More Examples
`while-loop` and `for-loop`

Due to limited lecture time, please find more examples in books and try to learn more unless you have strong programming background

Outline

- **Example 1:**
  Find the largest number among the input values
- **Example 2:**
  Print out numbers with alternating patterns
- **Example 3:**
  Compute the sum of digits of a positive integer
- **Example 4**:
  Find a solution by testing all possible values

Example 1

- **Objective:** To read N integers from the user one by one, and print out the largest number among the inputs.
  - The value of N will first be read from the user.
  - The N numbers are separated by whitespace characters.

Example 1: Any idea?

- Read the input values from user one by one.
- Use a variable to remember the largest value ever read so far. Name it as "max".
- For each input value, if it is larger than "max", replace the value of "max" by this input value.
- How do we choose an initial value for "max" so that its value will get updated when we encounter a larger input?
  - Approach 1: Select the smallest possible value
  - Approach 2: Use the 1st input value

Example 1: Solution (Approach #1)

```
1   #include <stdio.h>
2   #include <limits.h>
3
4   int main( void )
5   {
6       int max   ,         // To remember the largest input value
7           input ,         // To hold each input value temporarily
8           N     ,         // Number of inputs to be read from user
9           i     ;
10
11      printf( "N = ? "  );
12      scanf( "%d" , & N );
13
14      max = INT_MIN ;     // Start "max" with the smallest value,
15                          // we will update its value when we
16                          // encounter a larger input.
17
```

Need to add "`#include <limits.h>`" to use **`INT_MIN`** (a predefined constant that represents the smallest possible value of type `int`)

Example 1: Solution (Approach #1)

```
18      for ( i = 0 ; i < N ; i++ )
19      {
20          scanf( "%d" , & input );
21          if ( max < input )
22              max = input ;
23      }
24
25      printf( "The largest number is %d.\n" , max );
26      return 0 ;
27  }
```

```
N = ? 5
6
2
-10
99
11
The largest number is 99.
```

Example 1: Solution (Approach #2)

```
1   #include <stdio.h>
2
3   int main( void )
4   {
5       int max   ,         // To remember the largest input value
6           input ,         // To hold each input value temporarily
7           N     ,         // Number of inputs to be read from user
8           i     ;
9
10      printf( "N = ? "  );
11      scanf( "%d" , & N );
12
13      scanf( "%d" , & input );
14      max = input ;       // So far we have seen only one input,
15                          // so let it be the largest number.
16                          // We will update "max" when we
17                          // encounter a larger input.
```

Example 1: Solution (Approach #2)

```
18      // Process the remaining N-1 input values
19      for ( i = 0 ; i < N-1 ; i++ )
20      {
21          scanf( "%d" , & input );      // don't miss &
22          if ( max < input )
23              max = input ;
24      }
25
26      printf( "The largest number is %d.\n" , max );
27      return 0 ;
28  }
```

```
N = ? 5
6
2
-10
99
11
The largest number is 99.
```

Example 2

- **Objective:** To print out the first N numbers in the following number series:

  1 2 -3 -4 5 6 -7 -8 9 10 -11 -12 …

Notice that these numbers have the following pattern:

+ve +ve -ve -ve +ve +ve -ve -ve …

and the pattern "+ve +ve -ve -ve" repeats for every four numbers.

Example 2: Solution

```
1   int i , N ;
2
3   printf( "N = ? "  );
4   scanf( "%d" , & N );     // don't miss &
5
6   for ( i = 1 ; i <= N ; i++ )
7   {
8       if ( i % 4 == 1 || i % 4 == 2 )
9           printf( "%d " , i );
10      else
11          printf( "%d " , -i );
12  }
```

i	1	2	3	4	5	6	7	8	9	10	11	12	13
i % 4	1	2	3	0	1	2	3	0	1	2	3	0	1

The remainders, i % 4, also repeat every four numbers

Example 3

- **Objective:** To compute the sum of digits of a positive integer.

- This example aims to show how to use a while-loop to achieve the objective by repeatedly
  - Extracting the last digit from the number,
  - Adding the value of the extracted digit to a variable, and
  - Removing the last digit from the number.

Example 3: Solution

```
1   int i , num ;
2   int digitSum ;                  // To store the sum of digits of "num"
3
4   printf( "num = ? " );
5   scanf( "%d" , & num );          // don't miss &
6
7   digitSum = 0 ;
8   while ( num > 0 )               // Eventually num will become 0 in loop
9   {
10     digitSum += num % 10 ;       // Add the last digit of num to digitSum
11     num = num / 10 ;             // Remove the last digit from num
12  }
13
14  printf( "The sum of digits is %d.\n" , digitSum );
15
16
17
```

Note: We assume the input is a non-negative integer.

Example 4

- **Objective:** To find all possible sets of integers that satisfy the following equality:

$$x^2 + y^2 + z^2 = 1000000, \qquad 0 \le x, y, z \le 1000$$

- One possible (quick and dirty) solution is to try all possible values for x, y, and z

Example 4: Solution

What is its runtime complexity?
How about times we run the innermost body?

Any strategies to modify the code to improve its efficiently?

```
1   int x , y , z ;
2
3   for ( x = 0 ; x <= 1000 ; x++ )
4   {
5       for ( y = 0 ; y <= 1000 ; y++ )
6       {
7           for ( z = 0 ; z <= 1000 ; z++ )
8           {
9               if ( x*x + y*y + z*z == 1000000 )
10                  printf( "%d, %d, %d\n" , x , y , z );
11          }
12      }
13  }
```

Hint: when you reach the inner loop for y, your code should know the value of x. Given such x, what are the possible y value?

Suggestions

- The examples provided here are limited by the time we have in the class
- You should look for more examples to learn from…
  in the books, in the Internet, etc.