CSE 220 – C Programming
Fall 2015
Selection Statements
Notes

• Homework 5 will be posted today later.

• Your midterm exam is on October 14.
Cascading if statements
Recap

```java
if (BooleanExpression) {
    statement or block 1
} else {
    statement or block 2
}
```
Cascading if-else statements

• The if-else statement allows a choice to be made between two possible alternatives

• Sometimes you need to be able to test a series of conditions

Example: Suppose we need a program segment to read a number \( x \) and display its sign.

\[
\begin{align*}
x < 0 & \quad \text{printf("- \n");} \\
x = 0 & \quad \text{printf("0 \n");} \\
x > 0 & \quad \text{printf("+\n");}
\end{align*}
\]
Cascading if-else statements

Example: Suppose we need a program segment to read a number $x$ and display its sign.

$$x < 0 \quad \text{printf("- \n")};$$

$$x = 0 \quad \text{printf("0 \n")};$$

$$x > 0 \quad \text{printf("+ \n")};$$

if ($x < 0$)
    printf("- \n");
if ($x == 0$)
    printf("0 \n");
if ($x > 0$)
    printf("+ \n");
Cascading if-else statements

**Example:** Suppose we need a program segment to read a number \( x \) and display its sign.

\[
\begin{align*}
\text{if } (x < 0) & \quad \text{printf(”- \n”);} \\
\text{if } (x == 0) & \quad \text{printf(”0 \n”);} \\
\text{if } (x > 0) & \quad \text{printf(”+ \n”);} \\
\end{align*}
\]

\[
\begin{align*}
\text{if } (x < 0) & \quad \text{printf(”- \n”);} \\
\text{else if } (x == 0) & \quad \text{printf(”0 \n”);} \\
\text{else if } (x > 0) & \quad \text{printf(”+ \n”);} \\
\end{align*}
\]
Cascading if-else statements

• Roots of a quadratic equation: 

\[ ax^2 + bx + c \]

\[ \Delta = \text{delta} = b^2 - 4ac \]

\[ \Delta > 0 \quad \text{two roots} \]
\[ \Delta = 0 \quad \text{single root} \]
\[ \Delta < 0 \quad \text{no root at all} \]
Cascading if-else statements

- Roots of a quadratic equation:

\[ \Delta = \text{delta} = b^2 - 4ac \]

```c
if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");

if (delta < 0)
    printf("No real roots\n");
else if (delta == 0)
    printf("Exactly one real root\n");
else
    printf("Two distinct roots\n");
```
Cascading if-else statements

**cascading if-else statements** = **nested compound if-else statements**

```
if (delta < 0)
    printf("No real roots\n");
else{
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");
}
```

The else part of **first if** is a compound statement.

```
if (condition1)
    statement1;
else{
    if (condition2)
        statement2;
    else{
        if (condition3)
            statement3;
        ...
    }
}
```
Cascading if-else statements

- **delta < 0**
  - Yes: printf("no roots")
  - No: delta == 0

- **delta == 0**
  - Yes: printf("one root")
  - No: printf("two roots")

Note: No need to check
Cascading if-else statements

```c
if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");
```

Diagram:

- If `delta < 0`, print "No real roots".
- If `delta == 0`, print "Exactly one real root".
- If neither condition is met, print "Two distinct roots". 

1. Delta < 0: Print "no roots".
2. Delta == 0: Print "one root".
3. Delta not equal to 0: Print "two roots".
Cascading if-else statements

```c
if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");
```

Diagram:
- If `delta < 0`, print "No real roots\n".
- Else, if `delta == 0`, print "Exactly one real root\n".
- Else, print "Two distinct roots\n".

Flowchart:
- If `delta < 0`, go to "No roots".
- If `delta == 0`, go to "one root".
- If `delta > 0`, go to "two roots".
if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");
Exercise

if (age < 60)
   printf("You are young\n");
else
   if (age < 70)
      printf("In your 60s\n");
   else
      if (age < 80)
         printf("In your 70s\n");
      else
         if (age < 90)
            printf("In your 80s\n");
         else
            if (age < 100)
               printf("In your 90s\n");
            else
               printf("Really?\n");

• Draw the flow-chart on a piece of paper
• Highlight the path for age = 88
• Write your name, PID, and section #, turn in your paper on Wednesday
Cascading if-else statements

```c
if (age < 40)
    printf("You are young\n");
else if (age < 50)
    printf("You are in your forties\n");
else if (age < 60)
    printf("You are in your fifties\n");
else if (age < 70)
    printf("You are in your sixties\n");
else if (age < 80)
    printf("You are in your seventies\n");
else if (age < 90)
    printf("You are in your eighties\n");
else if (age < 100)
    printf("You are in your nineties\n");
else
    printf("Really?\n");
```

```c
if (age < 40)
    printf("You are young\n");
else
    if (age < 50)
        printf("In your 40s\n");
    else
        if (age < 60)
            printf("In your 50s\n");
        else
            if (age < 70)
                printf("In your 60s\n");
            else
                if (age < 80)
                    printf("In your 70s\n");
                else
                    if (age < 90)
                        printf("In your 80s\n");
                    else
                        printf("In your 90s\n");
                        else
                            printf("Really?\n");
```
Example

```c
int a, b, c, z;
printf("Enter 3 integers:\n");
scanf("%d%d%d", &a,&b,&c);

if (a < b)
    if (b<c)
        z = c;
    else
        z = b;
else
    if (a > c)
        z = a;
    else
        z = c;
```
Example

```c
int a, b, c, z;
printf("Enter 3 integers:\n");
scanf("%d%d%d", &a,&b,&c);

if (a > b && a > c)
    z = a;
else if (b > c)
    z = b
else
    z = c;
```
Why cascading if-else statements?

if (x < 0)
    sign = -1;
else if (x == 0)
    sign = 0;
else if (x > 0)
    sign = 1;

if (x < 0)
    sign = -1;
else if (x == 0)
    sign = 0;
else if (x > 0)
    sign = 1;

1 Readability

2 Efficiency
Dangling else problem

```c
if (salary < 200,000)
    if (salary > 50,000)
        printf("You are doing alright\n");
else
    printf("You make a lot of money\n");
```

- What is the output if salary is
  - $100,000 →
  - $250,000 →
  - $10,000 →
  - $20
Dangling else problem

if ( salary < 200,000)
    if (salary > 50,000)
        printf(“You are doing alright\n”);
else
    printf(“You make a lot of money\n”);

• What is the output if salary is
  $100,000 → You are doing alright
  $250,000 → ???
  $10,000   → You make a lot of money
Dangling else problem

```c
if ( salary < 200,000)
    if (salary > 50,000)
        printf("You are doing alright\n");
else
    printf("You make a lot of money\n");
```

- **else** is matched to the nearest **if**
  - $100,000 → You are doing alright
  - $250,000 → ???
  - $10,000 → You make a lot of money
Dangling else problem

if (salary < 200,000) {
  if (salary > 50,000) {
    printf("You are doing alright\n");
  }
}

/* The "scope" of the second if is now limited within the */
/* red curly brackets, so it is hidden from the following else */
else {
  printf("You make a lot of money\n");
}

• Use {} even with one statement only
Dangling else problem

if ( salary < 200000)
  
  if (salary > 50000)
    printf("You are doing alright\n");
  else
    printf("You make a lot of money");

• Else is matched to the nearest if

Use {} even with one statement only
Exercise

• What is the output?
  
  ```c
  if (10 < j < 20) {
    printf("Between 10 and 20\n");
  }
  ```

• Write a single expression whose value is -1, 0, 1 if x is less than y, equal to y, or greater than y respectively
  
  ```c
  x > y? 1 : (x < y ? -1 : 0)
  ```
Ternary Operator
Ternary operator

```
expr1 ? expr2 : expr3;
```

The expression `expr1` will be evaluated always. Execution of `expr2` and `expr3` depends on the outcome of `expr1`.

```
if CONDITION
STATEMENT 1;
else
STATEMENT 2;
```

```
int k;
if (a > b)
    k = a;
else
    k = b;
int k = a > b ? a : b;
k = maximum(a, b)
```
Ternary operator

CONDITION ? STATEMENT 1 : STATEMENT 2;

L-VALUE = CONDITION STATEMENT 1 STATEMENT 2
Ternary operator

CONDITION ? STATEMENT 1 : STATEMENT 2;

The output is the value of STATEMENT which is evaluated based on the CONDITION!

L-VALUE = CONDITION

Optional: you can either use or ignore it
Example: Number is Odd or Even

```c
#include<stdio.h>

int main()
{
    int num;

    printf("Enter the Number : ");
    scanf("%d",&num);

    flag = ((num%2==0)?1:0);

    if(flag==0)
        printf("\nEven");
    else
        printf("\nOdd");
}
```

The ternary operation also returns the value evaluation of statement being executed as OUTPUT.
Example: Number is Odd or Even

```c
#include<stdio.h>

int main()
{
    int num;

    printf("Enter the Number : ");
    scanf("%d", &num);

    flag = ((num%2==0)?1:0);

    if(flag==0)
        printf("\nEven");
    else
        printf("\nOdd");
}
```

```c
#include<stdio.h>

int main()
{
    int num;

    printf("Enter the Number : ");
    scanf("%d", &num);

    (num%2==0)?printf("Even"):printf("Odd");
}
```
Exercise

• What does this expression do?

\[
\text{int } k = ( i > 0 \ ? \ 1 : -1) * i;
\]

• Write a single expression whose value is -1,0,1 if x is less than y, equal to y, or greater than y respectively

\[
x > y ? 1 : (x < y ? -1 : 0)
\]
Conditional Expressions

\[\text{int } k = (i > 0 ? 1 : -1) \times i;\]

• Can use compound statement?
• Make programs shorter but harder to read
• Use with simple expressions
Problem

• Complete the following program, such that:
  – if the user types in a lowercase letter, convert it to uppercase and print out;
  – if the user types in an uppercase letter, convert it to lowercase and print out;
  – print out “It is not an alphabetic character” if the user’s input is neither an uppercase nor a lowercase letter.
Problem

* Approach: flow-chart
  - if the user types in a lowercase letter, convert it to uppercase and print out;
  - if the user types in an uppercase letter, convert it to lowercase and print out;
  - print out “It is not an alphabetic character” if the user’s input is neither an uppercase nor a lowercase letter.
Problem

User input

lowercase letter ?
Yes
Convert to uppercase

No

uppercase letter ?
Yes
Convert to lowercase

No

print ...

if (...) {
    ...
}
else if (...) {
    ...
}
else {
    ...
}
Character types (I)

• Char: single character
• ASCII code:
  – 7bit code, 128 characters
  – A is 1000001 (=65)
  – B is 1000010 (=66)
• Treated like integers
  – char c = 65, char c = ‘A’
  – c += 1 => c become ‘B’
  – c += ‘a’ - ‘A’ => ‘b’
  – char d = 32, char d = ‘ ‘
  – ‘a’”’z’/’X’

<table>
<thead>
<tr>
<th>int</th>
<th>char</th>
<th>int</th>
<th>char</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>0</td>
<td>90</td>
<td>Z</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>97</td>
<td>a</td>
</tr>
<tr>
<td>57</td>
<td>9</td>
<td>98</td>
<td>b</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>A</td>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Character types (II)

• Write an expression to test whether a character $ch$ is lowercase:
  \[ ch \geq 'a' \&\& ch \leq 'z' \]

• Write an expression to convert a lowercase character $ch$ to uppercase:
  \[ ch = ch - 32; \]
Switch Statement
Sometimes it is necessary to branch in more than two directions.

We do this via multiple selection.

The multiple selection mechanism in C is the switch statement.
Multiple Selection with if

if (day == 1) {
    printf("Monday\n");
} else if (day == 2) {
    printf("Tuesday\n");
} else if (day == 3) {
    printf("Wednesday\n");
} else if (day == 4) {
    printf("Thursday\n");
} else if (day == 5) {
    printf("Friday\n");
} else if (day == 6) {
    printf("Saturday\n");
} else if (day == 7) {
    printf("Sunday\n");
} else {
    printf("I don’t know\n");
}

Switch case statements are a **substitute** for long if statements that compare a variable to several "integral" values.
Switch- Flowchart

Fig. 4.8  |  switch multiple-selection statement with breaks.

Adapted from Deitel & Deitel, C How to Program, 6th ed., p. 111
The **switch** statement provides another way to decide which statement to execute next.

The **switch** statement evaluates an expression, then attempts to match the result to one of several possible cases.

The match must be an exact match.

```plaintext
switch ( expression ){
    case value1 :
        statement-list1;
        break;
    case value2 :
        statement-list2;
        break;
    case value3 :
        statement-list3
        break;
    case ... 
}
```
Switch Statement Syntax

• Each case contains a value and a list of statements

• The flow of control transfers to statement associated with the first case value that matches

```java
switch (expression) {
    case value1:
        statement-list1
        break;
    case value2:
        statement-list2
        break;
    case value3:
        statement-list3
        break;
    case ...
}
```
The *break statement* can be used as the last statement in each case's statement list.

A *break statement* causes control to transfer to the end of the switch statement.

If a *break statement* is not used, the flow of control will continue into the next case.

```java
switch (expression) {
    case value1:
        statement-list1
        break;
    case value2:
        statement-list2
        break;
    case value3:
        statement-list3
        break;
    case ...
}
```
Switch Statement Syntax

• A `switch` statement can have an optional `default` case

• The default case has no associated value and simply uses the reserved word `default`

• If the default case is present, control will transfer to it if no other case value matches

• If there is no default case, and no other value matches, control falls through to the statement after the switch
Switch Statement

if (day == 1) {
    printf("Monday\n");
} else if (day == 2) {
    printf("Tuesday\n");
} else if (day == 3) {
    printf("Wednesday\n");
} else if (day == 4) {
    printf("Thursday\n");
} else if (day == 5) {
    printf("Friday\n");
} else if (day == 6) {
    printf("Saturday\n");
} else if (day == 7) {
    printf("Sunday\n");
} else {
    printf("I don't know\n");
}

switch (day) {
    case 1: printf("Monday\n"); break;
    case 2: printf("Tuesday\n"); break;
    case 3: printf("Wednesday\n"); break;
    case 4: printf("Thursday\n"); break;
    case 5: printf("Friday\n"); break;
    case 6: printf("Saturday\n"); break;
    case 7: printf("Sunday\n"); break;
    default: printf("I don't know\n"); break;
}
Switch versus Cascaded if-else

A nested if-else structure is just as efficient as a switch statement.

BUT

• Easier to read
• Faster
Switch Statement Syntax

⚠️ The expression of a `switch` statement must result in an **integral type**, meaning an integer (`byte`, `short`, `int`, `long`) or a `char`.

⚠️ It **cannot** be a `boolean` value or a floating point value (`float` or `double`).

⚠️ The implicit boolean condition in a `switch` statement is **equality**.

⚠️ A switch statement will compile without a default case, but always consider using one.

🚫 You cannot perform relational checks with a `switch` statement.
Switch Statement

```c
switch (day) {
    case 1: printf("Monday\n");
       break;
    case 2: printf("Tuesday\n");
       break;
    case 3: printf("Wednesday\n");
       break;
    case 4: printf("Thursday\n");
       break;
    case 5: printf("Friday\n");
       break;
    case 6: printf("Saturday\n");
       break;
    case 7: printf("Sunday\n");
       break;
    default: printf("I don’t know\n");
       break;
}
```

- Switch must be followed by int (or char)
- No braces after case label
- Constant expressions:
  - 1, 4+3,
  - x + 1: if x is defined by a constant macro: #define x 5
- Duplicate labels not allowed
- `break`: exit the switch statement
- `default`: executed if no match
- Can group case labels
Switch Statement

```c
switch (day) {
    case 1: case 2: case 3: case 4: case 5:
        printf("Week day\t");
    case 6: case 7:
        printf("Weekend\t");
    default: printf("I don’t know\t");
}
```

• If day is 3:
  Weekday       Weekend       I don’t know

• If day is 6:
  Weekend       I don’t know
Switch Statement

```c
switch (day) {
    case 1: case 2: case 3: case 4: case 5:
        printf("Week day\n");
        break;
    case 6: case 7:
        printf("Weekend\n");
        break;
    default: printf("I don’t know\n");
        break;
}
```
Summary

- If statement
- Conditional statement
- Switch statement