1. char pet_type; // C for cat
20  ...
30  cout << "Before if\n";

General Form:
35  // if pet_type is equal to 'C'
40  if (pet_type == 'C')
50  cout << "Cat\n"; // Indentation
60  cout << "After if\n";

Condition or Logical Expression: expression that is either true or false
2. General Form:

```cpp
if (condition)  
    statement; // True path
else  
    statement; // False path
```

```
cout << "Before if\n";
if (pet_type == 'C')  // Note
    cout << "Cat\n";  // indentation
else
    cout << "Dog\n";
```

cout << "After if\n";

The false path is optional.

The if is sometimes known as the **Selection Construct.**
3.  
130  char pet_type;
140  int daily_rate;
150  ...
160  if (pet_type == 'D')
170  {
180     cout << "Dog\n";  // true path
190     daily_rate = 30;
200  }
210  else
220  {
230     cout << "Cat\n";  // false path
240     daily_rate = 20;
250  }

**Compound Statement or Block:** Two or more statements enclosed in braces

4.  
**Review:**

**Condition or Logical Expression:** expression that is either true or false

**Relational Expression:** a condition that states a relationship between 2 data items

Syntax of Relational Expression:
  data-item-1  relational operator  data-item-2

**Relational Operators**  |  **Meaning**
--- | ---
== | is equal to
!|= | is not equal to
<  | is less than
<= | is less than or equal to
>  | is greater than
>= | is greater than or equal to
Examples of relational expressions:

Assume:

```c
char pet_type;
int a, b, c, d;
double x, y, z;
```

<table>
<thead>
<tr>
<th>data-item-1</th>
<th>relational operator</th>
<th>data-item-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>pet_type</td>
<td>==</td>
<td>'C'</td>
</tr>
<tr>
<td>pet_type</td>
<td>==</td>
<td>'D'</td>
</tr>
<tr>
<td>x</td>
<td>&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>x</td>
<td>&gt;=</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>&lt;</td>
<td>y + z</td>
</tr>
<tr>
<td>a + b</td>
<td>&lt;=</td>
<td>24</td>
</tr>
<tr>
<td>a + b</td>
<td>!=</td>
<td>c / d</td>
</tr>
</tbody>
</table>

**Relational Operators:**

- `<`  `<=`  `>`  `>=`  Higher Precedence
- `==`  `!=`  Lower Precedence

Arithmetic operators have a higher precedence than relational operators.

Can compare:

- variable with constant  ➔ Most common form
- variable with variable
- variable with expression
- expression with constant
- expression with expression

Should compare like types, that is, int with int, double with double, etc.
1. Review **Condition**: expression that is either true or false

**Logical Operators**: Operators that combine simple conditions into compound conditions (more complex conditions)

2. **&& Logical AND**: Binary logical operator requiring both operands to be true for the compound condition to be true

```cpp
int grade; // Valid range is 0 to 100, inclusive
...
if (grade >= 0 && grade <= 100)
    cout << "Valid test grade\n";
else
    cout << "Invalid test grade\n";
```

For example:

<table>
<thead>
<tr>
<th>grade</th>
<th>grade</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>80</td>
<td>T</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>F</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>F</td>
</tr>
</tbody>
</table>

3. **|| Logical OR**: Binary logical operator requiring at least one or both operands to be true for the compound condition to be true

```cpp
int age;
char gender;
...
if (age > 50 || gender == 'F')
```

For example:

<table>
<thead>
<tr>
<th>age</th>
<th>gender</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>25</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>60</td>
<td>M</td>
<td>T</td>
</tr>
<tr>
<td>25</td>
<td>M</td>
<td>F</td>
</tr>
</tbody>
</table>
4. **Logical NOT**: Unary logical operator that reverses (negates) the truth value

There are only two truth values: **true** and **false**

<table>
<thead>
<tr>
<th>Logical NOT</th>
<th>Equivalent Simple Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>!true</td>
<td>false</td>
</tr>
<tr>
<td>!false</td>
<td>true</td>
</tr>
</tbody>
</table>

```c
80 int x;
81 // Compound condition
82 if (!(x == 5)) if (x != 5)
```

- a. !(4 == 5) 4 ! = 5 Assume x contains 4
  - !(F) T
  - F

- b. !(5 == 5) 5 != 5 Assume x contains 5
  - !(T) F

5. Precedence:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Logical NOT</td>
</tr>
<tr>
<td>* / %</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td>+ -</td>
<td>Relational Operators</td>
</tr>
<tr>
<td>== !=</td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
</tr>
</tbody>
</table>

if (a == b || c > d && e != f) is equivalent to if (a == b || (c > d && e != f))

6. Let a and b represent any condition, for example, a can represent (age <= 25)

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

7. if (a == b && c > d && e != f && g < h && i >= j && k <= l)