

CSE1301
Computer Programming
Lecture 8
Booleans

2

Topics

- Boolean
- Type `int` as Boolean
- Boolean expressions
- Boolean Operators
- Precedence
- Common mistakes

3

Boolean

- A special “type” with only two values:
 - `false` and `true`.
- Used to implement conditions
 - for selection and looping in an algorithm
- Boolean expressions
 - represent statements which are either strictly true or strictly false

4

Boolean Algebra --- History

- George Boole, 1815-1864
 - Initially self-guided studies of languages and philosophy
 - With 16 assistant teacher at private school
 - With 20 opened a school and taught himself mathematics
 - Published in “Cambridge Mathematical Journal” with 24
- The *Mathematical Analysis of Logic* was published in 1847
 - Casts logical reasoning in the form of algebra
 - + is OR, * is AND
 - 0 is FALSE, 1 is TRUE
- *An Investigation of the Laws of Thought* (1854) extends the algebra

5

Type `int` as Boolean

- In C, integers are used as Booleans
- Integer value `0` is `false`.
- Any `non-zero` integer value is `true`.

6

Example: What is the output?

```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    int whatever = 0;

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

7

Example: What is the output?

```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    int whatever = 1;

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

8

Example: What is the output?

```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    int whatever = -100;

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

9

Example: What is the output?

```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    int whatever = 'A';

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

10

Example: What is the output?



```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    int whatever = 0.003;

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

11

Example: What is the output?



Behavior is "undefined."

```
#include <stdio.h>

/* Test some Booleans. */

int main()
{
    float whatever = 0.003;

    if (whatever)
    {
        printf("Is that true, Homer?\n");
    }
    else
    {
        printf("No, Marge.\n");
    }
    return 0;
}
```

12

Boolean Expressions

- ...are either strictly true or strictly false.
- ... can be evaluated to determine their true or falsehood.
- A Boolean expression which evaluates to **true** has integer value **1**; otherwise, **0**.

Remember: any non-zero value is taken interpreted as true

13

Boolean Operators

- ...are used in forming Boolean expressions.
- ...are also known as “logical” operators
- And (&&)
- Or (||)
- Not (!)
- Equality (==)
- Inequality (!=)
- Comparison (<, >, <=, >=)

14

And

- True only if both Boolean arguments are true.

Examples:

```
1 && 2
1 && 0 && -1
healthy && wealthy && wise
```

15


And

- True only if both Boolean arguments are true.

Examples:

not to be confused with “bitwise AND” (&)

```
1 && 2
1 && 0 && -1
healthy && wealthy && wise
```



16

Or

- True if either Boolean argument is true (or both are true).

Examples:

```
1 || 2
11 || 0 || 1
good || bad || ugly
```

17


Or

- True only if either Boolean argument is true (or both are true).

Examples:

not to be confused with “bitwise OR” (|)

```
1 || 2
1 || 0 || 1
good || bad || ugly
```



18

Not


- True only if the single Boolean argument is false.

Examples:


```
! 1
! 0
! happy
```

19

Reminder: Gates



AND Gate



OR Gate

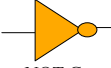
A	B	A AND B	A OR B
0	0		
0	1		
1	0		
1	1		

mult

add

20

Reminder: Gates



NOT Gate

A	NOT A
0	
1	

21

Equality

- True only if both arguments have identical values.

Examples:

1 == 2

1 == 0

42 == 42

truth == beauty

22

Equality

- True only if both arguments have identical values.

Examples:


1 == 2

1 == 0

== 42

truth == beauty

not to be confused with assignment (=)



23

Inequality

- True only if arguments have different values.

Examples:

1 != 2

1 != 0

42 != 42

truth != beauty

24

Comparison

- True only if the specified relationship holds

Examples:

1 < 2

0 > 1

42 <= 42

age >= 18

25

Short-circuiting

- A complex Boolean expression is only evaluated as far as necessary

Examples:



```
1 || 2
0 || 1 || 2
1 && 0 && -1
```

26

Precedence

- Highest to lowest:
 - Brackets
 - Not (!)
 - Comparison (<, >, <=, >=)
 - Equality (==)
 - Inequality (!=)
 - And (&&)
 - Or (||)

Note: The assignment operator (=) is lower in order of precedence than Boolean / Logical operators.

Example:

```
#include <stdio.h>

int main()
{
    int age = 18;
    int haveMoney = 0;
    int haveCard = 1;
    float thirst = 0.31;
    int afterHours = 1;

    int result;

    result = age >= 18 && (haveMoney || haveCard)
        && thirst > 0.3 && ! afterHours;

    printf("%d\n", result);

    return 0;
}
```

bool.c

Common Mistakes

- Using = instead of ==

29

Example:

```
#include <stdio.h>

/* Probably the most common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if (score = 48 || score = 49)
    {
        printf("Almost!\n");
    }

    return 0;
}
```

boolerr1.c

Example:

```
#include <stdio.h>

/* Probably the most common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if (score = 48 || score = 49)
    {
        printf("Almos");
    }

    return 0;
}
```

Usual error message (if any):
"LValue required..."

boolerr1.c


Example:

```
#include <stdio.h>

/* Probably the most common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if (score == 48 || score == 49) 
    {
        printf("Almost!\n");
    }

    return 0;
}
```

boolerr1.c

Common Mistakes

- Using = instead of ==
- Multiple comparisons

33

Example:

```
#include <stdio.h>

/* Another common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if ( 0 < score < 48 )
    {
        printf("Fail\n");
    }

    return 0;
}
```

boolerr2.c

Example:

```
#include <stdio.h>

/* Another common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if ( 0 < score < 48 )
    {
        printf("n");
    }
    0 or 1

    return 0;
}
```

boolerr2.c

Example:

```
#include <stdio.h>

/* Another common C error. */

int main()
{
    int score;
    always 1
    scanf("%d", &score);

    if ( 0 < score < 48 )
    {
        printf("n");
    }
    0 or 1

    return 0;
}
```

boolerr2.c


Example:

```
#include <stdio.h>

/* Another common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if ( 0 < score && score < 48 ) 
    {
        printf("Fail\n");
    }

    return 0;
}
```

boolerr2.c

True and False as Constants

- ‘C’ does not provide constants for TRUE/FALSE (other than 0 and 1)
- You can make use of the **pre-processor**
- Use
 - `#define TRUE 1`
 - `#define FALSE 0`
 whenever you need such constants

38

Exclusive OR

- True if “at most one” of two alternatives is true
- False if neither is true
- False if both are true!
- Define!

A xor B :

Do you need brackets?

39

Two out of three...

- True if and only if exactly two of A,B,C are true
- Define!
- *Is there a better way?*

40

Simplifying & Checking Boolean Expressions

- Use Truth Tables

A && !B || B && !A

A	B	A xor B
0	0	0
1	0	1
0	1	1
1	1	0

A	B	
0	0	
1	0	
0	1	
1	1	

- Transform Expressions using Boolean Algebra

41

Axioms of Boolean Algebra

- Commutative
 - $(A \ \&\& \ B) \Leftrightarrow (B \ \&\& \ A)$
 - $(A \ \|\| \ B) \Leftrightarrow (B \ \|\| \ A)$
- Associative
 - $A \ \&\& \ (B \ \&\& \ C) \Leftrightarrow (A \ \&\& \ B) \ \&\& \ C$
 - $A \ \|\| \ (B \ \|\| \ C) \Leftrightarrow (A \ \|\| \ B) \ \|\| \ C$

42

Axioms of Boolean Algebra (cont)

- Distributive
 - $A \ \&\& \ (B \ \|\| \ C) \Leftrightarrow (A \ \&\& \ B) \ \|\| \ (A \ \&\& \ C)$
 - $A \ \|\| \ (B \ \&\& \ C) \Leftrightarrow (A \ \|\| \ B) \ \&\& \ (A \ \|\| \ C)$
- ... plus some “technicalities”

43

De Morgan's Law

- From the axioms of Boolean algebra it follows that:
 - $\neg (A \parallel B) \Leftrightarrow \neg A \ \&\& \ \neg B$
 - $\neg (A \ \&\& \ B) \Leftrightarrow \neg A \parallel \neg B$
- You can use all rules above to simplify / verify expressions.
- Exercise: simplify $\neg (A \parallel \neg B) \ \&\& \ A$

44

Conditional Expressions

- We can write an expression such that its value depends on a TRUTH value

```
Condition ? Expr2 : Expr3
```

- A ternary operator
- For example:

```
int z,a,b;  
...  
z = (a>b) ? a : b ;
```

45

Summary

- Boolean
- Type **int** as Boolean
- Boolean expressions
- Boolean Operators
- Precedence
- Common mistakes
- Some Boolean algebra

46

Readings

This Lecture:

- King: Section 5.1
- D&D: Sections 2.6, 3.4 – 3.6, 4.10
- Kernighan & Ritchie: 2.6, 2.12

47