Twenty Questions (more or less)

I have three coins, but one of them is fake!
The fake is heavier than the others.
You have a scale. What’s the smallest number of weighings you can use to find the fake?

Finding the Fake Coin

Weigh two coins against each other:
– if the left one is heavier, it’s the fake
– otherwise, if the right is heavier, it’s the fake
– otherwise, the remaining one is the fake

How can we do this in C?

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 (uses some future concepts)
– 4.7: Nested if statements

Overview

Conditional execution
if statement
A strange bit of syntax: {Compound statements}
Conditional expressions
Relational and logical operators

Control Flow of our Scales

Here’s our algorithm:
control of the process... flows from box to box.
The algorithm is clearly stated and deterministic.
The computer should be able to do it!
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:

```c
{  int i = 1;  i = i + 1;  printf("%d", i);  }
```

But... what are those {}? Let’s come back to that!

Conditional Control Flow

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

Conditional Execution

Conditional statements allow 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 it’s a 9:30 class, prop your eyelids open; otherwise (it’s 11:30), gnaw on your arm while you wait for lunch.

“Compound statements”

Before we get into writing conditionals in C... Groups statements so that they are treated as a single statement:

```c
{  statement1;  statement2;  ...
}  indicates sequential control flow!
```

Also called a “block.”

You’ve seen this before...

```c
int main(void) {
    printf("Hello, world!
    return 0;
}
```

Now, detour over. But keep this in mind.

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.

“Nested” means “put inside one another.”
The statement is executed if the condition is true. Otherwise, the statement is skipped.

---

Conditional ("if") Statement

```c
if (temperature > 98.6) {
    printf("You have a fever.
");
    printf("Go see the doc.
");
}
```

---

Blocks are Back!

To perform multiple statements conditionally, we use a compound statement!

```c
if (temperatur > 98.6) {
    printf("You have a fever.
");
    printf("Go see the doc.
");
}
```

---

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: `<` , `<=` , `>` , `>=` , `==` , `!=`  

What should we call `==`?

---

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 1 (and 1 is TRUE)
TRUE is also any other non-zero value...
But relational ops will always give 1 for TRUE (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

C represents these with “Boolean” operators.

Boolean operators: && || !

#define TRUE 1
#define FALSE 0

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

Finding Absolute Value

Problem: Compute the absolute value |x| of x
Put the answer in variable abs.

if (x >= 0) {
abs = x;
} else {
if (x < 0) {
abs = -x;
} else {
abs = -x;
}
Which of these is right?

History Break

When we write conditions, we use “Boolean algebra”, the symbolic representation of logic.
This algebra is at the heart of everything computers do! Every operation is eventually calculated in terms of Boolean algebra.

Guess when George Boole invented it…

Almost 150 years ago, in 1854.

Picture/autobiography thanks to math dept.
U. of St. Andrews, Scotland

if - else

Example: print error message if condition is false:

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

if-else Control Flow
Nested if statements

```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 \) ;
    }
}
```

Any statement can go inside an if statement.

Therefore, an if statement can go inside an if statement.

Direct Solution

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

Mutually exclusive conditions - only one will be true.

Tax Table Example

Problem: Print the % tax based on income:

<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>

Cascaded ifs

```c
if ( income < 15000 ) {
    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." );
    }
}
```

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:
The World's Last C Bug

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

Bug! != is used instead of ==

This is not a syntax error, so the compiler will not report any errors and the program can execute
Pitfalls of if, Part II

No:
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 ");
}

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

Pitfalls of if, Part IV

Beware == and != with doubles:

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.

QOTD Tomb Raider:
Find and Destroy the Dead Code!

“Dead” code is code that will never be executed no matter how you run the program.

Find the “dead” code in the following…

    if (speed >= 0) {
        printf("You don’t go backward.\n");
        if (speed == -1) {
            printf("Wait! I was wrong!\n");
        } else if (speed > 0) {
            printf("You go forward.\n");
        } else if (speed < 0) {
            printf("You go backward.\n");
        } else {
            printf("What did you do?!\n");
        }
    }