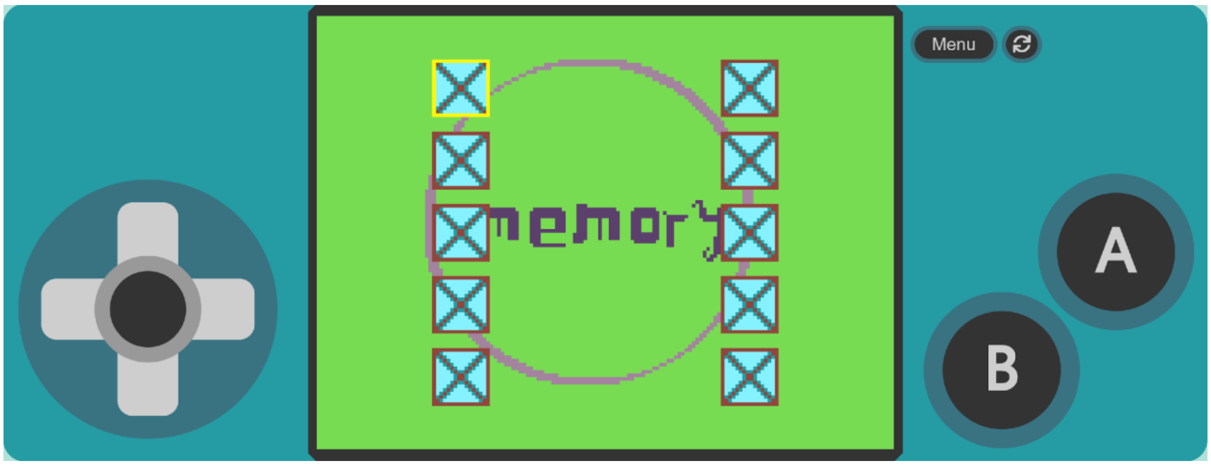
THIRD ARRAYS TASK



Description.

In this project, we will create a variation of the classic game "Memory," where the player has a table with cards and has to select two cards with the same image in consecutive turns.

We will use many of the concepts we have learned so far, with a primary focus on the "Array" section, where we will store the positions of each card, the related images, and more.

In order to do that, we use MakeCode Arcade to create the game.

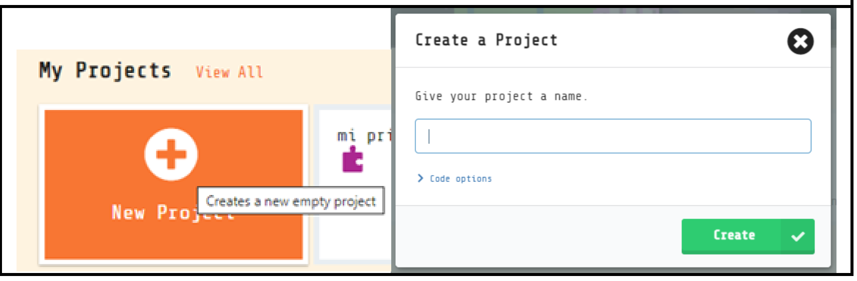
Goals.

- Create a sprite that serves as a cursor for navigating between different options.
- Use arrays to achieve the desired result, both for generating random content and associating images with specific positions.

Programming guide.

NEW PROJECT

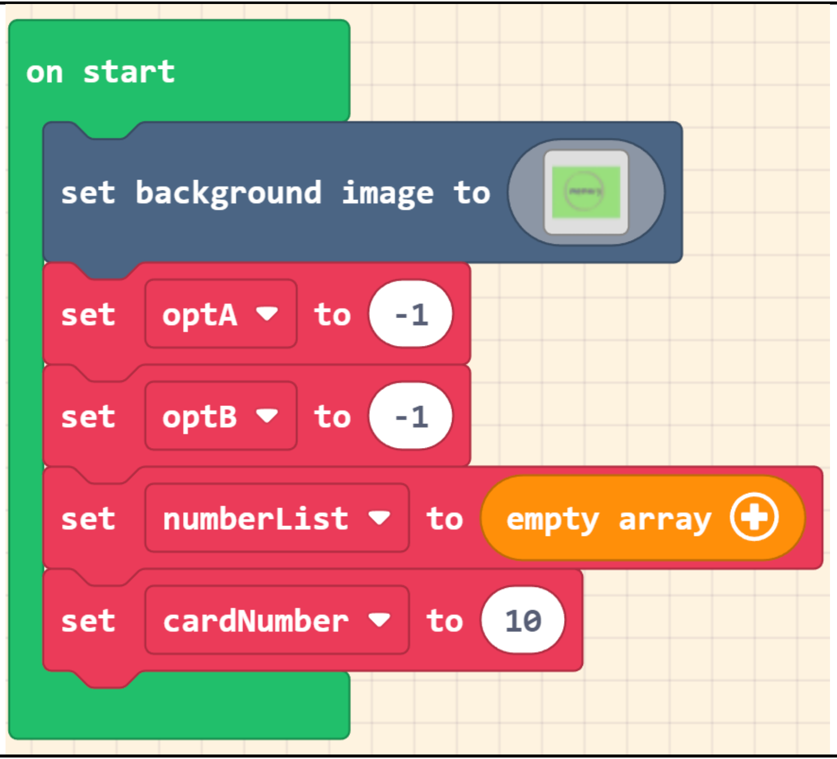
We start creating a project, we should stablish the name, for example ¨Picking up food¨ and then press ¨create¨ button.



MAIN PROGRAMMING

ON START GAME CREATION

Let's start by programming the game background and setting initial values. The variables "optA" and "optB" will be used to compare the choices made by the player and determine if they have made a correct match. They will be initially set to -1, as it represents a value when no option has been chosen. The variable "numberList" will be declared as an array and will determine the position of each card.
The variable "cardNumber" will set the total number of cards in the game.
Here's an example of how you can initialize these variables in MakeCode Arcade:



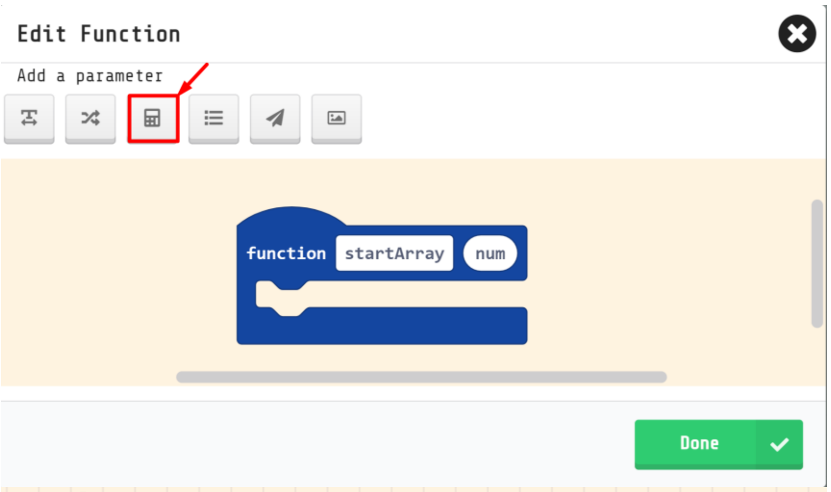
```
on start
    set background image to
    set optA to -1
    set optB to -1
    set numberList to empty array
    set cardNumber to 10
```

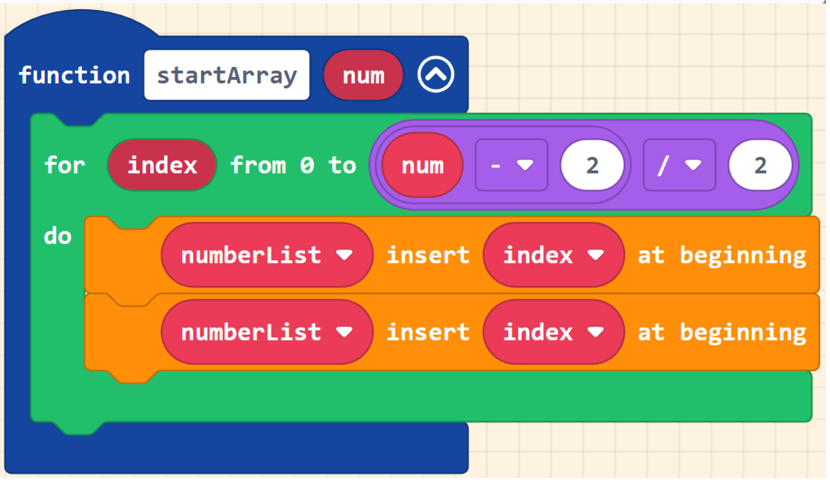
START ARRAY NUMBERLIST

Let's create a function that takes a numeric parameter. The parameter we'll use later is cardNumber.

Inside this function, we'll use a for loop that iterates half the number of cards used.
We'll insert the value of index twice in a row into the numberList array, duplicating the value. In this case, once the loop is finished, the values in numberList will be:

numberList = [0, 0, 1, 1, 2, 2, 3, 3, 4, 4]



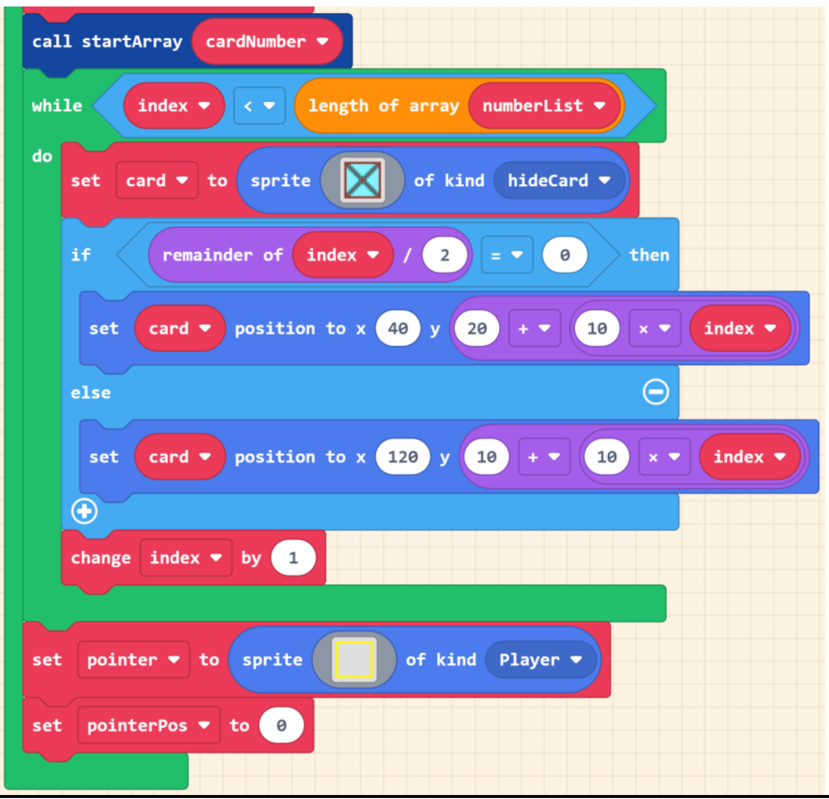


```
function startArray num
    for index from 0 to num - 2 / 2
    do
        numberList insert index at beginning
        numberList insert index at beginning
```

CARD PLACEMENT

Let's continue in the **on start** block and add a call to the **generateNumberList** function at the end, passing **cardNumber** as the argument.
Next, we'll program a **while** loop that will create and position the cards in the game. We'll use a formula to adjust the cards' positions on the y-axis, and on the x-axis, we'll ensure they are evenly spaced based on whether the index is even or odd. We'll increase the value of **index** by 1 in each iteration of the loop to move through the array.

Additionally, we'll create a sprite called "pointer" for the card selection.



```
call startArray cardNumber
while index < length of array numberList
do
    set card to sprite of kind hideCard
    if remainder of index / 2 = 0 then
        set card position to x 40 y 20 + 10 × index
    else
        set card position to x 120 y 10 + 10 × index
    change index by 1
set pointer to sprite of kind Player
set pointerPos to 0
```

RANDOM GENERATION OF CARD POSITIONS

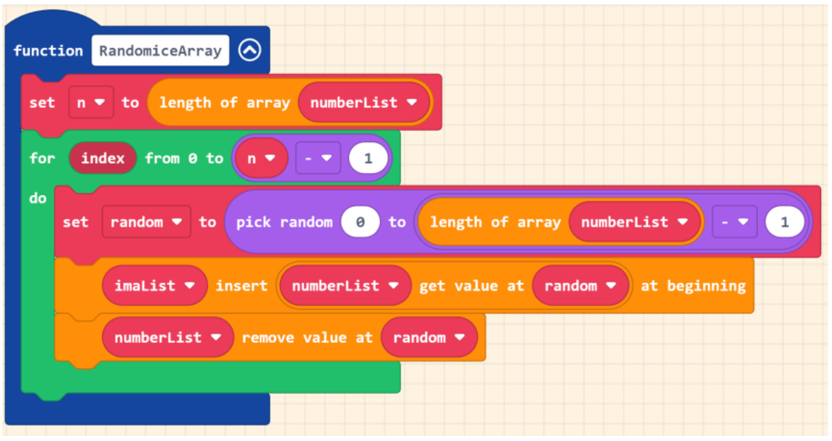
We will create the "randomizeArray" function to randomly arrange each card.

The variable "n" will hold the size of the number of cards we are using.

We will use a "for" loop that will increase the value of "index" starting from 0 up to the number of cards minus 1. Remember that if we have 10 cards, the loop will run 10 times.

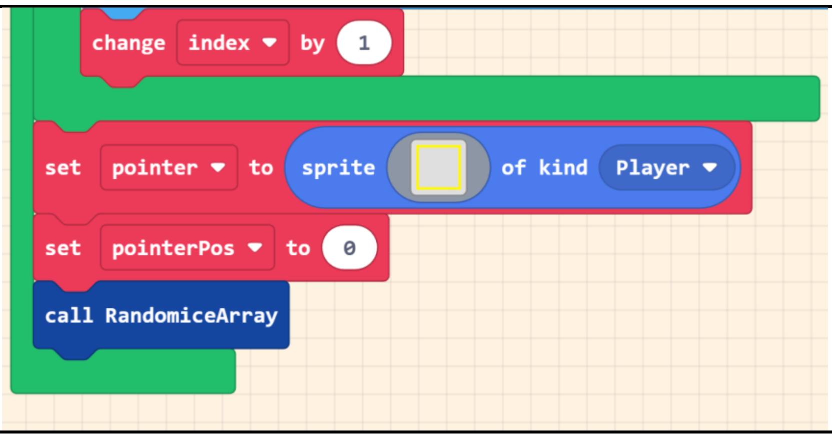
We will declare the "imaList" array, which will randomly place the values from "numberList".

To avoid conflicts with the input values, we will clear the used values.



```
function RandomiceArray
  set n to length of array numberList
  for index from 0 to n - 1
  do
    set random to pick random 0 to length of array numberList - 1
    imaList insert numberList get value at random at beginning
    numberList remove value at random
```

We will call the function in "randomizeArray" to activate it.



```
    change index by 1
  set pointer to sprite [ ] of kind Player
  set pointerPos to 0
  call RandomiceArray
```

With this programming, we will have established the quantity of cards in the game, their positions, and what each of them hides.