COMP1511 - Programming Fundamentals

Term 1, 2019 - Lecture 9
Stream B

What did we cover last week?

Theory of a Computer

- Turing machines
- Processors (CPU) and Memory (RAM, Disks etc)

Arrays

- Collections of variables

Functions

- Separate code that we can run as part of our programs

What are we covering today?

More Arrays and Functions

- Putting Arrays in other Arrays
- Using multiple functions in a program
- Using a function from a standard library

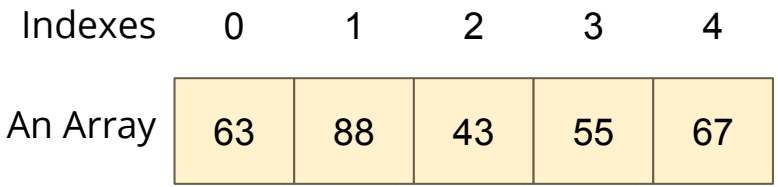
The first Assignment

- An explanation of the game Coco
- An explanation of what is expected in the assignment

Recap of Arrays

A collection of variables

- All elements of the array are the same type
- Declared using a fixed sized and type
- Elements are accessed via their index (an integer)

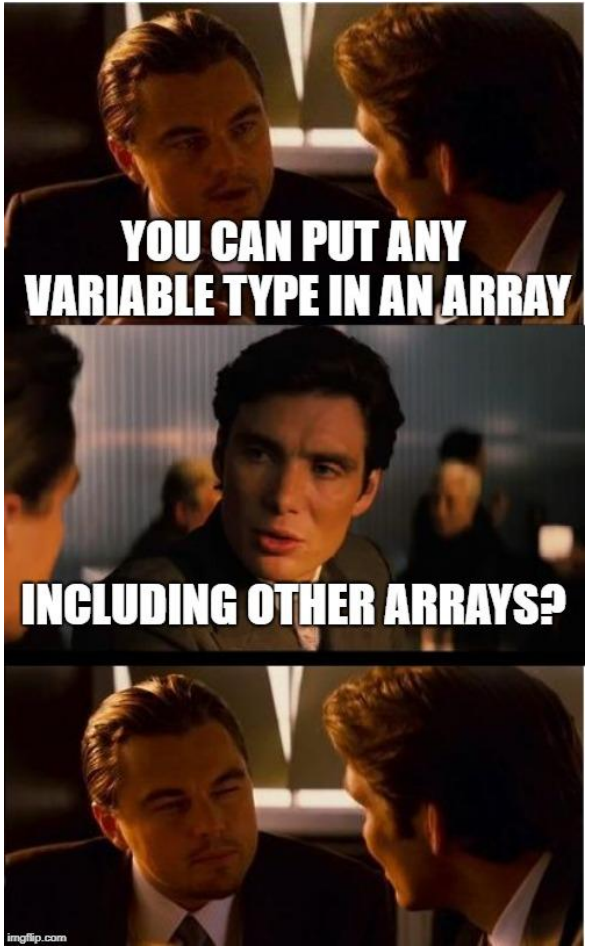


Arrays inside Arrays

An Array is a type of variable

An Array can contain any type of variable

- Arrays can be put inside other arrays!
- We call these multi-dimensional arrays
- Think of them as a grid, two or more dimensions



Two Dimensional Arrays

Arrays inside arrays

- Can be thought of like a grid
- The outer array contains arrays
- Each array is a column of the grid
- Addressed using a pair of integers like coordinates
- All inner arrays are of the same type

Indexes	0	1	2	3	4
0	63	88	43	55	67
1	63	88	43	55	67
2	63	88	43	55	67

A 2D Array

Two Dimensional Arrays in Code

```
int main (void) {
    // declare a 2D Array
    int grid[4][4] = {0};

    // assign a value
    myArray[1][3] = 3;
    // test a value
    if (myArray[2][0] < 1) {
        // print out a value
        printf("The bottom left square is: %d", myArray[0][3]);
    }
```

Recap of Functions

Code outside of our main that we can use (and reuse)

- Has a name that we use to call it
- Has an output type and input parameters
- Has a body of code that runs when it is called
- Uses return to exit and give back its output

Functions in Code

```
// a function declaration
int add (int a, int b);

int main (void) {
    int firstNumber = 4;
    int secondNumber = 6;
    // use the function here
    int total = add(firstNumber, secondNumber);
    return 0;
}

// the function is defined here
int add (int a, int b) {
    return a + b;
}
```

Why use functions?

Why do we separate code into functions?

Saves us from repeating code

- Instead of replicating code, we can write it once
- This also makes the code much easier to modify

Easier to organise code

- Complex functionality can be hidden inside a function
- The flow of the program can be read easily with clear function names

C Libraries

We've already used stdio.h several times

- C has other standard libraries that we can make use of
- The simple C reference in the Weekly Tests has some information
- math.h is a useful library of common maths functions
- stdlib.h has some useful functions
- Look through the references (also available in many places online)
- Don't worry if you don't understand the functions yet, some of them have no context in the programming we're doing so far

Using Libraries

```
// include some libraries
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

int main (void) {
    int firstNumber = -4;
    int secondNumber = 6;

    // change a number to its absolute value
    firstNumber = abs(firstNumber);

    // calculate a square root
    int squareRoot = sqrt(firstnumber);
    printf("The final number is: %d", squareRoot);
    return 0;
}
```

Let's work with 2D Arrays and Functions

I would like to make a simple game called "The Tourist"

- The world is a square grid
- The tourist can move up, down, left or right
- Be able to print out the world, including the location of the tourist and where they've been
- The tourist likes seeing new things . . .
- Score points for visiting unique places
- Lose points for revisiting old places

Break it down!

As usual, let's start simple and add features

- Make a square grid world
- Track the location of the tourist
- Be able to output what the world is

Make the Square Grid World

```c
#include <stdio.h>

#define WIDTH 8
#define HEIGHT 6

int main (void) {
    int grid[WIDTH][HEIGHT] = {0};
    int posX = 0, posY = 0;

    return 0;
}
```

Make a function that displays the grid

```
void displayGrid(int width, int height, int grid[width][height],
                 int posX, int posY) {
    int y = 0;
    while (y < height) {
        int x = 0;
        while (x < width) {
            if (x == posX && y == posY) { // The Tourist's location
                printf("T");
            } else { // print . for empty space
                printf(".");
            }
            x++;
        }
        printf("\n");
        y++;
    }
    return;
}
```

Add that function to the program

```c
#include <stdio.h>

#define WIDTH 8
#define HEIGHT 6

void displayGrid(int width, int height,
                 int grid[width][height], int posX, int posY)

int main (void) {
    int grid[WIDTH][HEIGHT] = {0};
    int posX = 0, posY = 0;

    // show the current status
    displayGrid(WIDTH, HEIGHT, grid, posX, posY);

    return 0;
}
```

Break Time

Arrays

- Two dimensional arrays

Functions

- Functions from libraries

Help Sessions, Weekly Tests

The course is speeding up a bit now, so feel free to use the help!

Help Sessions

- Tuesday, Wednesday, Thursday 6-8pm, Bugle and Horn Labs
- Friday 3-5pm, Viola and Cello Labs

Weekly Tests

- Don’t forget about these!
- They’re worth a few marks and are a good indicator of your progress

Continuing with our Tourist

Next Steps

- Let’s add movement
- Then track where the Tourist has been using the map
- After that, we total the score

Looping

- We can loop repeatedly for “turns” to allow the user to input directions

Movement - this code will loop

```
printf("Please enter a numpad direction or 0 to exit: ");
int input;
scanf("%d", &input);
if (input == 4) {          // left
    posX--;
} else if (input == 8) { // up
    posY--;
} else if (input == 6) { // right
    posX++;
} else if (input == 2) { // down
    posY++;
} else if (input == 0) { // exit
    exit = 1;
} else {                   // invalid
    printf("Input is not a numpad direction, please use 2,4,6 or 8\n");
}
```

Tracking the Tourist using the Map

Add 1 to each location we visit in the grid

```
// loop and let the user control the Tourist's movement
int exit = 0;
while (!exit) {
    // mark the location as having been visited by incrementing
    grid[posX][posY]++;

    // show the current status
    displayGrid(grid, posX, posY);

    printf("Please enter a numpad direction or 0 to exit: ");

    // Movement code from previous slide goes here . . .
```

Counting Score

```
int calculateScore(int width, int height, int grid[width][height]) {
    int score = 0;
    int y = 0;
    while (y < height) {
        int x = 0;
        while (x < width) {
            if (grid[x][y] == 1) { // only visited once
                score++;
            } else if (grid[x][y] > 1) { // visited too many times
                score--;
            }
            x++;
        }
        y++;
    }
    return score;
}
```

The Tourist Game

This is now roughly complete (but without proper testing!)

- We can move the tourist
- We can track where we've been and how many times
- We can display our visits as well as current location
- We can calculate our score

We're going to need some testing though!

- Try different inputs
- Try moving around a bit

Testing!

Moving around and seeing what works

- Use the controls to move around the map
- Try entering some integers that aren't the movement

What issues do we find?

Walking off the edge of the map

Our Tourist can walk outside of the bounds of our arrays!

This will run every time we move to make sure we stay on the map

```
// Check if we've walked off the map
if (posX < 0) {
    posX = 0;
} else if (posX >= WIDTH) {
    posX = WIDTH;
}
if (posY < 0) {
    posY = 0;
} else if (posY >= HEIGHT) {
    posY = HEIGHT;
}
```

Assignment one - Coco

The first assignment has been released and it's based on a card game

- Coco is a card game invented just for this assignment
- It has many similarities to existing card games
- It's designed for four players
- It falls within the "trick taking" family of games

You will be writing a Coco player

- The assignment is to write a program that can play Coco

How does the Game Coco Work?

There are 40 cards, numbered 10-49

- Each person is dealt 10 cards
- Each round each player will take turns playing a single card
- So there will be 10 rounds of the game in total
- Each round of four cards in the centre is called a trick
- The trick will be “taken” by one player (more on this later)

Discards after the hand is dealt

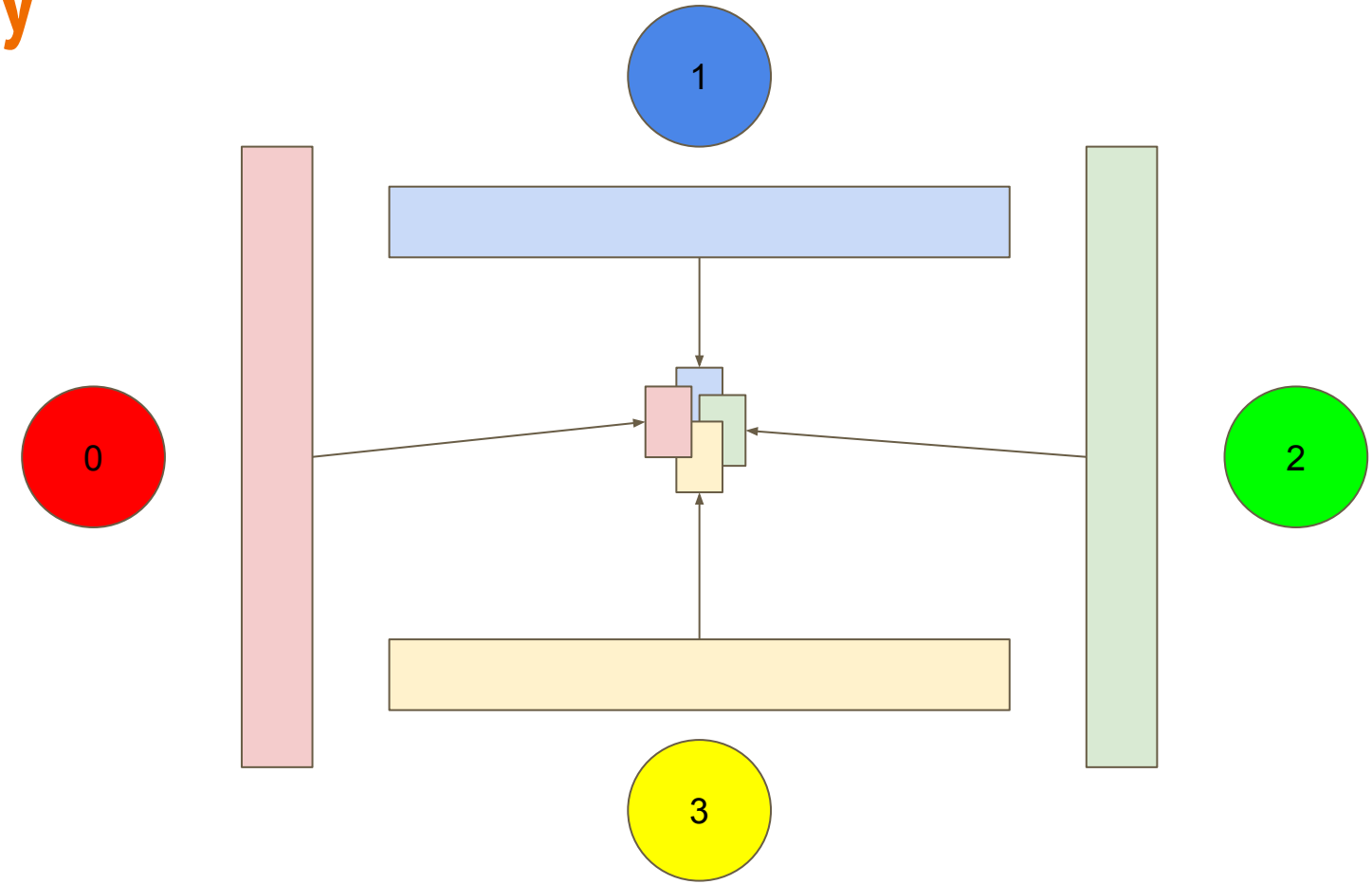
- After receiving 10 cards, every player will take 3 that they don’t want and pass them to the player on their left

Flow of Play

In this case . . . Player 1 goes first, then 2, 3 then 0.

Each plays one card from their hand onto the table.

Rules will decide which player ends up keeping the “trick”



What types of cards are there?

We use mathematical properties to differentiate cards

- Prime numbers are special cards
- 42 is called “The Douglas” and has special rules
- All other cards are related to each other based on cocompositeness

Cocomposites

- Any two cards that share a factor are cocomposite
- Eg: All even numbers are cocomposite (sharing factor 2)

Rules for which cards to play

The first card in a round decides how everyone else must follow

- If the first card is prime, all players must play a prime
- If the first card is another number, all players must play a cocomposite

- If a player can't follow the leader, they can play any card in their hand

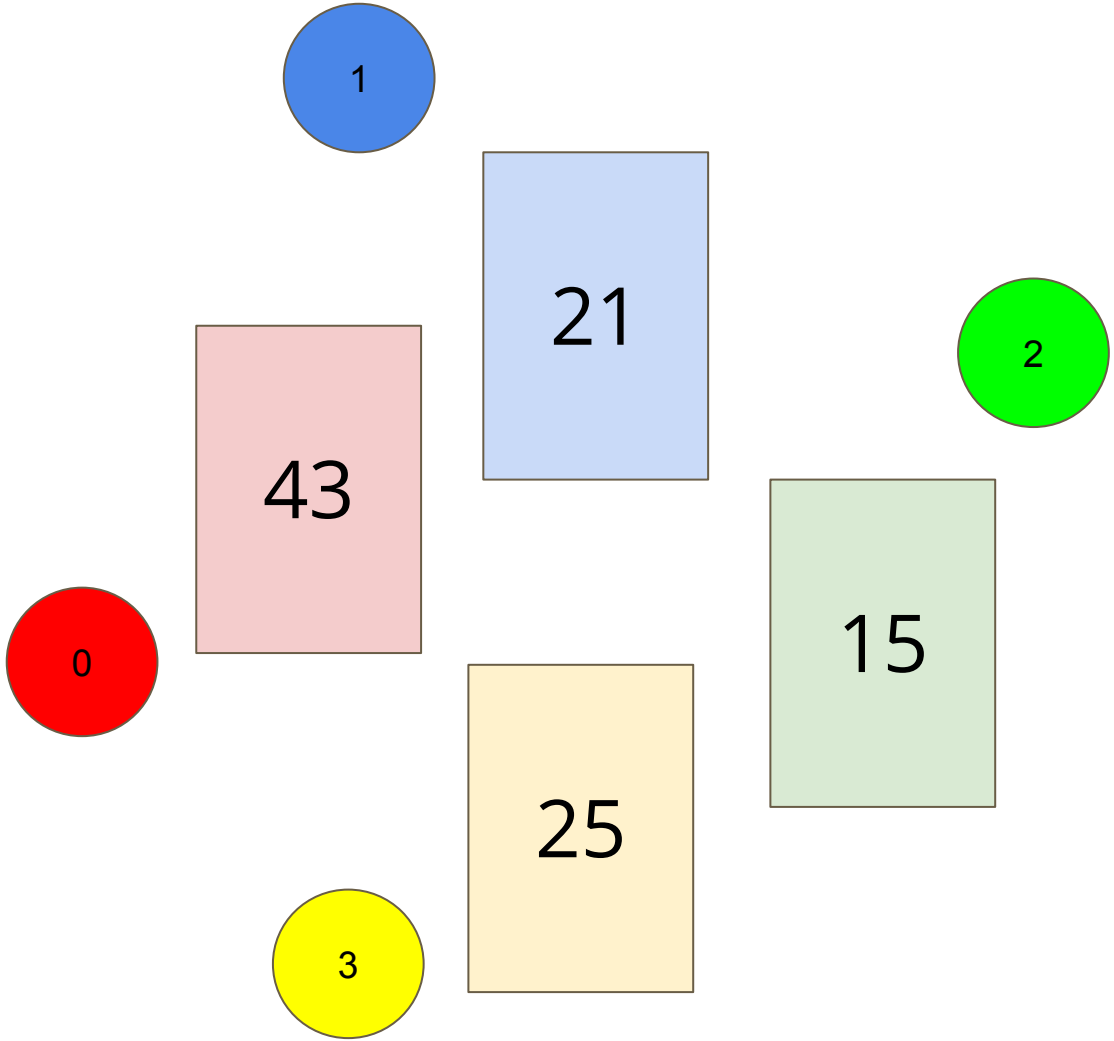
Leading with a Prime

- A player can only start a round with a prime if a prime number has been played in one of the earlier rounds

A single round

Player 2 (green) goes first

- 2 plays 15
- 3 is next and plays 25 (shared factor 5)
- 0 doesn't have any cards with factors 3 or 5, so they play 43, a prime
- 1 plays 21 (shared factor 3)



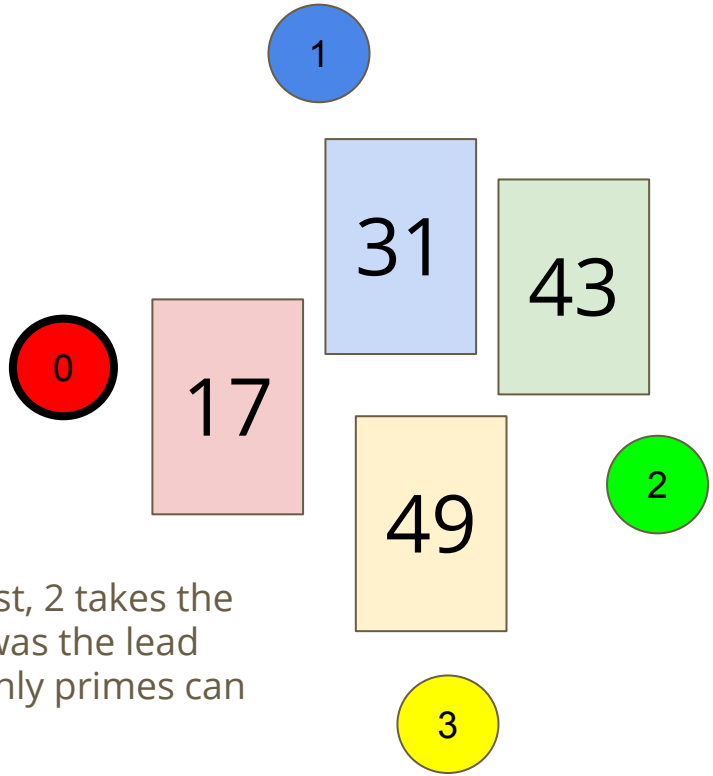
Who takes a trick?

One player will take the trick at the end of the round

- The highest card that correctly followed the lead card will take the trick
- If the first card was prime, the highest prime takes the trick
- If the first card was not prime, the highest cocomposite takes the trick
- The first card can possibly be the highest!

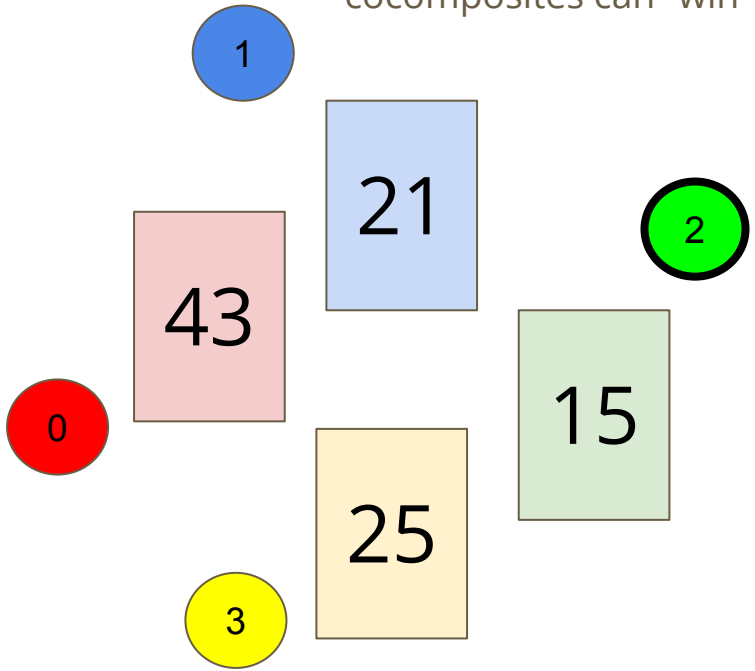
Taking a trick does not put the cards in your hand, they're just assigned to you in terms of points (more on this later)

Example rounds



0 goes first, 2 takes the trick. 17 was the lead card so only primes can “win”

2 goes first, 3 takes the trick. 15 was the lead card so only cocomposites can “win”



Scoring

Points are bad!

- The objective is to gain the least penalty points
- Penalty points are given to whoever takes tricks with primes in them
- For each prime card you take, you get one penalty point
- If you take The Douglas (card 42), you get 7 penalty point

Winning involves avoiding taking tricks with primes or The Douglas in them

Code for Coco

On Thursday:

- What is your Coco program expected to do?
- How to get started with the Assignment
- How we will be running the Assignment in terms of Tournaments and Marking

What did we learn today?

Arrays

- Two dimensional arrays

Functions

- Libraries
- Why we use functions

Coco

- How to play the game, Coco
- How the Assignment will run