CSE 142
Computer Programming I

Complex Conditions

Overview

Concepts this lecture
- Complex conditions
- Boolean operators
- Negating a condition
- Truth tables
- DeMorgan's laws

Complex Conditionals

if I have at least $15 or you have at least $15, then we can go to the movies

if the temperature is below 32 degrees and it's raining, then it's snowing

if it's not the case that it's Saturday or Sunday, then it's a work day

Boolean Operators in C

Complex conditionals often involve words like AND, OR, and NOT, and sometimes TRUE or FALSE

The Boolean operators AND, OR, and NOT have these symbols in C:

&&	||	!
and	or	not

As we know, TRUE and FALSE are not built-in concepts in C. You can define symbols:

```
#define  TRUE    1
#define  FALSE   0
```


Complex Conditionals in C

if I have at least $15 or you have at least $15, then we can go to the movies:

```
if (myMoney>=15.0 || yourMoney>=15.0)
        canGoToMovies = TRUE;
```

if the temperature is below 32 degrees and it's raining, then it's snowing:

```
if (temperature<32.0 && raining) snowing = TRUE;
```


An Example with !

if it's not the case that it's Saturday or Sunday, then it's a work day:

```
weekday = FALSE;
if (!(today==6 || today==7))
        weekday = TRUE;
if (weekday) mustWork = TRUE;
```

Conditional Expressions

Review: Like arithmetic expressions, conditional expressions have a value:

TRUE (non-zero) or FALSE (zero)

values are actually *int*

When using relational (<, ==, etc.) and Boolean (&&, ||, !) operators: TRUE is 1; FALSE is 0

Can be used in int expressions:

```
m = (z >= 0.0) ;
```


Nested *if* vs. AND (&&)

```
if ( age < 25 ) {
    if ( sex == 'M' ) {
        insurance_rate = insurance_rate * 2 ;
    }
}
```

```
if ( (age < 25)   &&   (sex == 'M') ) {
    insurance_rate = insurance_rate * 2 ;
}
```


Precedence of &&, ||, !, >, etc.

High (Evaluate First) Low (Evaluate Last)

! Unary -	* / %	- +	< > <= >=	== !=	&&	||

```
a = 2;
b = 4;
z = (a + 3 >= 5  &&  !(b < 5)) || a * b + b != 7 ;
```



```
z = (a + 3 >= 5  &&  !(b < 5)) || a * b + b != 7
z = (a + 3 >= 5  &&  ! 1) || a * b + b != 7
z = (a + 3 >= 5  &&  0) || a * b + b != 7
z = (5 >= 5  &&  0) || a * b + b != 7
z = (1  &&  0) || a * b + b != 7
z = 0 || a * b + b != 7
z = 0 || 8 + b != 7
z = 0 || 12 != 7
z = 0 || 1
z = 1
1
```

a = 2
b = 4

Negating Conditions

Suppose we want a while loop to terminate as soon as either x is 17 or x is 42

Which is it?

```
while (x!=17 || x!=42) …
while (x!=17 && x!=42) …
```

either way? something else?

Truth tables and DeMorgan's laws give us tools for answering such questions

Truth Tables for && and ||

A "truth table" lists all possible combinations of values, and the result of each combination

P Q	P && Q	P || Q
T T	T	T
T F	F	T
F T	F	T
F F	F	F

P and Q stand for any conditional expressions

Truth Table for NOT (!)

P	!P
T	F
F	T

I-13

NOT (!) Example

```
int high_risk ;
high_risk = (age < 25  &&  sex  ==  'M' ) ;
if ( high_risk ) {      /* Do nothing */
} else {
    printf ( "Cheap rates. \n") ;
}
```

```
if ( ! high_risk ) {
        printf ( "Cheap rates. \n") ;
}
```

P	!P
T	F
F	T

I-14

Equivalence of Complex Expressions

```
if ( ! (age < 25  &&  sex  ==  'M' ) )
        printf ( "Cheap rates. \n") ;
```

is equivalent to

```
if ( age >= 25  ||  sex  !=  'M' ) )
        printf ( "Cheap rates. \n") ;
```

Or is it?

I-15

DeMorgan's Laws

DeMorgan's laws help determine when two complex conditions are equivalent

They state:

! (P && Q)   is equivalent to (!P || !Q)
! (P || Q)   is equivalent to (!P && !Q)

This applies for any Boolean expressions P and Q, which might themselves be complex expressions

I-16

Proof of DeMorgan

Is it really true that !(P&&Q) == (!P || !Q) ?

P	Q	(P&&Q)	!(P&&Q)	!P	!Q	(! P || !Q)
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Exercise: Prove the other law

I-17

Proof of DeMorgan

Is it really true that !(P&&Q) == (!P || !Q) ?

P	Q	(P&&Q)	!(P&&Q)	!P	!Q	(! P || !Q)
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Exercise: Prove the other law

I-18

Solution To a Previous Question

We wanted a while loop to terminate as soon as either x is 17 or x is 42. I.e., loop should terminate if (x==17 || x==42)

So the loop condition is

while (! (x==17 || x==42) …

Using DeMorgan's laws, we can rewrite as

while (x != 17 && x != 42) …

A truth table would show that

while (x != 17 || x != 42)

is wrong!

I-19

Summary

Complex conditions are useful in while loops, for loops, if statements, and even in assignment statements

Operators &&, ||, and ! are part of C

TRUE and FALSE can be #defined

Truth tables and DeMorgan's laws help evaluate complex expressions

I-20