ual: creative computing institute

C++ Day 2

Working with Arrays

Date

Jasper Zheng

Before we start...

Working with Arrays

Recap: C++ is a statically-typed language

We specify the data type of each variable, and the type is fixed after we declared them

```
int my_number = 10;
string my_number_as_text = "Ten";
```

When declaring an array, it's the same. C++ arrays must be declared with both a type and a size at compile-time.

```
int myArray[] = {4, 5, 6, 10};
```

```
int myArry[10];
```

Before we start...

Working with Arrays

Access / Assign / Modify Elements

The index of an array starts from 0, if we want to access / assign the 3rd element in an array, we are looking for index 2.

```
int myArry[] = {1, 2, 3, 4, 5};
cout << myArry[2];
```

The output will be 3

```
int myArry[] = {1, 2, 3, 4, 5};
myArry[4] = 10;
```

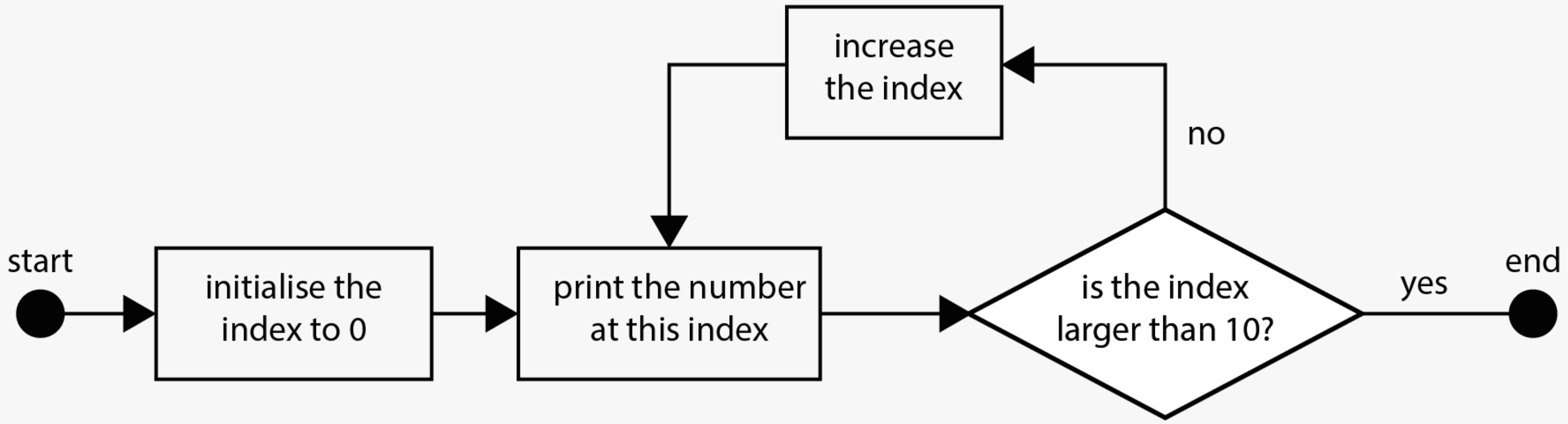
The new array will be `{1, 2, 3, 4, 10}`

Before we start...

Loops

Iterate Through an Array

If we have an array with 10 elements, and we want to create a function that print out each element one at a time.


start
initialise the index to 0
print the number at this index
is the index larger than 10?
yes
end
no
increase the index

A "For Loop" is what we need.

Before we start...

For Loops

```
int myNums[] = {1,2,3,4,5,6,7,8,9,10};

for (int i = 0; i < 10; i++){
    cout << myNums[i] << "\n";
}
```

`int i = 0;`

We use an integer variable `i = 0` as the initialiser of the for loop.

`i < 10;`

Within this condition, the loop will keep executing.

`i++`

This is executed every time after the code block has been executed.

Before we start...

For Loops

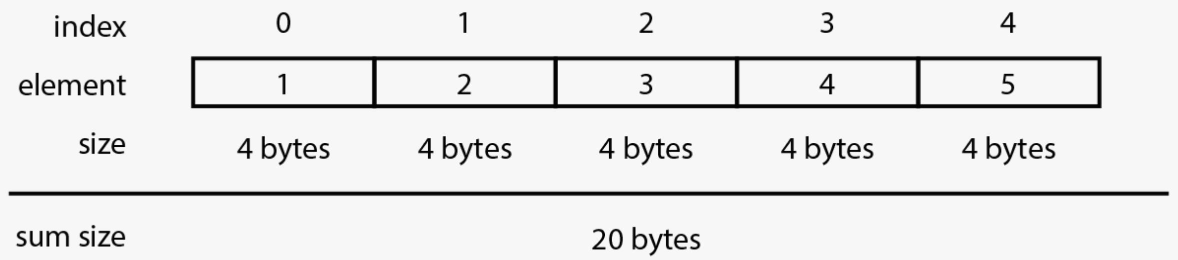
But what if the length of array change?

If we want our function to operate on array with different length, we will need to make the length as a variable.

But how to calculate the length of an array?

We know the array is taking 20 bytes of memory in our machine, and the array only has integers. We also learned from the Data Types chapter that an integer has 4 bytes.

Therefore we can calculate the length of our array: `length = 20 / 4 = 5`



Before we start...

For Loops

```
sizeof(myNums) / sizeof(myNums[0]);
```

`sizeof()` function will return the size of the input variable. We use the array's size divided by an element's size to calculate the length.

```
int myNums[] = {20,3,44,22,14,24,2};

int length = sizeof(myNums)/sizeof(myNums[0]);
printArray(myNums, length);
```

```
void printArray(int arr[], int length)
```

We write a function that takes both the array and it's length as parameters, and use the length as stopping condition of the for loop.

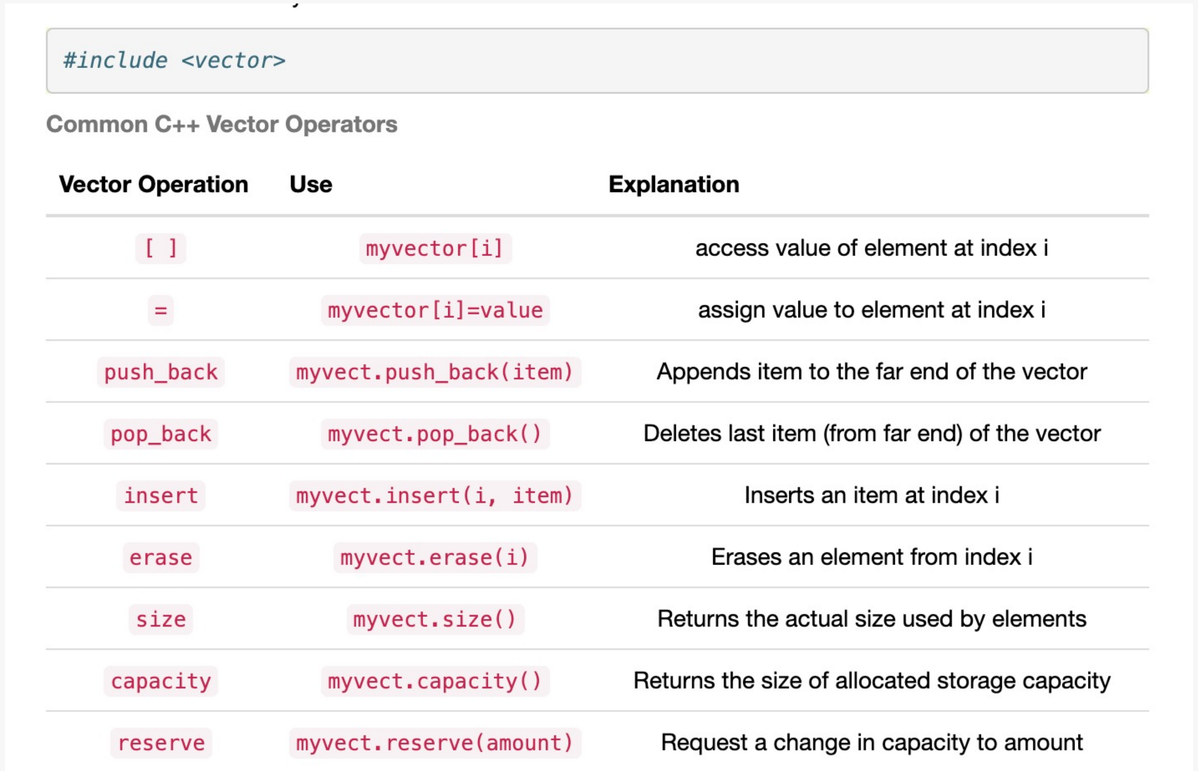
```
void printArray(int arr[], int length){
    for (int i = 0; i < length; i++){
        cout << arr[i] << " ";
    }
}
```

You may also see `void printArray(int *arr, int length)` this type of representation, it use star key to point the compiler to the array.

Before we start...

Vector

A type of collection with more friendly features



```
#include <vector>
```

Common C++ Vector Operators

Vector Operation	Use	Explanation
`[ ]`	`myvector[i]`	access value of element at index i
`=`	`myvector[i]=value`	assign value to element at index i
`push_back`	`myvect.push_back(item)`	Appends item to the far end of the vector
`pop_back`	`myvect.pop_back()`	Deletes last item (from far end) of the vector
`insert`	`myvect.insert(i, item)`	Inserts an item at index i
`erase`	`myvect.erase(i)`	Erases an element from index i
`size`	`myvect.size()`	Returns the actual size used by elements
`capacity`	`myvect.capacity()`	Returns the size of allocated storage capacity
`reserve`	`myvect.reserve(amount)`	Request a change in capacity to amount

Before we start...

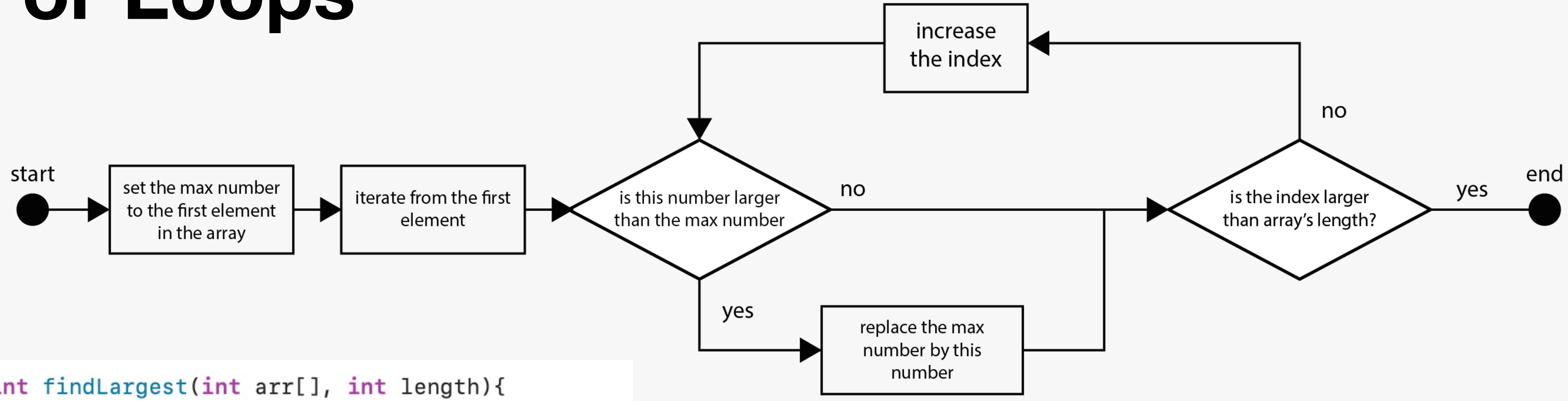
For Loops

Task: find largest number in an array

If we are given an array with integers, can we write a function that returns the largest number in that array?

Before we start...

For Loops


start
set the max number to the first element in the array
iterate from the first element
is this number larger than the max number
no
yes
replace the max number by this number
is the index larger than array's length?
no
increase the index
yes
end

```
int findLargest(int arr[], int length){
    int currentMax = arr[0];
    for (int i = 0; i < length; i++){
        if (arr[i] > currentMax){
            currentMax = arr[i];
        }
    }
    return currentMax;
}
```

Day 2 Activities

- Activity 1 - Arrays
- Activity 2 - For Loops
- Activity 3 - 2D Arrays and Nested Loops

Day 2 - Activity 1 & 2

Arrays and Loops

Arrays

Have a quick look at Chapter 5.2 Arrays. (Approx. 10 min)

For Loop

Have a quick look at Chapter 3 For Loops (Approx. 5 min)

Task 01 (approx. 15 min) :

Check the code for ‘find largest number’ example we have mentioned today. If you found it difficult to understand, try comparing it with the flowchart we made today.

An optional extension to this task: what if we’d like to find the second largest element in an array? what if we want to sort all elements into ascending order? You don’t need to make these in codes, but just think about it. [answers are included in the resources page]

Day 2 - Activity 1 & 2

Arrays and Loops

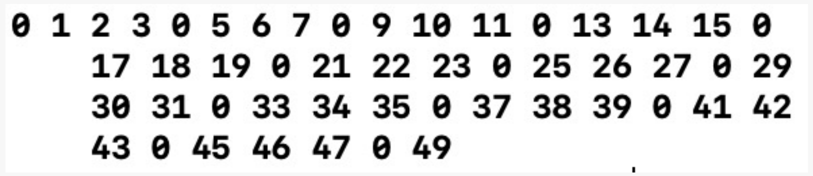
Task 02 (approx. 15 min) :

Initialise an array with the first 50 positive integers (use a loop to do it). Then change all numbers that are divisible by 4 to 0.

Then print the new array to the console.

Hint: try modulo operation

Solutions



```
0 1 2 3 0 5 6 7 0 9 10 11 0 13 14 15 0
    17 18 19 0 21 22 23 0 25 26 27 0 29
    30 31 0 33 34 35 0 37 38 39 0 41 42
    43 0 45 46 47 0 49
```

Day 2 - Activity 3

2D Arrays and Nested Loops

2D Arrays and Nested Loops (approx. 15 mins)

Watch and follow this tutorial from 2:45:21 to 2:54:55

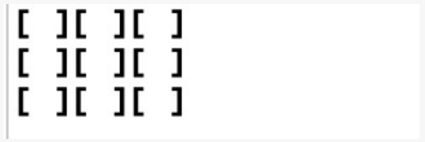
Day 2 - Activity 3

2D Arrays and Nested Loops

Task 03 [optional]: Tic-Tac-Toe (X's and O's Game)

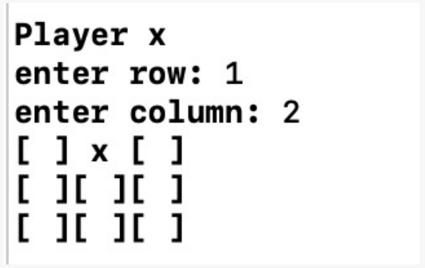
We are making a tic-tac-toe game in the console.

First print a grid to the console.

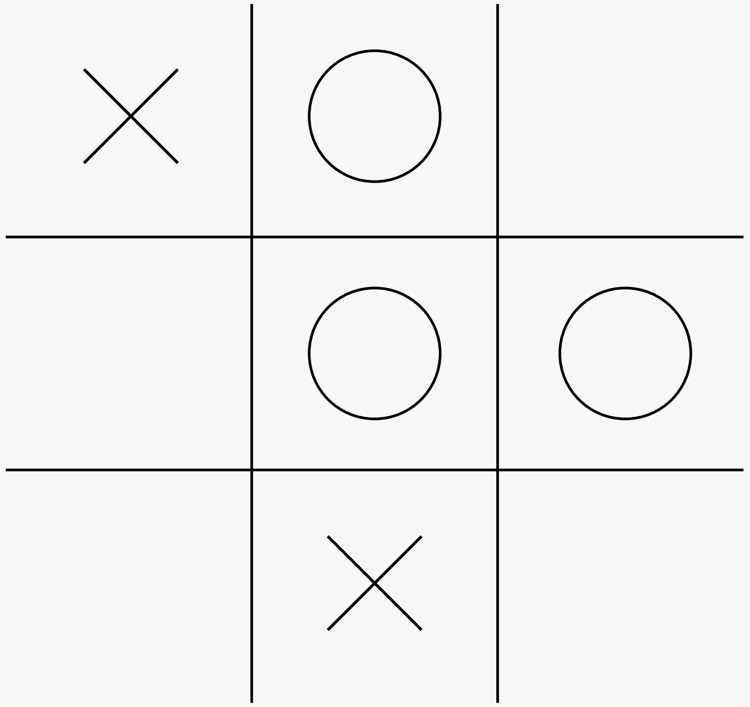


```
[ ][ ][ ]
[ ][ ][ ]
[ ][ ][ ]
```

Two players take turn entering row and column number to place their marker. After each marker is placed, print out the grid again.



```
Player x
enter row: 1
enter column: 2
[ ] x [ ]
[ ][ ][ ]
[ ][ ][ ]
```



Day 2 - Activity 3

2D Arrays and Nested Loops

Hints for Task 3

Eventually, the console may look something like this ->

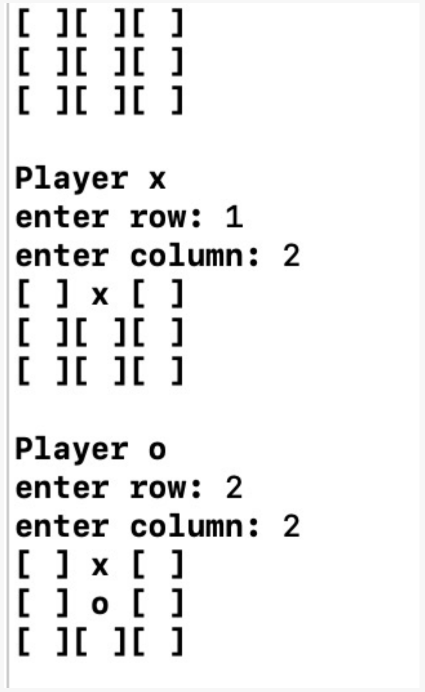
We may have a 2d array representing the grid, (e.g. 0 for empty, 1 for ‘x’, 2 for ‘o’). Player’s action will modify data in the array. Then we may use a function to print out the grid according to this array.

Our program will continuously take inputs from the player, so combine the code we’ve produced yesterday.

How do we decide whose turn is it? The modulo operator in task 2 may help.

Do we need to handle exceptions? What if the user enter an invalid row or column number, what if the gird is already taken by the other player?

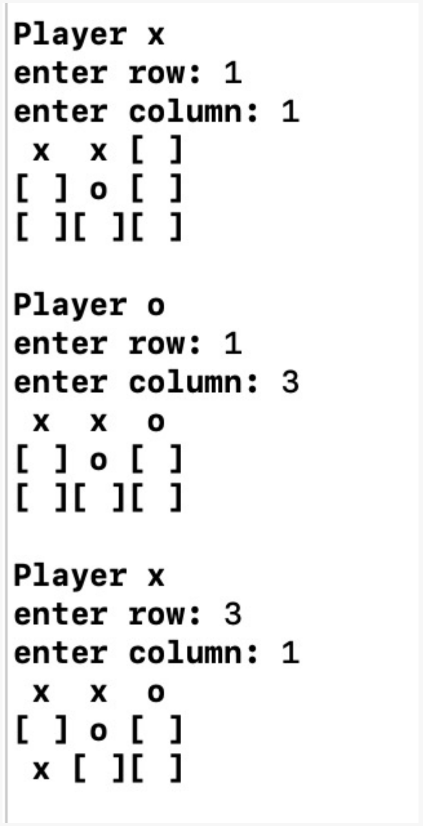
If you’re struggling with the code, don’t worry and take a step back, try to produce a flowchart at first, like the one we did in the for-loop section.



```
[ ][ ][ ]
[ ][ ][ ]
[ ][ ][ ]

Player x
enter row: 1
enter column: 2
[ ] x [ ]
[ ][ ][ ]
[ ][ ][ ]

Player o
enter row: 2
enter column: 2
[ ] x [ ]
[ ] o [ ]
[ ][ ][ ]
```



```
Player x
enter row: 1
enter column: 1
 x  x [ ]
[ ] o [ ]
[ ][ ][ ]

Player o
enter row: 1
enter column: 3
 x  x  o
[ ] o [ ]
[ ][ ][ ]

Player x
enter row: 3
enter column: 1
 x  x  o
[ ] o [ ]
 x [ ][ ]
```

Day 2 - Solutions

Task 02 – Solution Pt. 1

Initialise an array by loop

```
int myNums[50];

for (int i = 0; i < 50; i++){
    myNums[i] = i;
}
```

Day 2 - Solutions

Task 02 – Solution Pt. 2

modulo operator %

We use a modulo operator to figure out if a number is divisible by 4.

A modulo operator finds the remainder when an integer is divided by another.

e.g. `5 % 2 = 1` because 5 divides by 2 (twice), with 1 remaining.

Modulo operator is useful when the you have a group of tasks happening in turns.

```
int myNums[50];

for (int i = 0; i < 50; i++){
    if (i % 4 == 0){
        myNums[i] = 0;
    } else {
        myNums[i] = i;
    }
}
```

Day 2 - Solutions

Task 02 – Solution Pt. 3

If we have 5 players, their action starts in turn

4 % 4 = 0 because 4 divides by 4 (once), with 0 remaining. -> player 1's turn

5 % 4 = 1 because 5 divides by 4 (once), with 1 remaining. -> player 2's turn

6 % 4 = 2 because 6 divides by 4 (once), with 2 remaining. -> player 3's turn

7 % 4 = 3 because 7 divides by 4 (once), with $\underline{\text{3 remaining}}$. -> player 4's turn

8 % 4 = 0 because 8 divides by 4 (twice), with 0 remaining.

9 % 4 = 1 because 9 divides by 4 (twice), with 1 remaining.

10 % 4 = 2 because 10 divides by 4 (twice), with 2 remaining.

11 % 4 = 3 because 10 divides by 4 (twice), with $\underline{\text{3 remaining}}$.

Day 2 - Solutions

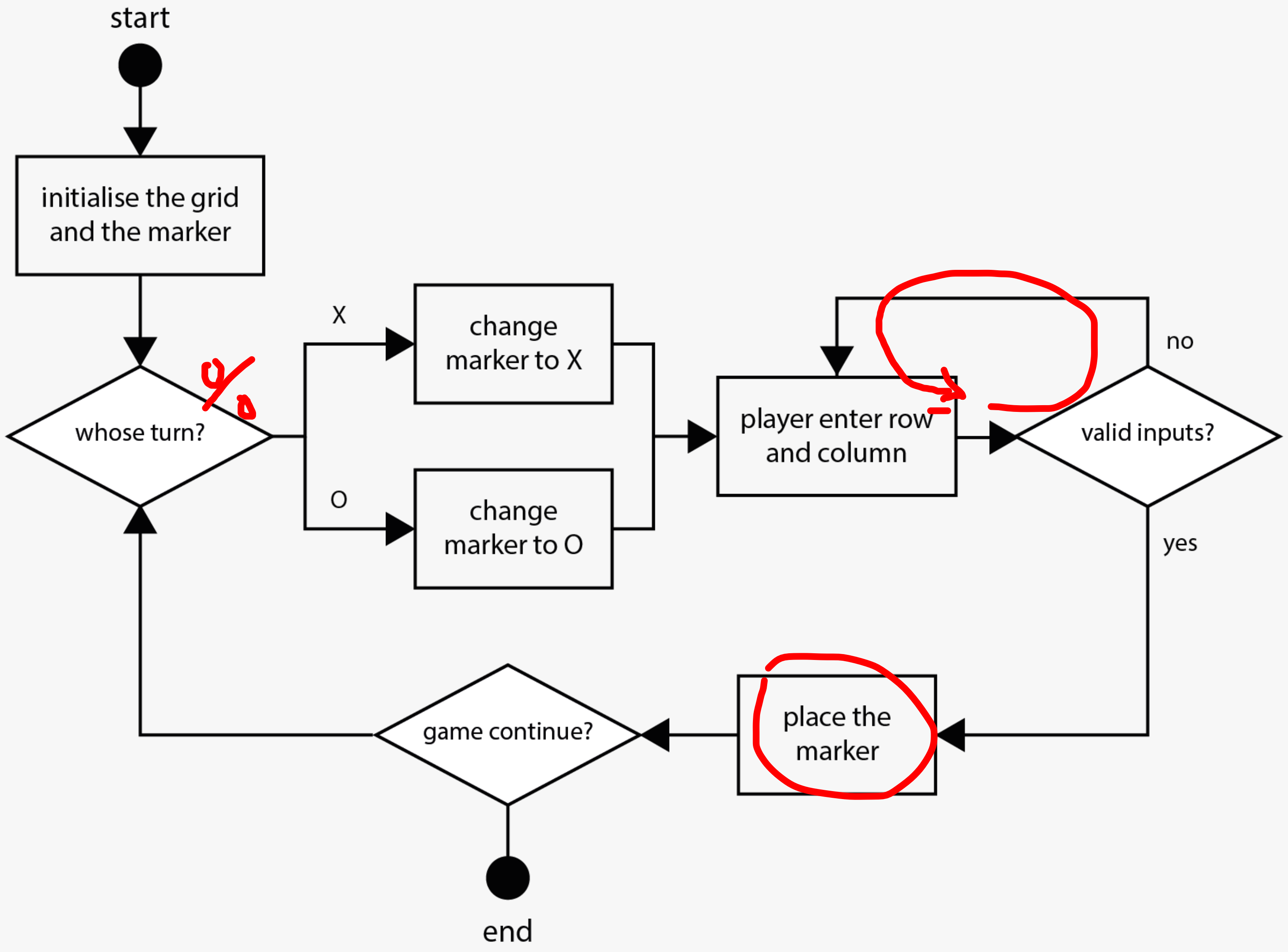
Task 03 – Solution Pt. 1

Key components

- Print grids
- Decide whose turn
- Take inputs
- Print grids again
- ...

Day 2 - Solutions

Task 03 - Solution Pt.2



Codes

Day 2 Resources

Codes for today

For Loop While Loop Array Vector

[optional] second largest element in an array:

https://www.geeksforgeeks.org/find-second-largest-element-array/

[optional] sort an array into ascending order:

In fact there are more than 40 types of sorting algorithm, classic methods include bubble sort, selection sort, merge sort... take a look at these animations, can you tell the different between them? which one is performing better?

[optional] 21min video explaining recursion (remember that very cool technique finding largest number without any loop?)

https://www.youtube.com/watch?v=ngCos392W4w

Day 2 de-brief

- How was today for you?
- What has gone well?
- What went as planned?
- What surprised you?
- Did you find today difficult?

- Share anything you made?
-
  Ask around the class and see if they have anything to share?

ual: creative computing institute

Day 2 Survey

https://artslondon.padlet.org/hbrueggemann/j2yr3zfwkap4v4rq

The password is **Jumpstart**.

Thank you, see you tomorrow at TIME