An elaborate programming example

- Uses ***while***, cannot be done using ***repeat***.
  - Can be done using ***for***.
- Algorithm is clever, correctness is not obvious.
- Exercise in algorithm design, including arguing correctness.

Euclid’s algorithm for GCD

- Greatest common divisor (GCD) of positive integers m, n : largest positive integer p that divides both m, n.
- “Standard” method: factorize m,n and multiply common factors.
- Euclid’s algorithm (2300 years old!) is different and much faster.
- Program based on Euclid’s method will be much faster than program based on factoring.

Euclid's Algorithm

Basic Observation: If d divides both m, n, then d divides m-n also, assuming m > n.

- Proof: m=ad, n=bd, so m-n=(a-b)d.

Also true: If d divides m-n and n, then it divides m too.

- Proof: m-n=cd, n=ed, so m = (c+e)d
- m, n, have the same common divisors as m-n,n.
- The largest divisor of m,n is also the largest divisor of m-n,n.

Insight:

- Instead of finding GCD(m,n), we might as well find GCD(n, m-n).

Example

GCD(3977, 943)
=GCD(3977-943,943) = GCD(3034,943)
=GCD(3034-943,943) = GCD(2091,943)
=GCD(2091-943,943) = GCD(1148,943)
=GCD(1148-943,943) = GCD(205, 943)

We should realize at this point that 205 is just 3977 % 943.
So we could have got to this point just in one shot by writing
GCD(3977,943) = GCD(3977 % 943, 943)

Example

Should we guess that GCD(m,n) = GCD(m%n, n)?

This is not true if m%n = 0, since we have defined GCD only for positive integers. But we can save the situation, as Euclid did.

Euclid's theorem: Let m,n>0 be positive integers. If n divides m then GCD(m,n) = n. Otherwise GCD(m,n) = GCD(m%n, n).

Example continued

GCD(3977,943)

= GCD(3977 % 943, 943)

= GCD(205, 943) = GCD(205, 943%205)

= GCD(205,123) = GCD(205%123,123)

= GCD(82, 123) = GCD(82, 123%82)

= GCD(82, 41)

= 41 because 41 divides 82.

Exercise

- Use Euclid’s algorithm to find the GCD of 26, 42.

Algorithm

- input: values M, N which are stored in variables m, n.
- iteration : Either discover the GCD of M, N, or find smaller numbers whose GCD is same as GCD of M, N.
- Details of an iteration:
  - At the beginning we have numbers stored in m, n, whose GCD is the same as GCD(M,N).
  - If n divides m, then we declare n to be the GCD.
  - If n does not divide m, then we know that GCD(M,N) = GCD(n, m%n).
  - So we have smaller numbers n, m%n, whose GCD is same as GCD(M,N)

Program for GCD

```
main_program{
    int m, n; cin >> m >> n;
    while(m % n != 0){
        int nextm = n;
        int nextn = m % n;
        m = nextm;
        n = nextn;
    }
    cout << n << endl;
}
// To store n, m%n into m,n, we cannot
// just write m=n; n=m%n;
// Can you say why?  Hint: take an example!
```

Remark

- We have defined variables ***nextm, nextn*** for clarity. We could have done the assignment with just one variable as follows.
  int r = m%n; m = n; n = r;
- It should be intuitively clear that in writing the program, we have followed the idea from Euclid's theorem.
- However, it is possible to make a mistake in "following the idea"...

Exercise: Find the mistake in this program

```
main_program{
    int m, n; cin >> m >> n;
    while(m % n != 0){
        int nextm = n;
        int nextn = m % n;
        m = nextn;
        n = nextm;
    }
    cout << n << endl;
}
```

What we discussed

- Euclid’s algorithm for determining the GCD of two positive integers.
- Program appears reasonably easy to write, but there is room to make mistakes.
- NEXT: How do we check correctness of our program and avoid mistakes.