16.070 Introduction to Computers and Programming

April 25 **Recitation 11** **Spring 2002**

Topics:

- Review for PS 9
    - ADT's
    - Handy Board Specs
- Part A – common problems
- Part B – getting started
- Other

Review for PS 9

Abstract Data Types:

- What is a Node?
    - Contains data
    - Points to the next node
    - Basic building block of ADT's
- What is a List?
    - Basic ADT
    - Has a Head, Nodes, and End of List (Optional)
    - Can Insert and Delete (Anywhere)
- What is a Queue?
    - Insert only to Tail
    - Delete only from Head
    - First-In-First-Out
- What is a Stack?
    - Insert only to Head
    - Delete only from Head
    - First-In-Last-Out
- What is a binary tree?
    - List where each node links to two other nodes
    - Head = Root
    - Branches
    - Last node on a branch = Leaf
    - Often ordered – optimizes searches

Hand Exercise 1: Insert '3' at the head of the list <head>1 10 4 15 <tail>

Hand Exercise 2: Delete '10' from the list.

Handy Board Specifications

Total RAM: _32KB____ = 2^15__ memory locations, each containing _8_ bits

Variable types and sizes on the Handy Board
Hint: Read the HB manual.

HB

Type	Size	Min Value	Max Value
int	16 bits	-32768	32767
long	32 bits	-2147483648	21474836437
float	32 bits	10^-38	10^38
char	8 bits	-128	127

What are the corresponding values with Visual C on the 33-202 workstations?
Hint: Use the sizeof() function, and the constants in the limits.h and float.h libraries.

VC

Type	Size	Min Value	Max Value
int	32 bits	-2147483648	21472836437
long	32 bits	-2147483648	21472836437
float	32 bits	10^-38	10^38
char	8 bits	-128	127

Part A – common problems

Serial Timing

- The workstation DOES NOT WAIT for incoming data. If there is nothing on the serial line, then sio() will return a zero and the variable you were reading into will not change. To make sure you read new data with the sio() function, you can use a loop. For example:

```
while(!sio(1,&inbyte, 1){}
```

Com Port

- The communication port (com port) your Handy Board is connected to may change from computer to computer. You can find out which com port you are connected to by either reading the label on the plug on the back of your workstation, or by checking which com port Interactive C is working on.

- Once you determine the correct com port, you should
  - o open “serial_ws.h”
  - o find the line that says “const LPCSTR LPSZPORTNUM = "com1";”. It should be right below the large section of comments.
  - o Change the string constant to “com1” or “com2” as appropriate.
  - o Save and close “serial_ws_h”.

Velocity Encoding

- Your controller should interpret a 13 in the velocity bits (telemetry byte of 01101???, where ?’s are wildcards) as the nominal speed.
- Chose the actual nominal speed (m/s) for your simulator.
- Encode that nominal speed as a binary 13 (01101)
- Scale all other speeds accordingly.

If your simulator generates the following velocities, how do you transmit them to the controller Handy Board and what does the Handy Board decode them as?

Simulator Velocity	Telemetry Byte	Handy Board Velocity
15		
2.6		
0.01		
-10		

General Tips

Step through your code by hand.

- Take one set of test cases, and walk though your code keeping track of all your variables.
- Pay particular attention to any symbolic constants and the IR code and Velocity values you get from the telemetry byte.
- Check where you write to the serial line, where you read to the serial line, where you set the command output byte value, and how you indicate the next state.

Part B

Step 1: Read the Instructions

Step 2: Interpret the Instructions

How do you get a command on the simulator?

What do you do with the command byte?

Command Byte	Action Taken
0	
1	
2	
4	
8	
32	
64	
128	

What telemetry do you send?

How do you send telemetry?

How do you use parity and the resend bit?

What are the simulation initial conditions?

When does the simulation end?

Does your simulation need any file or screen or graphical output?

Step 3: Get Your Simulator to Work

- If you had problems with your simulator from problem set 7, use these simplified test files to isolate your simulator's behavior. They should help you see which aspects of your simulator aren't working. Also, use the visual C debugger and break points to see how your variables change line by line (see lab week 10).

- Note: If you chose to use the solutions to problem set 7, BE VERY CAREFUL. There are extra variables and functions in the solution that are used to error-check the file input. When you start modifying the code, MAKE SURE YOU KNOW WHAT EVERY VARIABLE IS USED FOR AND WHERE IT IS USED.

File name(s)	Initial conditions	File Input	Expected Output

nothrust_B.dat
nothrust_C.dat
	Theta = Any
	Vx=Vy=0
	X=any, Y=any
		Time stamp and 0's in the thrust (and rotate) columns	Position and velocity should always be at initial conditions

allthrust_B.dat
allthrust_C.dat
	Theta = 0degrees
	Vx=Vy=0
	X=any, Y=any
		Time stamp and 1's in the thrust column (0's in the rotate column)	Only x-position and x-velocity should change. Velocity should increase by the same amount each time step.

allthrust_B.dat
allthrust_C.dat
	Theta = 90degrees
	Vx=Vy=0
	X=any, Y=any
		Time stamp and 1's in the thrust column (0's in the rotate column)	Only y-position and y-velocity should change. Velocity should increase by the same amount each time step.

rotateinplace_C.dat	Theta = Any
	Vx=Vy=0
	X=any, Y=any
		Time stamp, 0's in the thrust column, 0's in the rotate column	Angle will change but position and velocity will remain at initial conditions. Angle should increase by the same amount each time step.

Step 4: Implement everything in step 2

- Add one thing at a time to your simulator.

- Test that one thing thoroughly before moving on to the next task. Doing this will make part B go a lot smoother.

- Keep a record of any testing you do. That test information will also be useful in part D.

Example: Getting commands from the Handy Board

Because your controller software waits for a telemetry byte, you may want to use a simplified test harness to test the command input to your simulator.

The file located at http://web.mit.edu/16.070/www/labs/hb_testB.c is a test harness for sending commands to the Handy Board. This code sends the Thrust Forward command byte to the workstation when the start button is pressed, and sends the Thrust Backward command byte to the workstation when the stop button is pressed. The program will loop until the knob is moved to the 255 position.

Set up the Handy Board

- Load the serial_hb.c and the hb_testB.c files onto your Handy Board and close Interactive C.

Set up the simulator program

- Open a working copy of the simulator program, either your own or the solutions, in Visual C.
- Include the serial_ws.h library. Note: check what directory you have the serial_ws.h file saved in.
- Find the section of code where the commands are read from the input file. You may delete or comment out all code pertaining to the input file if you like.
- In the place that you used to read from the input file (fscanf(input_file, “%d %d %d”, time, thrust, rotate) or similar), write a loop that will wait for an input from the serial line. You will need to declare an unsigned char to hold the input. We will call it “command” in this exercise.

- Print the number you receive to the screen. This may help with debugging later on.
- Next, write an if-statement (or series of if-, else if-statements, or a function) that will translate command (which has a value of 0-128, see table above) into the appropriate thrust and rotate commands.

Run the programs

- Run the test harness on the Handy Board.
- Compile and run your modified simulator program. It should pause waiting for a Handy Board input.
- Try pressing the start and stop button and see how the simulator responds. Is everything connected correctly?
- Try running your simulator with the graphics package and sending commands with the Handy Board.