Week 4

Approaches – Psuedo Code

Last week

- Introduced the basic concepts
  - Sequential
  - Conditional branching
  - Looping

This week

- Methods to describe problems

Problem 2.2

- A routine to calculate a tip based on the overall service.

  - No tip if the service is poor
  - 10% tip if the service is okay
  - 15% tip if the service is good or better

Problem 2.2 – Problems

- What is poor, okay or good mean?
- How do we quantify these?
  - We could use marks out of 10.
    - <3 for poor
    - 3<=marks<=8 for okay
    - >8 for good
- What is our tip a percentage of?
  - We know this is going to be a percentage of the price of the bill.

Inputs needed

- Price of the meal
- Marks for service

Outputs

- Price + tip

Rules

- <3 for poor
- 3<=marks<=8 for okay
- >8 for good

User enters the price on the keyboard.

User enters the marks out of 10

If mark is less than 3 then Tip is 0;

If 3<=marks<=8 then Tip is the Price*0.1;

If marks>8 then Tip is Price*0.15

Display Tip
Psuedo code

- Input price
- Input marks
- If marks<3 for Tip=0;
- If 3<=marks<=8 then Tip =Price*0.1;
- If marks>8 then Tip=Price*0.15
- Display Tip

This is the start of psuedo code
- writing out our steps to be taken; in a form that is almost a programming language.
- We can refine this further

Further refinement

- Display message to the user to enter the price
- Input from the keyboard and stored in variable price
- Display message to the user to enter the marks
- Input from the keyboard and stored in variable marks
- If marks<3 for Tip=0;
- If 3<=marks<=8 then Tip =Price*0.1;
- If marks>8 then Tip=Price*0.15
- Display Tip

Set up variables price, marks;
Display “Please enter the price”;
Input price;
Display “Please enter the marks”;
Input marks;
If marks<3
  Tip=0;
If 3<=marks<=8
  Tip =Price*0.1;
If marks>8
  Tip=Price*0.15;
Display Tip;

Advantages

- Through incremental change it can fairly intuitive to go from plain English descriptions to almost programming code.

Disadvantages

- Describing loops and the relationships between statements in loop can be unclear.
- The way it is used can often be specific to the language used to write the programs.
Problem 4.1
- Returning to the square tracing project, but this time let's use a loop.
  - buggy.forward1(1000);
  - buggy.trturn(1000);
  - buggy.forward1(1000);
  - buggy.trturn(1000);
  - buggy.forward1(1000);
  - buggy.trturn(1000);
  - buggy.forward1(1000);
  - buggy.trturn(1000);
  - The lines below are repeated 4 times:
    - forward1(1000);
    - trturn(1000);
  - So if put these in a loop it could get them to repeat as many times as we want.

First pass
Loop four times
- buggy moves forward for 1000 milliseconds;
- buggy performs a tight turn to the right for 1000 milliseconds;
End loop

How does the routine know how many times to loop?

Second pass
counter=1;
Loop while counter <=4
  - forward1(1000);
  - trturn(1000);
  - counter=counter+1;
End loop

This is an acceptable place to stop with this problem.

Java Code
```java
public class week4_2{
    public static void main(String[] args)
    {
        int counter=1;
        robot_1 buggy=new robot_1();
        while (counter <=4)
        {
            buggy.forward1(1000);
            buggy.trturn(1000);
            counter=counter+1;
        }
    }
}
```

Problem 4.3
- Write a routine to get the robot to follow a black line on a white background.
- Two light sensors (left and right ones)
  - If a sensor is over the line it produces a true output.
  - The sensors are independent of each other so can produce different results depend on line position.
public class line_follower {
    public static void main (String[] args) {
        robot2 dick= new robot2();
        for(;;) {
            if ((dick.checkLight(1)==true)&&(dick.checkLight(2)==true)) {
                //if both sensor are on the line what action do you want to do?
            }
            if ((dick.checkLight(1)==false)&&(dick.checkLight(2)==false)) {
                //if both sensor are off the line what action do you want to do?
            }
            if ((dick.checkLight(1)==true)&&(dick.checkLight(2)==false)) {
                //if left sensor is on the line and right sensor is off what action do you want to do?
            }
            if ((dick.checkLight(1)==false)&&(dick.checkLight(2)==true)) {
                //if right sensor is on the line and left sensor is off what action do you want to do?
            }
        }
    }
}