Supplements for Chapter 4, Control Statements: Part 2

C++ How to Program, Late Objects Version, 7/e

4.5 while vs. do…while

```
total = 0; grade = 0;
counter = 0;
cout << "Enter grade, ";
cout << "-1 to end: ";
cin >> grade;
while (grade != -1) {
  total = total + grade;
  counter = counter + 1;
  cout << "Enter grade, ";
  cout << "-1 to end: ";
  cin >> grade;
}
```

- Duplicate input statements !!
- Initial values are all zero.

```
total = 0; grade = 0;
counter = -1;
do {
  total = total + grade;
  counter = counter + 1;
  cout << "Enter grade, ";
  cout << "-1 to end: ";
  cin >> grade;
} While (grade != -1);
```

- No duplicate input statements !!
- Initial counter starts from -1.

Appendix Usage of Infinite Loop

- Infinite loops are helpful when the termination condition is generated inside the loop

```
while (1) {
  ......
  ans = a * b;
   if (ans == 0) break;
   ......
}
```

1 (non-zero value) means always TRUE

- Should be used with `break` to terminate the loop
  - Make sure the condition will eventually become TRUE
- If sentinel-controlled loop can be used instead, use it !!
  - Infinite loops are not easy to debug

Appendix Nested Loops

- Nested loops (loop inside a loop) are allowed in C/C++

```
for (i=0; i<n; i++) {
    ......
    for (j=0; j<m; j++)
    {  ......  }
}
```

outer loop: run inner loop for n times

inner loop: repeated actions

- Similar to migrating 1-dimensional problems into multi-dimensional problems
  - One loop: f(0), f(1), f(2), …
  - Two loops: f(0,0), f(0,1), …, f(0,n), f(1,0), f(1,1), …
- The most important thing:
  - Obtain the changing rules of the running index

Nested Loops: Examples (1/3)

- Execute multi-dimensional operations

```
for (i=1; i<=4; i++) {
    cout << "i=" << i << ":\n";
    for (j=1; j<=3; j++) {
       cout<<i<<"x"<<j<<"="<<i*j;
    }
    cout << endl;
}
```

```
i=1:
1x1=1 1x2=2 1x3=3
i=2:
2x1=2 2x2=4 2x3=6
i=3:
3x1=3 3x2=6 3x3=9
i=4:
4x1=4 4x2=8 4x3=12
```

- Please pay special attention to the index changing sequence
  - Column first in this case
    (1,1) -> (1,2) -> (1,3) -> (2,1) -> ...
  - So, column is changed in the inner loop

Nested Loops: Examples (2/3)

- Repeat a loop for n times

```
for (i=1; i<=5; i++) {
   for (j=1; j<=6; j++)
   {  cout << "*";  }
   cout << endl;
}
```

6 stars

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

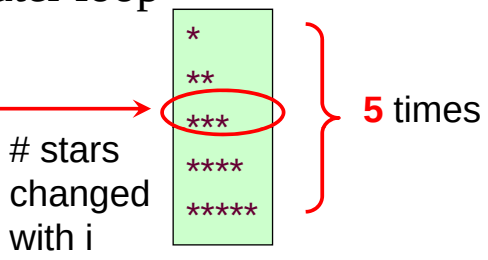
5 times

- Inner loop control the repeated actions
  - Print 6 stars in this case
- Outer loop control the number of times
  - Print 5 rows of stars in this case

Nested Loops: Examples (3/3)

- Inner loop is controlled by outer loop

```
for (i=1; i<=5; i++)
{  for (j=1; j<=i; j++)
   {  printf("*");  }
   printf("\n");
}
```



- Outer loop control the number of rows
  - Print 5 rows of stars in this case
- Inner loop control the number of stars
  - Change its termination condition
  - i=1 --> for (j=1;j<=1;j++) --> 1 star
  - i=2 --> for (j=1;j<=2;j++) --> 2 stars
  - .....

break in Nested Loops

- In nested loops, *break/continue* can only affect the most inner loop where the *break/continue* stands

```
for (i=1; i<=5; i++)
{  for (j=1; j<=3; j++)
   {  printf("(%d,%d) ",i,j);
      if (i==3) break;
   }
   printf("\n");
}
```

break inner loop j only

(1,1) (1,2) (1,3)
(2,1) (2,2) (2,3)
(3,1)
(4,1) (4,2) (4,3)
(5,1) (5,2) (5,3)

Jump out the inner loop and start from here

- If *break* is used to skip the following *switch* statements, it has no effects on the outside loop
  - One-time use only

```
18  /* loop until user types end-of-file key sequence */
19  while ( ( grade = getchar() ) != EOF ) {
20
21     /* determine which grade was input */
22     switch ( grade ) { /* switch nested in while */
23
24        case 'A': /* grade was uppercase A */
25        case 'a': /* or lowercase a */
26           ++aCount; /* increment aCount */
27           break; /* necessary to exit switch */
28
```