CS1010E TOPIC 4: REPETITION

Siau-Cheng KHOO

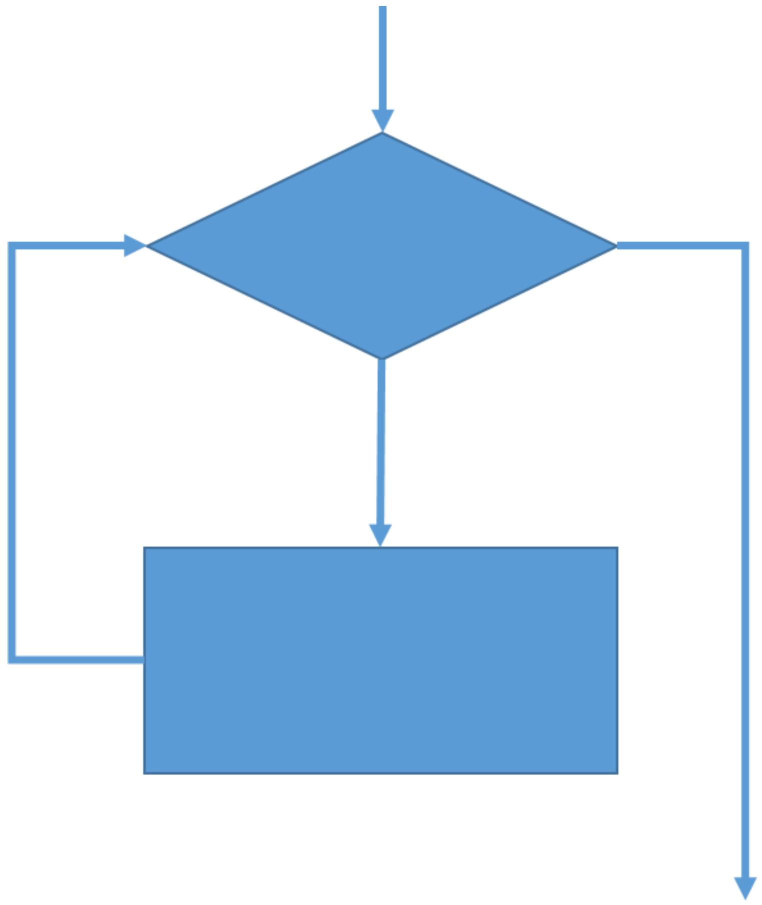
Block COM2, Room 04-11, +65 6516 6730

www.comp.nus.edu.sg/~khoosc

khoosc@nus.edu.sg

Semester II, 2017/2018

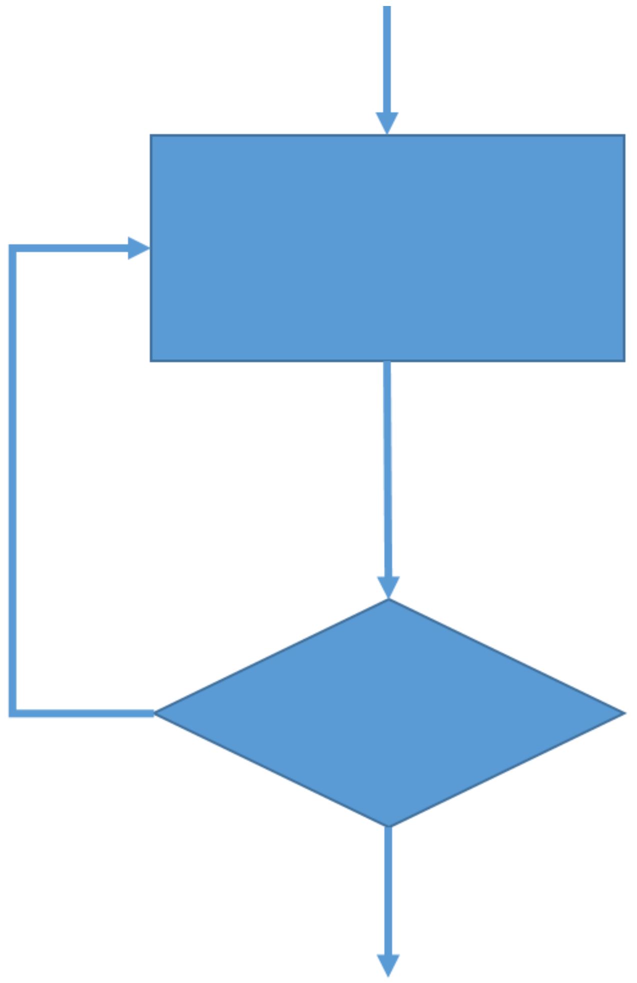
Lecture Outline



while

do-while

for

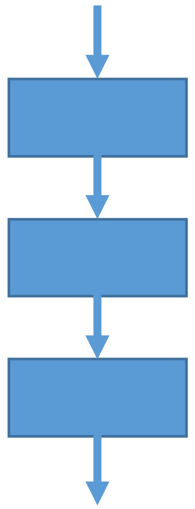


Various Control Structures in a program

Sequence

assignments

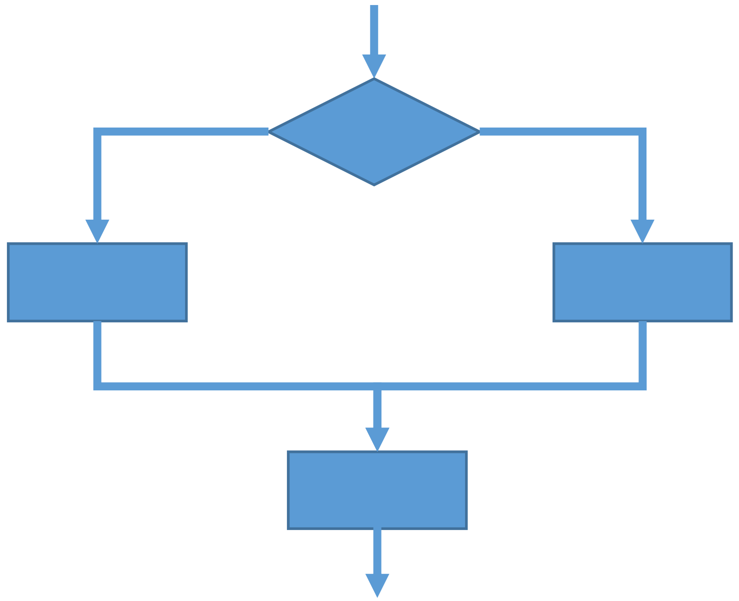
✓



Selection

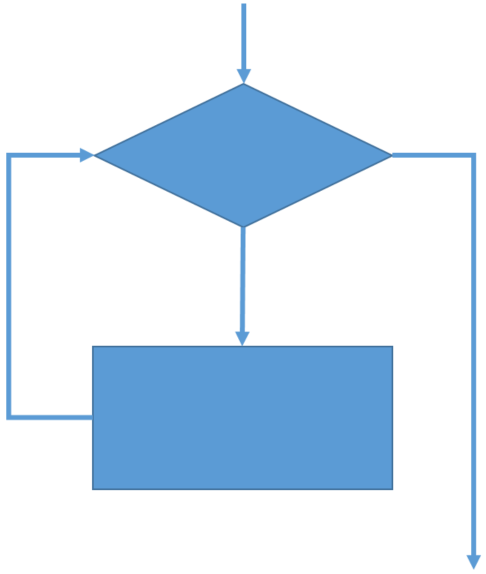
if-else, switch

✓



Repetition

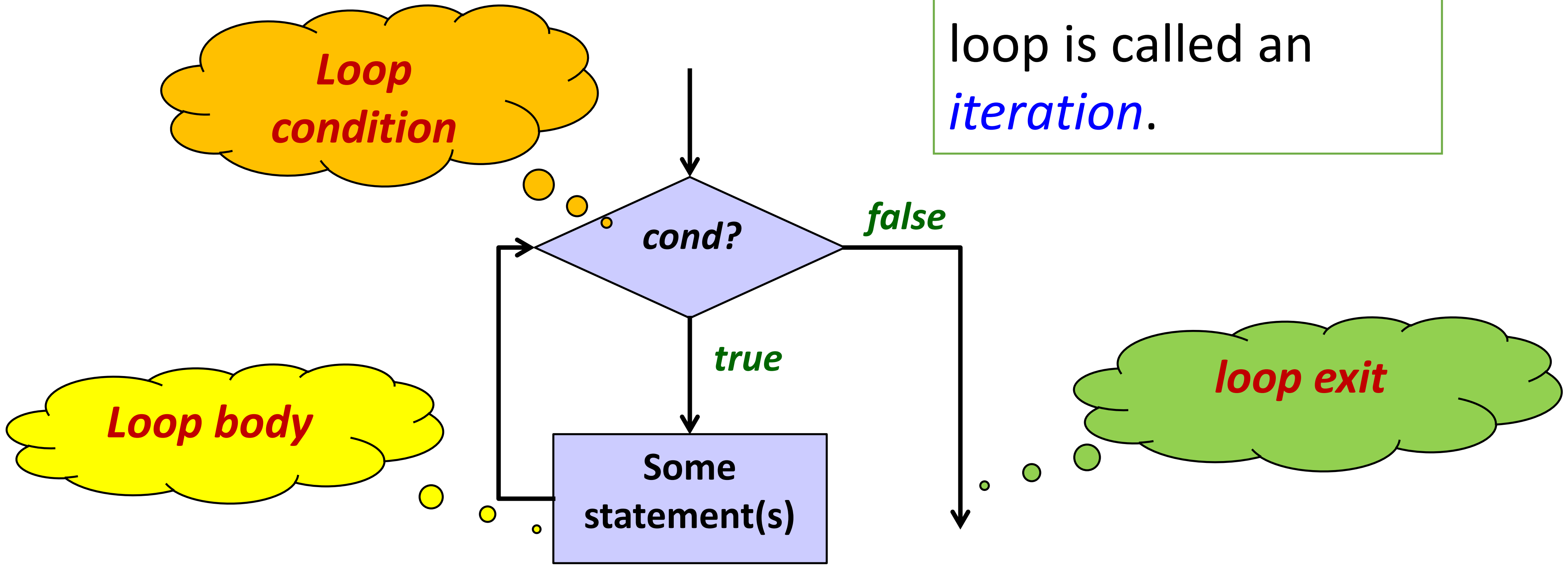
while, do-while, for



Structured programs

Single entry, single exit.

Loop Structures



Each round of the loop is called an *iteration*.

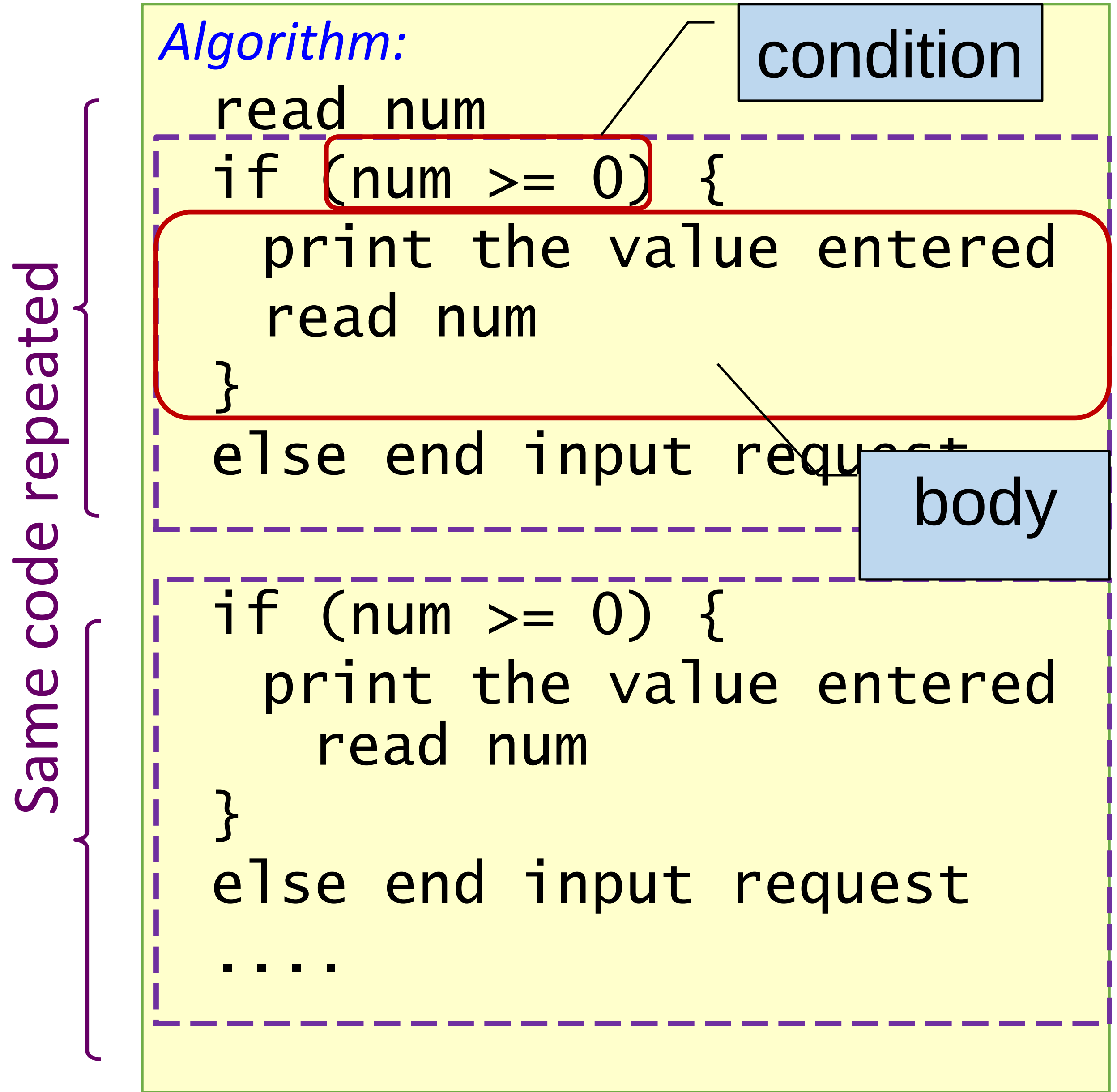
Loop demo

- Keep prompting the user to input a non-negative integer, and output that integer.
- Halt the loop when the input is negative.

```
Enter a number: 12
You entered: 12
Enter a number: 0
You entered: 0
Enter a number: 26
You entered: 26
Enter a number: 5
You entered: 5
Enter a number: -1
```

- Key observations:
  - You keep repeating a task while certain condition is met; in other word, you repeat a tast until the condition **is not met**.
  - You **do not know** beforehand how many iterations there will be.

Loop Demo

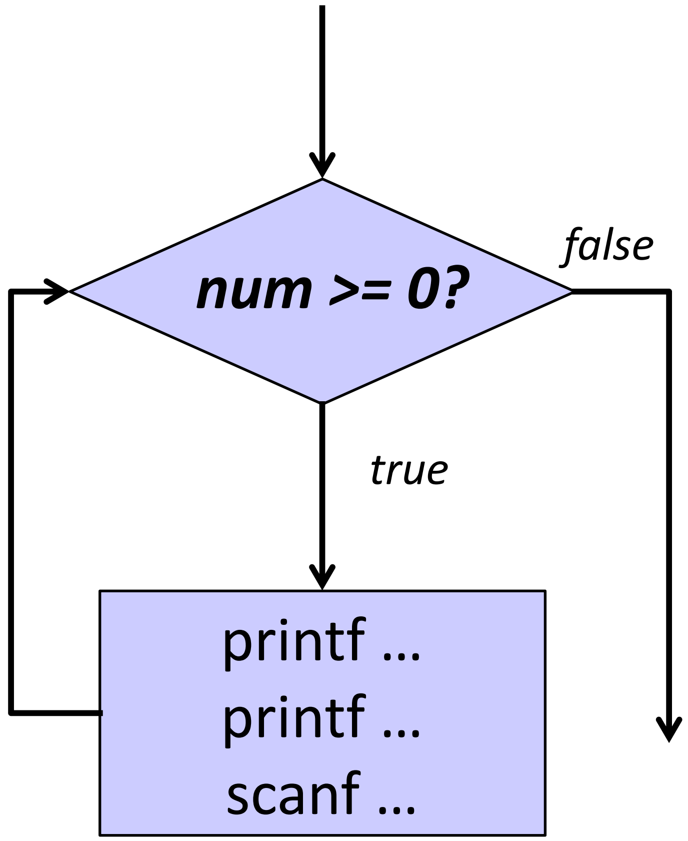


Algorithm:

```
  read num
  if (num >= 0) {
    print the value entered
    read num
  }
  else end input requ

  if (num >= 0) {
    print the value entered
      read num
  }
  else end input request
  ....
```

```
Enter a number: 12
You entered: 12
Enter a number: 0
You entered: 0
Enter a number: 26
You entered: 26
Enter a number: 5
You entered: 5
Enter a number: -1
```

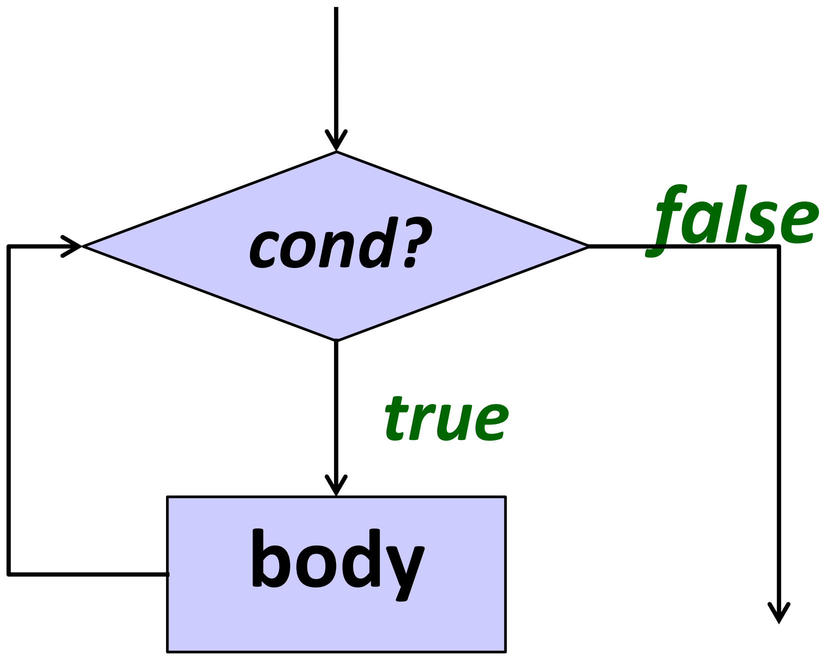
Loop Demo



```
#include <stdio.h>

int main(void)  {
  int num;

  printf("Enter a number: ");
  scanf("%d", &num);
  while (num >= 0) {
    printf("You entered: %d\n", num);
    printf("Enter a number: ");
    scanf("%d", &num);
  }

  return 0;
}
```

The `while` Loop

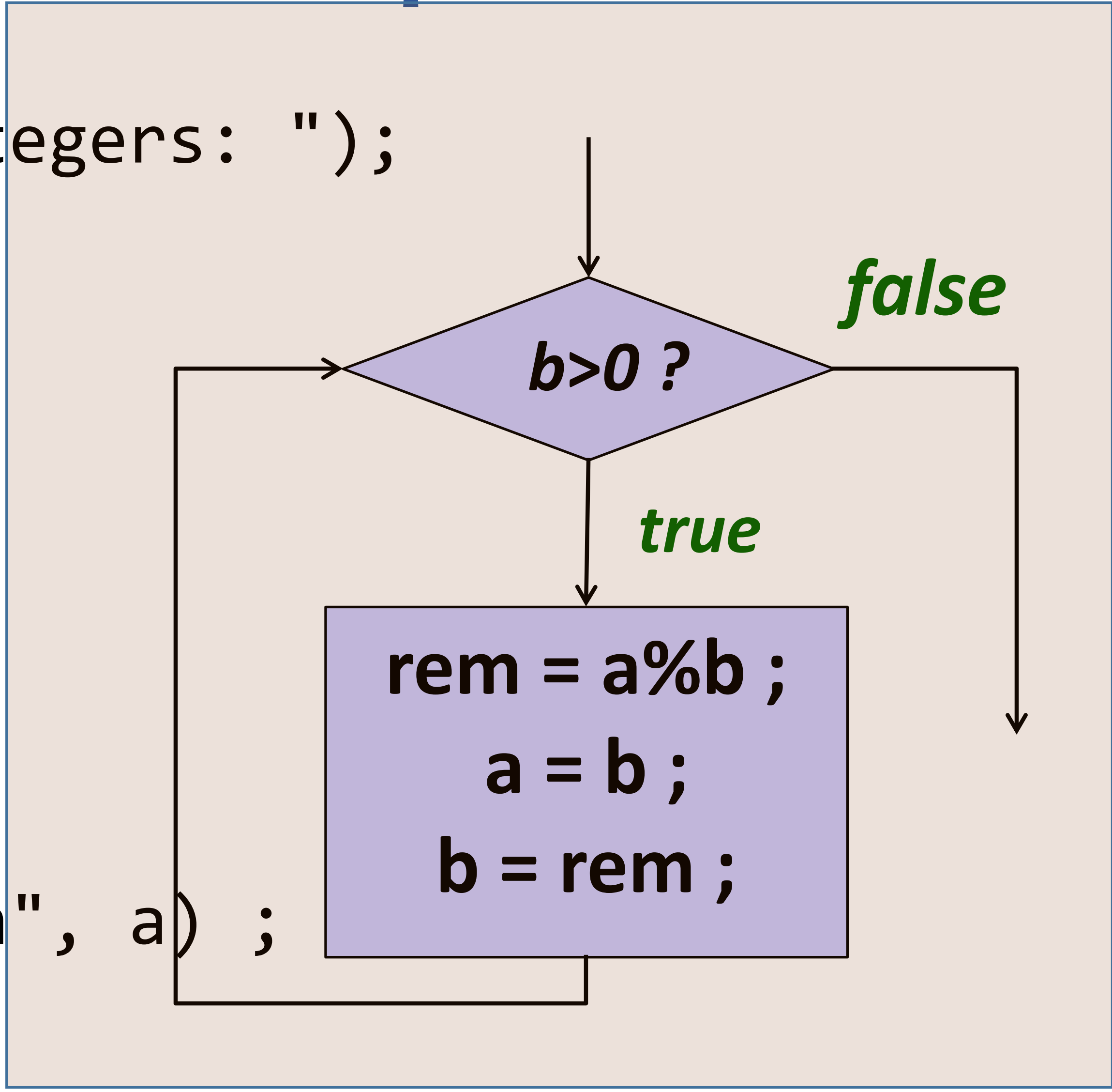
```
while ( condition )
{
    // loop body
}
```



If condition is true, execute loop body; otherwise, terminate loop.

Recall: While statement in GCD Computation

```
printf("Enter two non-negative integers: ");
scanf("%d %d", &a, &b) ;

while (b>0) {
     rem = a % b;
     a = b;
     b = rem;
}

printf(“The result of gcd is %d.\n", a) ;
```



Find Maximum input value using while-statements

PROBLEM

- Keep prompting the user to input a non-negative integer, and print that integer.
- Halt the loop when the input is negative.
- Print the maximum integer input.

```
Enter a number: 12
Enter a number: 0
Enter a number: 26
Enter a number: 5
Enter a number: -1
The maximum number is 26
```

Find Maximum input value using while-statements

```
maxi = 0;
read num;
if (num >= 0) {
  if (maxi < num)
    maxi = num;
  read num;
}
else . . .
if (num >= 0) {
  if (maxi < num)
    maxi = num;
  read num;
}
else . . .
...
print maxi;
```

Algorithm:

```
maxi = 0;
read num;
while (num >= 0) {
  if (maxi < num)
    maxi = num;
  read num;
}

print maxi;
```

Tracing while-loop

```
int a = 1;
while (a*a < 100) {
   printf("%d ", a);
   a *= 2;
}
printf("\n");
```

Iteration	a	(a*a < 100)	printf output	a'
0	1			
1	1	True	1	2
2	2	True	2	4
3	4	True	4	8
4	8	True	8	16
5	16	False		

While statements – your turn now!

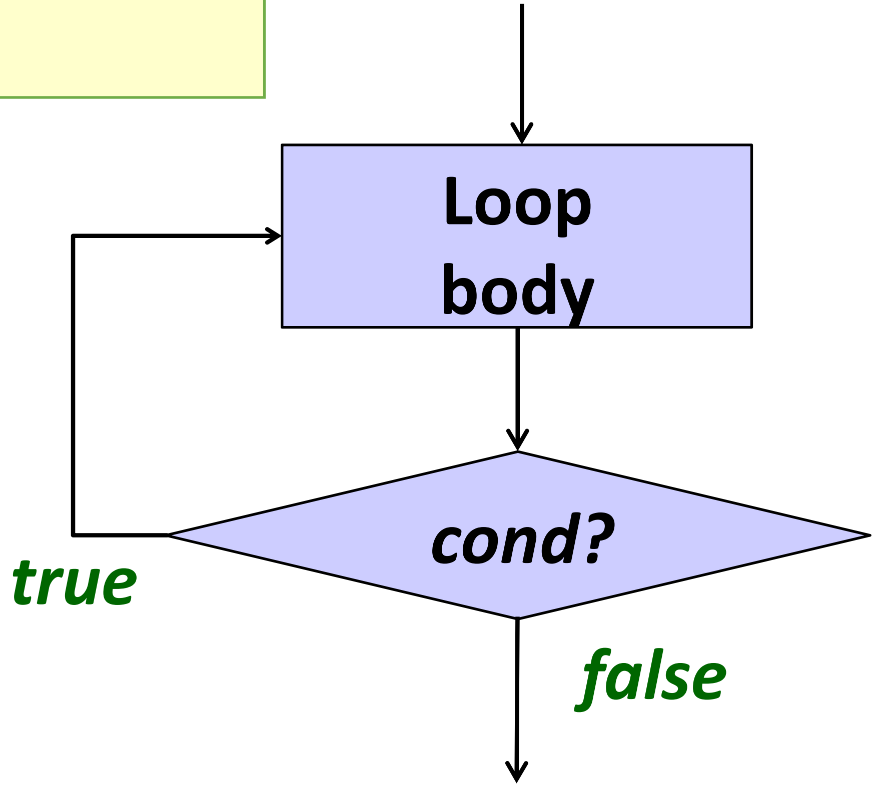
```
// pseudo-code
a = 2;
b = 7;
while (a == b) {
  print a;
  a = a + 2;
}
```

```
// pseudo-code
a = 2;
b = 7;
while (a != b) {
  print a;
  a = a + 2;
}
```

Do-While statements

```
do
{
   // loop body
}  while ( condition );
```

Execute loop body at least once.



An Example for Using do-while statements

Example: Count the number of digits in an integer.

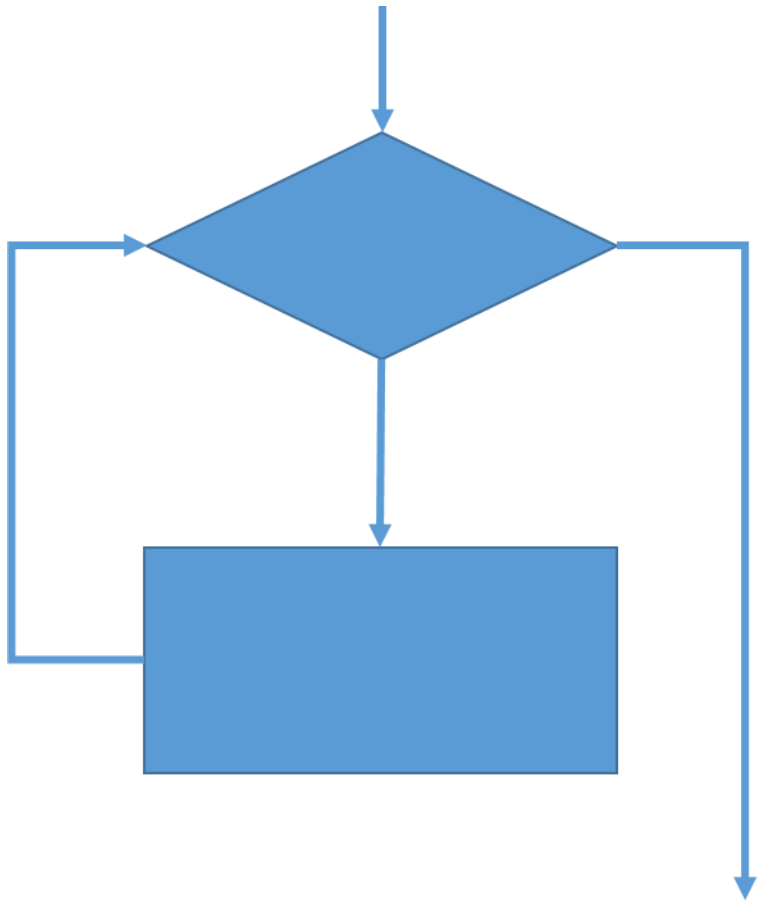
```
// Precond: n > 0
int count_digits(int n)  {
  int counter = 0;

  do {
    counter++;
    n /= 10;
  } while (n > 0);

  return counter;
}
```

```
do
{
   // loop body
}  while ( condition );
```

The Halting problem

```
//
int does_it_halt(program p, input inp)  {
   …
   …
   …
   if (willHalt) {
      print “program p halts on input inp” ;
   }
  else {
      print “program p does NOT halt on input inp” ;
   }

   return 0;
}
```

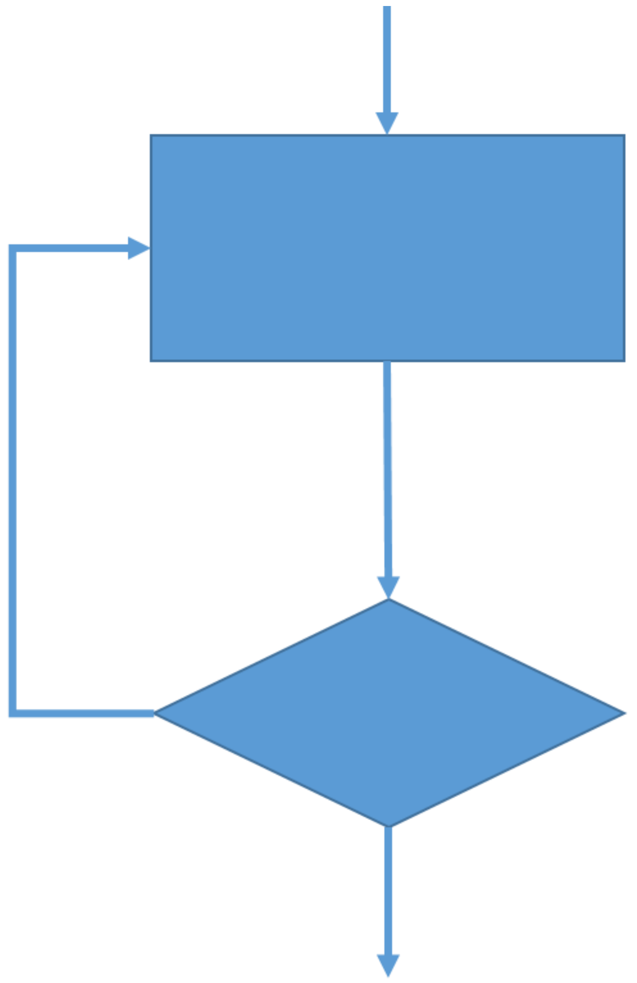
Lecture Outline



while

do-while

for



The *for* loop

```
for ( initialization; condition; update )
{
    // loop body
}
```

Initialization: initialize the **loop variable**

Condition: repeat loop while the condition on **loop variable** is true

Update: change value of **loop variable**

The *for* loop

Example: Print numbers 1 to 10

```
int n;
for (n=1; n<=10; n++) {
  printf("%3d", n);
}
```

Steps:

```
1.n=1;
2.if (n<=10) {
        printf(…);
        n++;
      Go to step 2
    }
```

3.Exit the loop

Tracing *for* loop

```
1.  int i = 1, sum = 0;
2.  for (  ; sum < 20; i*=2) {
3.    sum += i;
4.    printf("i=%d, sum=%d\n",
               i, sum);
   }
5. printf("Final i=%d\n", i);
6. printf("Final sum=%d\n",
              sum);
```

Iter	i@line4	sum@line4
1	1	1
2	2	3
3	4	7
4	8	15
5	16	31
6	**32**	

```
Final i=32
Final sum=31
```

Break from the current loop – break

```
sum = 0;
for (k = 1; k <= 5; k++) {
  scanf("%lf", &x) ;
[redacted]
  sum += x ;
}
printf(“Sum =%f\n", sum);
```

Break from the current loop – break

```c
sum = 0;
for (k = 1; k <= 5; k++) {
  scanf("%lf", &x) ;
  if (x > 10.0) {
       break ;
  }
  sum += x ;
}
printf(“Sum =%f\n", sum);
```

k	x	sum
1	0.5	0.5
2	7.5	8.0
3	8.5	16.5
4	10.5	

```
Sum = 16.5
```

It works for all types of loops: `while`, `do-while`, `for`.

Break from the current Iteration – continue

```
sum = 0;
for (k = 1; k <= 5; k++) {
  scanf("%lf", &x) ;
  if (x > 10.0) {
        continue ;
  }
  sum += x ;
}
printf(“Sum =%f\n", sum);
```

k	x	sum
1	0.5	0.5
2	7.5	8.0
3	8.5	16.5
4	10.5	16.5
5	9.5	26.0

```
Sum = 26.0
```

It works for all types of loops: `while`, `do-while`, `for`.

Nested Loops

$$\sum_{i=1}^{10}\sum_{j=i}^{10}(i+j) = ?$$

```
(1+1) + … + (1+10)
+ (2+2) + … + (2+10)
+ …
+ (9+9) + (9+10)
+ (10+10)
```

```
sum = 0;
for (i=1; i<=10; i++) {

  for (j=i; j<=10; j++) {
    sum = sum + (i + j);
  }

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

Tracing Nested Loop – Try it at home

```
for (p=0; p<6; p++) {
  if (p%2 == 0) {
    for (q=4; q>0; q--)
      printf("p = %d, q = %d\n", p, q);
  }
  else {
    for (q=p; q<20; q+=5)
      printf("p = %d, q = %d\n", p, q);
  }
}
```

```
p = 0, q = 4
p = 0, q = 3
p = 0, q = 2
p = 0, q = 1
p = 1, q = 1
p = 1, q = 6
p = 1, q = 11
p = 1, q = 16
p = 2, q = 4
p = 2, q = 3
p = 2, q = 2
p = 2, q = 1
p = 3, q = 3
p = 3, q = 8
p = 3, q = 13
p = 3, q = 18
```

```
p = 4, q = 4
p = 4, q = 3
p = 4, q = 2
p = 4, q = 1
p = 5, q = 5
p = 5, q = 10
p = 5, q = 15
```

Break in Nested Loop

In a nested loop, *break* **only breaks out of the inner-most loop** that contains it.

```
// with 'break' in a nested loop
printf("With 'break' in a nested loop:\n");
for (i=1; i<=3; i++) {
  for (j=1; j<=5; j++) {
    printf("%d, %d\n", i, j);
    if (j==3)
      break;
    printf("Ya\n");
  }
}
```

```
With ...
1, 1
Ya
1, 2
Ya
1, 3
2, 1
Ya
2, 2
Ya
2, 3
3, 1
Ya
3, 2
Ya
3, 3
```

Problem #6

Given a positive integer *n*, output an integer obtained by reversing the positioning of all the digits in *n*.

```
Enter a positive integer: 28943
The reverse integer is: 34982
```

Problem #6 – Design and Analysis

Given a positive integer *n*, output an integer obtained by reversing the positioning of all the digits in *n*.

Given $a_3\, a_2\, a_1\, a_0$, we would like to return $a_0\, a_1\, a_2\, a_3$.
As a number, this is:

$$a_0 * 10^3 + a_1 * 10^2 + a_2 * 10^1 + a_3 * 10^0$$

$$3498 = 3 * 10^3 + 4 * 10^2 + 9 * 10^1 + 8 * 10^0$$

Problem #6 – Design and Analysis

Given a positive integer *n*, output an integer obtained by reversing the positioning of all the digits in *n*.

Given $a_3\, a_2\, a_1\, a_0$, we would like to return $a_0\, a_1\, a_2\, a_3$. As a number, this is:

$$a_0 * 10^3 + a_1 * 10^2 + a_2 * 10^1 + a_3 * 10^0$$

$$\boxed{= \quad a_3 + 10*(a_2 + 10*(a_1 + 10*(a_0)))}$$

$$\underbrace{a_3 + 10*(\underbrace{a_2 + 10*(\underbrace{a_1 + 10*(a_0)))}_{}}_{}}_{}$$

Problem #6 – Refinement

```
Enter a positive integer: 28943
The reverse integer is: 34982
```

Horner's Rule:

$$a_0 * 10^3 + a_1 * 10^2 + a_2 * 10^1 + a_3 * 10^0$$
$$= \mathbf{a_3 + 10*(a_2 + 10*(a_1 + 10*(a_0)))}$$

```
res ← 0 ;
if (n <= 0) return res ;
res ← (n % 10);   a_0
n ← n / 10 ;
if (n <= 0) re a_1 n res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
```

```
if (n <= 0) ret a_2 n res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
if (n <= 0) ret a_3 n res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
if (n <= 0) return res ;
```

Problem #6 – Refinement

```
Enter a positive integer: 28943
The reverse integer is: 34982
```

Horner's Rule:

$$a_0 * 10^3 + a_1 * 10^2 + a_2 * 10^1 + a_3 * 10^0$$
$$= a_3 + 10*(a_2 + 10*(a_1 + 10*(a_0)))$$

```
res ← 0 ;
if (n <= 0) return res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
if (n <= 0) return res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
```

```
if (n <= 0) return res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
if (n <= 0) return res ;
res ← (n % 10) + res * 10 ;
n ← n / 10 ;
if (n <= 0) return res ;
```

Problem #6 – Implementation

Horner's Rule:

$$a_0 * 10^3 + a_1 * 10^2 + a_2 * 10^1 + a_3 * 10^0$$
$$= a_3 + 10*(a_2 + 10*(a_1 + 10*(a_0)))$$

```
// Precond: n > 0
1.int reverse(int n) {
2.   int res = 0 ;
3.   while (n > 0) {
4.      res = (n % 10) +
5.            10 * res ;
6.      n /= 10 ;
   }
7.   return res;
}
```

n = 1234   **res = 0**

iter	n@Line6	res@Line6
1	123	4
2	12	43
3	1	432
4	0	4321
5	-	-

Problem #7 – Hi-Lo Game

- The program holds a secret number between 1 and 100
  - Hopefully, every round of game the secret number changes
- Game begins:
  - User makes a guess about the number
  - The program responses according to whether user's guess is too low, too high from the secret number, or BINGO!
  - User can have up to 10 guesses
- The program asks if the user wishes to play another round.

Problem #7 – Hi-Lo Game

```
int tries = 0 ;
printf("\nGuess a number between 1 and 100 inclusive.\n");
do {
   tries++;
   printf("Enter your guess [%d]: ", tries);
   scanf("%d", &guess);
   if (guess < secret) {
      printf("Your guess is too low!\n");
   }
   else if (guess > secret) {
      printf("Your guess is too high!\n");
   }
} while ( (tries < TRIES) && (guess != secret) );
```

Problem #7 – Hi-Lo Game

```
void play_a_game(int);
```

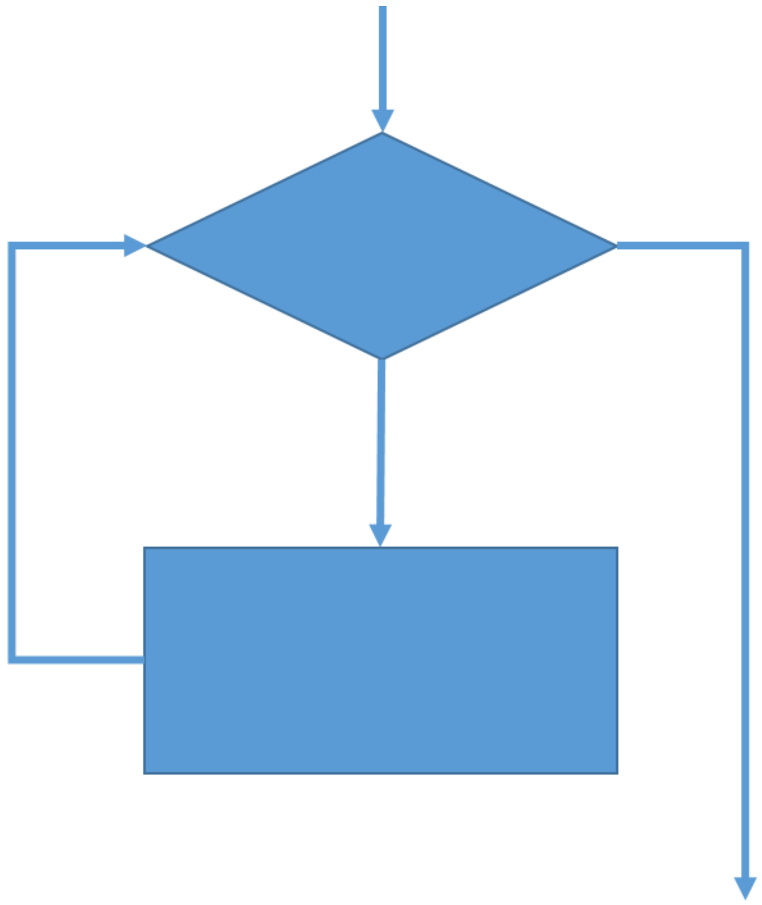
```
    void play_a_game(int secret) {
        int guess;
        int tries = 0 ;
        printf("\nGuess a number between 1 and 100 inclusive.\n");
        do {
           tries++;
           printf("Enter your guess [%d]: ", tries);
           scanf("%d", &guess);
           if (guess < secret) {
              printf("Your guess is too low!\n");
           }
           else if (guess > secret) {
              printf("Your guess is too high!\n");
           }
        } while ( (tries < TRIES) && (guess != secret) );
        if (guess == secret) {
            printf("Congratulations!"
                   "You did it in %d steps", tries);
        } else {
            printf("Too bad. The number is %d. "
                   "Better luck next time!\n", secret);
        }
    }
```

```c
int main(void) {
   int secret;
   int response;
   printf("*** Welcome to the HiLo game! ***\n\n");
   srand(time(NULL)) ;

   do {
     secret = rand()%100 + 1;
     play_a_game(secret);
     printf("Do you want to play again "
            "(0 means 'no'; 1 means 'yes')?\n");
     scanf("%d", &response);
   } while (response == 1);

   printf("\n*** Thanks for playing. Bye! ***\n");
   return 0;
}
```

Random Number Generation

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define TRIES 10

void play_a_game(int);

int main(void) {
  int secret;
  int response;
  printf("*** Welcome to the HiLo game! ***\n\n");
  srand(time(NULL)) ;
  do {
    secret = rand()%100 + 1;
    . . .
    return 0;
}
```

Summary



while

do-while

for