Can you figure them out?

- 26 L of the A (26 Letters of the Alphabet)
- 24 H in a D
- 7 W of the W
- 12 S of the Z
- 18 H on a G C

Branching Statements
Enumerated Types

Luke Sathrum – CSCI 20

Today's Class

- Branching Statements
  - `if`
  - `else`
  - `if else-if else` Statement
  - `switch`
- Enumeration / Enumerated Types

Branching Statements: If and If-Else

if-else Statement

- Choose between 2 alternative statements
- Based on a Boolean Expression
- Syntax

```
if (Boolean Expression)
    Yes/true Statement;
else
    No/false Statement;
```

`if`-`else` Statement

```
if (count > 5)
  cout << "The count is greater than 5.";
else
  cout << "The count has not reached 5.";
```

- Note
  - Boolean Expression is in parentheses
  - Only one statement after the `if` or `else`
  - The statement is moved over (2 spaces)

`if-else` Example

- We need to compute a week’s salary
- Must take into account overtime
- Up to 40 hours formula

```
    rate * hours
```

- Over 40 hours formula

```
    (rate * 40) + (1.5 * rate * (hours – 40))
```

- How do we do this?

`if-else` Example Code

```
if (hours > 40)
  gross_pay = (rate * 40) + (1.5 * rate * (hours - 40));
else
  gross_pay = rate * hours;
```

- Try it
  1. `hours = 40; rate = 30;`
  2. `hours = 50; rate = 30;`

Compound Statements

- We use braces **{ }** when we want more than one statement after the **if** or the **else**

```
if (hours > 40) {
    gross_pay = (rate * 40) + (1.5 * rate * (hours - 40));
     cout << "You worked a long week.";

} else {
    gross_pay = rate * hours;
    cout << "Ever think about working overtime?";
}
```

Omitting the `else`

- Sometime you only want to execute one of the alternatives in an `if`-`else` statement
- We can use an `if` statement to do this

```
if (sales >= minimum)
  salary = salary + bonus;
```

- If the Boolean Expression is `false` then the `if` statement is not executed

Omitting the `else` - Note

- Don’t forget only the 1st statement after the `if` is executed unless you have the statements in `{ }`

```
if (sales >= minimum)
   salary = salary + bonus;
   cout << "You are receiving a bonus";
```

- What happens if `sales >= minimum` is `true`?
- What if it is `false`?

Omitting the else - Note

- How you would actually write it

```
if (sales >= minimum) {
  salary = salary + bonus;
  cout << "You are receiving a bonus";
}
```

Summary

- You can make decisions using **Branching Statements**
  - `if` statement
  - `if-else` statement
- They decide based on Boolean Expressions
- They only work for the 1st line of code following them
  - Can use curly braces for more than one line

Sample Code

- **`if`** and **`if-else`** statements
  - `if_else.cpp`

Branching Statements: If, Else-If and Else

`if else-if else`

- Make a decision on more than one Boolean Expression
- We add the following to our syntax
    - `else if (Boolean_Expression)`
- Goes through `if else-if else` until an expression is `true`
- Goes to `else` if no expression evaluates to `true`

if else-if else

- We'll add another case to our payroll example

```
if (hours > 40)
 gross_pay = (rate * 40) + (1.5 * rate * (hours-40));
else if (hours > 0)
 gross_pay = rate * hours;
else
  cout < "You did not work any hours this week";
```

Notes on `if else-if else`

- There is a space between the `else` and the `if`
- You can use braces with these as well
- The final `else` can be omitted

Summary

- Can use an `if else-if else` to make a decision on more than one Boolean Expressions
- The `else` is not needed

Sample Code

- `if else-if else` Statement
  - `if_else-if_else.cpp`

Branching Statements: Switch Statements and Enumerated Types

`switch` Statement

- Like the `if else-if else` it can implement multiway branching
- Very useful for menu options

switch Statement Syntax

```
switch (Controlling Expression) {
  case literal_1:
    statement_1;
    statement_2;
    break;
  case literal_2:
    statement;
    break;
  default:
    statement;
}
```

switch Statement Example

```
int vehicle_class;
double toll;
cout << "Enter vehicle
class: ";
cin >> vehicle_class;
```

```
switch (vehicle_class) {
  case 1:
    toll = 0.50;
    break;
  case 2:
    toll = 1.50;
    break;
  default:
    cout << "Unknown
vehicle!";
}
```

Notes on Switch Statements

- Use **break** statement to end case
- **default** statement to handle all other cases
- You can combine 2 cases

```
case 'A':
case 'a':
  cout << "Excellent. "
         << "No need to take the final";
  break;
```

Enumerated Types

- Value is defined by a list of constants of type **`int`**
- Handy when we use **`switch`** statements
- Use only for labels, don’t do arithmetic with them
- Name the labels like constants
- Example

```
enum Direction {kNorth = 1, kSouth = 3, kEast = 5, kWest = 7};
```

Enumeration Types

- If we leave off the assignment statements then the values are assigned in order starting at 0
- Example

```
enum Direction {kNorth, kSouth, kEast, kWest};
```

is the same as

```
enum Direction {kNorth = 0, kSouth = 1, kEast = 2, kWest = 3};
```

Summary

- `switch` statements are great for menu choices
- They have a few keyword associated with them
  - `case`
  - `break`
  - `default`
- Don’t forget they have curly braces
- We can name a group of literals using enumeration

Sample Code

- **`switch`** statement in action with enumerated types
  - `switch.cpp`

Review

- Branching Statements
  - if
  - if-else
  - if else-if else
  - switch
- Enumerated Types