ual: creative computing institute

C++ Day 2

Arrays and Loops

2023

Jasper Zheng

Outlook day 2

- Recap at yesterday's tasks

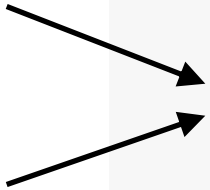
- Arrays and Loops
  - Iterating Through an Array
  - Example task: finding the largest number
- 2D Array (Nested array)
- Modulo Operator (%)
- Tic-Tac-Toe

Common Questions

Functions in C++

Parameter types really matters

```
29  void calculate(int number){
30      cout << "Integer as input";
31  }
32
33  void calculate(double number){
34      cout << "Double as input";
35  }
```



These are different functions

Actually it's called function polymorphism (A function behaves differently in different situations).

Further reading: [Polymorphism in C++]

Common Questions

Return Multiple Variables

`struct` is what you'll need. Structures are a way to group several related variables into one place.

```
struct Velocity {
    double magnitude;
    bool direction;
};

Velocity getVelocity(){

    Velocity myVelocity;
    myVelocity.magnitude = 2.0;
    myVelocity.direction = true;

    return myVelocity;
}
```

Before we start...

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";
```

It's the same for arrays. C++ arrays must be declared with both a type and a size:

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

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

Arrays

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

Access Elements

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

The output will be 3

Assign / Modify Elements

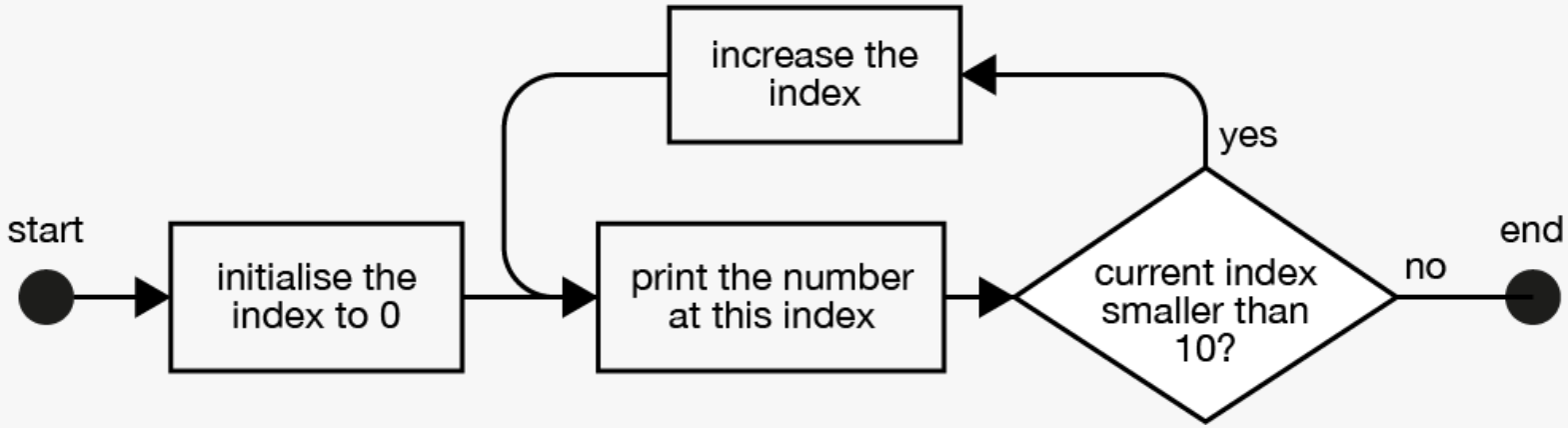
```
int myArry[] = {1, 2, 3, 4, 5};
myArry[4] = 10;
```

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

Loops

Iterate Through an Array

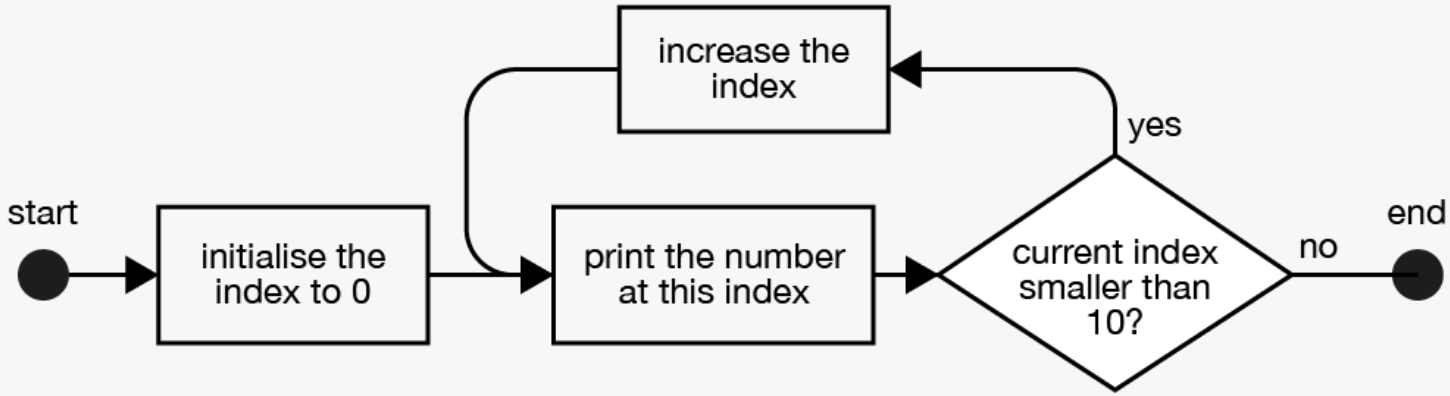
If we have an array with 10 elements, and we want to print out each element one at a time:



A "For Loop" is what we need.

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;`

If this condition is true, the loop will keep executing.

`i++`

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

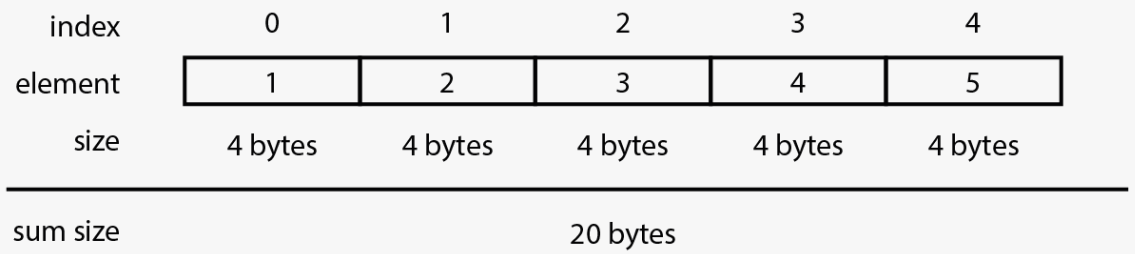
For-Loops

But what if the length of array change?

If we want our function to work for arrays with different length, we will need to make the length as a variable.

But how to calculate the length of an array?

- We can know the total amount of memory taken by an array (e.g. 20 bytes).
- We also learned from the Data Types chapter that an integer takes 4 bytes of memory.



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

For-Loops

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

`sizeof()` function will return the size of the input variable (i.e. the amount of memory used by the variable). Then divide the array's size 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 a stopping condition of our loop.

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

You may also see this type of representation:

```
void printArray(int *arr, int length)
```

It use star key to point the compiler to the array.

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 number of elements in the vector.
`capacity`	`myvect.capacity()`	Returns the size of allocated storage capacity
`reserve`	`myvect.reserve(amount)`	Request a change in capacity to amount

[Vectors in C++]

Find the Largest Number

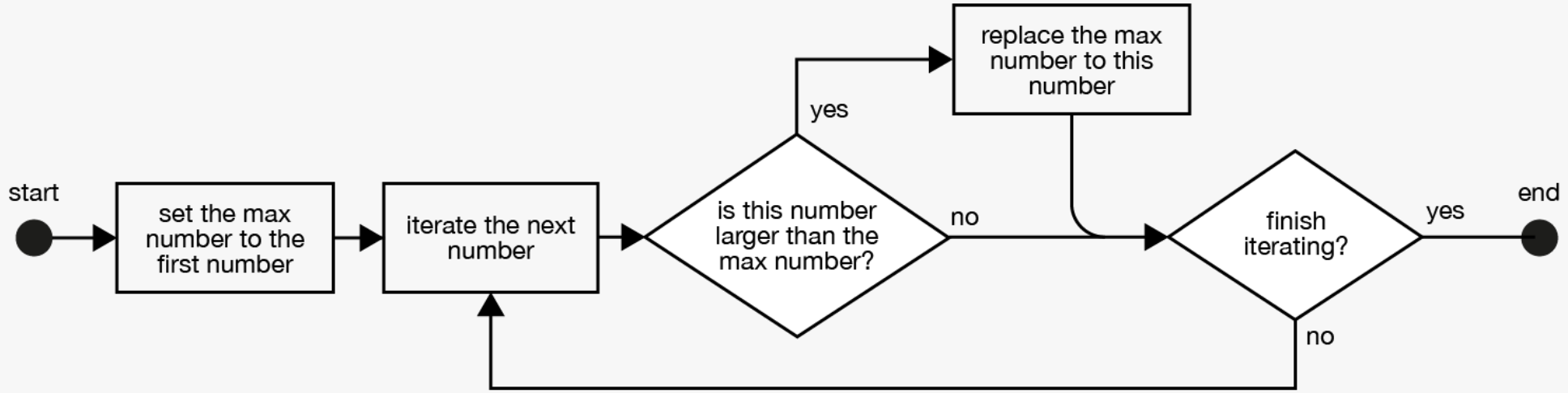
Example task: find the largest number in an array

If we are given an array with 10 integers, can we write a function that returns the largest number in that array?

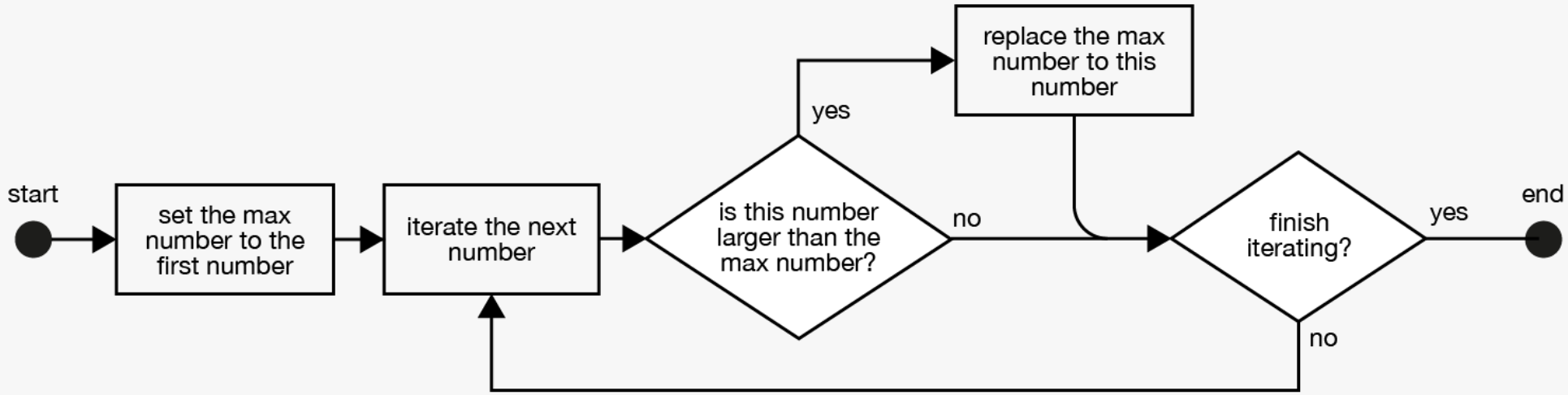
Find the Largest Number

Example task: find the largest number in an array

If we are given an array with 10 integers, can we write a function that returns the largest number in that array?



Find the Largest Number




```
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;
}
```

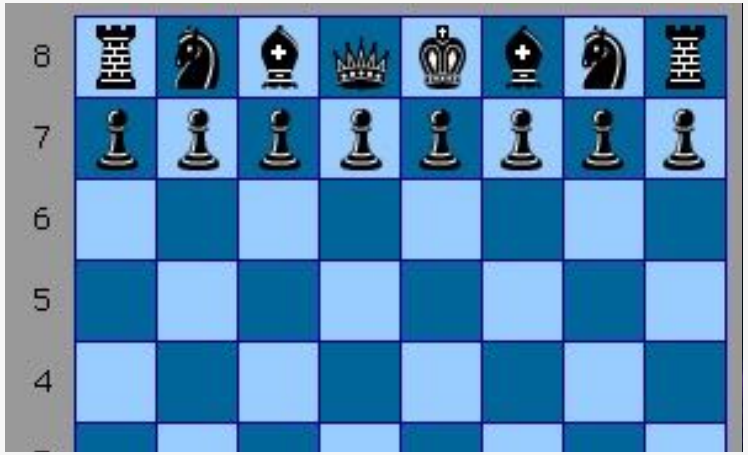
Full Code

2D (Multi-Dimensional) Arrays

An array of arrays

```
10      int grid[2][4] = {
11          {0, 1, 2, 3},
12          {4, 5, 6, 7}
13      };
```

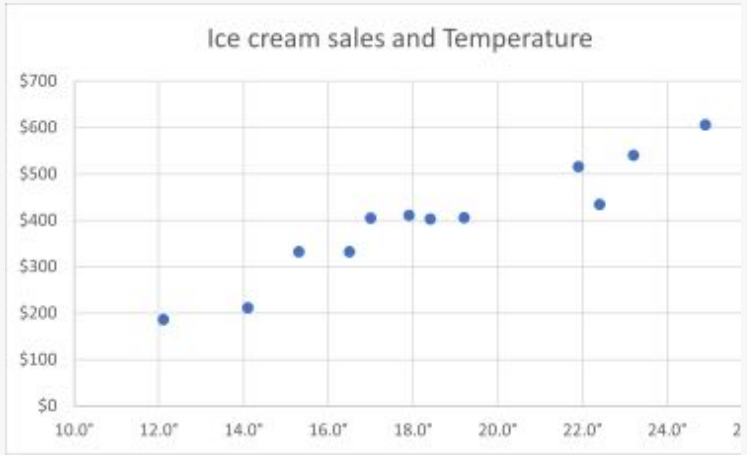
```
grid[0][0];  // 0
grid[0][3];  // 2
grid[1][2];  // 6
```



Chessboard

```
int board[8][8];
```

each representing a cell in the grid

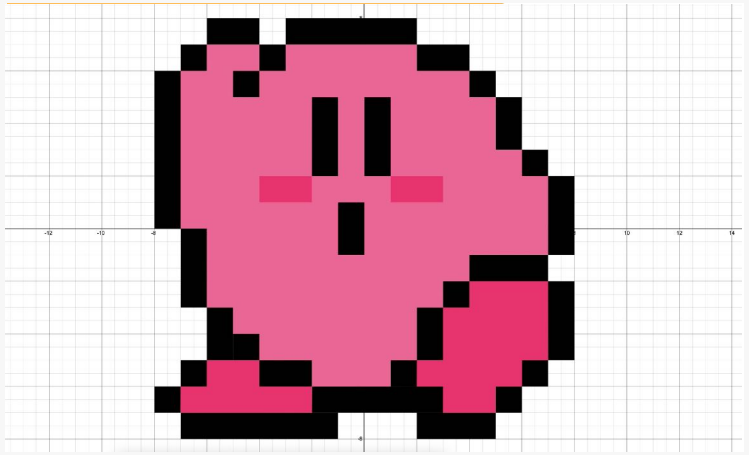


Ice Cream Sales and Temperature

```
double sales[n][2];
```

n: number of days

2: temperature and number of sales at that day



Images

```
int image[height][width];
```

grayscale images

```
int image[height][width][3];
```

RGB images (3 refers to colour channels)

Accessing 2D Arrays

A loop in a loop (nested loop)

```
10      int grid[2][4] = {
11          {0, 1, 2, 3},
12          {4, 5, 6, 7}
13      };
```

```
15      for (int r = 0; r < 2; r++){
16          for (int c = 0; c < 4; c++){
17              cout << grid[r][c] << endl;
18          }
19      }
```

Daily Code Jumpstart Choreography

	Mon	Tues	Wed	Thurs	Fri
9am-10am	---	Coaching aims	Daily Aims and Objectives	Daily Aims and Objectives	---
10am-13oo	---	Self-study Time	Self-study Time	Self-study Time	---
Break	---	Social Lunch	Social Lunch	Social Lunch	---
14oo-16oo	---	Self-study Time	Self-study Time	Self-study Time	---
18oo-19oo	---	QandA with Coach	QandA with Coach	QandA with Coach	---

Day 2 Tasks

- Task 1 – Arrays (approx. 5 mins)
- Task 2 - For Loops (approx. 30 mins)
- Task 3 - 2D Arrays and Nested Loops (approx. 60 mins)

Task 1 & 2

Arrays and Loops

Task 1 - Arrays

Have a quick look at Chapter 5.2 Arrays. (Approx. 5 min)

Task 2.1 - For Loop

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

Task 2.2

Arrays and Loops

Task 2.2 (approx. 10 min):

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

Optional extensions 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]

Task 2.3

Arrays and Loops

Task 2.3 (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.

Ideal output:

```
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
```

Hint: try modulo operator

Solution

Task 3.1

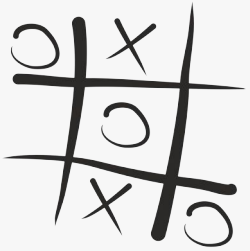
2D Arrays and Nested Loops

Task 3.1: 2D Arrays and Nested Loops (approx. 10 min):

- Option 1: Read this chapter on C++ Multi-Dimensional Arrays
- Option 2: Watch and follow this tutorial from 2:45:21 to 2:54:55

Task 3.2

Tic-Tac-Toe



Task 3.2: Tic-Tac-Toe (X's and O's Game)

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

Step 1: First print a grid to the console.

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

Step 2: The program asks the 'x' player to place a mark by entering a row and a column number.

After a mark is placed, print out the updated grid.

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

Step 3: The program asks the 'o' player to place a mark by entering a row and a column number. Then print out the grid again.

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

Step 4: Two players take turns placing their marks until the grid is filled.

(Our program doesn't need to be able to decide who's the winner at this stage; we'll try to do that tomorrow.)

Task 3.2

Tic-Tac-Toe

An ideal console interaction:

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

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 [ ][ ]

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

```
Player x
enter row: 2
enter column: 3
 x  x  o
 o  o  x
 x [ ][ ]

Player o
enter row: 3
enter column: 2
 x  x  o
 o  o  x
 x  o [ ]

Player x
enter row: 3
enter column: 3
 x  x  o
 o  o  x
 x  o  x
```

Task 3.2

Tic-Tac-Toe

Hints for Task 3.2

- We may have a 2d array representing the grid, (e.g. 0 for empty, 1 for 'x', 2 for 'o'). Players' actions 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 players, so combine the code we produced yesterday.
- How do we decide whose turn it is? The modulo operator in task 2 may help.
- Do we need to handle exceptions: What if users 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.

Solution

Task 2.3 – Solution

Task 2.3 - Initialise an array with the first 50 positive integers, then change all numbers that are divisible by 4 to 0.

Step 1 - Initialise an array by a loop

```
int myNums[50];

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

Solution

Task 2.3 – Solution

Task 2.3 - Initialise an array with the first 50 positive integers, then change all numbers that are divisible by 4 to 0.

Step 2 - Modulo operator %

```
int myNums[50];

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

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.

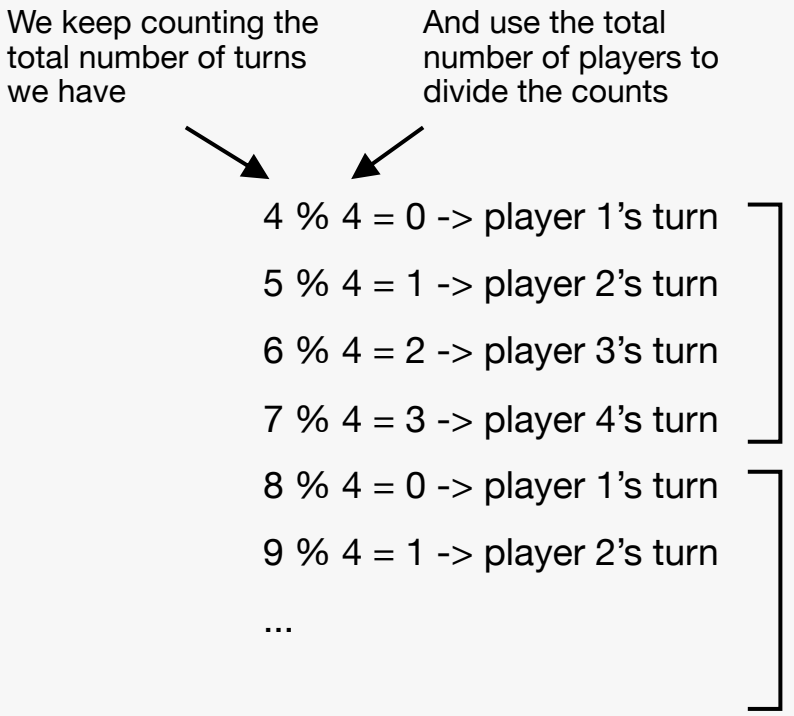
Solution

Task 2.3 – Solution

Modulo operator %

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

e.g. If we have 5 players, their action starts in turn:



Solution

Tic-Tac-Toe – Solution

Task 3.2: Tic-Tac-Toe (X's and O's Game)

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

Step 1: First print a grid to the console.

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

Step 2: The program asks the 'x' player to place a mark by entering a row and a column number.

After a mark is placed, print out the updated grid.

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

Step 3: The program asks the 'o' player to place a mark by entering a row and a column number. Then print out the grid again.

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

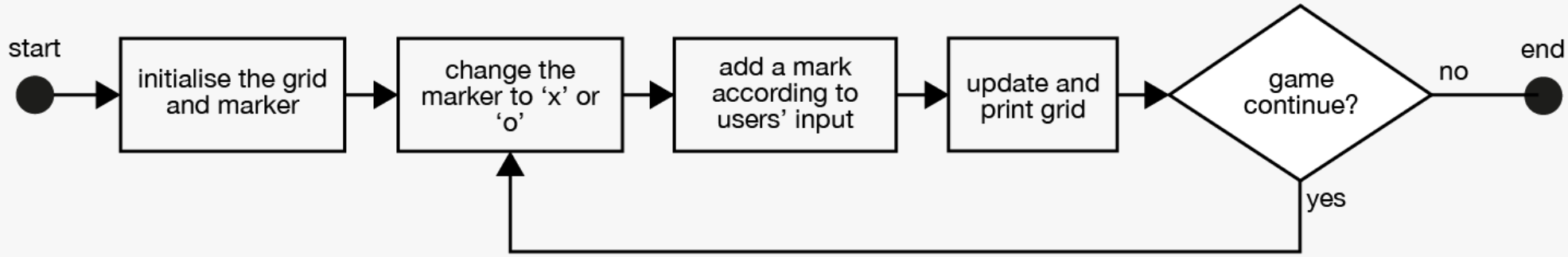
Step 4: Two players take turns placing their marks until the grid is filled.

Solution

Tic-Tac-Toe – Solution

Key components:

- Print grids
- Decide whose turn
- Take inputs
- Update grids



Solution

Tic-Tac-Toe – Solution

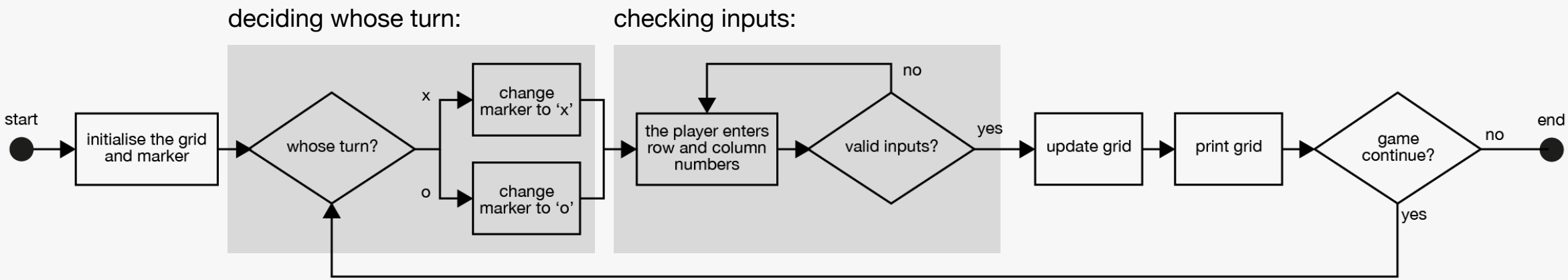
An overall structure of our program:

```
int grid[3][3];       // grid: 0 for empty; 1 for player x; 2 for player o;
int marker;           // marker: 1 for player x, 2 for player o;

void initialiseGrid();
void showGrid();
bool checkInput(int x, int y);

int main() {
    initialiseGrid();
    showGrid();
    for (int i = 0; i < 9; i++){

        //TODO:
        // • change marker
        // • get and check input
        // • update grid

        showGrid();
    }
}
```

Solution

Tic-Tac-Toe – Solution

Add details to our flowchart:



Solution

Tic-Tac-Toe – Solution

Fill the `//TODO` section:

```
for (int i = 0; i < 9; i++){
    marker = i % 2 + 1;
    while (true) {
        if (marker == 1){
            cout << "Player x \n";
        } else {
            cout << "Player o \n";
        }
        cout << "enter row: ";
        cin >> y;
        cout << "enter column: ";
        cin >> x;
        if (checkInput(x, y)){
            break;
        }
    }
    grid[y-1][x-1] = marker;
    showGrid();
}
```

decide whose turn

check inputs

take inputs

update and display the grid

[Full Codes]

Day 2 Resources

Codes

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