Informatik - AS19

Exercise 3: Boolean expressions & Basic loops

Handout: 30. Sep. 2019 06:00

Due: 7. Okt. 2019 18:00

Task 1: Expression Evaluation

Open Task

This task is a text based task. You do not need to write any program/C++ file: the answer should be written in main.md *(and might include code fragments if questions ask for them).*

Task

Which of the following expressions evaluate to true, which to false?

1. `3 >= 3`
2. `true || false && false`
3. `(true || false) && false`
4. `3 > (1 < true)`
5. `8 > 4 > 2 > 1`
6. `2 < a < 4` (a is a variable of type int)

Task 1.5: two-complement integer representation (Optional)

Open Task

This task is a text based task. You do not need to write any program/C++ file: the answer should be written in main.md *(and might include code fragments if questions ask for them).*

Task

Now we assume an architecture using 4-bit arithmetics using two's complement representation of integers. Convert the following binary numbers to signed decimal numbers (`0b` is a prefix that indicates binary encoding):

1. `0b0001`
2. `0b0101`
3. `0b0111`
4. `0b1000`
5. `0b1010`
6. `0b1111`

Task 2: From Natural Language to C++

Open Task

This task is a text based task. You do not need to write any program/C++ file: the answer should be written in main.md *(and might include code fragments if questions ask for them).*

Task

Translate the following natural language expressions to C++ expressions. Assume that all the variables are non-negative numbers or boolean (of value `true` or `false`).

Example: a is greater than 3 and smaller than 5. $\Longrightarrow$ **Solution**: `a > 3 && a < 5`

1. a greater than b and the difference between a and b is smaller than 15.
2. a is an even natural number greater than 3.
3. a is at most 5 times greater than b and at least 5 times greater than c.
4. Either a and b are both false or c is true, but not both.
5. a is false and b is zero.

Task 3: From decimal to binary representation

Open Task

Task

Write a program that inputs a natural, i.e., `unsigned int`, number `n` and outputs the binary digits of `n` in the *correct* order (i.e., starting with the most significant bit). Do not output the leading zeros.

Restrictions: you cannot assume that *int* is 32 bits (ie. could be much smaller or much larger) and only the `iostream` standard library header is allowed. No arrays are permitted.

Task 4a: Fibonacci primes

Open Task

Task

Fibonacci numbers are the integers in the following sequence:
$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$. Each number is the sum of the two previous numbers.

Fibonacci primes are Fibonacci numbers that are also prime numbers. Write a program that asks the user for an integer m and then computes and prints all Fibonacci primes between 0 and m (including). Print each number on a new line.

Finally, on a new line print the total number of Fibonacci primes found.

Task 4b: Fibonacci overflow check

Open Task

Task

Fibonacci numbers are the integers in the following sequence:
$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$ Each number is the sum of the two previous numbers.

Fibonacci numbers grow fast, thus they can easily exceed the value range of a 32-bit number. Think of a general way how you can check whether the result of an addition would exceed the range (overflow) of a 32-bit number **without actually performing the addition causing the overflow**.

Write a program that asks the user for an integer n and then prints the first n *Fibonacci numbers*. Print each number on a new line. Use an `unsigned int` (32-bit) to represent the current Fibonacci number. Using the check described above, if calculating the next Fibonacci number would exceed the range representable by an `unsigned int` (32-bit), exit the loop.

Finally, on a new line print the total number of Fibonacci numbers printed x, and the number of Fibonacci numbers requested n in the format: `x of n`.

Restrictions: the program *must not* rely on the knowledge of its final result: in particular, hard-coding the largest 32-bits Fibonacci number, the number of digits that it has or the total number of Fibonacci numbers representable with 32-bits is not allowed. Moreover, it is not allowed to perform the addition that causes an overflow. The violation of these restrictions will result in 0 points, even if tests pass.

Hint: you should know the biggest positive integer you can represent with 32 bits... you are allowed to use this!