CSE 142
Programming I

Conditionals

Chapter 4

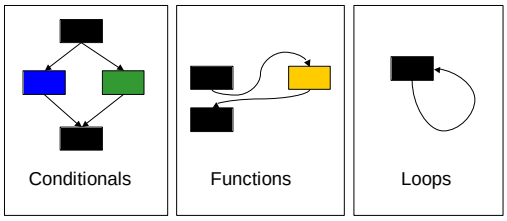
- Read Sections 4.1–4.5, 4.7–4.9
- The book assumes that you've read chapter 3 on functions
  - ➤Read it anyway, you should do fine
  - ➤Their order is a little screwy...

Preview of Things to Come

- Control flow is the order in which statements are executed
- Until now, control flow has been sequential:

Preview Prologue

- We'll look at ways to change the flow:



Conditionals

- Conditionals let the computer choose an execution path depending on the value of an expression
  - ➤Print an error if the withdrawal amount is more then what is in the account
  - ➤Add one to my age if it is my birthday
  - ➤If my grade is greater than 3.5, then celebrate

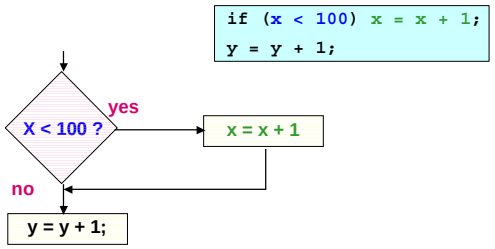
Conditional ("if") Statement

```
if (condition) statement;
```

- The statement is executed *only* if the condition is true.

```
if (age >= 21)
    printf("Have a beer!\n");
```

Conditional Flow Chart



Conditional Expressions

- Also called “logical”, or “Boolean” expressions
- Make use of relational operators
- A relational operator compares two values
- Examples:

```
(x < 30)
(12 > y)
```


Relational Operators

In Math	In C	In English
<	<	Less Than
>	>	Greater Than
=	==	Equal To
≤	<=	Less Than or Equal To
≥	>=	Greater Than or Equal To
≠	!=	Not Equal

Some Conditional Expressions

```
air_temperature > 80

98.6 == body_temperature

marital status == 'M'

weight >= 8000
```


Multiple Actions

- What if we want to do more things at once?
  - ➤Use a compound statement!
- Replace the statement with several statements surrounded by braces: { }
  - ➤Sometimes called a block
  - ➤Indent each block!

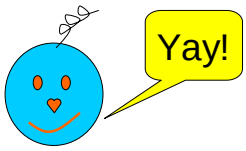
Example: Checking for Error

```
printf("Enter divisor: ");
scanf("%lf", &divisor);
if (0 == divisor){
  printf("Can't divide by 0!\n");
  exit(0);
}
```

No ;

Announcement!

- Now you know enough to do homework 1



Value of Boolean Expressions

- Remember, expressions are things in C that have values
- What's the value of a conditional expression?

Values of Boolean Expressions

- Conditional expressions are either true or false
- C doesn't have a Boolean Type
- C fakes it using integers!
  - ➤0 means false
  - ➤non-zero (usually 1) means true

Examples:

- Do these examples make sense?
- If so, what are the values assigned?

```
foo = 7 < 0;

bar = 8 != 3;
```


More Strangeness!

- In fact, this is an *expression*:

```
x = 6 + 7
```

- If it is an expression, then it has a value
- The value of an assignment is the value assigned

The Value of an Assignment

- What does this do?

```
int foo, bar;

foo = (bar = 6);
```

Why Am I Telling You This?!

- It's easy to confuse = and ==
- If you confuse them, the program will compile and run, but it won't do what you want it to
- Watch out for this error!

Strange Example I

- What does this do?

```
if (x = 7)
    printf("X is equal to 7\n");
```


Strange Example II

- What does this do?

```
x == PI * radius * radius;
```


Lesson:

- C is very picky
  - ➤Getting *one* character wrong will often make your program work incorrectly!
- You'll have to get very good at finding these stupid little errors
  - ➤Practice practice practice!

One More Strange Point

- Do not use `==` or `!=` with doubles
  - ➤doubles may not have exact values, only approximations
  - ➤`==` and `!=` are only make sense if the values are exact
  - ➤`<`, `>`, etc. are fine for use with doubles

Complex Conditionals

- We'd like more expressive power
  - ➤If I have at least $15 or you have at least $15, then we can go to the movies.
  - ➤If you're in Guggenheim 224 and it's 12:00 on Friday, then you're in CSE 142.
  - ➤If you're in CSE 142 and your name is Isaac, then you're the lecturer.

Boolean Operators

- Boolean operators act on Boolean values to produce new Boolean values

C	English
`&&`	And
`||`	Or (sort-of…)
`!`	Not

Complex Conditionals in C

```
if ((myMoney   > 15.0) ||
    (yourMoney > 15.0)){…}

if ((location == 224) &&
    (time     == 12 )){…}

if (!(initial == 'I')){…}
```


Truth Tables for `&&`, `||`

- A truth table lists all possible combinations of values and their result

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

Or isn't Always Like in English!

- In English if "this fruit is an apple or it is an orange", then it cannot be both apple and orange
- In C,

```
(fruit == 'A') || (fruit == 'O')
```

is true if either half is true!

Truth Table for !

- && and || are binary operators
- ! is a unary operator that inverts the value

P	!P
T	F
F	T

An Example Expression

- foo is an integer that is not 5, 6, or between 10 and 20

```
!((foo == 5) || (foo == 6) ||
  ((foo > 10) && (foo < 20)))
```

DeMorgans' Laws

- Convert between and expressions and or expressions
- Example:

```
!((age < 25) && (sex == 'M'))
```

is equivalent to

```
((age >= 25) || (sex != 'M'))
```


DeMorgan's Laws

- DeMorgan's Laws tell us some legal conversions:

```
!(P && Q)  <-------->  (!P || !Q)

!(P || Q)  <-------->  (!P && !Q)
```


Proof of DeMorgan's Laws

- Proof using a truth table:

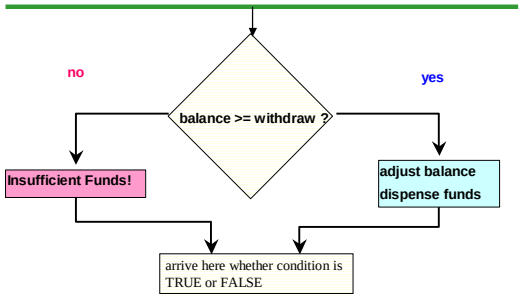
P	Q	(P && Q)	!(P && Q)	!P	!Q	(!P || !Q)
T	T					
T	F					
F	T					
F	F					

Else: The Other Half of If

- else lets you do something if the condition was false

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


if-else Control Flow



Nested ifs

```
#define BILL_SIZE  20
if (balance < withdrawal) {
   printf("Insufficient funds!\n");
} else {
    if (withdrawal < BILL_SIZE)
       printf ("Try a larger amount. \n");
    else balance = balance - withdrawal;
}
```

Tax Example

- 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%

Simple 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.");
}
```

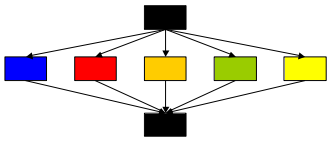

Cascaded ifs

```
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." );
}
```


Another Type of Conditional!

- The Switch Statement:



Longwinded if

```
/* How many days in a month? */
if   ( month ==  1 ) {          /* Jan */
   days = 31 ;
} else if ( month ==  2 ) {     /* Feb */
   days = 28 ;
} else if ( month ==  3 ) {     /* Mar */
   days = 31 ;
} else if ( month ==  4 )       /* Apr */
   days = 30 ;
...       /* need 12 of these */
```


Better Code:

```
if ( month==9 || month==4 || /* Sep, Apr */
     month==6 || month==11 ){/* Jun, Nov */
   days = 30;
} else if ( month == 2 ) {   /* Feb */
   days = 28 ;
} else {
   days = 31;   /* All the rest  */
}
```

Even Better Code:

```
switch ( month ) {
   case 2:   /* February  */
      days = 28 ;
      break ;
   case 9:   /* September */
   case 4:   /* April     */
   case 6:   /* June  */
   case 11:  /* November  */
      days = 30 ;
      break ;
   default: /* All the rest have 31 */
      days = 31 ;
}
```


switch

```
switch (control expression){
   case-list-1
      statements 1
      break;
   case-list-2
      statements 2
      break;
   …
   default:
      statements
}
```


switch Pitfalls

- The type of the control expression must be int or char
- The cases must be constant
- The switch statement falls through

Falling Through

```
switch (month) {
   case 2:
      days = 28 ; /* break missing */
   case 9:
   case 4:
   case 6:
   case 11:
      days = 30; /* break missing */
   default:
      days = 31 ;
}
```


Conditionals Summary

- `if` lets the execution branch
  - ➤complex conditions are put together with `&&`, `||`, and `!`
- `else` does something if the condition was false
- `switch` can be used in *some* situations to do a multiple branch
  - ➤control-expression must be int or char
  - ➤cases must be constants