CS1100 – Introduction to Programming

Trimester 3, April – June 2021

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lecture 14

Testing if a number is prime

A number n is prime if it has no other divisors other than one and itself.

Algorithm: Check, for every number m in the range 2 to $n - 1$, whether m divides n or not. If none divides, then you can declare that it is a prime number. If one of them divides, then you can declare right away that is is a composite number.

Pseudocode:

- Start checking from 2 to $n - 1$.
- If any of the above divides n, declare "not prime!"
- Else declare "prime".

Testing if a number is prime

```
scanf("%d", &n);
i = 2; flag = 0;
while (i < n) {
    if (n % i == 0) {
       flag = 1;
       break;
    }
    i = i+1;
}
if (1 == flag)
   printf("not prime\n");
else
   printf("prime\n");
```

- see the initialization, termination.
- (1 == flag)
- use of break.

Nested For Loop for Finding Prime Numbers

Find the prime numbers from 2 to 100

```
#include <stdio.h>

int main () {

   /* local variable definition */
   int i, j;

   for(i = 2; i<100; i++) {

      for(j = 2; j <= (i/j); j++)
      if(!(i%j)) break; // if factor found, not prime
      if(j > (i/j)) printf("%d is prime\n", i);
   }

   return 0;
}
```

Finding min of n integers

- Take n from input.
- initialize counter to count n (in some way!)
- scan input, modify min (if needed).

Finding min of n integers

```
#include<stdio.h>
main() {
    int n; int currInt;
    int a; int min;

    scanf("%d",&n);
    a = 1;
    while (a <= n) {
       scanf ("%d", &currInt);
       if (a == 1) {
           min = currInt;
       }
       if (currInt < min) {
           min = currInt;
       }
       a++;
    }
    printf("min = %d\n", min);
}
```

Points to remember

- Is counter updated?
- Corner cases: a single input, no input?
- min occurs as the first or last element.

- When control is at the scanf statement, we are scanning the *a*-th input.
- Just before the statement a++; we have computed min of first *a* elements given by user.

Finding min of positive integers : terminated by a negative integer

```
#include<stdio.h>
main() {
    int n; int currInt;
    int min;

    scanf("%d",&currInt);
    min = currInt;
    while (currInt >= 0) {
       scanf ("%d", &currInt);
       if (currInt < min) {
           min = currInt;
       }
    }
    printf("min = %d\n", min );
}
```

What is the output of this program? Always gives a negative value.

Finding min of positive integers : terminated by a negative integer

```
#include<stdio.h>
main() {
    int n; int currInt;
    int min;

    scanf("%d",&currInt);
    min = currInt;
    while (currInt >= 0) {
       scanf ("%d", &currInt);
       if (currInt <  0) break;

       if (currInt < min) {
           min = currInt;
       }
    }
    printf("min = %d\n", min );
}
```

- What happens when first input is negative?
- Add a check in the end.

Finding GCD of two integers

Given positive integers x and y, output the GCD of x and y.

Idea

- Let z be min of x and y.
- for i = 1 to z
    - check if i divides both x and y.
    - output largest such i as gcd.

Finding GCD of two integers

Given positive integers x and y, output the GCD of x and y.

```
if (x < y)
   z = x;
else z = y;
// z contains min of x and y

gcd = 1; i = 1;
while (i<=z) {
    if ((x % i == 0) && (y % i == 0)) {
       gcd = i;
    }
    i++;
}
```

Finding GCD of two integers

Given x and y, output the GCD of x and y.

Idea2 by Euclid

- If y divides x we are done!
- Else there is a smaller problem to solve!

$$\boxed{\text{gcd(x, y)} = \text{gcd(x-y, y)}}$$

- Needs proof!

Finding GCD of two integers – Euclid's algorithm

$$\begin{aligned}
gcd(1034, 237) &= gcd(797, 237)\\
&= gcd(560, 237)\\
&= gcd(323, 237)\\
&= gcd(86, 237) && \text{next?}\\
&= gcd(86, 151)\\
&= gcd(86, 65)\\
&= gcd(21, 65)\\
&= gcd(21, 44)\\
&= gcd(23, 44)\\
&= gcd(23, 21)\\
&= gcd(2, 21)\\
\ldots &= 1
\end{aligned}$$

Finding GCD of two integers

Given x and y, output the GCD of x and y.

Idea2

by Euclid

- If x % y == 0, we are done!
- Else modify x and y suitably.
    - x = x % y;
    - What if x < y?
    - Exchange x and y.

Finding GCD of two integers

Euclid's algorithm

```
#include<stdio.h>
int main() {
    int x, y;
    int temp;
    scanf("%d %d", &x, &y);
    if (x < y) {
        temp = x; x = y; y = temp;
    }
    // Assume x  >= y.
    while ( x % y != 0) {
         x = x % y;
         printf ("x = %d, y = %d\n", x, y);
         if (x < y) {
            temp = x; x = y; y = temp;
         }
    }
    printf("gcd of input numbers is %d \n", y);
    return 0;
}
```

Learnings so far...

- Examples: Finding min of positive integers, testing primality, finding gcd using simple and Euclid's method.
- Our problems naturally needed loops.
- break is a useful way to terminate out of the loop.

A very important and useful learning: Power of a clever algorithm.