CSE1142 –Repetitive Structures in C

Sanem Arslan Yılmaz

Some of the slides are from:
CMPE150 – Boğaziçi University
Deitel & Associates

Agenda

- Types of loops
- For loop
  - Syntax
  - Example
- While loop
  - Syntax
  - Example
- Do-while loop
  - Syntax
  - Example
- Break statement
- Continue statement

Types of loops

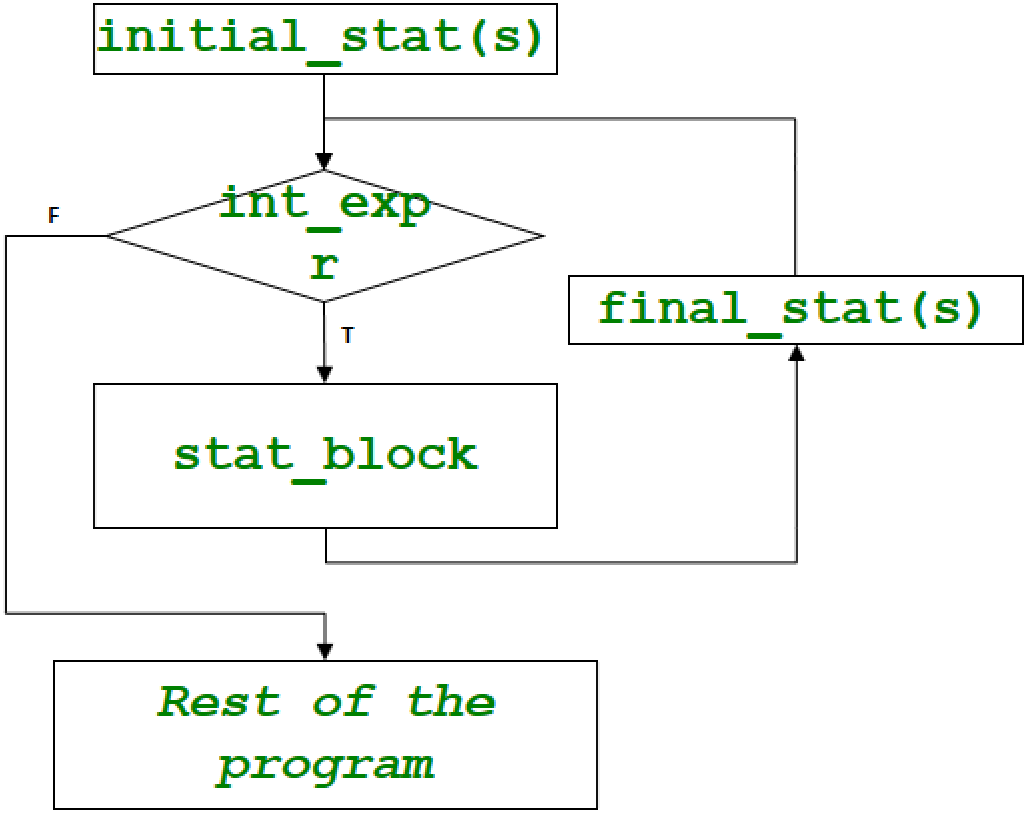
- There are three types of loops:
  - "for" loop
  - "while" loop
  - "do-while" loop

- You can implement anyone of these loops using the others (and possibly an additional if statement and duplication of some statements).
  - The idea is to use the type of loop that is best suited for your problem.

For loop syntax

- Syntax:

```
for (initial_stat(s);int_expr; final_stat(s))
    stat_block
```



For loop

- Note that initial and final statements as well as the integer expression are optional according to the syntax.

  - If `initial_stat(s)` is missing, start directly with the comparison.

  - If `final_stat(s)` is missing, go directly to the comparison after the execution of the statement block.

  - If `int_expr` is missing, the comparison is always true.
    - Make use of the `break` statement.

For loop – Example I

- Print all numbers between 1-10

```
int main(void)
{
    unsigned int counter;

   // initialization, iteration condition, and increment
   //  are all included in the for statement header.
   for ( counter = 1; counter <= 10; ++counter) {
      printf("%u\n", counter);
   }
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

For loop – Example II

- Find the sum of all even numbers between 2-100

```
int main(void)
{
   unsigned int sum = 0, number; // initialize

   for (number = 2; number <= 100; number += 2) {
      sum += number; // add number to sum
   }

   printf("Sum is %u\n", sum);
}
```

Output:

```
Sum is 2550
```

For loop – Example III

- Find a^b. (a and b integers)

```
int a, b, result=1, i;

scanf("%d %d", &a, &b);

for (i=0; i<b; i++)
        result *= a;

printf("Result:%d",result);
```

Output

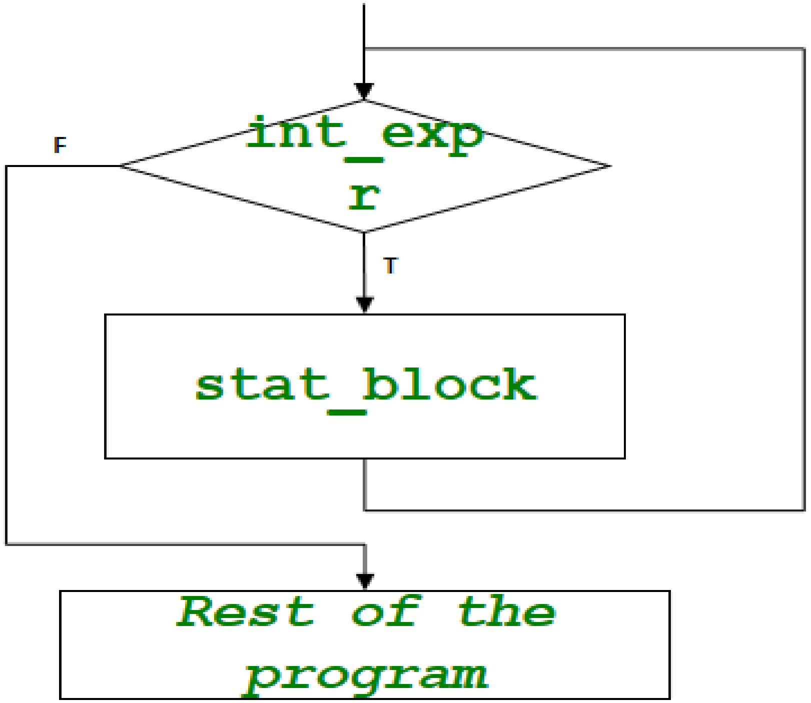
```
3
5
Result:243
```

While loop syntax

- Use while loop if the statement block should be executed as long as a condition holds.

- Syntax:

```
while (int_expr)
    stat_block
```



While loop – Example I

- Print all numbers between 1-10

```
int main(void)
{
   unsigned int counter = 1; // initialization

   while (counter <= 10) { // iteration condition
      printf ("%u\n", counter);
      ++counter; // increment
   }
}
```

While loop – Example II

- Find the average of a sequence of integers terminated with a negative value.

```
int sum=0, n, count=0;
float avg;

scanf("%d", &n);
while (n>=0)
{
        sum += n;
        count++;
        scanf("%d", &n);
}
avg = (count) ? (float)sum/count : 0;
printf("Average:%f",avg);
```

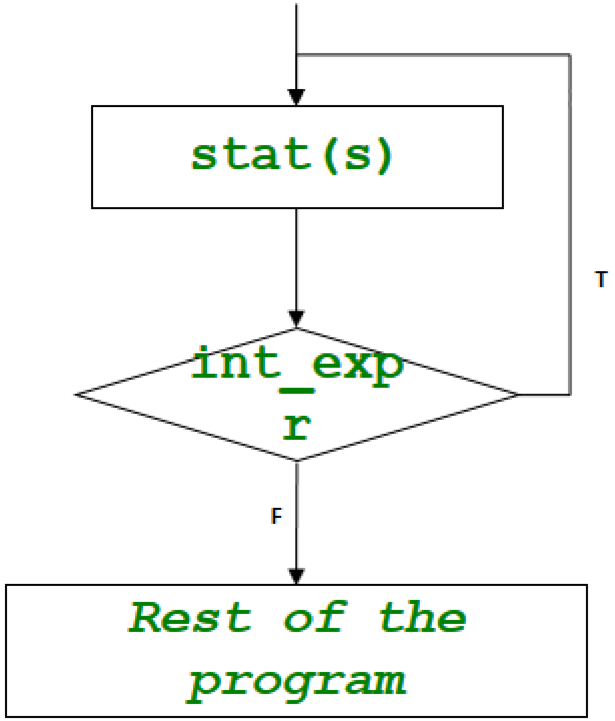
Output

```
1
2
3
4
5
Average:3.000000
```

Do-while loop syntax

- Similar to while loop.
  - Only difference: Condition is checked after the execution of every iteration.
- Syntax:

```
do
{
    stat(s)
}
while (int_expr);
```



Do-while loop – Example I

- Print all numbers between 1-10

```
int main(void)
{
   unsigned int counter = 1;

   do {
      printf("%u\n", counter);
   } while (++counter <= 10);
}
```

Break statement

- It is possible to terminate a loop prematurely.

- Remember the **`break`** statement we discussed before.
  - **`break`** breaks the <u>innermost</u> loop or switch statement.

Break - Example

- Break when number is equal to 5

```
int main(void){
   unsigned int x; // declared here so it can be used after loop
   // loop 10 times
   for (x = 1; x <= 10; ++x) {
      // if x is 5, terminate loop
      if (x == 5) {
         break; // break loop only if x is 5
      }
      printf("%u\n", x);
   }
   printf("\nBroke out of loop at x == %u\n", x);
}
```

Output:

```
1
2
3
4

Broke out of loop at x == 5
```

Continue statement

- It is possible to skip the rest of an iteration and continue with the next iteration (if any).
  - In the for loop, `continue` jumps to the final statement.
  - In the while and do-while loops, `continue` jumps to the condition expression.

Continue - Example

- Continue when number is equal to 5

```
int main(void){
     unsigned int x;
   // loop 10 times
   for (x = 1; x <= 10; ++x) {
      // if x is 5, continue with next iteration of loop
      if (x == 5) {
         continue; // skip remaining code in loop body
      }
      printf("%u\n", x); // display value of x
   }
   printf("\nUsed continue to skip printing the value 5");
}
```

Output:

```
1
2
3
4
6
7
8
9
10

Used continue to skip printing the value 5
```

Example - 1

- Read an integer and print its digits in reverse order.

```
#include <stdio.h>

int main()
{
   int num, digit;

   scanf("%d", &num);

   while (num)
   {
      digit = num % 10;
      num /= 10;
      printf("%d", digit);
   }
   return 0;
}
```

Output:

```
123456
654321
```

Example - 2

- Draw a square or rectangle. Take lenght and width as input

```
int main(void){
   unsigned int x, y, i, j;

   // prompt user for input
   printf("Enter two unsigned integers in the range 1-20: ");
   scanf("%u%u", &x, &y); // read values for x and y

   for (i = 1; i <= y; ++i) { // count from 1 to y
         for (j = 1; j <= x; ++j) { // count from 1 to x
                  printf("*");
         }
         printf("\n");
   }
}
```

Output:

```
Enter two unsigned integers in the
range 1-20: 5
10
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

Example - 3

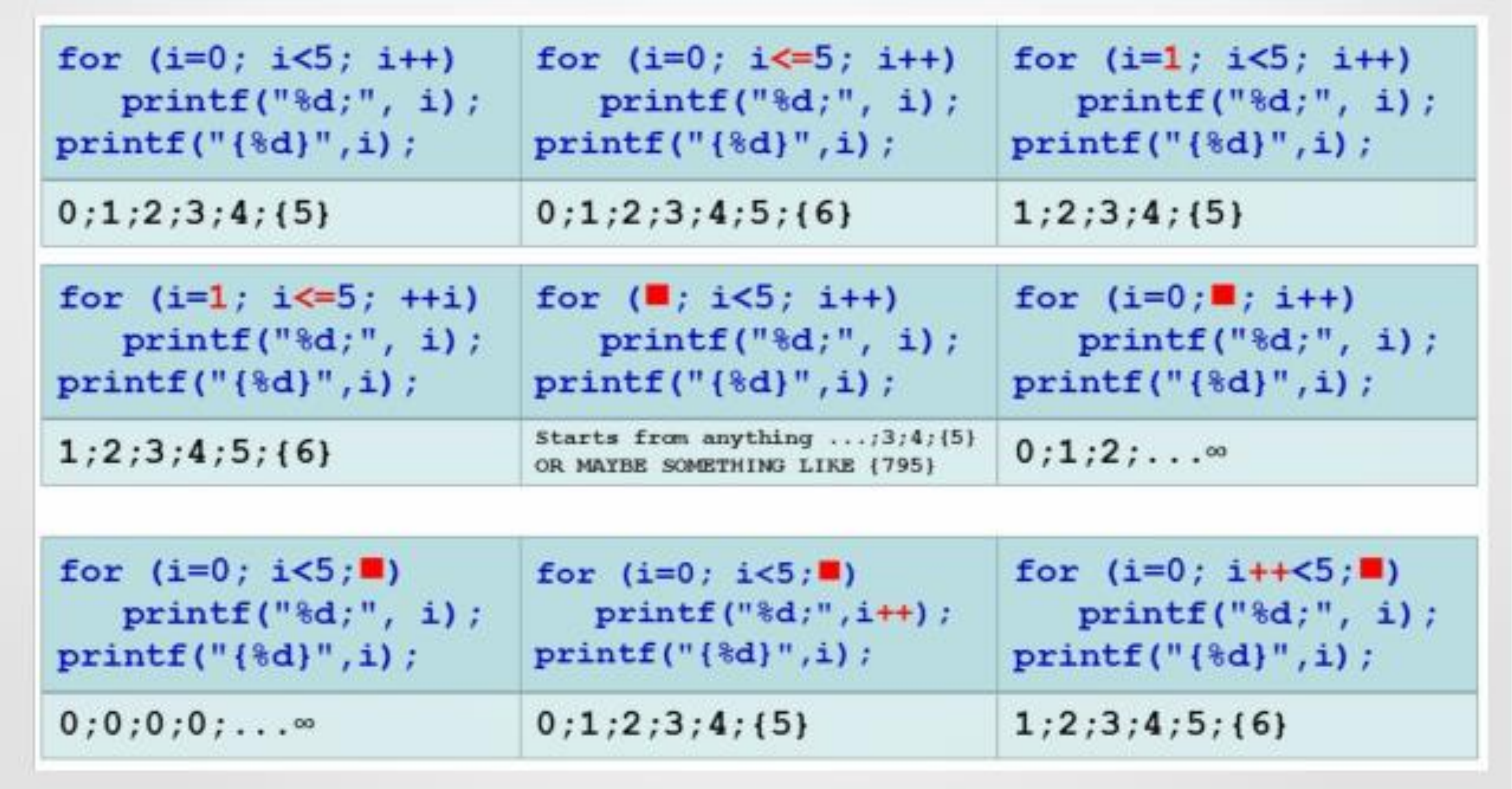
- Draw a triangle. Take lenght as input

```
int main(void){
  unsigned int x, i, j;

  // prompt user for input
  printf("Enter length: ");
  scanf("%u", &x); // read value for x

  for (i = 1; i <= x; i++) { // count from 1 to x
        for (j = i; j <= x; j++) { // count from i to x
                printf("*");
        }
        printf("\n");
  }
}
```

Output:

```
Enter length: 5
*****
****
***
**
*
```

Consider the differences between the following for loops



`for (i=0; i<5; i++)`
`printf("%d;", i);`
`printf("{%d}",i);`
	`for (i=0; i<=5; i++)`
	`printf("%d;", i);`
	`printf("{%d}",i);`
		`for (i=1; i<5; i++)`
		`printf("%d;", i);`
		`printf("{%d}",i);`

0;1;2;3;4;{5}	0;1;2;3;4;5;{6}	1;2;3;4;{5}

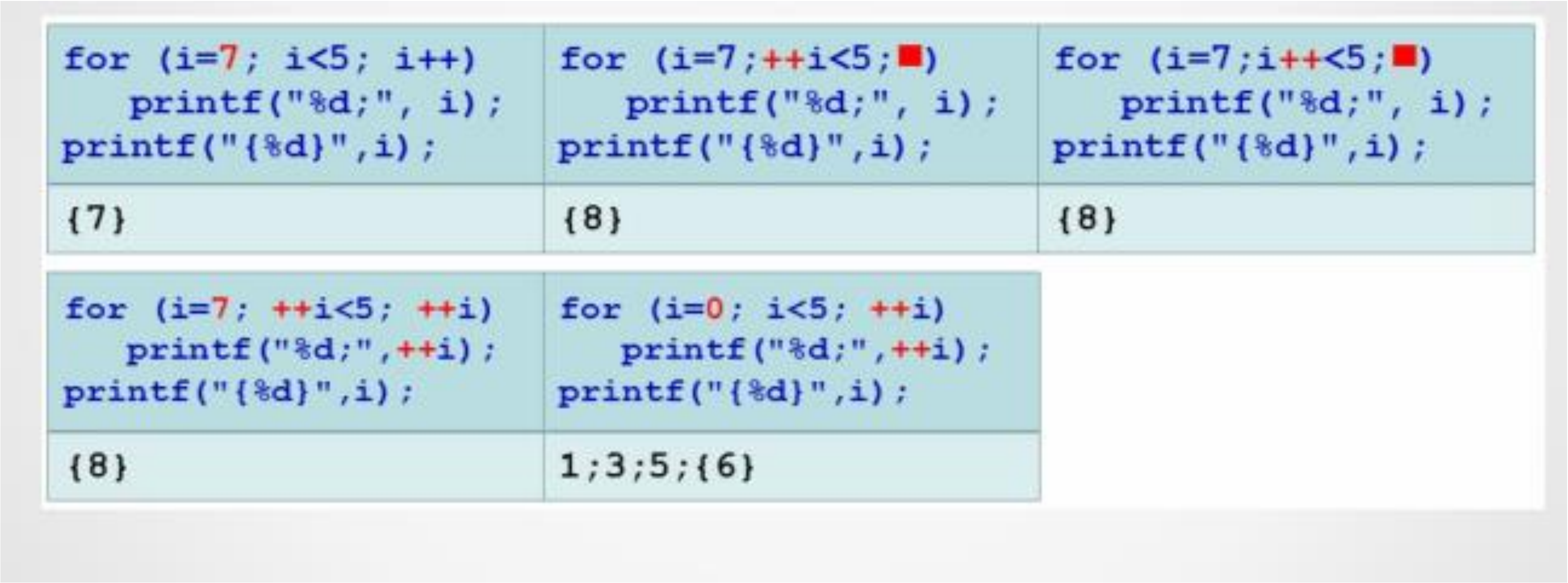
`for (i=1; i<=5; ++i)`
`printf("%d;", i);`
`printf("{%d}",i);`
	`for (■; i<5; i++)`
	`printf("%d;", i);`
	`printf("{%d}",i);`
		`for (i=0;■; i++)`
		`printf("%d;", i);`
		`printf("{%d}",i);`

1;2;3;4;5;{6}	Starts from anything ...;3;4;{5}
	OR MAYBE SOMETHING LIKE {795}
		0;1;2;...∞

`for (i=0; i<5;■)`
`printf("%d;", i);`
`printf("{%d}",i);`
	`for (i=0; i<5;■)`
	`printf("%d;",i++);`
	`printf("{%d}",i);`
		`for (i=0; i++<5;■)`
		`printf("%d;", i);`
		`printf("{%d}",i);`

0;0;0;0;...∞	0;1;2;3;4;{5}	1;2;3;4;5;{6}

Examples



```
for (i=7; i<5; i++)
   printf("%d;", i);
printf("{%d}",i);
```

```
{7}
```

```
for (i=7;++i<5;■)
   printf("%d;", i);
printf("{%d}",i);
```

```
{8}
```

```
for (i=7;i++<5;■)
   printf("%d;", i);
printf("{%d}",i);
```

```
{8}
```

```
for (i=7; ++i<5; ++i)
   printf("%d;",++i);
printf("{%d}",i);
```

```
{8}
```

```
for (i=0; i<5; ++i)
   printf("%d;",++i);
printf("{%d}",i);
```

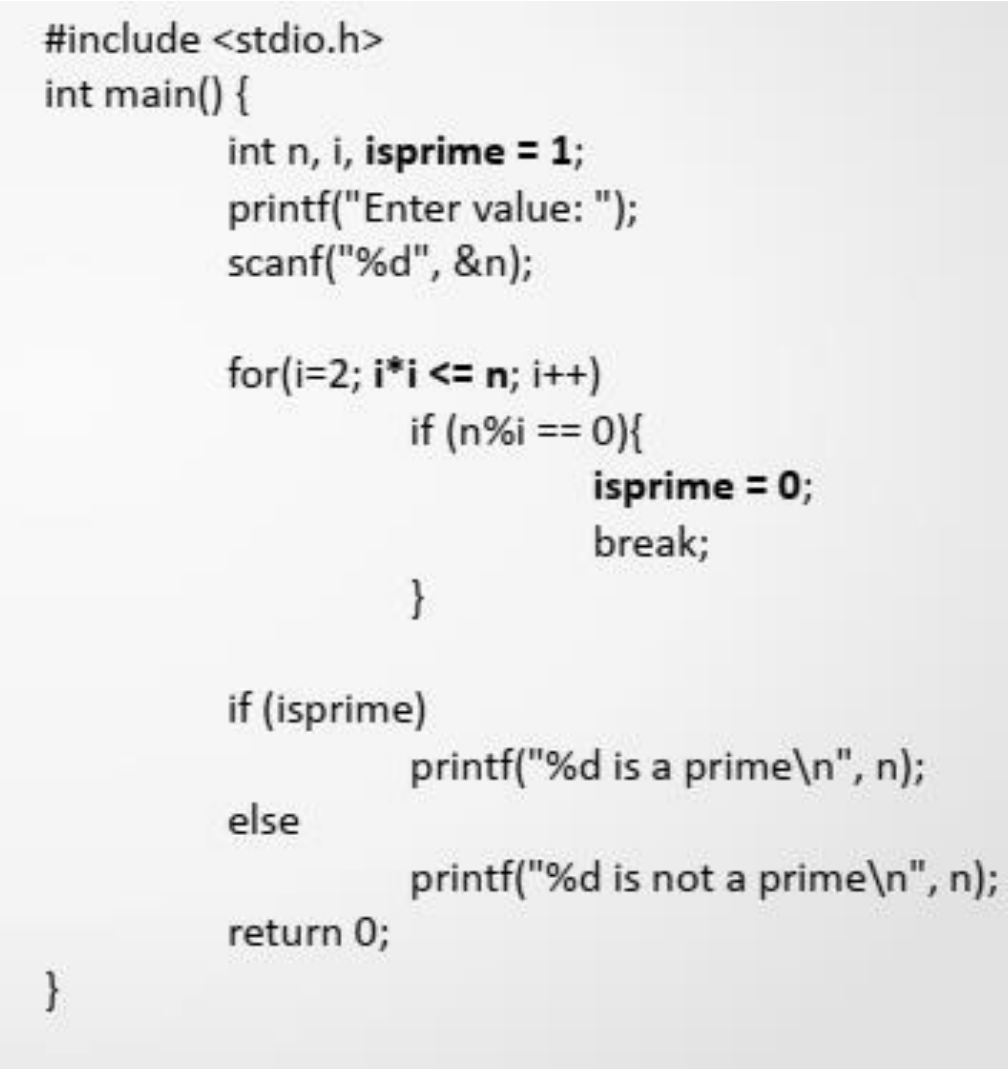
```
1;3;5;{6}
```

Example: Is it a prime?

- We start by assuming the number is prime.

- We scan every integer smaller than the square root of **n**.

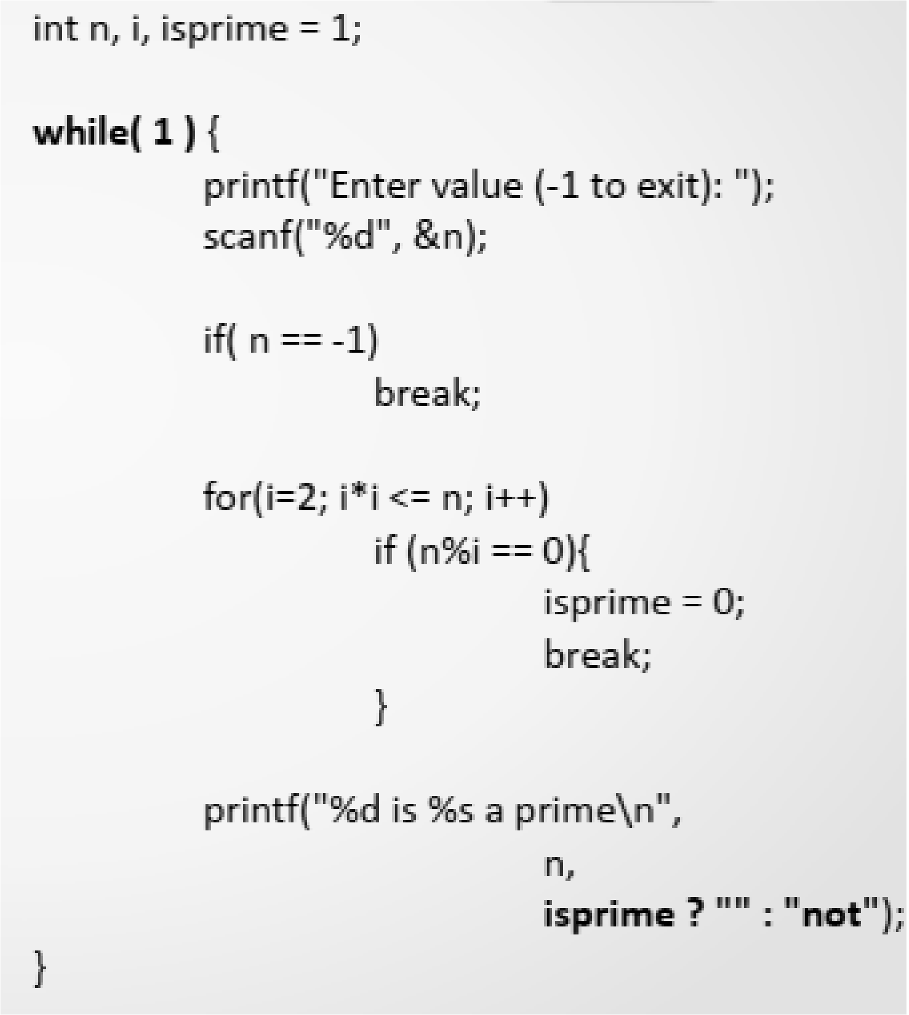
- If any **i** divides **n**, we decide that **n** is not a prime.



```
#include <stdio.h>
int main() {
        int n, i, isprime = 1;
        printf("Enter value: ");
        scanf("%d", &n);

        for(i=2; i*i <= n; i++)
                if (n%i == 0){
                        isprime = 0;
                        break;
                }

        if (isprime)
                printf("%d is a prime\n", n);
        else
                printf("%d is not a prime\n", n);
        return 0;
}
```

Example: Is it a prime?

- Slight modifications:
  - Use an upper loop to read input many times.
  - Use ternary if-else in printf

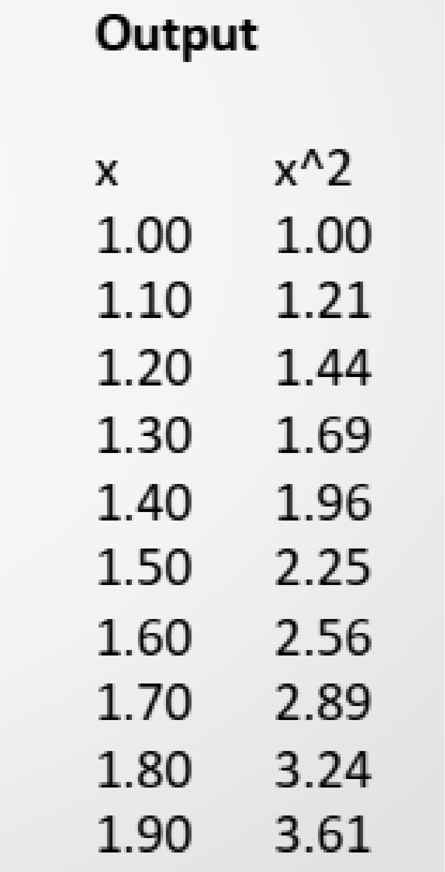


```
int n, i, isprime = 1;

while( 1 ) {
        printf("Enter value (-1 to exit): ");
        scanf("%d", &n);

        if( n == -1)
                break;

        for(i=2; i*i <= n; i++)
                if (n%i == 0){
                        isprime = 0;
                        break;
                }

        printf("%d is %s a prime\n",
                        n,
                        isprime ? "" : "not");
}
```

Floats and round-off errors

- Suppose we want to print out a table of squares of floatingpoint numbers 1.0, 1.1, … , 2.0. Display two digits after the decimal point.

```
float x, dx = 0.1;
printf("x\tx^2\n");
for(x = 1.0; x<= 2.0; x += dx)
        printf("%.2f\t%.2f\n", x, x*x);
```



Output

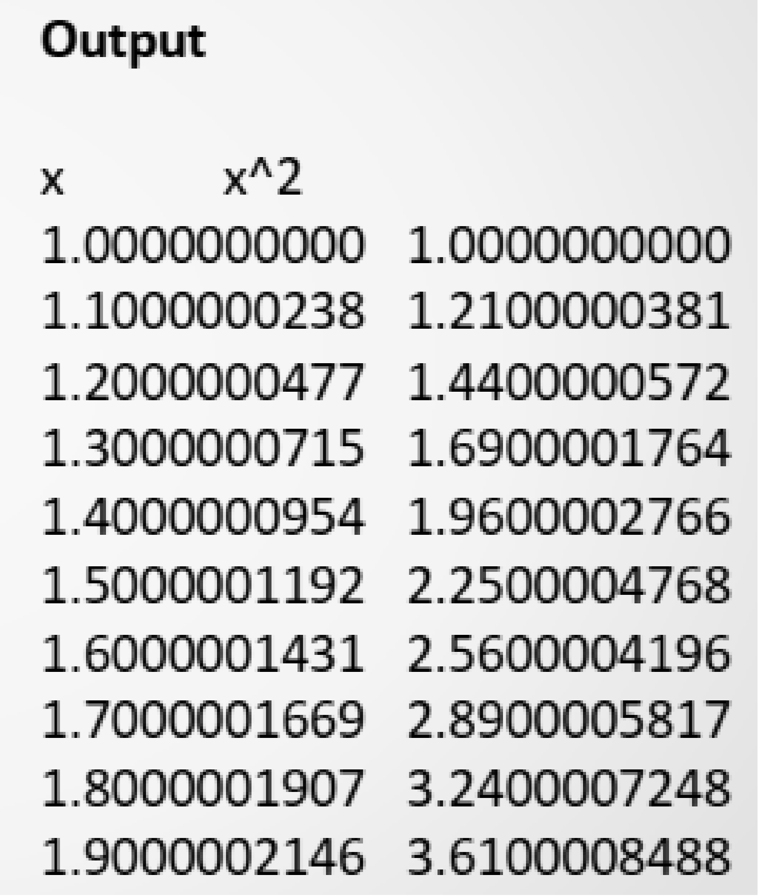
```
x       x^2
1.00    1.00
1.10    1.21
1.20    1.44
1.30    1.69
1.40    1.96
1.50    2.25
1.60    2.56
1.70    2.89
1.80    3.24
1.90    3.61
```

- The output stops at 1.90 even though we included 2.0 in the range.

Floats and round-off errors

- Let's display 10 digits after the decimal point to see why.

```
float x, dx = 0.1;
printf("x\tx^2\n");
for(x = 1.0; x<= 2.0; x += dx)
   printf("%.10f\t%.10f\n", x, x*x);
```



```
Output

x           x^2
1.0000000000  1.0000000000
1.1000000238  1.2100000381
1.2000000477  1.4400000572
1.3000000715  1.6900001764
1.4000000954  1.9600002766
1.5000001192  2.2500004768
1.6000001431  2.5600004196
1.7000001669  2.8900005817
1.8000001907  3.2400007248
1.9000002146  3.6100008488
```

- The results are off because the floating-point representation is inexact.

- Repeated additions increase the error.

Floats and round-off errors

- To be able to display the final value, increase the range by dx

```
float x, dx = 0.1;
printf("x\tx^2\n");
for(x = 1.0; x<= 2.0 + dx; x += dx)
        printf("%f\t%f\n", x, x*x);
```

Output

```
x         x^2
1.000000     1.000000
1.100000     1.210000
1.200000     1.440000
1.300000     1.690000
1.400000     1.960000
1.500000     2.250000
1.600000     2.560000
1.700000     2.890001
1.800000     3.240001
1.900000     3.610001
2.000000     4.000001
```