3.10 Switch statements

Switch statement

A ***switch*** statement can more clearly represent multi-branch behavior involving a variable being compared to constant values. The program executes the first ***case*** whose constant expression matches the value of the switch expression, executes that case's statements, and then jumps to the end. If no case matches, then the ***default case*** statements are executed.

PARTICIPATION ACTIVITY 3.10.1: Switch statement.

1 2 3 2x speed

```
switch (a) {
  case 0:
     // Print "zero"
     break;

  case 1:
     // Print "one"
     break;

  case 2:
     // Print "two"
     break;

   default:
     // Print "unknown"
     break;
}
```

Output

a = 1 → one

a = 5 → unknown

If no case matches, then the default case statements are executed.

Feedback?

PARTICIPATION ACTIVITY 3.10.2: Switch statement.

numItems and userVal are int types. What is the final value of numItems for each userVal?

```
switch (userVal) {
   case 1:
      numItems = 5;
      break;

   case 3:
      numItems = 12;
      break;

   case 4:
      numItems = 99;
      break;

   default:
      numItems = 55;
      break;
}
```

1) userVal = 3;

12

Check Show answer

Correct

12

The case with expression 3 will execute numItems = 12 and then break.

2) userVal = 0;

55

Check Show answer

Correct

55

No case expression is 0, so the default case executes numItems = 55.

3) userVal = 2;

55

Check Show answer

Correct

55

No case expression is 2, so the default case executes numItems = 55. The fact that 2 is between 1 and 3 is irrelevant.

Feedback?

Multi-branch if-else statement

A switch statement can be written using a multi-branch if-else statement, but the switch statement may make the programmer's intent clearer.

```
if (dogYears == 0) {           // Like case 0
   // Print 0..14 years
}
else if (dogYears == 1) {      // Like case 1
   // Print 15 years
}
...
else if (dogYears == 5) {      // Like case 5
   // Print 37 years
}
else {                         // Like default case
   // Print unknown
}
```

Switch statement general form

The switch statement's expression should be an integer or char. The expression should not be a string or a floating-point type. Each case must have a constant expression like 2 or 'q'; a case expression cannot be a variable.

The order of cases doesn't matter assuming break statements exist at the end of each case. The earlier program could have been written with case 3 first, then case 2, then case 0, then case 1, for example (though that would be bad style).

Good practice is to always have a default case for a switch statement. A programmer may be sure all cases are covered only to be surprised that some case was missing.

Construct 3.10.1: Switch statement general form.

```
switch (expression) {
   case constantExpr1:
      // Statements
      break;

   case constantExpr2:
      // Statements
      break;

   ...

   default: // If no other case matches
      // Statements
      break;
}
```

Feedback?

Figure 3.10.1: Switch example: Estimates a dog's age in human years.

```
#include <iostream>
using namespace std;

/* Estimates dog's age in equivalent human years.
   Source: www.dogyears.com
*/

int main() {
   int dogAgeYears;

   cout << "Enter dog's age (in years): ";
   cin  >> dogAgeYears;

   switch (dogAgeYears) {
      case 0:
         cout << "That's 0..14 human years." << endl;
         break;

      case 1:
         cout << "That's 15 human years." << endl;
         break;

      case 2:
         cout << "That's 24 human years." << endl;
         break;

      case 3:
         cout << "That's 28 human years." << endl;
         break;

      case 4:
         cout << "That's 32 human years." << endl;
         break;

      case 5:
         cout << "That's 37 human years." << endl;
         break;

      default:
         cout << "Human years unknown." << endl;
         break;
   }

   return 0;
}
```

```
Enter dog's age (in years): 4
That's 32 human years.

...

Enter dog's age (in years): 17
Human years unknown.
```

Feedback?

zyDE 3.10.1: Switch statement: Numbers to words.

Extend the program for dogYears to support age of 6 to 10 years. Conversions are 8:52, 9:57, 10:62.

Load default template

Load default template...

```
#include <iostream>
using namespace std;

/* Estimates dog's age in equivalent huma
   Source: www.dogyears.com
*/

int main() {
   int dogAgeYears;

   cout << "Enter dog's age (in years): "
   cin  >> dogAgeYears;

   switch (dogAgeYears) {
      case 0:
         cout << "That's 0..14 human year
         break;

      case 1:
         cout << "That's 15 human years."
         break;
```

7

Run

Feedback?

Omitting the break statement

Omitting the ***break*** statement for a case will cause the statements within the next case to be executed. Such "falling through" to the next case can be useful when multiple cases, such as cases 0, 1, and 2, should execute the same statements.

The following extends the previous program for dog ages less than 1 year old. If the dog's age is 0, the program asks for the dog's age in months. Within the `switch (dogAgeMonths)` statement, "falling through" is used to execute the same display statement for several values of dogAgeMonths. For example, if dogAgeMonths is 0, 1 or 2, the same statement executes.

A common error occurs when the programmer forgets to include a break statement at the end of a case's statements.

Figure 3.10.2: Switch example: Dog years with months.

```
Enter dog's age (in years): 0
Enter dog's age in months: 7
That's 5..9 human years.

...

Enter dog's age (in years): 4
FIXME: Do earlier dog year cases.
```

```
#include <iostream>
using namespace std;

int main() {
   int dogAgeYears;
   int dogAgeMonths;

   cout << "Enter dog's age (in years): ";
   cin >> dogAgeYears;

   if (dogAgeYears == 0) {
      cout << "Enter dog's age in months: ";
      cin  >> dogAgeMonths;

      switch (dogAgeMonths) {
         case 0:
         case 1:
         case 2:
            cout << "That's 0..14 human months." << endl;
            break;

         case 3:
         case 4:
         case 5:
         case 6:
            cout << "That's 1..5 human years." << endl;
            break;

         case 7:
         case 8:
            cout << "That's 5..9 human years." << endl;
            break;

         case 9:
         case 10:
         case 11:
         case 12:
            cout << "That's 9..15 human years." << endl;
            break;

         default:
            cout << "Invalid input." << endl;
            break;
      }
   }
   else {
      cout << "FIXME: Do earlier dog year cases." << endl;
      switch (dogAgeYears) {
      }
   }

   return 0;
}
```

Feedback?

PARTICIPATION ACTIVITY 3.10.3: Switch statement.

userChar is a char and encodedVal is an int. What will encodedVal be for each userChar value?

```
switch (userChar) {
   case 'A':
      encodedVal = 1;
      break;

   case 'B':
      encodedVal = 2;
      break;

   case 'C':

   case 'D':
      encodedVal = 4;
      break;

   case 'E':
      encodedVal = 5;

   case 'F':
      encodedVal = 6;
      break;

   default:
      encodedVal = -1;
      break;
}
```

1) userChar = 'A'

1

Check Show answer

Correct

1

The first case assigns 1 to encodedVal and then breaks out of the switch statement.

2) userChar = 'B'

2

Check Show answer

Correct

2

The second case assigns 2 to encodedVal and then breaks out of the switch statement.

3) userChar = 'C'

4

Check Show answer

Correct

4

Because the case for 'C' has no break statement, execution falls through to the next case 'D' and executes that case's statements (without comparing userChar to 'D'). That case assigns 4 to encodedVal and then breaks out of the switch.

4) userChar = 'E'

6

Check Show answer

Correct

6

The statement in case 'E' assigns 5 to encodedVal. But that case has no break statement. So after

assigning 5, execution proceeds to the statements in case 'F', which assigns 6 to encodedVal, and then breaks out of the switch.

5) userChar = 'G'

-1

Check Show answer

Correct

-1

Because no explicit case matches 'G', execution proceeds to the default case, which assigns -1 to encodedVal.

Feedback?

CHALLENGE ACTIVITY 3.10.1: Rock-paper-scissors.

Write a switch statement that checks nextChoice. If 0, print "Rock". If 1, print "Paper". If 2, print "Scissors". For any other value, print "Unknown". End with newline.

```
9     /* Your solution goes here  */
10    switch (nextChoice) {
11    case 0:
12      cout << "Rock"<< endl;
13       break;
14
15    case 1:
16       cout << "Paper"<< endl;
17       break;
18
19    case 2:
20       cout << "Scissors"<< endl;
21       break;
22
23    default:
24       cout << "Unknown"<< endl;;
25       break;
26 }
27
28    return 0;
29 }
```

Run ✓ All tests passed

✓ Checking for nextChoice = 0

Your output `Rock`

✓ Checking for nextChoice = 1

Your output `Paper`

✓ Checking for nextChoice = 2

Your output `Scissors`

✓ Checking for nextChoice = 3

Your output `Unknown`

Feedback?

CHALLENGE ACTIVITY 3.10.2: Switch statement to convert letters to Greek letters.

Write a switch statement that checks origLetter. If 'a' or 'A', print "Alpha". If 'b' or 'B', print "Beta". For any other character, print "Unknown". Use fall-through as appropriate. End with newline.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5     char origLetter;
6
7     cin >> origLetter;
8
9     /* Your solution goes here  */
10    switch (origLetter) {
11   case 'A':
12    case 'a':
13      cout << "Alpha"<< endl;
14       break;
15    case 'B':
16    case 'b':
17       cout << "Beta"<< endl;
18       break;
19
20    default:
21       cout << "Unknown"<< endl;;
```

Run ✓ All tests passed

✓ Testing with origLetter = 'a'

Your output `Alpha`

✓ Testing with origLetter = 'A'

Your output `Alpha`

✓ Testing with origLetter = 'b'

Your output `Beta`

✓ Testing with origLetter = 'B'

Your output `Beta`

✓ Testing with origLetter = 'z'

Your output `Unknown`

Feedback?