Notes part 6

Exploring C++20. Ch. 12. Conditions and Logic

• `bool` defaults as `int` in `cout, string` if with `format` (p.77)
– use `std::boolapha` for bool as text string in `cout`
– does not spill over from `function` to caller code
– disable with `std::noboolalpha`
– use `(int)false` to get numerical value with `format`

```
void testBoolNumerisk() {
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "Bool", "Numerical");
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "true", static_cast<int>(true));
  std::cout << std::format("* {:^9} * {:^14} *\n", "false", static_cast<int>(false));
  std::cout << std::format("**{:^9}***{:^14}**\n\n", border1, border2);
}

void testBoolAlphaNumerisk() {
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "Bool", "Alphanumerical");
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "true", true);
  std::cout << std::format("* {:^9} * {:^14} *\n", "false", false);
  std::cout << std::format("**{:^9}***{:^14}**\n\n", border1, border2);
}

void testConditional() {
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "If", "Output");
  std::cout << std::format("**{:^9}***{:^14}**\n", border1, border2);
  std::cout << std::format("* {:^9} * {:^14} *\n", "true", (true ? "true" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "false", (false ? "false" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "42", (42 ? "42" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "0", (0 ? "0" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "42.4242", (42.42 ? "42.42" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "0.0", (0.0 ? "0.0" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "-0.0", (-0.0 ? "-0.0" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "-1", (-1 ? "true" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "'\\0'", ('\0' ? "'\\0'" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "'\\1'", ('\1' ? "'\\1'" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "\"1\"", ("1" ? "\"1\"" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "\"hello\"", ("hello" ? "\"hello\"" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "std::cout", (std::cout ? "std::cout" : "(no output)"));
  std::cout << std::format("* {:^9} * {:^14} *\n", "cin:", (std::cin ? "cin" : "(no output)"));
  std::cout << std::format("**{:^9}***{:^14}**\n\n", border1, border2);
}
```

• in conditional statements *zero values*, *empty strings*, and *invalid streams* are `false`