Program Control Flow
2/21/01 Lecture #7 16.070

• Sequential - Straight line programs
  ➢ Statements executed in order in which they appear, without branching or repetition
  ➢ Most programming problems are not so simple

• Conditional Branching - enables program to make decisions
  ➢ Decide whether or not to execute sequence of statements based on value of an expression
  ➢ Use if task consists of doing one of two subtasks, but not both, depending on a condition (TRUE or FALSE)

• Iterative - loops (Friday's lecture)
  ➢ Perform subtask for as long as condition remains TRUE
  ➢ After subtask is completed, check condition again
Basic Constructs of Structured Programming

Sequential

- Do first part to completion
- Do second part to completion

Conditional

- Test condition
- Sub-task 1
- Sub-task 2

Iterative

- Test condition
- True
- Sub-task
- False
Conditional Branching - Selection Statements

• The *if* Statement

```
if (condition)
    action;
```

```
if (<selection expression>)
    statement1;
```

➢ Test condition is written as a selection expression; i.e., combination of operators and operands

➢ If selection expression evaluates to TRUE, execute statement1

➢ Any non-zero value is considered TRUE

```
if (x <= 10)
    y = 3 * x + 5;
```

➢ Expression `(x <= 10)` evaluates to 1 if TRUE, 0 if FALSE

➢ `y = 3 * x + 5` will be executed if `(x <= 10)` evaluates to 1 (i.e., TRUE)
Selection Statements - the if Statement

• Compound statements, or blocks, group simple statements into one entity

\[
\text{if} \ (<\text{selection expression}>)
\{
\quad \text{statement1;}
\quad \text{statement2;}
\quad \text{statement3;}
\}
\]

➢ Execution of all statements in block are based on single test condition captured in the selection expression

\[
\text{if} (x \neq 10)
\{
\quad z = z + 1;
\quad w = w - 2;
\}
\]
Conditional Branching - Two-Way Selection Statement

• The *if-else* statement performs two-way selection

\[
\text{if} (\langle \text{selection expression} \rangle) \\
\hspace{1cm} \text{statement1; /* Action if: executes if expression = T */} \\
\text{else} \\
\hspace{1cm} \text{statement2; /* Action else: Executes if expression = F */} \\
\hspace{1cm} \text{statement3; /* Always executes */}
\]

- Mutually exclusive outcomes
  - Perform one set of actions if condition is TRUE
  - Perform another set of actions if FALSE

\[
\text{if} (x) \\
\hspace{1cm} y = y + 2; \\
\text{else} \\
\hspace{1cm} \{ \\
\hspace{2cm} y - 2; \\
\hspace{2cm} z = z/2; \\
\hspace{1cm} \}
\]
Use of Two-Way Selection Statement

• The *if-else* statement is often used to test validity of data prior to use
  ➢ Check for possible bad situations during execution
    – Divide by zero
    – Square-root of negative number

```c
    double num;
    printf("Enter a non-negative number: ");
    scanf("%lf", &num);
    if (num >= 0)
        printf(The square root is: %f\n", sqrt(num));
    else
        printf("Input Error: Number is negative.\n");
    /* end if */
```

➢ Performs error checking

➢ Style comment: Note that non-error case appears first, error case appears second -- provides visual clue that error case is the uncommon one.
Conditional Branching - Nested if Statements

- The if-else statement can be contained within other if-else statements

\[
\text{if (} <\text{selection expression1}> \text{)} \\
\hspace{1cm} \text{if (} <\text{selection expression2}> \text{)} \\
\hspace{2cm}\text{statement1;} \quad /* \text{Executes if expression1 and expression2 are T} */ \\
\hspace{1cm} \text{else} \\
\hspace{2cm}\text{statement2;} \quad /* \text{Executes if exp1 = T and exp2 = F} */ \\
\hspace{1cm} \text{else} \\
\hspace{2cm}\text{statement3;} \quad /* \text{Executes if expression1=F} */ \\
\text{/* end if */}
\]
Nested if Statements - Example

- Write a function to determine which of two circles is bigger, given their radii. If first circle is bigger, return 1; if second circle is bigger, return 2; if invalid input, return 0.

```c
int bigger_circle (float radius1, float radius2)
{
    /* the radii must be greater than zero */
    if ((radius1 > 0) && (radius2 > 0))
        if (radius1 > radius2)
            return 1;
        else
            return 2;
    /* end inner if */
    else
        return 0; /* input error */
    /* end outer if */
}
```
Use of *if-else* Statements

- C syntax rule for the *if-else* statement: an *else* is associated with the closest unassociated *if*

```c
if (x == 10)
    if (y > 5)
        z = z * z;
else
    z = 0;
```

- Without this rule, it would be unclear whether *else* associates with outer *if* or inner *if*

- Exercise 1: Use braces to ensure *else* is associated with inner *if*
- Exercise 2: Use braces to force *else* to be associated with outer *if*
Conditional Branching - Multi-way Selection Statement

- Some programs require testing for multiple conditions; e.g., perform different actions based on the value of a variable
- The \texttt{if-else if} statement performs multi-way selection
- Performs a series of tests, usually based on the same variable

\begin{verbatim}
If (light == green)
go();
else if (light == yellow)
slow_down();
else if (light == red)
stop();
else
  (printf "light is broken\n");
\end{verbatim}
Nested *if* and *else-if* Statements - Example

- Write a function that accepts three integers, and returns the one that has the smallest value

```c
int min (int a, int b, int c);
{
    if (a < b)
        if (a < c)
            return a; /* a < b and a < c */
        else
            return c; /* a < b and c < a */
    /* end inner if */
    else if (b < c);
        return b; /* b < a and b < c */
    else
        return c; /* b < a and c < b */
    /* end outer if */
}
```
Conditional Branching - Multi-way Selection Statement

- Caution: Same variable is not required for each condition
- Offers flexibility as well as a way to get into trouble
- Completeness issue

```c
if (light == green)
    go();
else if (road == wet)
    slow_down();
else if (light == red)
    stop();
else
    (printf "light is broken\n");
```

- What case is missing? What happens if that case is TRUE?
Conditional Branching - The *switch* Statement

- Performs a series of tests based on the same variable

  ```
  switch (<expression>) {
    case <label-1>: statement1; /* statement1 is optional */
    case <label-2>: statement2; /* statement2 is optional */
    ...
    default: statementN; /* optional */
  }
  ```

- Unlimited number of execution paths based on value of single expression

- Execution continues where label = expression

- If none of the case labels match the *switch* expression, program executes at *default*

- If no *default* label exists, program exits the *switch* statement
The `switch` Statement - Example

- Driving directions to the airport

```java
switch (location) {
    case MIT: walk_to_Kendall();
    case kendall: board_redline();
    case redline: switch_greenline();
    case greenline: switch_blue_line();
    default: ask_directions();
}
```
Controlling Exit out of the `switch` Statement

- To force an exit from inside a `switch` statement, use `break` or `return`
- Break causes a jump to next line that follows the `switch` statement

```
switch (<expression>) {
    case <label-1>: statement1; break;
    case <label-2>: statement2; break;
    case <label-3>: statement3; break;
    default: statement4; break;
}
```

```
statement5;
```

- Note: No braces are required for multiple statements within a case condition
Example of switch Statement

• switch can be used to check for multiple matches of a variable

    switch (thrust_state)
    {
        case -1:
            printf ("Fuel tank empty.");
            break;
        case 0:
            printf ("Thrust is off.");
            break;
        default:
            printf ("Thrust is %d.", thrust_state);
            break;
    }
Review

• You now know how to write programs using simple selection structures: *if, if-else, if-else if, switch*

• Friday will be on loops - Chapter C9.

• Problem Set 1 has been graded. Pick up in TA office: 33-112