Nested Loops

- Sometimes you may want to place one loop inside another.
- You might want to count the number of occupants in each house on a street.
    - ➔ Step from house to house, and for each house you count the number of occupants.
- Going through all the houses could be an outer loop, and for each iteration of the outer loop you would have an inner loop that counts the occupants.

Example

```
for(int i = 1 ; i <= count ; ++i)
{
  sum = 0;                    // Initialize sum for the inner loop

  // Calculate sum of integers from 1 to i
  for(int j = 1 ; j <= i ; ++j)
    sum += j;

  printf("\n%d\t%d", i, sum);      // Output sum of 1 to i
}
```

Another Example (while inside a for)

```
for(int i = 1 ; i <= count ; ++i)
 {
   sum = 1;                  // Initialize sum for the inner loop
   j=1;                     // Initialize integer to be added
   printf("\n1");

   // Calculate sum of integers from 1 to i
   while(j < i)
   {
     sum += ++j;
     printf(" + %d", j);         // Output +j – on the same line
   }
   printf(" = %d", sum);         // Output  = sum
 }
```

Continue Statements

- Sometimes a situation arises where you do not want to end a loop, but you want to skip the current iteration
- The continue statement in the body of a loop does this.
    - ➔ All you need to do is use the keyword “continue;” in the body of the loop
- An advantage of using continue is that it can sometimes eliminate nesting or additional blocks of code .

- ➔ Can enhance readability when the statements are long or are deeply nested already.
- Don’t use continue if it complicates rather than simplifies the code.

Continue Example

```
enum Day { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday};

for(enum Day day = Monday; day <= Sunday ; ++day)
{
  if(day == Wednesday)
    continue;

  printf("It's not Wednesday!\n");
  /* Do something useful with day */
}
```

Break Statements

- Normally, after the body of a loop has been entered, a program executes all the statements in the body before doing the next loop test.
  - ➔ We learned how continue works.
  - ➔ Another statement named break alters this behavior.
- The break statement causes the program to immediately exit from the loop it is executing .
  - ➔ Statements in the loop are skipped, and execution of the loop is terminated.
  - ➔ If the break statement is inside nested loops, it affects only the innermost loop containing it .
  - ➔ Use the keyword “break;”
- Break is often used to leave a loop when there are two separate reasons to leave.
- Break is also used in switch statements.

Break example

```
while ( p > 0)
{
   printf("%d\n", p);
   scanf("%d", &q);
   while( q > 0)
   {
      printf("%d\n",p*q);
      if (q > 100)
         break;          // break from inner loop
      scanf("%d", &q);
   }
   if (q > 100)
      break;             // break from outer loop
   scanf("%d", &p);
}
```