5a. Branch statements

Topic: branch statements

• if
• if ... else
• Compound statements
• Relational operators
• Boolean variables
• Logical operators
• switch

Using if to validate input

```cpp
void main() {[age1.cpp]
    cout << "Enter your age: ";
    int age;
    cin >> age;
    if (age < 0)
        cout << "Invalid age\n";
}
```

• An if statement consists of the keyword if, a Boolean expression (condition) in parentheses, and one statement that is executed if the condition is true

if with compound statement

```cpp
int age;
    cin >> age;
if (age < 0)
{
    cout << "Invalid age\n";
    cin >> age;
}
```

• When multiple statements execute conditional on an if, they must be part of a compound statement

Omitting braces here would change meaning!

Compound statements and scope of access

```cpp
void main() {[expdate.cpp]
    int curr_year = 1998, expir_date = 2000;
    if (curr_year <= expir_date)
    {
        cout << "Amount of purchase: ";
        double amount = 0.0;
        cin >> amount;
    }
    cout << amount;
}
```

• The scope of access of an identifier is the block (compound statement) where it is declared
• For code above to work, put cout within braces

else saves the result of a test

```cpp
cout << "How many CDs? ";
    int quantity;
    cin >> quantity;
    if (quantity >= 5)
        cout << "Discount 10\n";
    if (quantity < 5)
        cout << "No discount\n";
```

Equivalent branch statement:

```cpp
if (quantity >= 5)
    cout << "Discount 10\n";
else
    cout << "No discount\n";[discount.cpp]
```

Tip: Check any file-opening operation before proceeding

```cpp
char line[80];
    ifstream infile(“X.TXT”, ios::nocreate);
if (infile.good())
    infile.getline(line,79);
else
    cout << “X.TXT not found.” << endl;
[readline.cpp]
```
What is the output, on inputs $a = 3$, $b = 5$?

```cpp
void main()
{
  int a,b;
  cin >> a >> b;
  if (a > 2)
    if (b < 4)
      cout << "OK";
    else
      cout << "a <= 2";
  cout << "Done";
}
```

• The code above has a logic error. What is it?

The pairing rule with `else`:

Each `else` is paired with the most recent unpaired `if` in the same scope.

*Example* (misleading code):

```cpp
if (test1)
  if (test2)
    statement1();
else
  statement2();
(If test2 fails, statement2 executes.)
```

Relational operators

equal-to $==$  not-equal-to $!=$
greater $>$  less-than-or-equal $<=$
less-than $<$  greater-or-equal $>=$

• Each operator has a complement

• Tip: don’t compare *floats or doubles* for equality

• Can’t compare C-style strings with any relational operators; use `strcmp`

Standard functions and `if`

```cpp
char input[20];
char name[20];

// declared in ctype.h
if (isdigit(input[0]))
  cout << "Input starts with digit";

// declared in string.h
if (strcmp(name,"M") > 0)
  cout << "Name is in M..Z";

if (! strcmp(command,"quit"))
  cout << "Command is 'quit'";
```

== is a relational operator; = is not

```cpp
void main()
{
  cout << "Enter your age: ";
  int age;
  cin >> age;  // Watch out!
  if (age = 0)
    cout << "Invalid\n";
}
```

• Assignment as an `if` condition is valid syntax
  but usually logic error

• With assignment of constant, in effect no test is performed

A Boolean variable (flag) stores a truth value

may be replaced by `int` a flag

• `bool` invalid = `(age < 0)`;

• `bool` is a standard ANSI/ISO C++ type with a range of values `{false, true}` (0, 1)

• Boolean variables hold values for later use
5a. Branch statements

### Logical operators form

**Boolean expressions**

<table>
<thead>
<tr>
<th>Operation</th>
<th>ANSI</th>
<th>C++</th>
<th>C Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>not</td>
<td>!</td>
<td>(price &gt; cost)</td>
</tr>
<tr>
<td>Conjunction</td>
<td>and</td>
<td>&amp;&amp;</td>
<td>a &gt; b &amp;&amp; b &gt; c</td>
</tr>
<tr>
<td>Disjunction</td>
<td>or</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Nested if's may express conjunction too:
  
  ```
  if (age > 0)
    if (age < 120)
      cout << "Valid age";
  
  The above is equivalent to
  ```

  ```
  if (age > 0 && age < 120)
    cout << "Valid age";
  ```

### Using logical operators

```cpp
void main()
{
  cout << "Enter 3 integers: ";
  int a, b, c;
  cin >> a >> b >> c;
  if (a == b && b == c)
    cout << "They're the same!" << endl;

  cout << "Enter your age: ";
  int age;
  cin >> age;
  bool impossible = (age < 0 || age > 120);
  if (!impossible)
    cout << "Thank you" << endl;
}
```

### Truth tables

<table>
<thead>
<tr>
<th>a</th>
<th>!a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

| a | b | a || b |
|---|---|-----|
| 0 | 0 | 0   |
| 0 | 1 | 1   |
| 1 | 0 | 0   |
| 1 | 1 | 1   |

### Exercise all paths in testing a module

- To test a program containing an `if` statement, you must use test data that will cause a `true` value and a `false` value to appear as the result of the `if` condition.
- With nested if's, fully exercising your code (getting test results that reflect execution of each statement in program) may require 4 tests, 8, etc.

### Multiway branches

**Nested if:**

```cpp
if (age == 1) cout << "Baby";
else if (age == 2) cout << "Toddler";
else if (age == 3) cout << "Preschooler";
else cout << "Other";
```

### The switch statement: guidelines

- Appropriate for mutually exclusive cases
- Test for exact matches, not ranges of values
- Selector should be an `int` or `char`
- Case labels may be stacked
- Don’t forget to use `break`!
- `default` triggers processing when no case label matches the selector
5a. Branch statements

Stacking case labels

```c
#include <ctype.h>
#include <iostream>

void main()
{
    char grade;
    cin >> grade;
    switch (toupper(grade))
    {
        case 'A':
            cout << "Honors" << endl; break;
        case 'B':
            cout << "Passing" << endl; break;
        case 'C':
            cout << "Failing" << endl; break;
        default:
            cout << "Invalid input" << endl;
    }
}
```

Syntax for branches

A branch statement consists of an `if` or `switch` statement that is followed by one or more alternative statements, each of which is preceded by a `case` label.

The subordinate statement in a switch statement is normally compound, with case labels, alternative statements, breaks, and a single optional default label.

Discussion problems

1. Write a program that accepts two floating-point numbers and displays the larger one.
2. Write a program that accepts
   (a) three numbers and displays the largest;
   (b) four numbers;
   (c) five numbers
3. Write a program that prompts for two integers and displays their quotient; show an error message if the divisor is 0.
4. Write a program that allows the user to input any two real numbers and then choose one of the four operations +, -, *, or /. Display the appropriate computed result. Use a switch statement to select an operation.