Programming Studio 2 – C++

Week 6 Class 2

Dr Ruwan Tennakoon
Dr Steven Korevaar

RMIT UNIVERSITY

Week 6 Class 2 Summary

Download the starter code on canvas (Modules -> Week6 -> Class 2)

Abstract data types (ADTs)

- Field, Agent, and Path classes

Program efficiency

- Time, Memory, Disk I/O, CPU operations, Parallel vs Sequential programming
- Scalability (Big O notation)

Live coding exercise:

- Make an agent follow a random path, then go back to the starting position.
- Linked Lists!

RMIT UNIVERSITY

Today's Project

Random Walk Algorithm

Move an agent across an environment by taking random steps until you reach a goal.

1. Randomly set the agent's starting point inside the environment.
2. Take a random step (up, down, left, or right)
3. Check if the gold block is within the "neighbourhood" of the agent (surrounding 8 squares)
4. If Yes: DONE, print the path.
5. If No: GOTO 2.

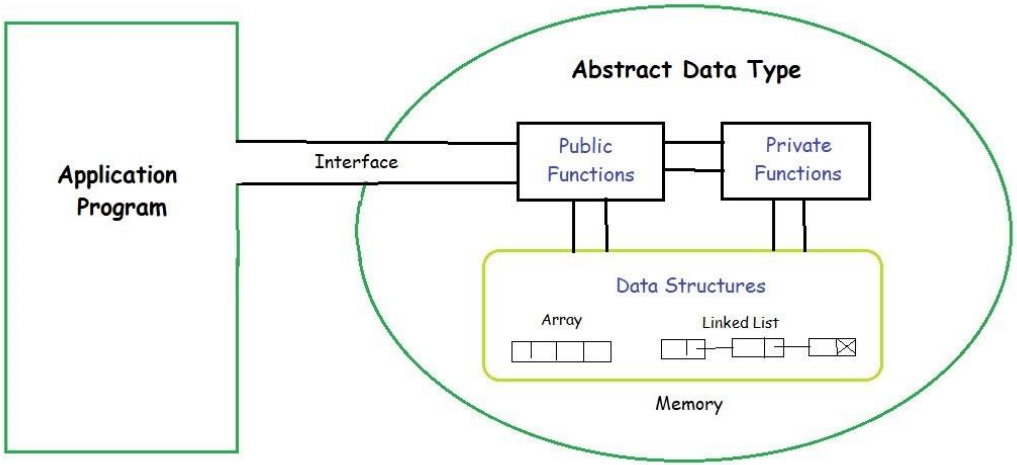
RMIT UNIVERSITY

Abstract Data Types

What is an ADT?

An object! An object that encapsulates data and provides an interface with which to use the data.

Make your assignment using these, this is a pretty big part of how we mark you.



Starter Code

Made up of 3 unfinished ADTs (classes):

Field

- Contains the information and procedures for building an environment
- Size, location, treasure location.
- Building the environment in Minecraft

Agent

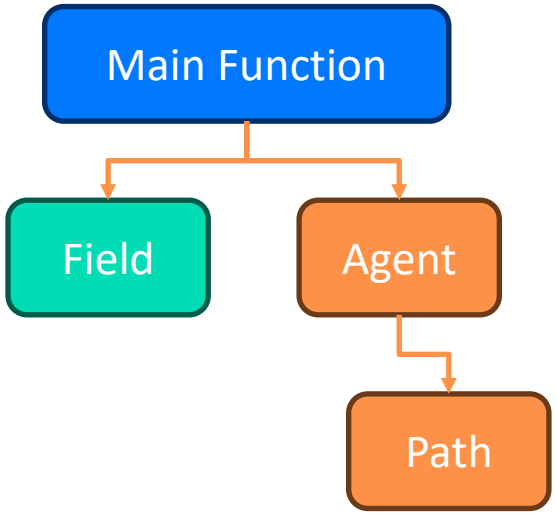
- Location, possible future moves, previous moves (path), checking environment for gold, and making moves

Path

- Contains a list of some sort with all the previous moves the agent has taken

Main File

- Responsible for creating instances of the ADTs
- Calling the functions in the appropriate order to solve the assigned problem



RMIT UNIVERSITY

Field.h

```
public:
    // Constructs a field by building concrete walls in a Minecraft world at a
    // given basepoint with sides equal to xLength and zLength.
    // Sets the treasure (Gold) at a random location within the field.
    Field(mcpp::Coordinate basePoint, unsigned int xLength, unsigned int zLength);
    ~Field();

    // Delete the existing walls and treasure and rebuild
    void resetField(void);

    // Provide a random location within the field
    mcpp::Coordinate getStartLocation(void) const;

private:
    // Data
    mcpp::Coordinate basePoint;
    unsigned int xLength;
    unsigned int zLength;
    mcpp::Coordinate treasureLoc;

    //private methods
    void BuildFence(mcpp::MinecraftConnection& mc);
    void eraseField();
    void SetupField(void);
```

RMIT UNIVERSITY

Agent.h

```
public:
    // Initialize an agent with ID at a given location.
    Agent(unsigned int id, mcpp::Coordinate location);

    ~Agent();

    //Move the agent randomly to one of the reachable adjacent blocks.
    //An adjacent cell is one within 1 unit in the x and y direction.
    // reachable if the cell is not more that 1 unit high to the current loc.
    // if no reachable locations, then returns false else returns true.
    bool randomStep(void);

    // Check the immediate neighbourhood for a given type of block
    // If the block is found, then treasureCoord is updated.
    bool isBlockInNeighborhood(mcpp::BlockType block, mcpp::Coordinate& treasureCoord);

    void Agent::printPath(void);

private:
    unsigned int id;
    mcpp::Coordinate location;
    bool checkMove(mcpp::Coordinate next);
    Path* path;
```

RMIT UNIVERSITY

Path.h

```
public:

    Path();
    ~Path();

    //Adds a node to path
    void pushCoordinate(mcpp::Coordinate loc);

    // return the current length of the path
    unsigned int getLength(void);

    /*
    Returns the last coordinate in path and removes it.
    Contract:
      Assume there are nodes in the path to pop
    */
    mcpp::Coordinate popCoordinate(void);


private:
    std::vector< mcpp::Coordinate* > savePath;
```

RMIT UNIVERSITY

Program Efficiency

How efficient is this random walk program?

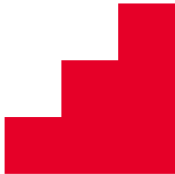
Program Efficiency

How efficient is this random walk program?

Not very? But why?

What aspects of efficiency are there?

RMIT UNIVERSITY



Program Efficiency

Process/Time Efficiency

How many “calculations” are required to complete a task given the input.

Memory Efficiency

How much memory is required to complete the task given the input.

Usually, we describe efficiency in terms of Big O Notation:

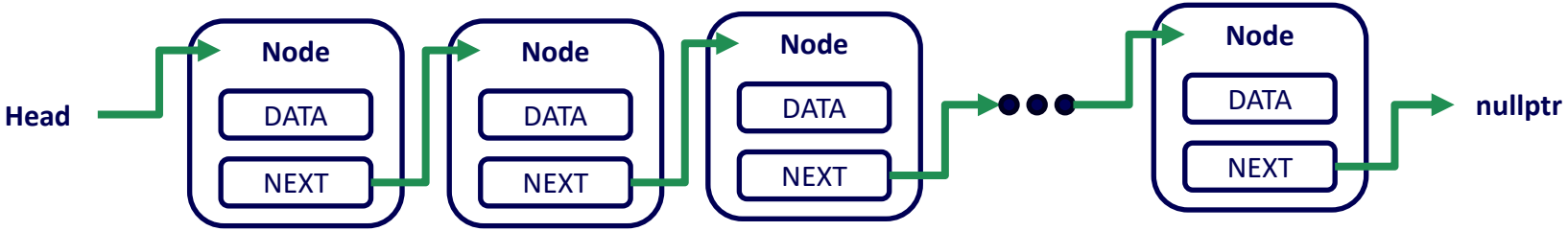
Basically: how many operations are required given an input size of n.

For example: **Bubble sort** has a worst-case complexity of $O(n) = n^2$: which, implies that for an input size of n , n^2 operations will need to be done to sort the n items.

RMIT UNIVERSITY

Linked Lists

A linked list is a sequence of “nodes”, where each node contains some data and a pointer to the next node.



Vectors and Linked Lists

Vectors:

Store all data sequentially in the heap.

Linked Lists:

Stores data non-sequentially, with pointers to the next element.

What kind of operations are better with vectors over linked lists?
Where do linked lists perform better than vectors?

Let's implement a linked list!

RMIT UNIVERSITY

Exercises for the day

1. **Go through and fix the ADTs/Classes in the starter code.**

2. **Implement a linked list instead of a vector in the starter code!**

3. **Prepare for checkpoint 2 tomorrow**

We will be checking **milestone 2** progress:

- Minimally functioning Base program: correctly functioning menu, reading the maze from the terminal, building the maze in Minecraft and placing the player in maze

RMIT UNIVERSITY