Lecture 4:
Selection Statements

Sarah Nadi
nadi@ualberta.ca
Department of Computing Science
University of Alberta

CMPUT 201 - Practical Programming Methodology
Winter 2018

[With material/slides from Guohui Lin, Davood Rafei, and Michael Buro. Most examples taken from K.N. King's book]

Agenda

- Selection statements
  - If statement
  - Switch/case statement
- Logical Expressions

Readings

- Chapter 5

Calculating Commission (motivating example)

- Page 81: Assume a broker charges commission according to the following table. Additionally, the minimum charge is $39.

Transaction size	*Commission rate*
Under $2,500	$30 + 1.7%
$2,500–6,250	$56 + 0.66%
$6,250–20,000	$76 + 0.34%
$20,000–50,000	$100 + 0.22%
$50,000–500,000	$155 + 0.11%
over $500,000	$255 + 0.09%

- Expected appearance:

```
Enter value of transaction: 30000
Commission: 166.0
```

Selection Statements

- allow the program to select a particular execution path from a set of alternatives

- `if` and `switch` statements are selection statements

- decision of which path to follow is based on a *logical expression*

Logical Expressions

- Statements that test the value of an expression
  - “true” (1) or “false” (0)
  - e.g., `if (10==11)`
- Logical expression can include:
  - Relational operators (<, >, <=, >=)
  - Equality operators (==, ! =)
  - Logical operators (!, &&, ||)

Boolean Values in C

- There is no built-in “bool” or “boolean” type in C. Basically 0 is false and any non-zero value is true

- In C99, there is a type _Bool, which is basically an integer type with values 0 (false) or 1 (true)

- `#include <stdbool.h>`
  - provides a macro bool for _Bool
  - macros for true and false (stand for 1 and 0, respectively)

```
bool flag; ...
flag = false;
...
flag = true;
```

The `if` Statement

- Form
  - `if ( expression ) statement` or
  - `if ( expression ) {statements}` — a sequence of multiple statements enclosed in curly brackets is called a *compound statement*
    - required use of parentheses to enclose the expression
    - when “true” (i.e., expression is non-zero), the statement(s) is/are executed
- e.g., testing whether 0 ≤ i < n

```
if (i >= 0 && i < n) statement /* i is in expected range */
if (i < 0 || i >= n) statement /* is is outside range */
```

Operators Revisited

Precedence	Name	Symbol(s)	Associativity
1	increment (postfix) decrement (postfix)	++ --	left
2	increment (prefix) decrement (prefix) unary plus unary minus **not**	++ -- + - **!**	right
3	multiplicative	* / %	left
4	additive	+ -	left
5	**relational**	**< > <= >=**	**left**
6	**equality**	**== !=**	**left**
7	**logical**	**&& ||**	**left**
5	assignment	= *= /= %= += –=	right

Short Circuit Evaluation

```
int i = 10;
int j = 2;
if ((i == 10) || (j < 5)){
    …
}
```

Short Circuit Evaluation

```
int i = 10;
int j = 2;
if ((i == 10) || (j < 5)){
    …
}
```

Will the expressions (i==10) and (j<5) both be evaluated?

Short Circuit Evaluation

```
int i = 10;
int j = 2;
if ((i == 10) || (j < 5)){
    …
}
```

Will the expressions (i==10) and (j<5) both be evaluated?

```
int i = 10;
int j = 2;
if ((i != 10) && (j < 5)){
    …
}
```

Cascaded form of an `if` statement

```
if ( expression ) {
 statements
}else if ( expression ) {
 statements
}
...
else if ( expression ) {
 statements
} else {
 statements
}
```

exactly one compound statement will get executed

Switch Statement

- Equivalent to a cascaded if statement

```
if (grade == 4) {
   printf("Excellent");
}
else if (grade == 3) {
   printf("Good");
}
else if (grade == 2) {
   printf("Average");
}
else if (grade == 1) {
   printf("Poor");
}
else if (grade == 0) {
   printf("Failing");
}
else {
   printf("Illegal grade");
}
```

```
switch (grade) {
   case 4:
            printf("Excellent");
            break;
   case 3:
            printf("Good");
            break;
   case 2:
            printf("Average");
            break;
   case 1:
            printf("Poor");
            break;
   case 0:
            printf("Failing");
            break;
   default:
            printf("Illegal grade");
            break;
}
```

Switch Statement

- Equivalent to a cascaded if statement

The break statement is very important!

```
if (grade == 4) {
   printf("Excellent");
}
else if (grade == 3) {
   printf("Good");
}
else if (grade == 2) {
   printf("Average");
}
else if (grade == 1) {
   printf("Poor");
}
else if (grade == 0) {
   printf("Failing");
}
else {
   printf("Illegal grade");
}
```

```
switch (grade) {
   case 4:
           printf("Excellent");
           break;
   case 3:
           printf("Good");
           break;
   case 2:
           printf("Average");
           break;
   case 1:
           printf("Poor");
           break;
   case 0:
           printf("Failing");
           break;
   default:
           printf("Illegal grade");
           break;
}
```

Ternary Operator

- `expr1 ? expr2 : expr3`
- A special if statement
- The ternary operator (? and :)
- Read as “if expr1 then expr2 else expr3”
- `i > j ? i-- : j++;` is equivalent to:

```
if (i > j)
    i--;
else
    j++;
```

Calculating Commission — Revisited

- Page 81: Assume a broker charges commission according to the following table. Additionally, the minimum charge is $39.

Transaction size	*Commission rate*
Under $2,500	$30 + 1.7%
$2,500–6,250	$56 + 0.66%
$6,250–20,000	$76 + 0.34%
$20,000–50,000	$100 + 0.22%
$50,000–500,000	$155 + 0.11%
over $500,000	$255 + 0.09%

- Expected appearance:

```
Enter value of transaction: 30000
Commission: 166.0
```

demo: commission.c

What is the Output of the Following?

```
int i = 7, j = 8, k = 9;
printf ("%d ", (i=j) || (j=k));
printf ("%d %d %d", i, j, k);
```

Answer using mentimeter

Printing a Date in Legal Form (p89)

- Appearance:

```
Enter data (mm/dd/yy): 02/05/16

Dated this 5th day of February, 2016.
```

- Things to think about:
  - 1st, 2nd, 3rd, 4th, …
  - displaying month as a number
  - displaying 16 as 2016

Group Exercise!