Goal:

To learn how **think** in the *procedural paradigm* and write code to implement a procedure in C.

Consider the following guessing game:

I pick a number from 0 to 500.

You guess a number and I tell you if it is equal to my number, greater than my number or less than my number.

You want to find my number as quickly as you can.

Write a paragraph carefully describing how you would proceed to guess my number.

I observe that in this game I will always know that the number is between some low number and some high number. Initially the low number is 0 and the high number is 500 by the rules of the game. I will choose to guess the mean of the low and high number. If my number is too low I will replace the low number with my guess. If my number is too high I will replace my high number with my guess. I will continue guessing the mean of the low and high numbers until I am correct.

Make an indication in your paragraph for each of the following:

- Each piece of information
- Each place the value of the information changes
- Each time a question is asked or information is needed

I observe that in this game I will always know that the number is between some low number and some high number. *Initially* the low number is 0 and the high number is 500 by the rules of the game. I will choose to guess the mean of the low and high number. *If* my number is too low I will replace the low number with my guess. *If* my number is too high I will replace my high number with my guess. I will *continue* guessing the mean of the low and high numbers *until* I am correct.

- Three integers are needed: guess, high and low
- Initially low = 0; high = 500; guess = (high+low)/2;
- Make the guess
- If the guess is correct then we are done, while it is not correct
- If the guess is high, then high = guess
- Otherwise the guess must be low, so low = guess
- Make guess = (high + low)/2

Try to write a list like this for your paragraph

- Three integers are needed: guess, high and low
- Initially low = 0; high = 500; guess = (high+low)/2;
- Make the guess
- If the guess is correct then we are done, while it is not correct
- If the guess is high, then high = guess
- Otherwise the guess must be low, so low = guess
- Make guess = (high + low)/2

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

The bold line is a C *comment.*

Single line comments begin with //

Multiple line comments start with /* and end with */

```c
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

The bold line is a C *statement*. C statements end in a semicolon.

The statement defines guess, high and low as integer variables.

C has the following primitive types for variables:

int, char, short, long, long long <- these are integer types

float, double <- these are floating point types

Integer types may be signed or unsigned:
short var1
unsigned short var2

Signed means they can take on negative values but the range cut in half

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

The bold lines *assign* values to the variables low, high and guess.

Variables in C cannot be assumed except in specific cases to be initialized to any known value.

You must initialize the value before you use the variable.

[Global variables in C are an exception to this in that they have a 0 initial value, but it is still considered good practice to initialize them explicitly.]

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

While is a *flow control* statement.

It tests the logical value of the statement inside the parentheses and while it is *true*. It allows the steps that are inside the braces {} to execute.

!= means not equal

This while loop calls a *function: testGuess(guess)*

The function takes a single variable (guess) as an *argument*

Not shown here the *implementation* of testGuess *returns* 0 if the guess is correct. It *returns* 1 if the guess is too high, and it *returns* -1 if the guess is too low.

The while loop will keep trying guess until it is correct!

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

if is a flow control statement to conditionally do the steps inside the braces once if the condition inside is true.

== asks if something is equal rather than assigning a value.

Here if the guess is too high (testGuess returns 1), then we change the value of high to the value of guess.

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

else is a flow control statement to conditionally do the steps inside the braces once if the condition of the preceding if statement was false.

If the guess had been correct the while loop would not have executed.

If the guess was too high then high = guess;

If this was *not* true then the statements inside the else are run: low = guess;

This is the only remaining option.

Last we must update the guess to the new guess.

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
      if(testGuess(guess) == 1)
      {
            high = guess;
      }
      else
      {
            low = guess;
      }
      guess = (high+low)/2;
      printf("%d\n", guess);

}
printf("The number was %d\n", guess);
```

We have added two more lines to the program to help us see both the work of the program and the correct number.

These lines call a built in function called printf, which stands for formatted print.

It prints a line out on the screen. %d is a placeholder for interpreting a variable and printing it as a decimal number.

\n is an *escape sequence* to the new line character which is input in a keyboard with the enter key.

The first printf function call will print each guess made.

The second printf function call will print the final correct number out.

- Three integers are needed: guess, high and low
- Initially low = 0; high = 500; guess = (high+low)/2;
- Make the guess
- If the guess is correct then we are done, while it is not correct
- If the guess is high, then high = guess
- Otherwise the guess must be low, so low = guess
- Make guess = (high + low)/2

```
//And now in C

int guess, high, low;
low = 0;
high = 500;
guess = (high+low)/2;

while(testGuess(guess) != 0)
{
        if(testGuess(guess) == 1)
        {
                high = guess;
        }
        else
        {
                low = guess;
        }
        guess = (high+low)/2;

}
```

```c
#include <stdio.h>
#define magicNumber 42

int testGuess(int guess)
{
    if(guess == magicNumber) return 0;
    else if(guess > magicNumber) return 1;
    else return -1;
}
int main(void)
{
    int low, high, guess;
    low = 0;
    high = 500;
    guess = (high + low)/2;
    while(testGuess(guess) != 0)
    {
        if(testGuess(guess) == 1)
        {
            high = guess;
        }
        else
        {
            low = guess;
        }
        guess = (high + low)/2;
        printf("%d\n", guess);
    }
    printf("The number is %d\n", guess);
    return 0;
}
```

Here is a full program!

#include is a way of adding other code to the program.
#include <stdio.h> adds standard I/O's functions which is where printf lives

#define is a way of creating a label for the stuff the right of the label (42 here). Thus everywhere magicNumber appears after this #define statement will be replaced with 42.

#define keeps your code readable and allows you to avoid magic numbers repeating in your code

int main(void) is the first code to execute in a normal C program

return as mentioned is the value the function gives back to the calling function.

```c
#include <stdio.h>
#define magicNumber 42

int testGuess(int guess)
{
    if(guess == magicNumber) return 0;
    else if(guess > magicNumber) return 1;
    else return -1;
}
int main(void)
{
    int low, high, guess;
    low = 0;
    high = 500;
    guess = (high + low)/2;
    while(testGuess(guess) != 0)
    {
        if(testGuess(guess) == 1)
        {
            high = guess;
        }
        else
        {
            low = guess;
        }
        guess = (high + low)/2;
        printf("%d\n", guess);
    }
    printf("The number is %d\n", guess);
    return 0;
}
```

The code is available at:
http://pages.hmc.edu/bbryce/first.c

Change the value of magicNumber and run the code on codepad.org, verify that it works

```c
#include <stdio.h>
#define magicNumber 42

int testGuess(int guess)
{
    if(guess == magicNumber) return 0;
    else if(guess > magicNumber) return 1;
    else return -1;
}
int main(void)
{
    int low, high, guess;
    low = 0;
    high = 500;
    guess = (high + low)/2;
    while(testGuess(guess) != 0)
    {
        if(testGuess(guess) == 1)
        {
            high = guess;
        }
        else
        {
            low = guess;
        }
        guess = (high + low)/2;
        printf("%d\n", guess);
    }
    printf("The number is %d\n", guess);
    return 0;
}
```

The code is available at:
http://pages.hmc.edu/bbryce/first.c

Change the value of magicNumber and run the code on codepad.org, verify that it works

Now change the algorithm. Instead of guessing the mean value, and using low and high, just guess in order from 0 until you reach the correct value. Remove any unused variables.