Chapter 3

Decisions and Loops

Relational Operators

<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

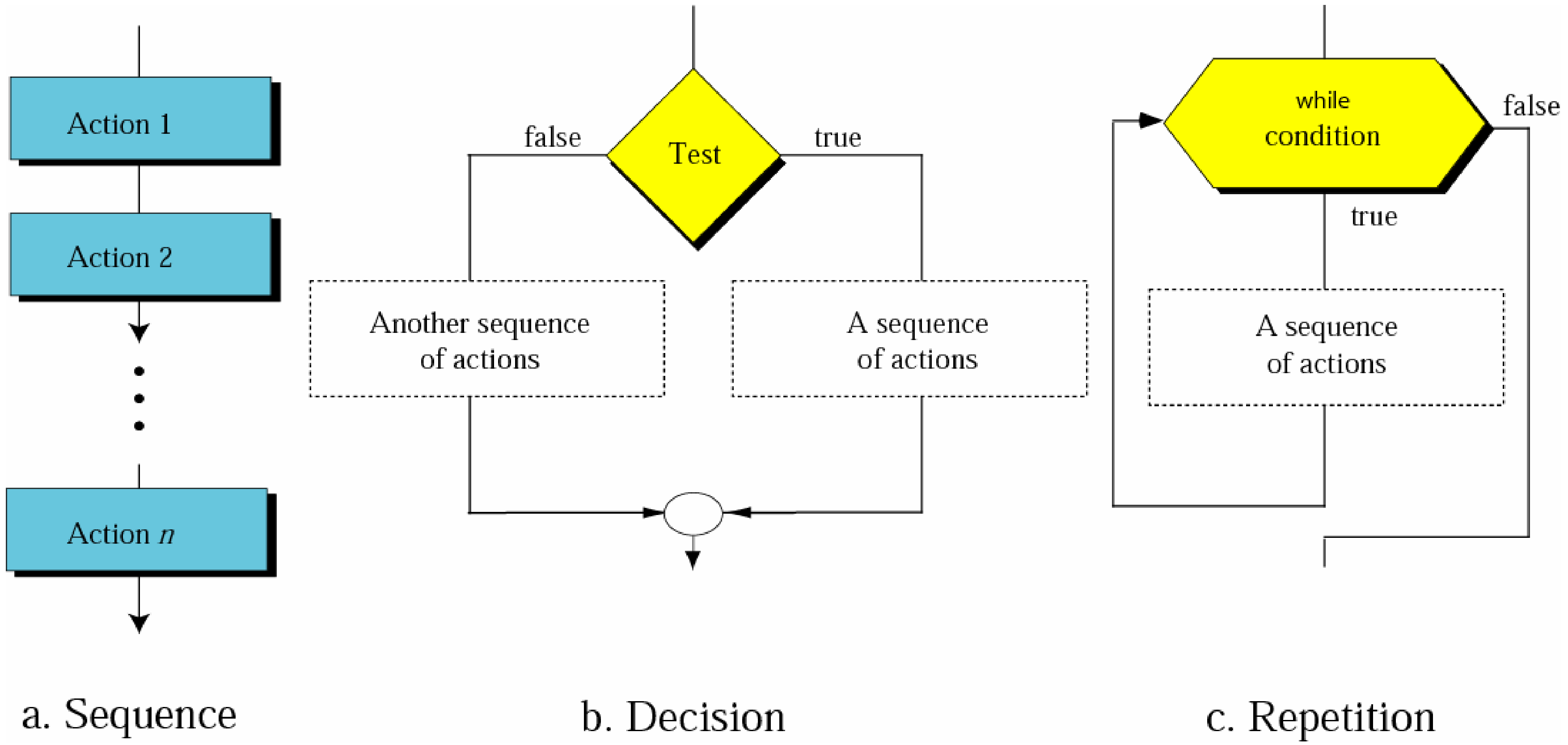
- Compare the values of two operands, and return
  - true
  - false

Example of Logical Expressions

- Suppose two integer variables
  i = 10, j = -5
- The following expressions are all true:
  - i > j
  - i != j
  - j > -8
  - i <= j + 15
- cout << (i < j)
  - Displays “0”   (implicit cast)
- cout << (i > j)
  - Displays “1”   (implicit cast)

Flowcharts for three constructs

- Review Forouzan's Chapter 8



a. Sequence b. Decision c. Repetition

The if Statement

- The condition to be tested appears in parenthesis

  -
    ```
    if (letter == ‘A’)
        cout << “Apple”;
    ```
- A block of statements between braces could replace the single statement.

  -
    ```
    if (letter == ‘A’)
    {
        cout <<  “Apple”;
        letter = ‘a’;
    }
    ```
- Nested if Statement (P.118)

The if … else … Statement

```
if (number % 2)
   cout << “Odd”
        << endl;
else
   cout << “Even”
        << endl;
```

- The condition express
  - `(number % 2)`
- is equivalent to
  - `(number %2 != 0)`
- A non-zero value is interpreted as `true` (implicit cast).
- A zero value result casts to `false`.

Logical Operators

-
  ```
  if ((letter >= ‘A’) && (letter <= ‘Z’))
      cout << “This is a capital letter.”;
  ```

&&	Logical AND
||	Logical OR
!	Logical negation (NOT)

The Conditional Operator

- `c = a>b ? a : b ;`
  // set c to the maximum of
  // a and b

```
if (a > b)
  c = a;
else
  c = b;
```

- Sometimes called the **ternary operator**.
  - condition ? expression1 : expression2

Output Control

```
cout << endl
     << “We have ” << nCakes
     <<  “cake”
     << ((nCakes > 1) ? “s.” : “.”)
     << endl;
```

- nCakes = 1
  - We have 1 cake.
- nCakes = 2
  - We have 2 cakes.

The switch Statement

```
if (option >= ‘a’ && option <= ‘z’)
   switch (option)
   {
     case ‘a’:
             cout << “Append”;
             break;
     case ‘d’:
             cout << “Delete”;
             break;
     case ‘q’:
             cout << “Quit”;
             break;

     default: cout << “You entered a wrong option.”;
   }
```

Saves the Trouble of Multiple-if

```
if (option == ‘a’)
  cout << “Append”;
else
  if (option == ‘d’)
     cout << “Delete”;
  else
     if (option == ‘q’)
          cout << “Quit”;
     else
          cout << “You entered a”
               << “ wrong option.”;
```

Ex3_06.cpp (P.133)

- An elegant example to demonstrate the power of C language.

```
switch (letter * (letter >= ‘a’ && letter <= ‘z’))
{
  case ‘a’:
  case ‘e’:
  case ‘i’:
  case ‘o’:
  case ‘u’: cout << “You entered a vowel.”;
           break;
  case 0: cout << “That is not a small letter.”;
           break;
  default: cout << “You entered a consonant.”;
}
```

Unconditional Branching

```
myLabel: cout << “myLabel is here”;
.
.
.
goto myLabel;
```

- Whenever possible, you should avoid using `gotos` in your program.

Loop (Ex3_07 in P.135)

```
  int i = 0, sum = 0;
  const int max = 5;

  i = 1;
KevinLabel:
  sum +=i;
  if (++i <= max)
     goto KevinLabel;

  cout << “sum=“ << sum << endl
     << “i = “ << i << endl;
```

i = [illegible], sum = [illegible]

The for Loop

```
for (i=1; i<=6; i++)
  cout << i << endl;
```

- 1
- 2
- 3
- 4
- 5
- 6

Using The for Loop for Summation

```
int i = 0, sum = 0;
  const int max = 5;

for (i=1; i<=max; i++)
  sum += i;
```

i = 5, sum = 15

Nested for Loop

```
const int N = 5;
int i, j;
for (i=1; i<=N; i++)
{
   for (j=1; j<=i; j++)
        cout << '*';
   cout << endl;
}
```

```
*
**
***
****
*****
```

Increment/Decrement of the Counter

```
for (i=1; i<=N; i++)
{
    for (j=1; j<=i; j++)
         cout << '*';
    cout << endl;
}

for (i=N; i>=1; i--)
{
    for (j=1; j<=i; j++)
         cout << '*';
    cout << endl;
}
```

```
*
**
***
****
*****
*****
****
***
**
*
```

Variation on the for Loop

- Declare the counter i within the loop scope
  - for (int i; i<=max; sum+= i++)
- The loop statement can be empty.
  - for (i=1; i<=max; sum+= i++)
        ;
- A block of statements between braces could replace the single loop_statement.
- Use the comma operator to specify several expressions:
  - for (i=0, power=1; i<=max; i++, power *=2)

break vs. continue

- The keyword `continue` allows you to skip the remainder of the current iteration in a loop and go straight to the next iteration.
- The keyword `break` provides an immediate exit from a loop.
- (See P.140 and P.142)

Other Types of Loop

- The while loop
  - while (condition)
    ```
    while (condition)
        loop_statement;
    ```
- The do-while Loop
  - do
    ```
    do
    {
        loop_statements;
    } while (condition);
    ```
  - Always executed at least once.
- You may see infinite loops like
  - while (true)
    ```
    while (true)
    {
        …
    }
    ```
  - while (1)
    ```
    while (1)
    {
        …
    }
    ```

Print as Hexadecimal (P.145)

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

int main()
{
   int a = 65;
   cout << hex << a << endl;
   cout << dec << a << endl;
   printf("%x \t %d \n", a, a);
}
```

C++/CLI Programming

- Reading Key Presses
  - Console::ReadKey
  - Ex3_16.cpp in P.154

- ConsoleKeyInfo class has three properties
  - Key – the key that was pressed
  - KeyChar – Unicode character code for the key
  - Modifiers – Alt, Shift, Control