions: the end of the file

- Once execution leaves the scope of a variable, the variable is de-allocated (memory freed)

i>Clicker #1

```
int main(void) {
    int x = 4;
    cout << x;
    if(x < 10) {
        int a = 3;
        cout << a;
    }
    cout << x << a;

    return 0;
}
```

What gets printed?

A) 4343
B) 440
C) 443
D) Error

i>Clicker #1

```
int main(void) {
    int x = 4;
    cout << x;
    if(x < 10) {
        int a = 3;
        cout << a;
    }
    cout << x << a;

    return 0;
}
```

Scope of a

What gets printed?

A) 4343
B) 440
C) 443
D) Error

a is not visible here. It is de-allocated (destroyed) after the execution leaves its scope

i>Clicker #2

```
void bar(void);

double foo(double value) {
    return sqrt(value);
}

int main(void) {
    double x = 49.0;
    …
    bar(foo(x));
    …
}
```

Is this program correct?

a) yes
b) no

i>Clicker #2

```
void bar(void);

double foo(double value) {
    return sqrt(value);
}

int main(void) {
    double x = 49.0;
    …
    bar(foo(x));
    …
}
```

No parameters declared, but bar() is called with one double argument

Is this program correct?

a) yes

[✓ b) no]

if-else == the way we speak

English

if condition,
    then statement(s),
otherwise,
    statements(s)

C++

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

if-else == the way we speak

English

if condition,
    then statement(s),
otherwise,
    statements(s)

C++

Good style: even if only 1 statement

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

if-else == the way we speak

English

if condition,
    then statement(s),
otherwise,
    statements(s)

C++

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

optional

i>Clicker #3

```
int x = 8;
if(x > 5) {
    cout << "more ";
}
if(x < 5) {
    cout << "less ";
}
else {
    cout << "not less";
}
```

What gets printed?

A) more
B) less
C) not less
D) more not less

i>Clicker #3

```
int x = 8;
if(x > 5) {
    cout << "more ";
}
if(x < 5) {
    cout << "less ";
}
else {
    cout << "not less";
}
```

true

What gets printed?

A) more
B) less
C) not less
D) more not less

i>Clicker #3

```
int x = 8;
if(x > 5) {
    cout << "more ";
}
if(x < 5) {
    cout << "less ";
}
else {
    cout << "not less";
}
```

false

What gets printed?

A) more
B) less
C) not less
D) more not less

i>Clicker #3

```
int x = 8;
if(x > 5) {
    cout << "more ";
}
if(x < 5) {
    cout << "less ";
}
else {
    cout << "not less";
}
```

Executes

What gets printed?

A) more
B) less
C) not less
D) more not less

i>Clicker #4

```
int x = 8;
if(x = 7) {
    cout << "equal";
}
else {
    cout << "not equal";
}
```

What gets printed?

A) equal
B) not equal
C) Nothing due to an error

i>Clicker #4

```
int x = 8;
if(x = 7) {
    cout << "equal";
}
else {
    cout << "not equal";
}
```

What gets printed?

A) equal
B) not equal
C) Nothing due to an error

Boolean Operators

We saw the **==** operator earlier, that says two numbers/strings are equal; here are all the operators

Expression	Meaning
• `(a == b)`	a is **equal to** b
• `(a != b)`	a is **not equal to** b
• `(a > b)`	a is **greater than** b
• `(a >= b)`	a is **greater than or equal to** b
• `(a < b)`	a is **less than** b
• `(a <= b)`	a is **less than or equal to** b

Boolean Operators

We can also combine multiple Boolean expressions:

Expression	Meaning
`(a && b)`	`both a and b are true`
`(a || b)`	`at least one of a and b is true`
`(!a)`	`not a`

i>Clicker #5

```
int x = 72;
if(x > 9 && x < 20 || x % 2 == 0) {
    x = x + 1;
}
else {
    x = 0;
}
```

What is the value of x after executing this code?

A) 72
B) 9
C) 0
D) 73

i>Clicker #5

```
int x = 72;
if(x > 9 && x < 20 || x % 2 == 0) {
    x = x + 1;
}
else {
    x = 0;
}
```

What is the value of x after executing this code?

A) 72
B) 9
C) 0
D) 73

i>Clicker #6 Exam F14

```
bool doIt = true;
if(!(!(doIt)) ) {
    cout << "Beware The Mummy! ";
}
else {
    cout << "Beware The Witch! ";
}
```

What does the above code snippet print?

A) `Beware The Mummy!`
B) `Beware The Witch!`
C) `true`
D) `false`
E) `Beware The Mummy! Beware The Witch!`

i>Clicker #6 Exam F14

```
bool doIt = true;
if(!(!(doIt)) ) {
    cout << "Beware The Mummy! ";
}
else {
    cout << "Beware The Witch! ";
}
```

What does the above code snippet print?

[✓ A) Beware The Mummy!]
B) Beware The Witch!
C) true
D) false
E) Beware The Mummy! Beware The Witch!

Complementing Expressions

```
if (x < 5) {
   cout << "x is less than 5";
}

if (x >= 5) {
   cout << "x is at least 5";

}
```

Complementing Expressions

```
if (x < 5) {
   cout << "x is less than 5";
}

~~if (x >= 5)~~ {
   cout << "x is at least 5";

}
```

Redundant code

Can be replaced by `else`

Complementing Expressions

```
if (x < 5) {
   cout << "x is less than 5";
} else {
   cout << "x is at least 5";

}
```

Doing this reduces the number of comparisons that the computer must perform.

And likely the number of mistakes you will make!

Complementing

Operator	Complement
`<`	`>=`
`>=`	`<`
`==`	`!=`

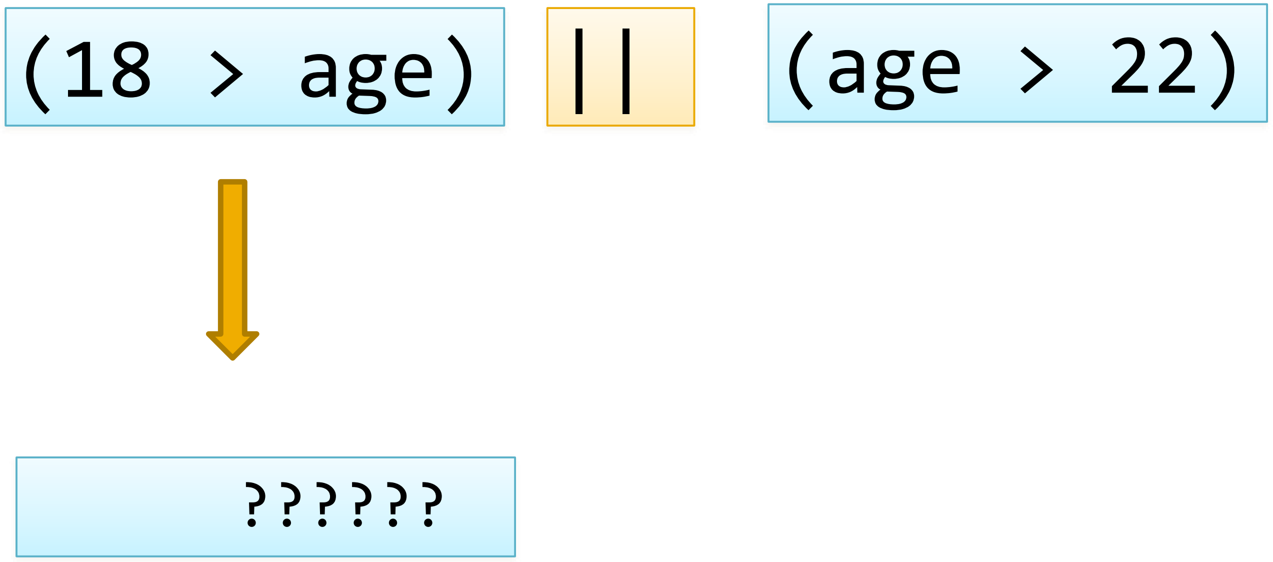
Expr	Complement
`x < 5`	`!(x < 5)`

better style: `x >= 5`

DeMorgan's Theorem

- To complement a compound logical expression:
  - First write complement of each sub-expression
  - Change each AND to OR and
    each OR to AND

Solve:



complement

Solve:

`(18 > age)` `||` `(age > 22)`

$$\Big\downarrow$$

complement

`(18 <= age)`

Solve:

complement

$$\begin{tikzcd}
\texttt{(18 > age)} \arrow[d] & \texttt{||} & \texttt{(age > 22)} \arrow[d] \\
\texttt{(18 <= age)} & & \texttt{??????}
\end{tikzcd}$$

Solve:

$$\begin{tikzcd}
\texttt{(18 > age)} \arrow[d, "\text{complement}"'] & \texttt{||} & \texttt{(age > 22)} \arrow[d] \\
\texttt{(18 <= age)} & & \texttt{(age <= 22)}
\end{tikzcd}$$

Solve:

$$\begin{tikzcd}
& \texttt{(18 > age)} \arrow[d] & \texttt{||} \arrow[d] & \texttt{(age > 22)} \arrow[d] \\
\text{complement} & \texttt{(18 <= age)} & \texttt{??} & \texttt{(age <= 22)}
\end{tikzcd}$$

Solve:

```
(18 > age) || (age > 22)
```

complement

$$\begin{tikzcd}
\texttt{(18 > age)} \arrow[d] & \texttt{||} \arrow[d] & \texttt{(age > 22)} \arrow[d] \\
\texttt{(18 <= age)} & \texttt{\&\&} & \texttt{(age <= 22)}
\end{tikzcd}$$

```
(18 <= age) && (age <= 22)
```

Solve:

$$\begin{tikzcd} \texttt{(18 > age)} \arrow[d, bend right, "\text{complement}"'] & \texttt{||} & \texttt{(age > 22)} \\ \texttt{(18 <= age)} & \texttt{\&\&} & \texttt{(age <= 22)} \end{tikzcd}$$

Important!

age = 12

$$\underbrace{\underbrace{\text{18 <= age}}_{\text{false (0)}} \text{ \&\& } \underbrace{\text{age <= 22}}_{\text{true (1)}}}_{\text{false}}$$

VS

$$\underbrace{\underbrace{\text{18 <= age}}_{\text{false (0)}} \text{ <= 22}}_{\text{true}}$$

Important!

age = 20

$$\underbrace{\underbrace{18 \le \text{age}}_{\text{true (1)}} \;\&\&\; \underbrace{\text{age} \le 22}_{\text{true (1)}}}_{\text{true}}$$

vs

$$\underbrace{\underbrace{18 \le \text{age}}_{\text{true (1)}} \le 22}_{\text{true}}$$

Important!

age = 25

$$\underbrace{\underbrace{18 <= age}_{\text{true (1)}} \;\&\&\; \underbrace{age <= 22}_{\text{false (0)}}}_{\text{false}}$$

VS

$$\underbrace{\underbrace{18 <= age}_{\text{true (1)}} <= 22}_{\text{true}}$$

ALWAYS True!!

Short Circuit Evaluation

```
(i == 5)  &&  (j > 10)
```

If the left operand of && is false

Right operand is not evaluated

and the entire expression is false

```
(i == 5)  ||  (j > 10)
```

If the left operand of || is true

Right operand is not evaluated

and the entire expression is true

Short Circuit Evaluation

```
if ((i == 5)  &&  (sqrt(j) > 10)
```

If the left operand of && is false

Right operand is not evaluated

and the entire expression is false

More efficient if left operand is inexpensive

Short Circuit Evaluation

```
int i = 0;
if ((i != 0)  &&  (j / i > 10))
```

If the left operand of && is false

Right operand is not evaluated

and the entire expression is false

Useful for error prevention

Short Circuit Evaluation

- Promotes efficiency
  - Put conditions that are less expensive first
- Useful in error prevention
- In case of AND &&:
  - Put the expression that is most likely **false** first
- In case of OR ||:
  - Put the expression that is most likely **true** first

Today

Comparing `string`

Comparing `double`

Nested if-else statements

`switch` statements

Comparing string variables

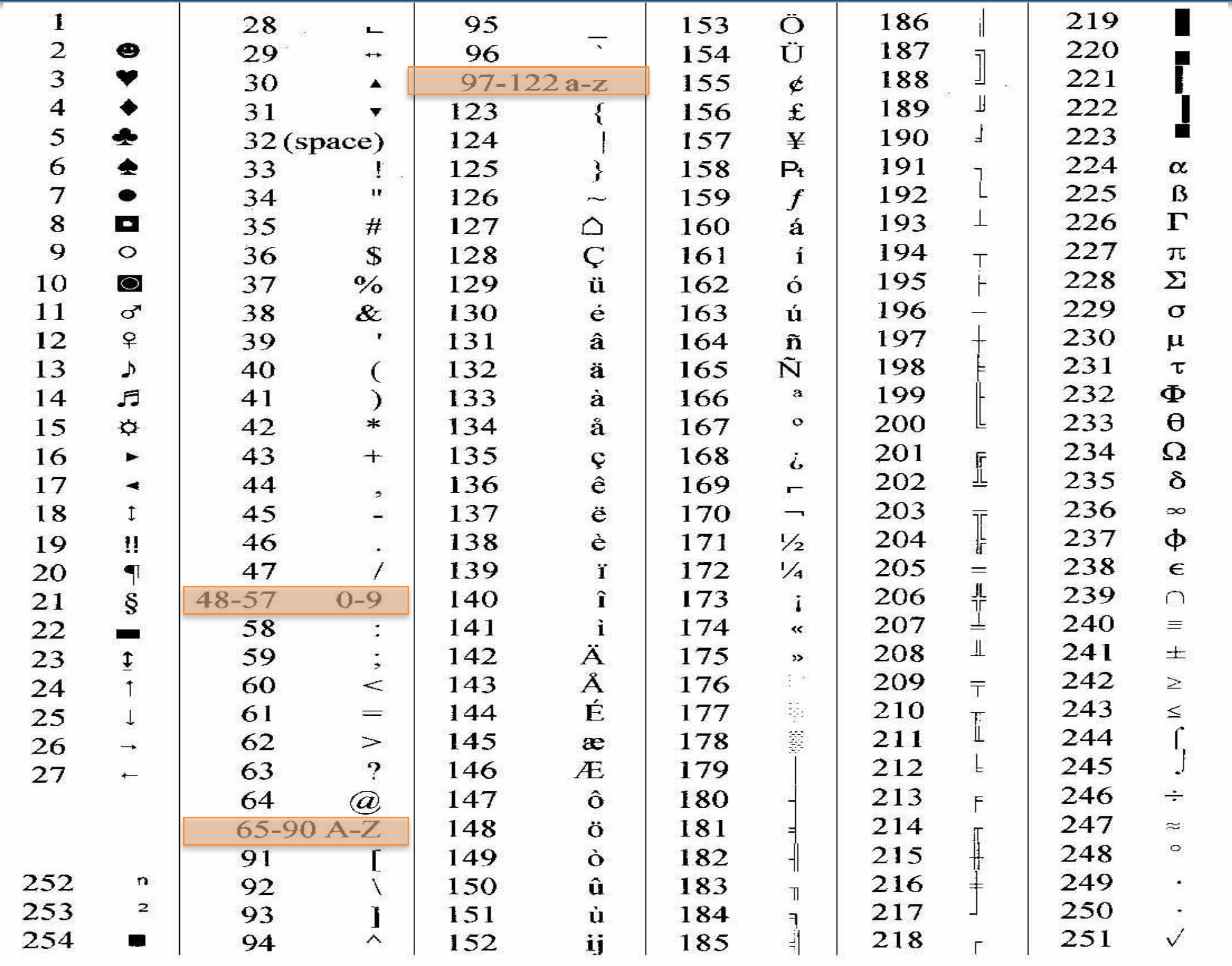
```
if(user == "Fred") {
    …
}
if("apples" > "oranges") {
    …
}
if("apples" > "Apples") {
    …
}
```

char by char: ASCII followed by length

```
if("apples" == "apples") {
    …
}
```

1. Compare two strings character-by-character
2. Compare using the ASCII value of each character
    1. 0-9 in order
    2. A-Z in order, but case sensitive
3. If all characters match, the longer string is greater

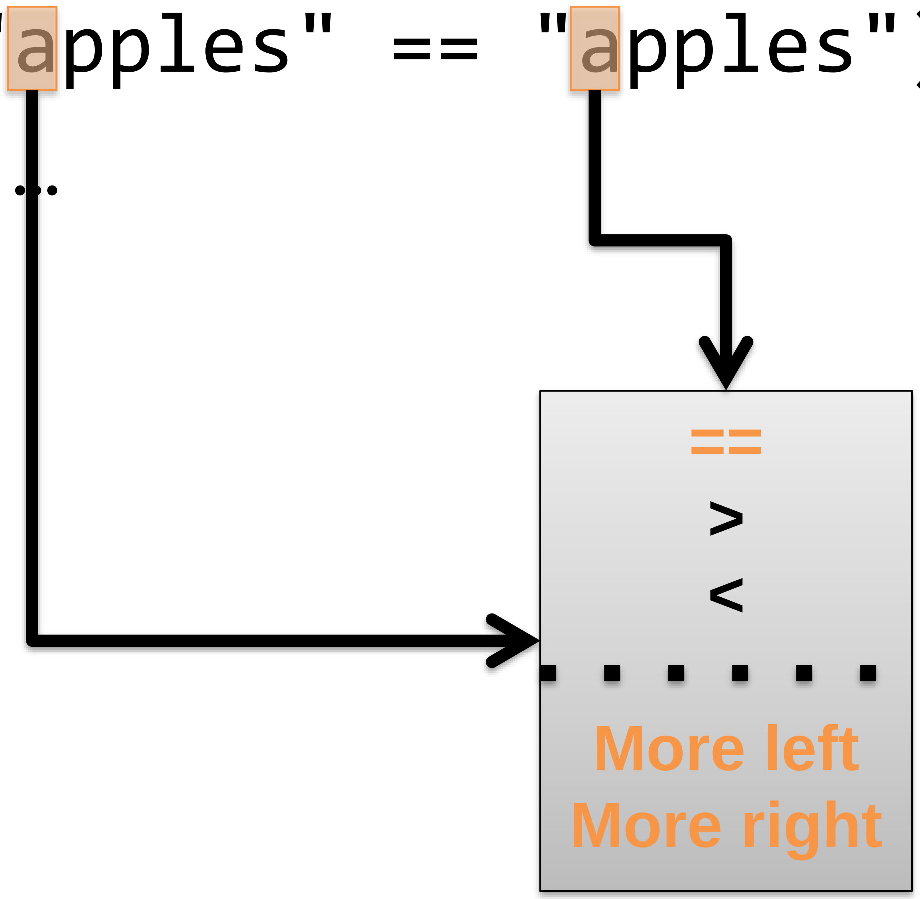
char by char: ASCII followed by length



1		28	∟	95	_	153	Ö	186	║	219	█
2	☻	29	↔	96	`	154	Ü	187	╗	220	▄
3	♥	30	▲	97-122 a-z		155	¢	188	╝	221	▌
4	♦	31	▼	123	{	156	£	189	╜	222	▐
5	♣	32 (space)		124	|	157	¥	190	╛	223	▀
6	♠	33	!	125	}	158	₧	191	┐	224	α
7	•	34	"	126	~	159	ƒ	192	└	225	ß
8	◘	35	#	127	⌂	160	á	193	┴	226	Γ
9	○	36	$	128	Ç	161	í	194	┬	227	π
10	◙	37	%	129	ü	162	ó	195	├	228	Σ
11	♂	38	&	130	é	163	ú	196	─	229	σ
12	♀	39	'	131	â	164	ñ	197	┼	230	µ
13	♪	40	(	132	ä	165	Ñ	198	╞	231	τ
14	♫	41	)	133	à	166	ª	199	╟	232	Φ
15	☼	42	*	134	å	167	º	200	╚	233	Θ
16	►	43	+	135	ç	168	¿	201	╔	234	Ω
17	◄	44	,	136	ê	169	⌐	202	╩	235	δ
18	↕	45	-	137	ë	170	¬	203	╦	236	∞
19	‼	46	.	138	è	171	½	204	╠	237	φ
20	¶	47	/	139	ï	172	¼	205	═	238	ε
21	§	48-57	0-9	140	î	173	¡	206	╬	239	∩
22	▬	58	:	141	ì	174	«	207	╧	240	≡
23	↨	59	;	142	Ä	175	»	208	╨	241	±
24	↑	60	<	143	Å	176	░	209	╤	242	≥
25	↓	61	=	144	É	177	▒	210	╥	243	≤
26	→	62	>	145	æ	178	▓	211	╙	244	⌠
27	←	63	?	146	Æ	179	│	212	╘	245	⌡
		64	@	147	ô	180	┤	213	╒	246	÷
		65-90 A-Z		148	ö	181	╡	214	╓	247	≈
		91	[	149	ò	182	╢	215	╫	248	°
252	ⁿ	92	\	150	û	183	╖	216	╪	249	∙
253	²	93	]	151	ù	184	╕	217	┘	250	·
254	■	94	^	152	ÿ	185	╣	218	┌	251	√

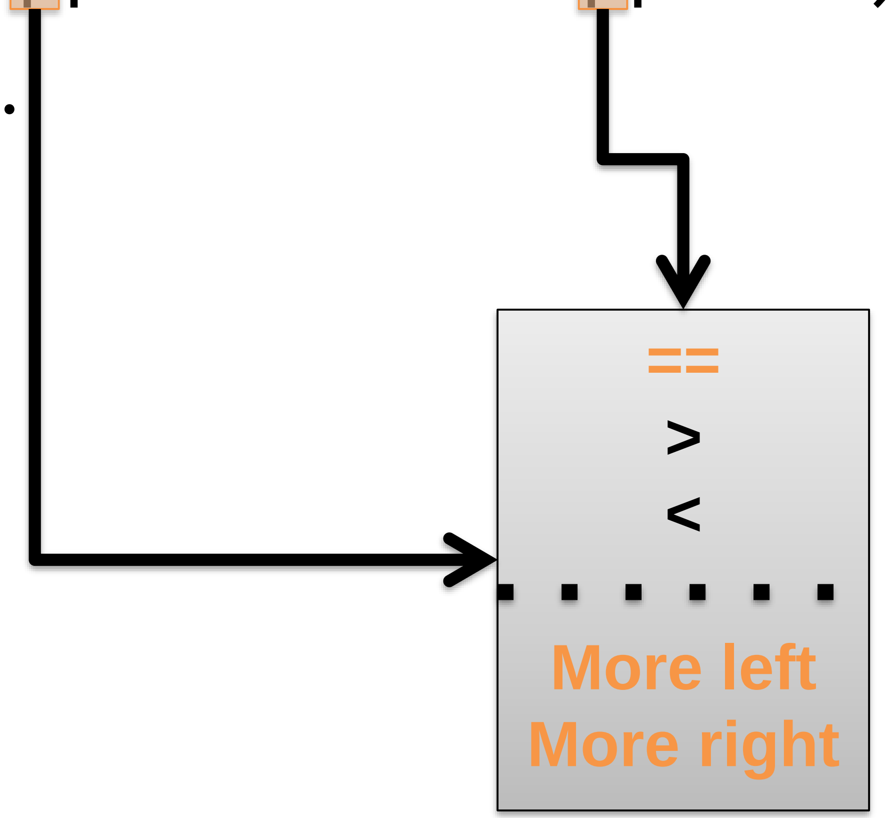
char by char: ASCII followed by length

```
if("apples" == "apples") {
      ...
}
```



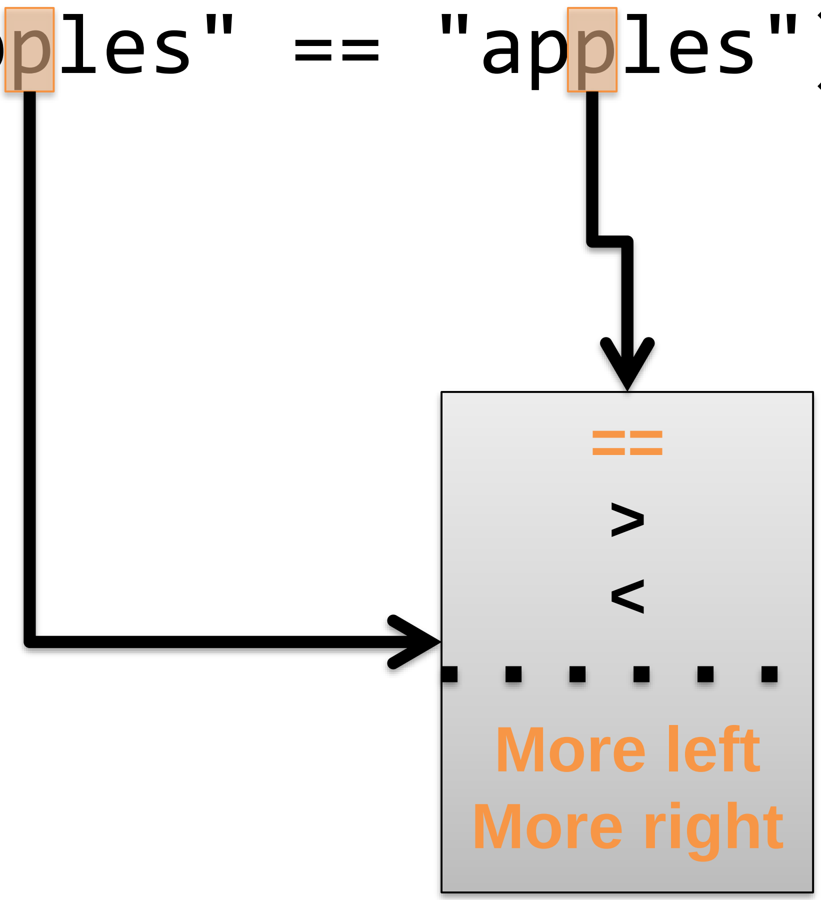
char by char: ASCII followed by length

```
if("apples" == "apples") {
      …
}
```



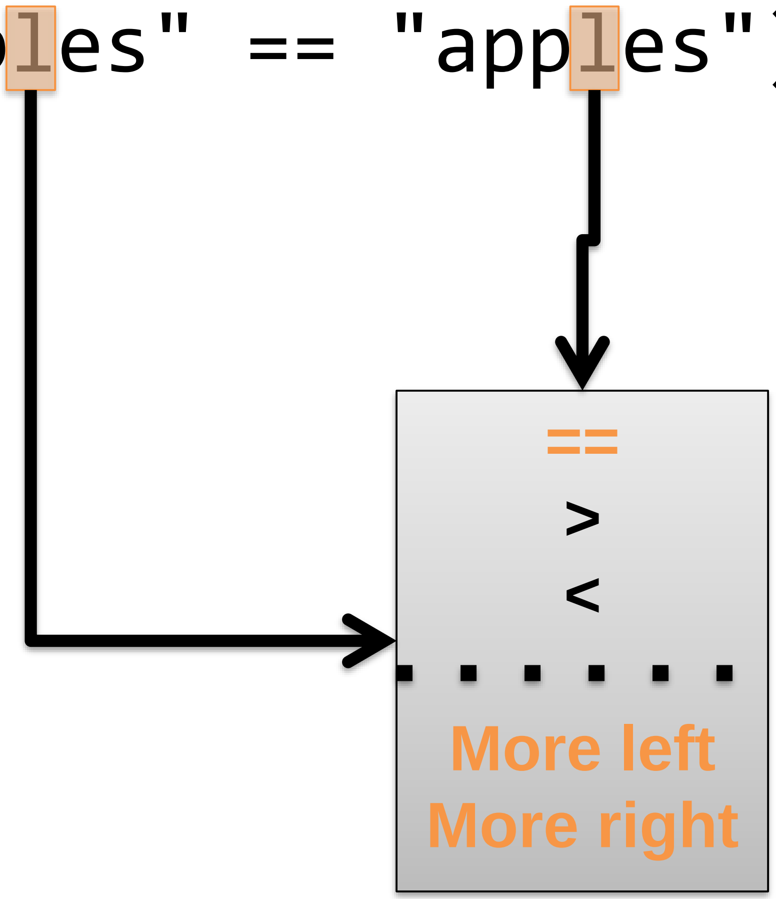
char by char: ASCII followed by length

```
if("apples" == "apples") {
      …
}
```



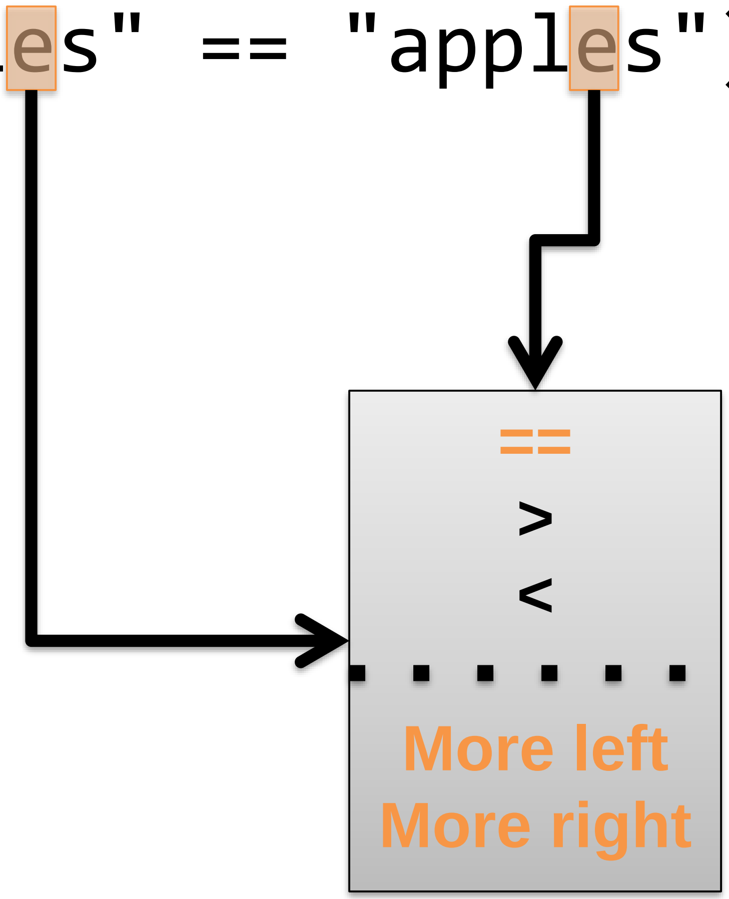
char by char: ASCII followed by length

```
if("apples" == "apples") {
      …
}
```



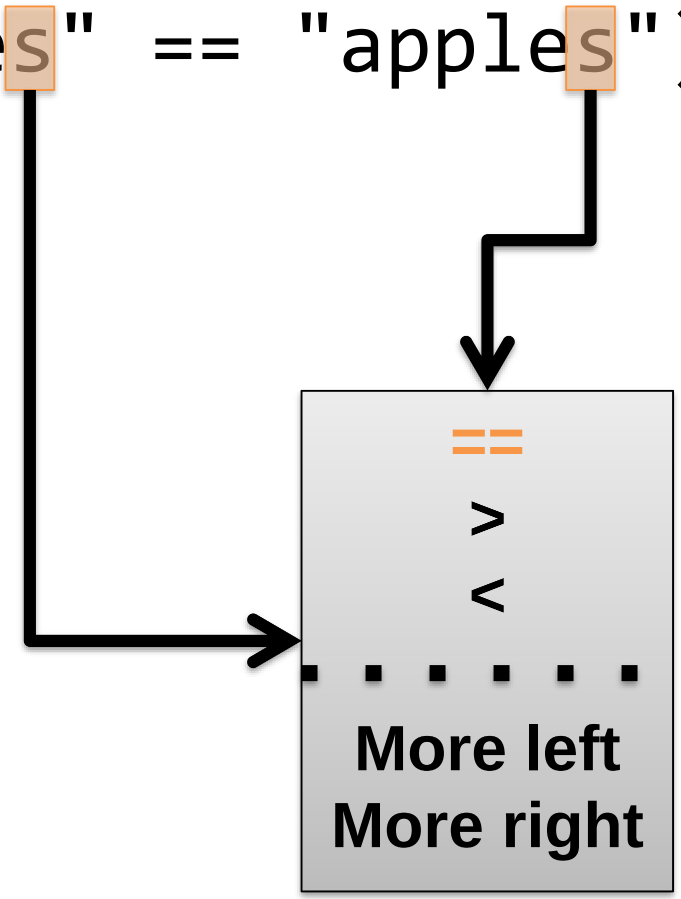
char by char: ASCII followed by length

```
if("apples" == "apples") {
        …
}
```



char by char: ASCII followed by length

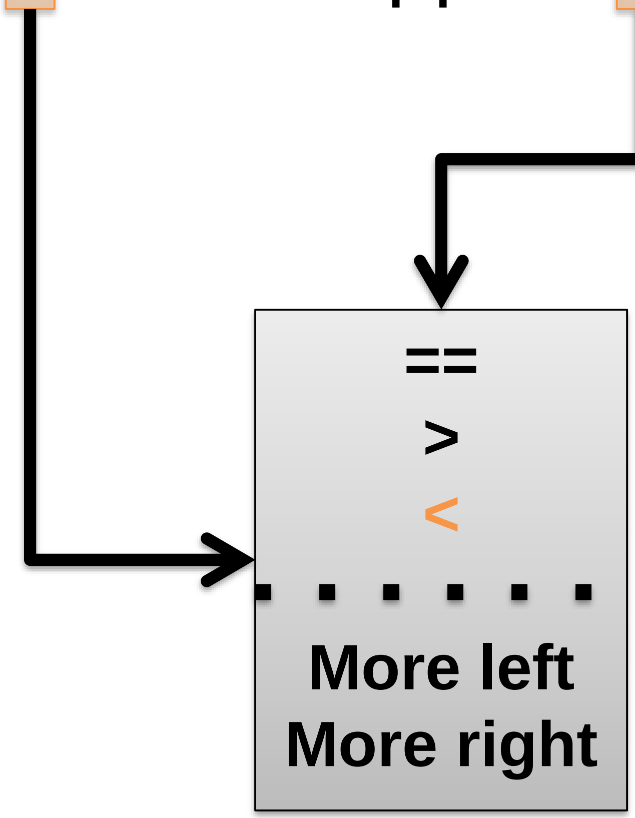
```
if("apples" == "apples") {
      …
}
```



Since no more chars left in either string, we stop and go with == as the result, which makes this condition true

char by char: ASCII followed by length

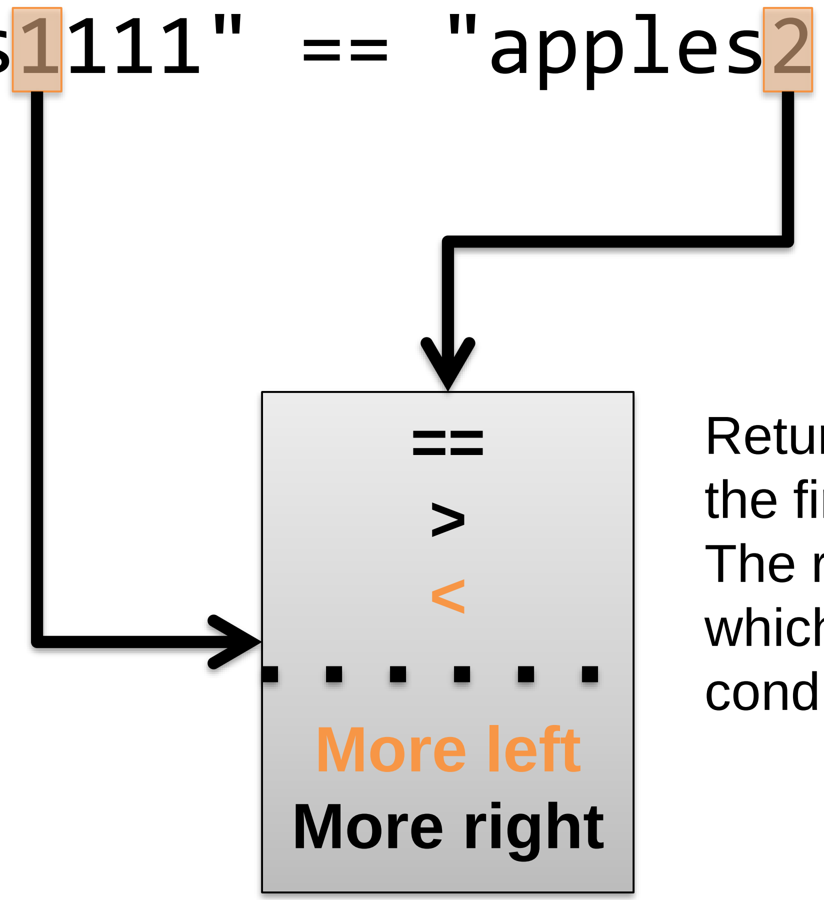
```
if("apples1" == "apples2") {
      …
}
```



Since no more chars left in either string, we stop and go with < as the result, which makes this condition false

char by char: ASCII followed by length

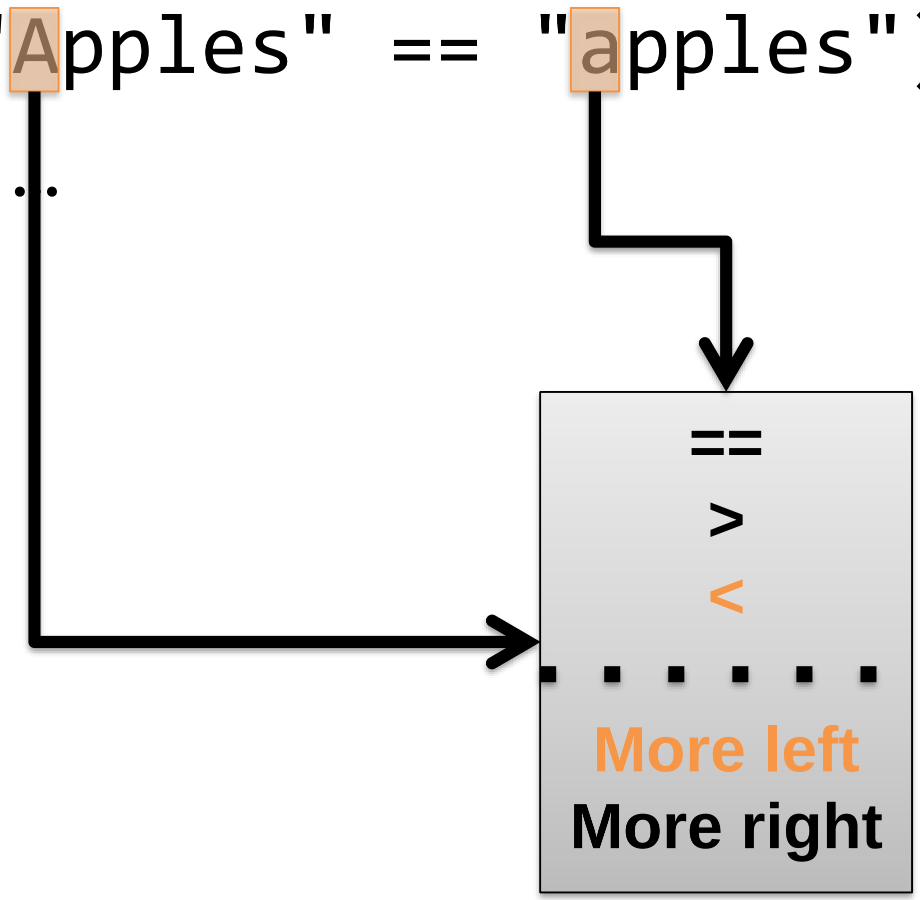
```
if("apples1111" == "apples2") {
      …
}
```



Return an answer at the first difference. The result is still <, which makes this condition false

char by char: ASCII followed by length

```
if("Apples" == "apples") {
    ...
}
```



Return an answer at the first difference. The result is still <, which makes this condition false

i>Clicker #7

```
"eeCS" > "EECS183"
```

This condition evaluates to:

```
A) true
B) false
```

i>Clicker #7

```
"eeCS" > "EECS183"
```

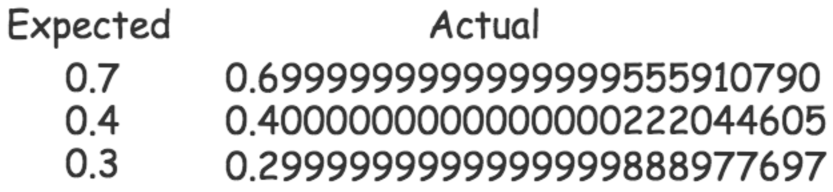
This condition evaluates to:

A) `true`

B) `false`

Comparing `double` variables

- Variables of the double type **MAY** hold inexact values



Expected	Actual
0.7	0.6999999999999999555910790
0.4	0.4000000000000000222044605
0.3	0.2999999999999999888977697

  - Imprecision accumulates
- This imprecision needs to be accounted for in comparison
  - Do **NOT** use
    - `==`
    - `!=`
  - Do use
    - `fabs(x – y) > .0001 to replace !=`
    - `fabs(x – y) < .0001 to replace ==`

Nesting if-else

if statement

Scope of **true** branch

```
if  ( schoolDay )
{
   // Do Something
}
```

If-else statement

```
           if  ( schoolDay )
Scope of   {
  true         // do something
 branch    }
           else
Scope of   {
 false         // do something else
 branch    }
```

Multiple Conditions

```
if  (schoolDay && wednesday) {
   // Do Something
}
```

Nested Conditionals

```
if  (schoolDay && wednesday) {
   // Do Something
}
```

Equivalent

```
if  (schoolDay) {
   if  (wednesday) {
      // Do Something
   }
}
```

Multi-condition if-else

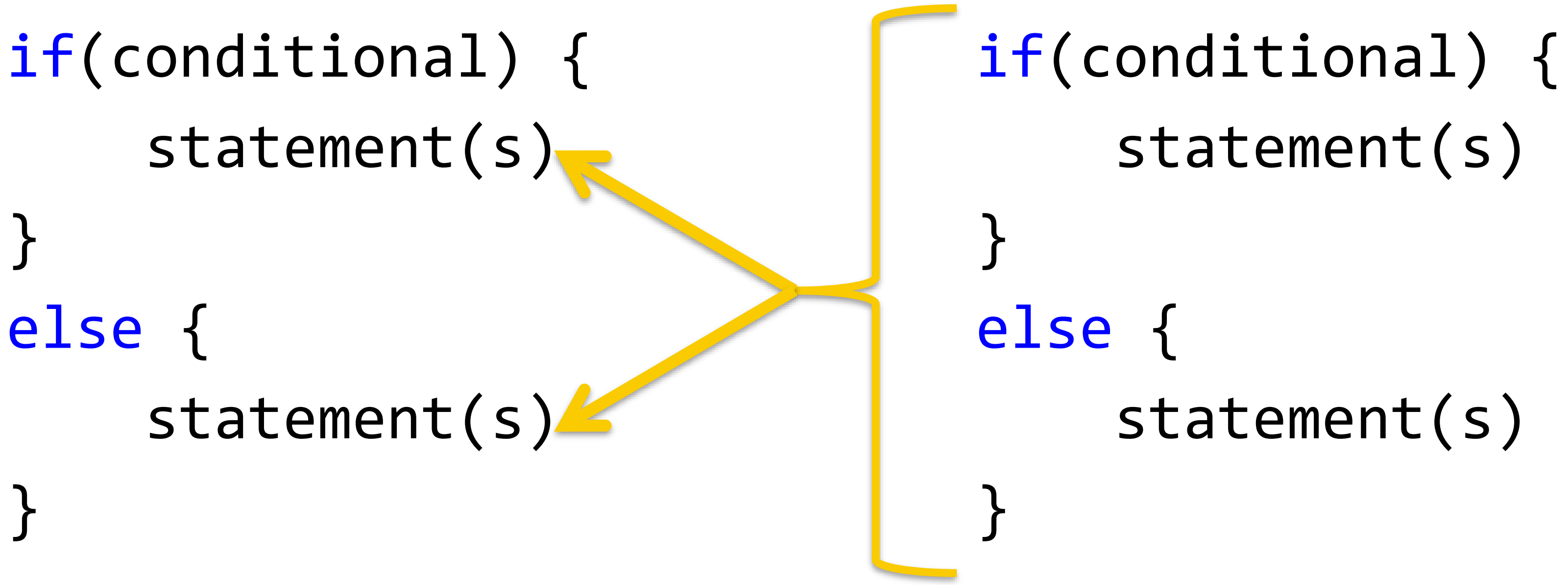
```
if(grade >= 90) {
    cout << "A";
}
if(grade < 90 && grade >= 80) {
    cout << "B";
}
if(grade < 80 && grade >= 70) {
    cout << "C";
}
if(grade < 70 && grade >= 60) {
    cout << "D";
}
if(grade < 60) {
    cout << "F";
}
```

Ugly, lots of code to write, and error prone

if-else is itself a statement

```
if(conditional) {
    statement(s)
}
else {
    statement(s)
}
```

if-else is itself a statement



```
if(conditional) {
    statement(s)
}
else {
    statement(s)
}
```

```
if(conditional) {
    statement(s)
}
else {
    statement(s)
}
```

if-else statements nest

```
if(grade >= 90) {
    cout << "A";
}
if(grade < 90 && grade >= 80) {
    cout << "B";
}
if(grade < 80 && grade >= 70) {
    cout << "C";
}
if(grade < 70 && grade >= 60) {
    cout << "D";
}
if(grade < 60) {
    cout << "F";
}
```

5 independent conditionals

if-else statements nest

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    }
}
if(grade < 80 && grade >= 70) {
    cout << "C";
}
if(grade < 70 && grade >= 60) {
    cout << "D";
}
if(grade < 60) {
    cout << "F";
}
```

4 independent conditionals

if-else statements nest

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        }
    }
}
if(grade < 70 && grade >= 60) {
    cout << "D";
}
if(grade < 60) {
    cout << "F";
}
```

3 independent conditionals

if-else statements nest

```cpp
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            }
        }
    }
}
if(grade < 60) {
    cout << "F";
}
```

2 independent conditionals

if-else statements nest

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

1 independent conditional
With up to 4 levels of nesting

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade < 90 && grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade < 80 && grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade < 70 && grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                if(grade < 60) {
                    cout << "F";
                }
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Remove redundant conditions

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Correct

Lots of
redundant conditions

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57
Output:

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57

Output:

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57

Output:

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57
Output:

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57
Output:

Execution of nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Input: grade = 57
Output: F

Nested if-else

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

There is a cleaner way to write this

***Special Case**

Multi-condition if-else with style

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Multi-condition if-else with style

```
if(grade >= 90) {
    cout << "A";
} else {
    if(grade >= 80) {
        cout << "B";
    } else {
        if(grade >= 70) {
            cout << "C";
        } else {
            if(grade >= 60) {
                cout << "D";
            } else {
                cout << "F";
            }
        }
    }
}
```

Style

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

Order matters

```
if(grade >= 60) {
    cout << "D";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 90) {
    cout << "A";
} else {
    cout << "F";
}
```

Order matters

```
if(grade >= 60) {
    cout << "D";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 90) {
    cout << "A";
} else {
    cout << "F";
}
```

Input: grade = 77
Output: ??

Order matters

```
if(grade >= 60) {
    cout << "D";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 90) {
    cout << "A";
} else {
    cout << "F";
}
```

Input: grade = 77
Output: ??

Order matters

```
if(grade >= 60) {
    cout << "D";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 90) {
    cout << "A";
} else {
    cout << "F";
}
```

Input: grade = 77
Output: D

`switch`

- Another selection technique to control the flow of execution of a program
- Similar to the special case of if-else if-else

- Why?

switch? Less typing, fewer errors

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

① Lots of typing the same variable name

- Chance for spelling error

switch? Less typing, fewer errors

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

① Lots of typing the same variable name

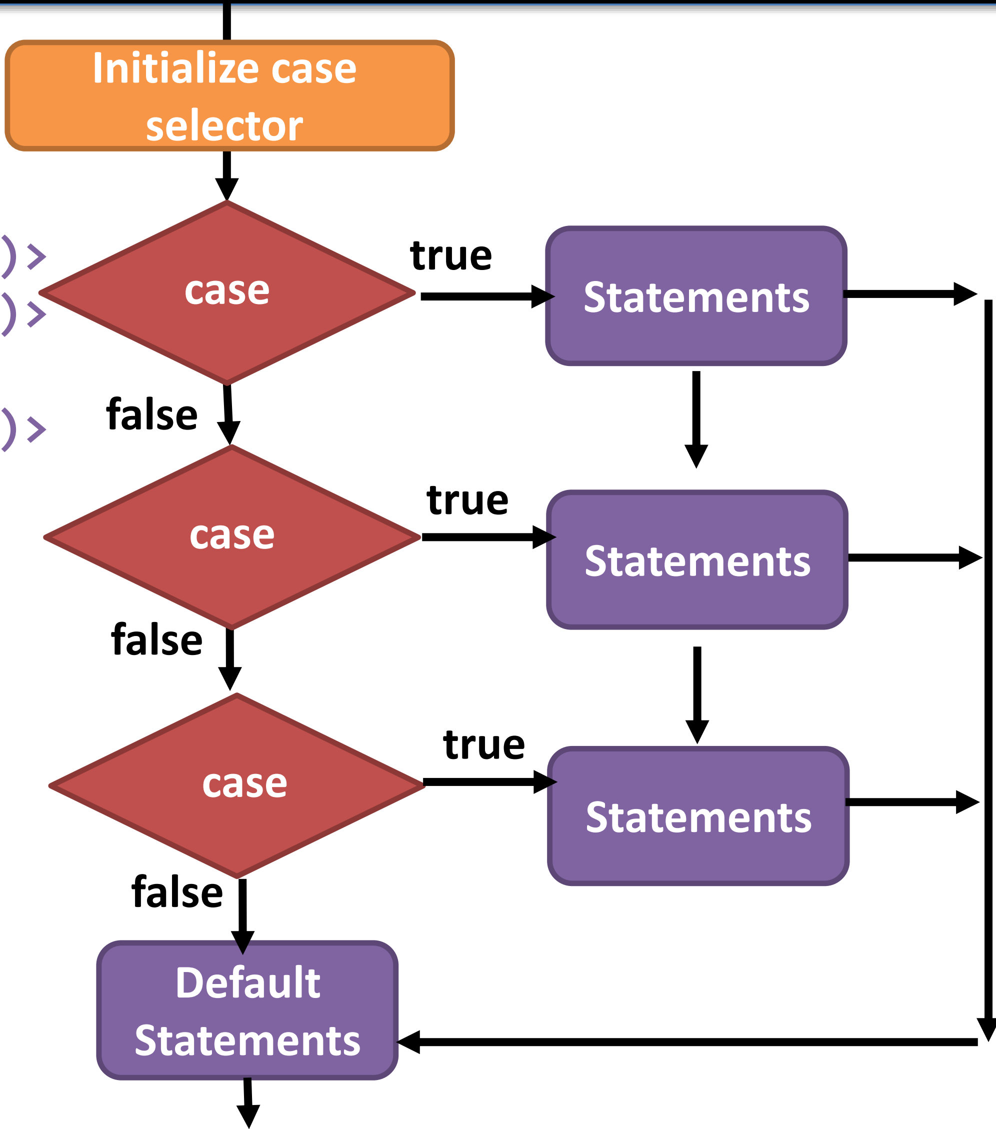
- Chance for spelling error

② Lots of typing the same comparison

- How hard would it be to find a failing test case for and debug a missing =?

switch syntax

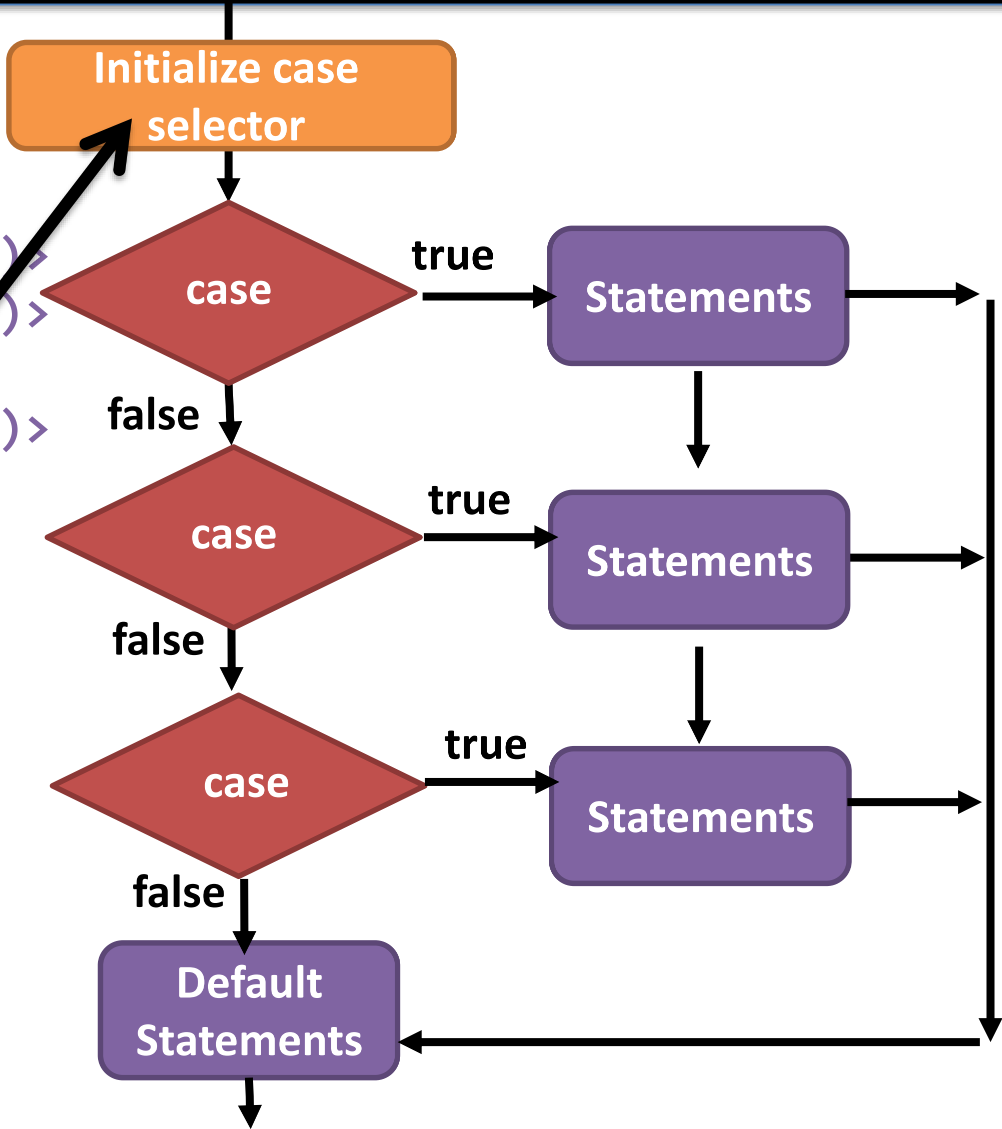
```
switch( <case-selector-expr> )
{
    case <label1> : <statement(s)>
    case <label2> : <statement(s)>
    …
    case <labelN> : <statement(s)>
    default: <statements(s)>
}
```



switch syntax

```
switch( <case-selector-expr> )
{
    case <label1> : <statement(s)>
    case <label2> : <statement(s)>
    …
    case <labelN> : <statement(s)>
    default: <statements(s)>
}
```

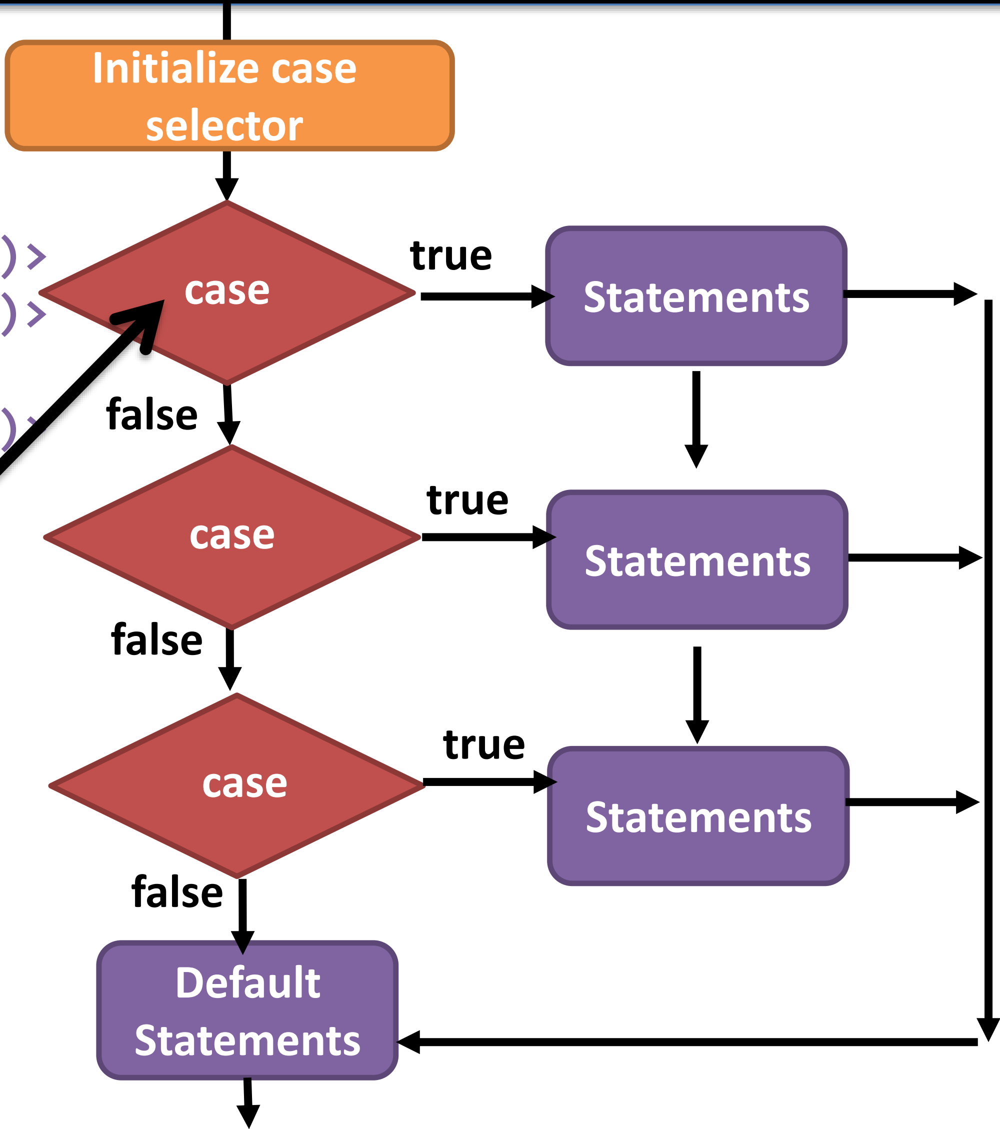
int or char



switch syntax

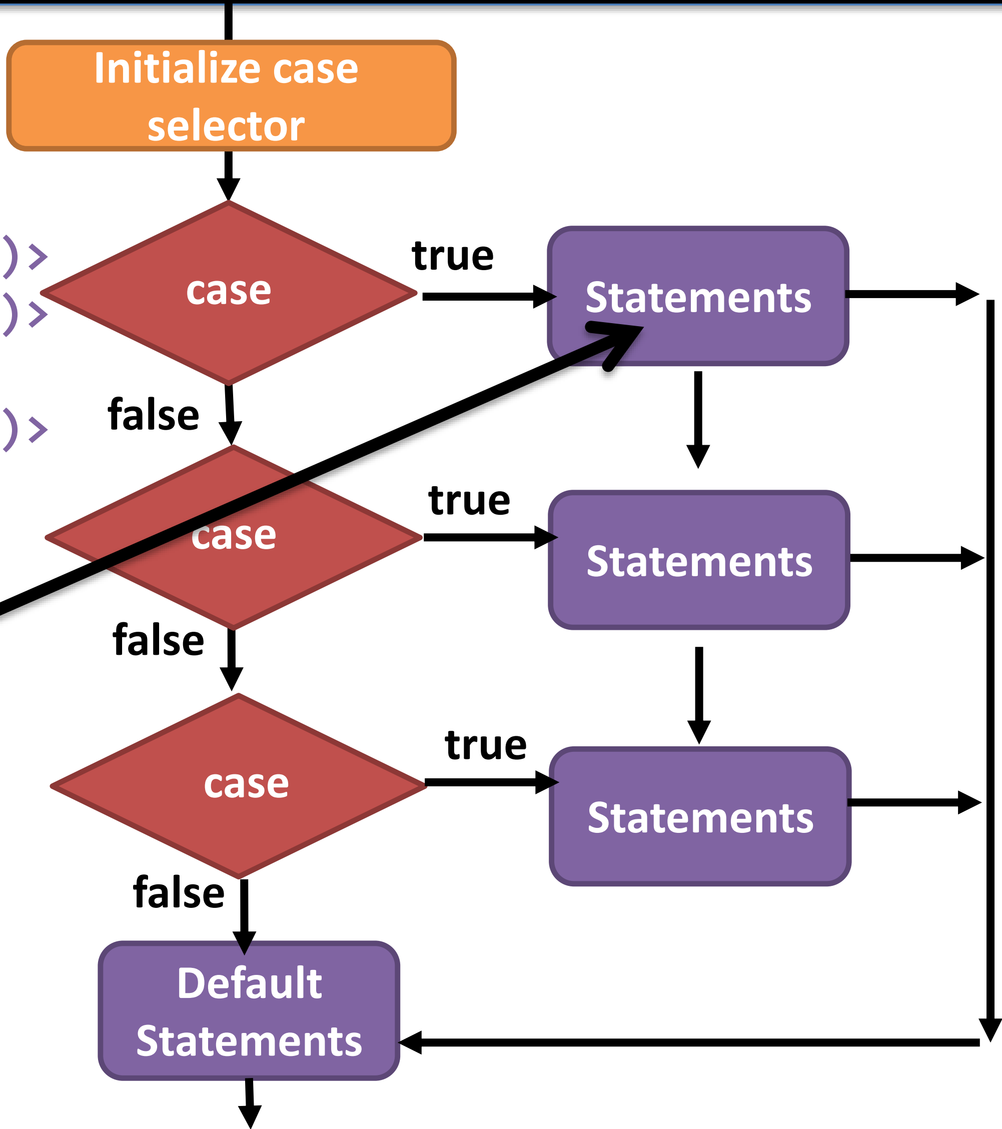
```
switch( <case-selector-expr> )
{
    case <label1> : <statement(s)>
    case <label2> : <statement(s)>
    …
    case <labelN> : <statement(s)>
    default: <statements(s)>
}
```

Integer literal, e.g., 7
Unique



switch syntax

```
switch( <case-selector-expr> )
{
    case <label1> : <statement(s)>
    case <label2> : <statement(s)>
    …
    case <labelN> : <statement(s)>
    default: <statements(s)>
}
```

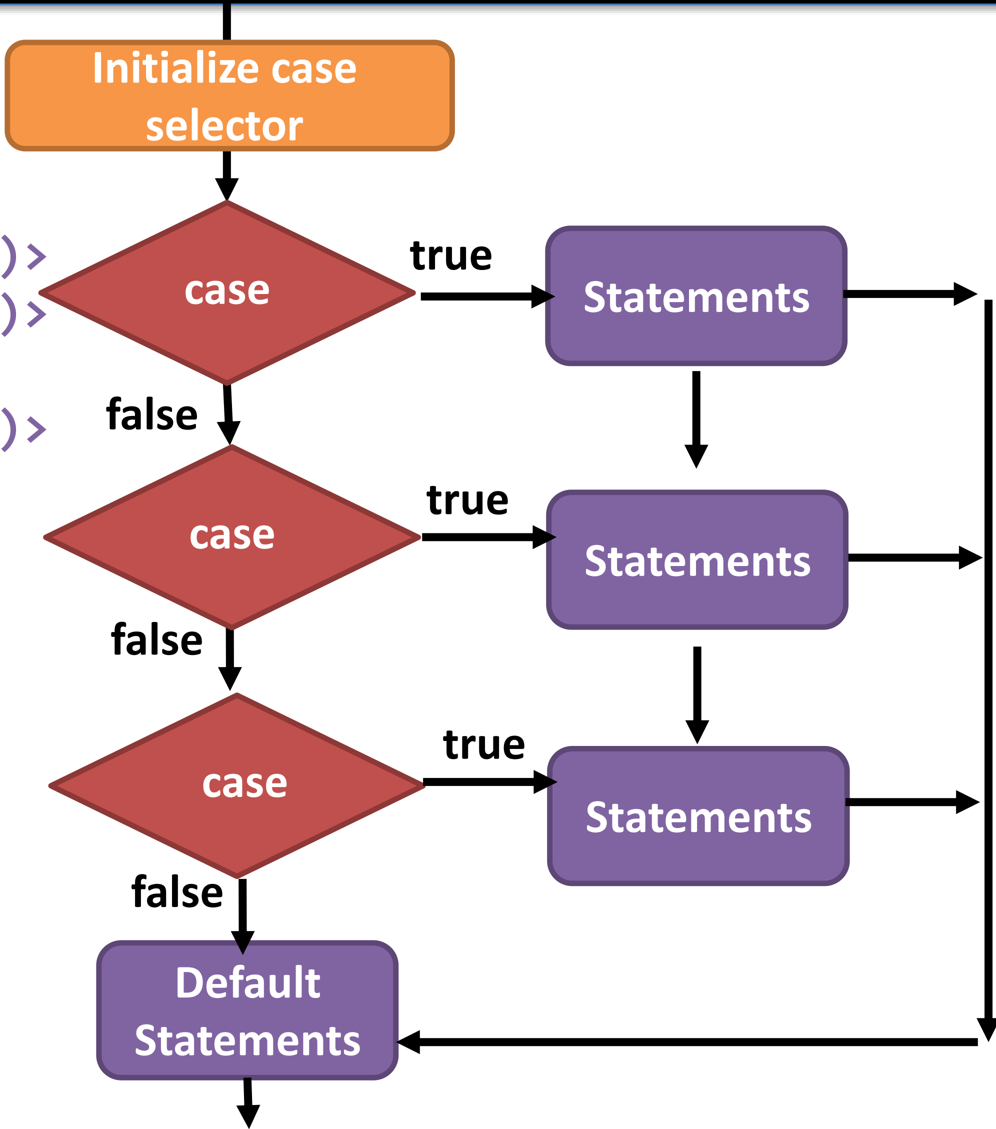


Zero or more
Any valid statement
break or fall through

switch syntax

```
switch( <case-selector-expr> )
{
    case <label1> : <statement(s)>
    case <label2> : <statement(s)>
    …
    case <labelN> : <statement(s)>
    default: <statements(s)>
}
```

If no case matches
Always have this



switch Example

```
int x;
...
switch(x) {
case 1:
    cout << "x is 1";
    break;
case 2:
    cout << "x is 2";
    break;
default:
    cout << "x is unknown";
}
```

if-else if-else vs. switch

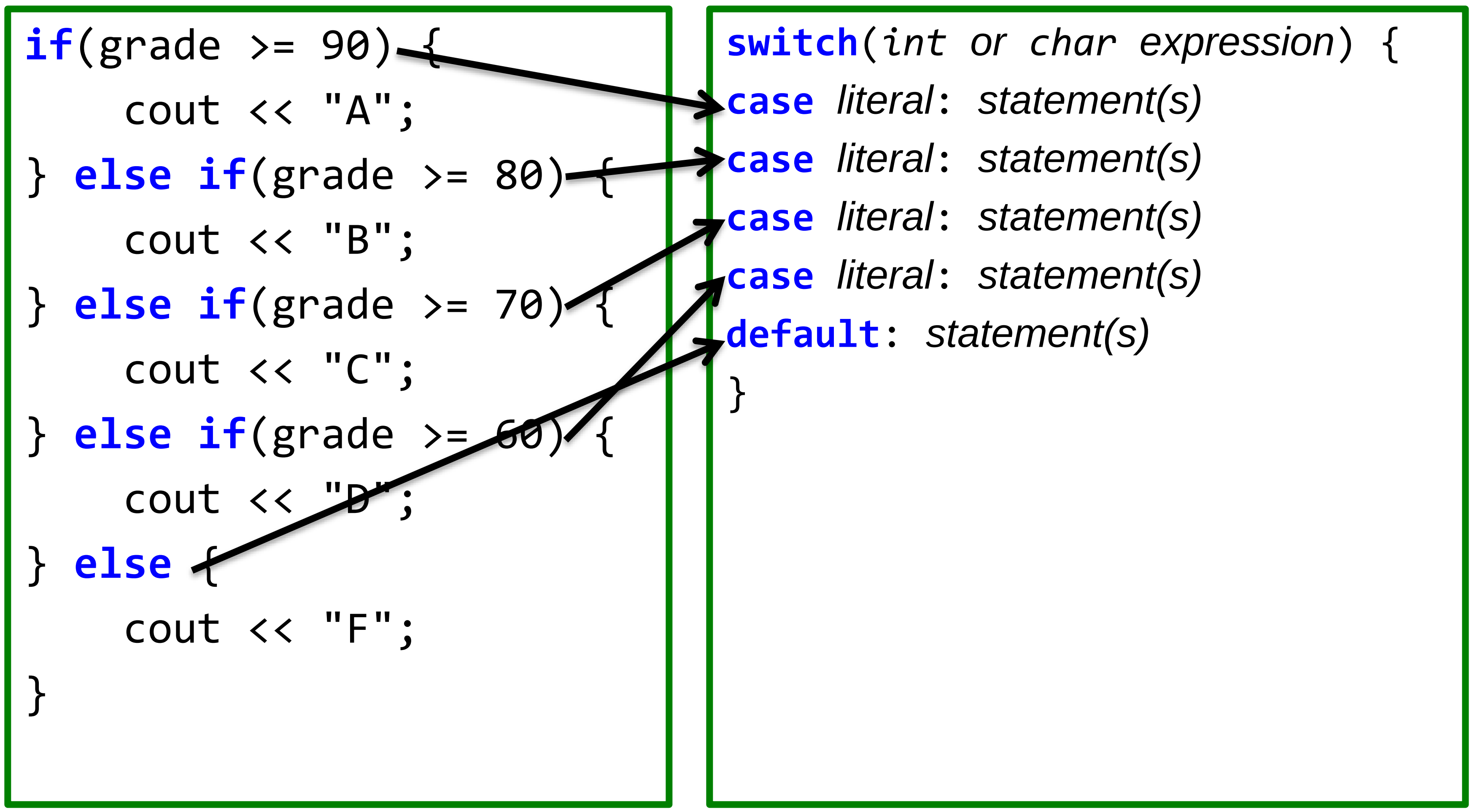
if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(int or char expression) {
case …
default: statement(s)
}
```

if-else if-else vs. switch



if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(int or char expression) {
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
default:  statement(s)
}
```

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(int or char expression) {
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
default:  statement(s)
}
```

int or char

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
case literal:  statement(s)
default:  statement(s)
}
```

int or char

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case literal: statement(s)
case literal: statement(s)
case literal: statement(s)
case literal: statement(s)
default: statement(s)
}
```

Integer literal, e.g., 7
Unique

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case 6:  statement(s)
case 7:  statement(s)
case 8:  statement(s)
case 9:  statement(s)
default:  statement(s)
}
```

Integer literal, e.g., 7
Unique

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case 6: statement(s)
case 7: statement(s)
case 8: statement(s)
case 9: statement(s)
case 10: statement(s)
default: statement(s)
}
```

i>Clicker #8

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case 6: statement(s)
case 7: statement(s)
case 8: statement(s)
case 9: statement(s)
```

What should I put for statement(s) for case 6?

A) nothing
B) `cout << "D";`
C) `cout << "F";`
D) `cout << "D"; break;`

i>Clicker #8

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

switch

```
switch(grade / 10) {
case 6:
    cout << "D";
    break;
case 7:  statement(s)
```

What should I put for statement(s) for case 6?

A) nothing
B) cout << "D";
C) cout << "F";
D) cout << "D"; break;

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    cout << "A";
} else if(grade >= 80) {
    cout << "B";
} else if(grade >= 70) {
    cout << "C";
} else if(grade >= 60) {
    cout << "D";
} els
    c
}
```

switch

```
switch(grade / 10) {
case 6:
    cout << "D";
    break;
case 7:
    cout << "C";
    break;
case 8:
    cout << "B";
    break;
case 9:
case 10:
    cout << "A";
    break;
default: statement(s)
}
```

Zero or more
Any valid statement
break or fall through

if-else if-else vs. switch

if-else if-else

```
if(grade >= 90) {
    c
} els
    c
} els
    c
} else if(grade >= 60) {
    cout << "D";
} else {
    cout << "F";
}
```

Always have default
Zero or more
Any valid statement
break or fall through

switch

```
switch(grade / 10) {
case 6:
    cout << "D";
    break;
case 7:
    cout << "C";
    break;
case 8:
    cout << "B";
    break;
case 9:
case 10:
    cout << "A";
    break;
default: cout << "F";
}
```

if-else if-else vs. `switch`

- Efficiency
  - `switch` typically runs faster
  - Not a big difference with modern compilers
- Generality
  - switch syntax limits applicability
    - (many things that **`if`** can do that **`switch`** can't)
    - can't `switch` on `double`, `string`, ranges (e.g., 1.0 thru 1.7), …
- Readability
  - `switch` table format very clear
- Error tradeoff
  - Missing `break` vs. spelling/copy-paste