C++ Programming

Logical Operators

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Logical operators

```
4  int main() {
5      int age = 30, salary = 7000;
6
7      bool result = (age > 25) && (salary < 8000);
8      cout<<result<<"\n";
9
10     cout<<(  (age > 25) && (salary > 9000) )<<"\n";
11
12     cout<<(  (age > 35) || (salary < 8500) )<<"\n";
13     cout<<(  (age > 35) || (salary > 9000) )<<"\n";
14
15     return 0;
16 }
```

Console | Problems | Tasks | Properties | Call Graph

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspace
1
0
1
0
```

- && for **AND**
- || for **OR**
- Precedence: && before ||
- Same concepts as truth table, but now in C++
- Using (), we force order

Mixing Logical Operators in C++

```
 3
 4  int main() {
 5      int age = 30, salary = 7000, weight = 110;
 6
 7      cout<<(  (age > 25) && (salary < 8000) && (weight < 150) )<<"\n";
 8      cout<<(  (age > 25) && (salary < 8000) && (weight > 200) )<<"\n";
 9
10      cout<<(  (age > 35) || (salary > 6000) || (weight > 200) )<<"\n";
11
12      cout<<(  (age > 35) && (salary > 6000) || (weight > 200) )<<"\n";
13      cout<<(  (age > 20) && (salary > 6000) || (weight > 200) )<<"\n";
14
15      return 0;
16  }
17
18
```

Console ⊠ Problems Tasks Properties Call Graph Search

<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/eclipse_cpp/ztemp/D

```
1
0
1
0
1
```

() applied first

05_6.cpp

```cpp
#include<iostream>
using namespace std;

int main() {
    int age = 30, salary = 7000, weight = 110;

    // ANDs are evaluated
    cout << ( age > 35 || salary > 6000  && weight > 200) << "\n";

    // () are evaluated FIRST even before some ANDS
    cout << ((age > 35 || salary > 6000) && weight > 200) << "\n";

    return 0;
}
```

Let’s try simplifying

- Let’s simplify this expression T && T && (F || (T && T)) || T
- T && T && (F || **(T && T)**) || T ⇒ **(T && T)** is the simplest (). Its value is T
- T && T && (**F || T**) || T ⇒ (**F || T**) is the simplest (). Its value is T
- T && T && T || T ⇒ No more (). Next is group ands
- **T && T && T** || T ⇒ **T && T && T** is group of ands. Evaluate to T
- T || T. Now final expression is set of conditions ORed ⇒ T

Short-Circuit Evaluation

- Stop evaluating when result is determined (**efficiency**)
- Let say we have an expression
- T || T && (F || (T && T)) || T ⇒ (T && T)
  - Do we really need to evaluate after the first T || <something>
  - No. According to OR table, this is **definitely** TRUE. STOP
- T && T && F && (F || (T && T) && (F || (T && T)))
  - Do we really need to evaluate after the first T && T && F && <something>
  - No. According to AND table, this is **definitely** False. STOP
- Note: In complex expression: some sub-groups are discarded, but still continue evaluating
  - Rule: Logically discard what can be discarded, following C++ precedence rules

Short-Circuit Evaluation

```
int x = 10;

// (x+= 50 > 10) is discarded
x < 100 || (x+= 50 > 10);

// (++x > 10) is discarded
x == 20 && (++x > 50);

// (++x > 10) is discarded, but (x > 0) eval
(x == 20) && (++x > 50) || (x > 0);

// X still 10
// All evaluated
(x == 10) && ((++x > 50) || (x > 0));

// X now 11
```

- Tips
  - Don’t make long expressions
  - Don’t change variables in expressions this way

Coding mistakes

- Writing < = NOT <= (extra spaces)
- Writing & NOT &&
- Writing & & NOT && (extra space)
- Writing | not ||
  - & and | are called bits operators (later topic)
- Writing = not ==
  - = is assignment. == is for comparing
- Writing ! result NOT !result (extra space)
- cout<<x < 5<<"\n";
  - Compiler get confused. Use () ⇒ cout<<(x < 5)<<"\n";
- Imbalanced expression: (T || (T && F) ⇒ (T || (T && F))

Precedence Table

Operators (ordered)	Associativity
++, -- (postfix)	left to right
++, -- (prefix), - (unary)	right to left
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
&&	left to right
||	left to right
= += -= *= /= %=	right to left

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."