Odds and endls

Comp Sci 1570 Introduction to C++

Outline

Conditional, Arithmetic if, or ternary operator

If/else statements in the following form:

```
if (condition)
    expression;
else
    other_expression;
```

can be rewritten as:

```
(condition) ? expression : other_expression;
```

Outline

The C++ syntax for the conditional operator is:


```
expression1 ? expression2 : expression3
```

- expression1 will first be evaluated as a Boolean expression: true or false.
- If it is true, the return value of this operator is expression2. Otherwise (false), expression3 is returned.
- The types of expression2 and expression3 must be identical.

Outline

1. Conditional operator
   - Syntax
   - Examples
2. Input and Output
   - Streams
   - cout
   - cin
3. Escape sequences
   - "\n" and endl
   - Other escape sequences

Examples


```
7==5 ? 4 : 3 // evaluates to 3
7==5+2 ? 4 : 3 // evaluates to 4
5>3 ? a : b // evaluates to the value of a
a>b ? a : b // evaluates to greater, a or b.
```

- The conditional operator (?:) (also known as the "arithmetic if" or conditional operator) is C++'s only ternary operator (it takes 3 operands).
- Because of this, it's also sometimes referred to as the "ternary operator"

C++ examples

- check out the code

Outline

1 Conditional operator
Syntax
Examples

2 Input and Output
Streams
cout
cin

3 Escape sequences
"\n" and endl
Other escape sequences

Outline

1 Conditional operator
Syntax
Examples

2 Input and Output
Streams
cout
cin

3 Escape sequences
"\n" and endl
Other escape sequences

Streams

- C++ uses a convenient abstraction called streams to perform input and output operations in sequential media such as the screen, the keyboard or a file.
- A stream is an entity that a program can either insert or extract characters to/from.
- There is no need to know details about the physical media associated with the stream or any of its internal specifications.
- All we need to know is that streams are a source/destination of characters, and that these characters are provided/accepted sequentially (i.e., one after another).

Outline

1 Conditional operator
Syntax
Examples

2 Input and Output
Streams
cout
cin

3 Escape sequences
"\n" and endl
Other escape sequences

cout

- The predefined object cout is an instance of ostream class.
- The cout object is said to be "connected to" the standard output device, which usually is the display screen.
- The cout is used in conjunction with the stream insertion operator, which is written as <<
- The << operator inserts the data that follows it into the stream that precedes it.

Outline

1 Conditional operator
Syntax
Examples

2 Input and Output
Streams
cout
cin

3 Escape sequences
"\n" and endl
Other escape sequences

cin

- The predefined object cin is an instance of istream class.
- In most program environments, the standard input by default is the keyboard, and the C++ stream object defined to access it is cin.
- For formatted input operations, cin is used together with the extraction operator, which is written as >> (i.e., two "greater than" signs).
- This operator is then followed by the variable where the extracted data is to be stored.
- The cin object is said to be attached to the standard input device, which usually is the keyboard.

Summary

New programmers often mix up:
std::cin, std::cout, <<, and >>

- std::cin and cout always go on the left-hand side of the statement.
- std::cout is used to output a value (cout = output)
- std::cin is used to get an input value (cin = input)
- << is used with std::cout, and shows the direction that data is moving from the r-value to the console. std::cout << 4 moves the value of 4 to the console
- >> is used with std::cin, and shows the direction that data is moving from the console into the variable. std::cin >> x moves the value from the console into x

Streams

- Abstractly, a **stream** is just a sequence of characters that can be accessed sequentially.
- **Input streams** are used to hold input from a data producer, such as a keyboard, a file, or a network.
  - For example, the user may press a key on the keyboard while the program is currently not expecting any input. Rather than ignore the users key-press, the data is put into an input stream, where it will wait until the program is ready for it.
- **Output streams** are used to hold output for a particular data consumer, such as a monitor, a file, or a printer.
  - For example, when writing data to an output device, the device may not be ready to accept that data yet – the printer may still be warming up when the program writes data to its output stream.
  - The data will sit in the output stream until the printer begins consuming it.

Outline

1 Conditional operator
Syntax
Examples
2 Input and Output
Streams
cout
cin
3 Escape sequences
"\n" and endl
Other escape sequences

Outline

"\n" and endl

- To insert a line break, a new-line character can be inserted at the exact position the line should be broken.
- In C++, a new-line character can be specified as \n (i.e., a backslash character followed by a lowercase n).

```
cout << "First sentence.\n";
cout << "Second sentence.\nThird sentence.";

cout << "First sentence." << endl;
cout << "Second sentence." << endl;
cout << "Third sentence." <<endl;
```

Prints:

First sentence.
Second sentence.
Third sentence.

Flushing buffers

- When output is streamed from the CPU to the screen, it is usually sent to a "buffer" a set of memory registers that hold the output temporarily while the CPU has "better things to do".
- Streaming output to the screen is a relatively slow process for the CPU, thus preventing it from doing more useful work.
- When the CPU has time to send that info to the screen, it will "flush" that buffer.
- The "manipulator" endl is streamed after the literal constant "hello" and it will flush the buffer before the CPU can do anything else no additional computing takes place until the output buffer is emptied.
- The stream's buffer (if any) is flushed with endl, which means that the output is requested to be physically written to the device, if it wasn't already.

Summary

- Both '\n' and std::endl will move the cursor to the next line.
- In addition, std::endl will also ensure that any queued output is actually output before continuing.
- So when should you use '\n' vs std::endl?
  - **Use std::endl** when you need to ensure your output is output immediately (e.g. when writing a record to a file, when updating a progress bar, or when **debugging**). Note that this may have a performance cost, particularly if writing to the output device is slow (e.g. when writing a file to a disk).
  - **Use '\n'** in other cases, for example in faster production code.

endl

- Check out code

Outline

1 Conditional operator
Syntax
Examples

2 Input and Output
Streams
cout
cin

3 Escape sequences
"\n" and endl
Other escape sequences

Other escape sequences

Name	Symbol	Meaning
Alert	\a	Makes an alert, such as a beep
Backspace	\b	Moves the cursor back one space
Formfeed	\f	Moves the cursor to next logical page
Newline	\n	Moves cursor to next line
Carriage return	\r	Moves cursor to beginning of line
Horizontal tab	\t	Prints a horizontal tab
Vertical tab	\v	Prints a vertical tab
Single quote	\'	Prints a single quote
Double quote	\"	Prints a double quote
Backslash	\\	Prints a backslash
Question mark	\?	Prints a question mark
Octal number	\(number)	Translates into char represented by octal
Hex number	\x(number)	Translates into char represented by hex number

Other escape sequences

- Check out code