ETH zürich

Session 02
Boolean Functions, Control Flow I

Informatik I

Lan Zhang

28.02.2025

Schedule

1. Review of Last Week
2. Main Topics
   - Boolean Expressions
   - For-Loops
3. In-Class Code Examples

Feedback on weekly exercises

1. Expressions

- ++PreIncrement: incremented value returned
- PostIncrement++: current value returned, incremented value at next call

3. Equivalent Resistance:

#1: Avoid floating-point operations by scaling the intermediate results and adjusting the final value to determine the rounding direction.

int rnd = (50 + ((100*(n % d)) / d) / 100; (if n%d < d/2 round down, else up)

#2: Integer division truncates the decimal part, add 1 before the division to ensure any fractional part is accounted for and round up the result.

Boolean Expressions

Conversion:

int → bool
x ≠ 0 → true
x = 0 → false

bool → int
true → 1
false → 0

In conditional statements, int values are converted to bool !

z.B. `int x = true + 3; → x == 4` `(int) bool b = 4; → b == true (bool)`

Short Circuit Evaluation:

```
false && (...) -> false
true || (...) -> true
```

- && and || evaluate the left operand first (left-associative)
- If entire result can be inferred from left expression alone, **the right is never evaluated**!
- `x == 1 || 1/(x-1) < 1 == true`

Boolean Expressions: Examples

Parenthesize:

```
3 < 4 + 1 && 2 < 3
3 == 3 * 2 || 5 <= 2+3 && true

true && false || !false


2 > 3 && 17 - 55 <= ++x % y



int x = 1;
!(1 < 2 && x == 1) + 1
```

Minimization of Boolean Expressions

Simplify the following expressions, where a and b are bool type and n is an int

1. `b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)`
2. `a && !(a == true) || (b == false)`
3. `!(a != false && !(b == false) == false)`

Iterations

For-loops:

loops over a code block until condition is no longer satisfied

```
for (initialisation; condition; expression) {
        statement;
        ...
}
```

```
(i) initialisation statement is executed.

(ii) if condition == true, statement is executed, expression is executed.
```

init and *expr* is optional (can be defined outside)

Go-to Instructions

- `break;` → exits the loop
- `continue;` → skips the rest of the loop and moves on to the next iteration

Loops: Examples

```
bool is_prime_1 (int n){
    for(int d = 2; n % d != 0; ++d);
    return (n==d);
}
bool is_prime_2 (int n){
    for(int d = 2; d <= n/2; ++d){
        if(n % d == 0) return false;
    }
    return true;
}
```

```
void prime_factors(int n){
    assert (n>=2);
    int d = 2;
    while(n > 1 || d < n){
        if (n % d == 0){
            cout << d;
            n /= d;
        } else {
            ++d;
        }
    }
}
```

In-Class Code Examples

Strange Sum: Given a positive natural number, output the sum of all positive odd numbers <= n and not divisible by 5.

Largest Power: Given a positive natural number, calculate the largest power of two (2,4,8,16,...) <= n.

In-Class Code Examples

C++ Debugging

- `std::cerr`
  - prints out debugging information, writes to error output stream instead of standard output stream
- `assert(expr);`
  - requires `#include<cassert>`
  - program is terminated if expression is not true → imposes restrictions on program
  - deactivate with `#define NDEBUG`
- multi-line comments with /* */
- comment out multiple lines with SHIFT + CTRL/CMD + 7
- search code with CTRL/CMD + F

C++ Error Types

- segmentation fault, undefined behavior, timeout, compiler/runtime errors (semantic or syntax errors)