CSE 142
Computer Programming I

Conditionals

Overview

Concepts this lecture

- Conditional execution
- *if* statement
- Conditional expressions
- Relational and logical operators

Related Reading

Read Sections 4.1-4.5, 4.7-4.9

- 4.1: Control structure preview
- 4.2: Relational and logical operators
- 4.3: *if* statements
- 4.4: Compound statements
- 4.5: Example
- 4.7: Nested *if* statements

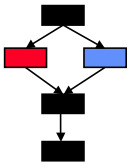
Control Flow

"Control flow" is the order in which statements are executed

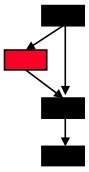
Until now, control flow has been sequential -- the next statement executed is the next one that appears, in order, in the C program

Conditional Control Flow

Choosing which of two (or more) statements to execute before continuing

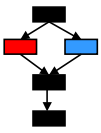


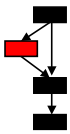
Choosing whether or not to to skip a statement before continuing



Conditional Execution

A conditional statement allows the computer to choose an execution path depending on the value of a variable or expression





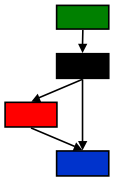
Examples in English

if the withdrawal is more than the bank balance, then print an error

if today is my birthday, then add one to my age

F-7

A More Complete Example



Advance to the intersection

If the light is red

Then stop until it turns green

Proceed through the intersection

F-8

In C: *if* Statement

Critical punctuation

```
if (condition) {
    statement;
}
```

The statement is executed if and only if the condition is true.

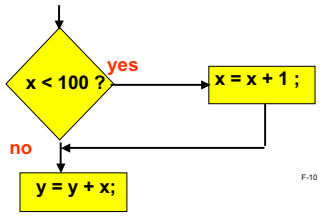
```
if (withdrawalAmount > balance) {
    printf( "Not enough money\n");
}
```

```
if (x < 100) {
    x = x + 1;
}
```

F-9

Conditional Flow Chart

```
if (x < 100) {
   x = x + 1 ;
}
y = y + x;
```



F-10

Conditions

```
if (condition) {
    statement;
}
```

The condition is also called a “logical” or “Boolean” expression

Made up of variables, constants, arithmetic expressions, and the relational operators

Math symbols:	$<$, $\leq$, $>$, $\geq$, $=$, $\neq$
in C:	<, <=, >, >=, ==, !=

F-11

Conditional Expressions

airTemperature > 80.0

98.6 <= bodyTemperature

maritalStatus == 'M'

divisor != 0

Such expressions are used in *if* statements and numerous other places in C.

F-12

Value of Conditional Expressions

What is the value of a conditional expression??

Answer: we think of it as TRUE or FALSE

Under the hood in C, it's really an integer:

FALSE ⇔ 0 (0 is FALSE and FALSE is 0)

Any value other than 0 ⇒TRUE

TRUE ⇒1 (e.g., (4 < 7) evaluates to 1)

F-13

Complex Conditions

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

F-14

Complex Conditionals in C

We use *Boolean operators* to express complex conditionals in C.

We'll say lots more about this later! For now, here is some information for reference.

Boolean operators && || !
and or not

```
int     TRUE = 1;
int     FALSE = 0;

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

F-15

Multiple Statements

What if there's more than one conditional action?

"If your temperature is high, then
you have a fever, and
you should take two aspirin, and
you should go to bed, and
you should call in sick tomorrow"

F-16

No Problem in C

```
if (condition) {
    statement1 ;
    statement2 ;
    ...
}
```

This is called a *compound* or *block statement*

F-17

Using a Compound Statement

```
if ( temperature > 98.6 ) {
    printf ( "You have a fever. \n" );
    aspirin  =  aspirin  –2 ;
    printf ("Go to bed\n");
    printf ("Sleep in tomorrow\n");
}
```

F-18

Another Compound Example

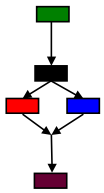
Cash machine program fragment:

```
if (balance >= withdrawal) {
    balance = balance – withdrawal;
    DispenseFunds(withdrawal);
}
```

What if () omitted?
What if {} omitted?

F-19

if…then…else, in English



Put 4 cups flour in a large mixing bowl

if using whole milk, then add two eggs,
otherwise add three eggs

Add a yard of butter

F-20

if…then…else in C

Example: Finding Absolute Value

Problem: Compute the absolute value |x| of x and put the answer in variable *abs*. Here are three solutions, all correct:

```
if (x >= 0) {
   abs = x;
}
if ( x < 0 ) {
   abs = -x;
}
```

```
abs = x;
if ( x < 0 ) {
   abs  = -x;
}
```

```
if (x >= 0) {
   abs = x;
} else {
   abs = -x;
}
```

F-21

If…then…else

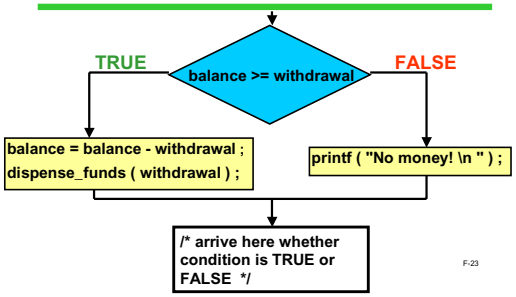
Print error message only if the condition is false:

```
if ( balance >= withdrawal ) {
    balance = balance - withdrawal ;
    dispense_funds ( withdrawal ) ;
} else {
    printf ( "Insufficient Funds! \n " ) ;
}
```

1, 2, 3, 4

F-22

If…then…else Control Flow



F-23

Nested *if* Statements

```
int  BILL_SIZE = 20;
if ( balance >= withdrawal ) {
    balance = balance - withdrawal ;
    dispense_funds ( withdrawal ) ;
} else {
    if ( balance >= BILL_SIZE ) {
        printf ( "Try a smaller amount. \n " ) ;
    } else {
        printf ( "Go away! \n" ) ;
    }
}
```

F-24

Nested *if*s , Part II

```
if ( x == 5 ) {
    if  ( y == 5 )  {
        printf ( "Both are 5. \n ") ;
    } else {
        printf ( "x is 5, but y is not. \n ") ;
    }
} else {
    if  ( y == 5 ) {
        printf ( "y is 5, but x is not. \n ") ;
    } else {
        printf ( "Neither is 5. \n ") ;
    }
}
```

F-25

Cascaded ifs
Tax Table Example

Problem: Print the % tax based on income:

income	tax
< 15,000	0%
15,000, < 30,000	18%
30,000, < 50,000	22%
50,000, < 100,000	28%
100,000	31%

F-26

One Solution

```
if ( income < 15000 ) {
    printf( "No tax." );
}
if ( income >= 15000 && income < 30000 ) {
    printf("18%% tax.");
}
if ( income >= 30000 && income < 50000 ) {
    printf("22%% tax.");
}
if ( income >= 50000 && income < 100000 ) {
    printf("28%% tax.");
}
if ( income >=100000) {
   printf("31%% tax.");
}
```

F-27

Mutually exclusive conditions - only one will be true, but that isn't immediately clear to a (human) reader

Cascaded ifs

```
if ( income < 15000 ) {
    printf( "No tax" );
} else {
    if ( income < 30000 ) {
        printf( "18%% tax." );
    } else {
        if ( income < 50000 ) {
            printf( "22%% tax." );
        } else {
            if ( income < 100000 ) {
                printf( "28%% tax." );
            } else {
                printf( "31%% tax." );
            }
        }
    }
}
```

```
if ( income < 15000 ) {
    printf( "No tax" );
} else if ( income < 30000 ) {
    printf( "18%% tax." );
} else if ( income < 50000 ) {
    printf( " 22%% tax." );
} else if ( income < 100000 ) {
    printf( "28%% tax." );
} else
    printf( "31%% tax." );
}
```

F-28

Order is important. Conditions are evaluated in order given.

Warning: Danger Ahead

The idea of conditional execution is natural , intuitive, and highly useful

However...

- Programs can get convoluted and hard to understand
- There are syntax errors that are easy to make but won't cause any compile-time errors.
- Worse, the syntax errors mean something in C, just not what you wanted.

F-29

Pitfalls of *if*, Part I

```
if  ( x = 10 ) {
    printf( "x is 10 " ) ;
}
```

Bug! = is used instead of ==

This is not a syntax error, so the compiler will not report any errors and the program can execute

F-30

The World's Last C Bug

```
status = check_radar ( ) ;
if (status = 1) {
    launch_missiles ( ) ;
}
```

F-31

Pitfalls of if, Part II

Not C *(thankfully, you'll get a compiler error in this case)*:

```
if  ( 0 <= x <= 10 ) {
      printf ( "x is between 0 and 10. \n " ) ;
}
```

Yes:

```
if  ( 0 <= x  &&  x <= 10 ) {
        printf ( "x is between 0 and 10. \n " ) ;
}
```

F-32

Pitfalls of if, Part III

& is different from &&
| is different from ||

& and | are not used in this class, but are legal C
If used by mistake, no syntax error, but program may produce bizarre results

F-33

Pitfalls of if, Part IV

Beware == and != with doubles:

```
double x ;
x = 30.0 * (1.0 / 3.0) ;
if ( x == 10.0 ) …  /* may or may not be true */
```

F-34

Next Time

We'll be discussing functions, a major topic of the course

F-35