Overview
Concepts this lecture
Conditional execution
if statement
Conditional expressions
Relational and logical operators
{Compound statements}

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 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
if the withdrawal is more than the bank balance, then print an error
if today is my birthday, then add one to my age
if using whole milk, add two eggs, otherwise add three eggs
Conditional ("if") Statement

if (condition) statement; The statement is executed if and only if the condition is true.

if (withdrawalAmount > balance)
    printf( "Not enough money\n\n");
if (temperature > 98.6)
    printf("You have a fever.\n\n");
if (x < 100) x = x + 1;

Conditional Flow Chart

if (x < 100) x = x + 1;
y = y + 1;

Conditions

In parentheses is a condition, also called a "logical" or "Boolean" expression
Made up of variables, constants, arithmetic expressions, and the relational operators

Math symbols: <, ≤, >, ≥, =, ≠ in C: <e, <=, >e, >=, ==, !=

Conditional Expressions

air_temperature > 80.0
98.6 <= body_temperature
marital_status == 'M'
divisor != 0

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

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 is 0 (and 0 is FALSE)
TRUE is any value other than 0 (and non-zero is TRUE)
1 is the result of a true relational operator (e.g., 4 < 7 evaluates to 1)
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

Complex Conditionals in C

We use Boolean operators to code 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

#define TRUE 1
#define FALSE 0

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

Multiple Actions

What if there’s more than one conditional action?

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

Compound Statement

Groups together statements so that they are treated as a single statement:

{ 
    statement1 ;
    statement2 ;
    ...
}

Also called a “block.”
Highly useful
Not just in conditionals, but many places in C

Using a Compound Statement

if (temperature > 98.6) {
    print(“You have a fever. ia”); 
    aspirin = aspirin - 2 ; 
    print(“Go to bed n”); 
    print(“Sleep in tomorrow n”);
}

Combining and Substituting Statements

You may use a compound statement anywhere that a single statement may be used

Anywhere that a statement is allowed in C, any kind of statement can be used
A compound statement can contain any number of statements (including 0)
Among other things, these principles imply that compound statements can be nested to any depth
Another Compound Example

Cash machine program fragment:

```
if (balance >= withdrawal) {
    balance = balance - withdrawal;
    dispense_funds(withdrawal);
}
```

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

Finding Absolute Value (1)

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;
```

Finding Absolute Value (2)

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;
```

Finding Absolute Value (3)

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;
```

if - else

Print error message only if the condition is false:

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

if-else Control Flow

```
if (balance >= withdrawal) {
    if (balance > withdrawal) {
        balance = balance - withdrawal;
        dispense_funds(withdrawal);
    } else {
        printf ("Insufficient Funds!\n");
    }
    dispense_funds(withdrawal);
} else {
    printf ("No money!\n");
}
```
Nested if Statements

```c
#define 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");
}
```

Nested ifs, Part II

```c
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");
}
```

Tax Table Example

<table>
<thead>
<tr>
<th>Income</th>
<th>Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 15,000</td>
<td>0%</td>
</tr>
<tr>
<td>15,000, &lt; 30,000</td>
<td>18%</td>
</tr>
<tr>
<td>30,000, &lt; 50,000</td>
<td>22%</td>
</tr>
<tr>
<td>50,000, &lt; 100,000</td>
<td>28%</td>
</tr>
<tr>
<td>100,000</td>
<td>31%</td>
</tr>
</tbody>
</table>

Direct Solution

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

Mutually exclusive conditions - only one will be true

Cascaded ifs

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

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 syntactic pitfalls to avoid
Pitfalls of if, Part I

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

Bug! = is used instead of ==

This is not a syntax error, so the program can execute

The World’s Last C Bug

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

Pitfalls of if, Part II

**No:**
```c
if ( 0 <= x <= 10 ) {
    printf ( "x is between 0 and 10. \n" ) ;
}
```

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

Pitfalls of if, Part III

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

& and | are not used in this class
If used by mistake, no syntax error, but program may operate incorrectly

Pitfalls of if, Part IV

Beware == and != with doubles:
```c
double x ;
x = 30.0 * (1.0 / 3.0) ;
if ( x == 10.0 ) ...```

Next Time

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

Many students find it intellectually challenging compared to the previous material